Feladatok

1. Aszimptotikus jelölések

1.1. Feladat Az f függvényre $f(n) = \Theta(n \log n)$. Lehetséges -e, hogy $f(5) = 5$? Lehetséges -e, hogy $f(n) = n$ minden prím esetén? A válaszokat indokoljuk!

1.2. Feladat Az f függvényre $f(n) = O(n \log n)$. Előfordulhatnak -e az alábbi esetek?

- $f(n) = n$, ha $n \leq 100$
- $f(n) = n^2$, ha $n \leq 100$
- $f(n) = n$ minden páros n esetén
- $f(n) = n^2$ minden páros n esetén

A válaszokat indokoljuk!

1.3. Feladat Adjunk meg olyan függvényt, amelyre $f(2n) = O(f(n))$.

1.4. Feladat Határozzuk meg az alábbi (f, g) függvénypárokban szereplő függvényekre milyen relációk írhatóak fel a $f = \Theta(g)$, $f = o(g)$, $f = O(g)$, $f = \omega(g)$, $f = \Omega(g)$ relációk közül!

- $(2^n, 2^{2n})$,
- $(n \log_{10} n, n \ln n)$,
- $(2^n, n!)$,
- $(n^n, n!)$,
- $(\sqrt{n^7}, n^3)$.

1.5. Feladat Legyenek $f(n)$ és $g(n)$ olyan pozitív függvények, amelyekre $f(n) = \Theta(g(n))$ teljesül. Legyen $h(n) = f(2n)$ és $p(n) = g(2n)$.

- Igaz -e, hogy $h(n) = \Theta(p(n))$?
- Igaz -e, hogy $h(n) = O(f(n))$? Mit mondhatunk abban az esetben, ha $f(n)$ monoton csökken?

- Igaz -e, hogy $h(n) = \Omega(f(n))$? Mit mondhatunk, ha $f(n)$ monoton növekszik?

1.6. Feladat Teljesül -e minden függvényre, hogy $f(n + 100) = O(f(n))$? Monoton növekvő vagy monoton csökkenő függvények esetén teljesül az $f(n + 100) = O(f(n))$ állítás?

1.7. Feladat Legyenek $g(n)$ és $h(n)$ olyan pozitív függvények, amelyekre $g(n) = \Theta(n)$ és $h(n) = \Theta(n)$ teljesül. Mely állítások igazak az alábbiak közül?

- Igaz -e, hogy $(g(n) + h(n)) = \Theta(n)$?
- Igaz -e, hogy $(|g(n) - h(n)| + 1) = \Theta(n)$?
- Igaz -e, hogy $(|g(n) - h(n)| + 1) = O(n)$?

2. Rekurziók megoldásai

2.1. Feladat Igazoljuk a helyettesítő módszerrel, hogy a $T(n) = 9T(n/3) + n$ rekurzióval megadott függvényre $T(n) = O(n^2)$. Az egészre kerekítésektől eltekinthetünk.

2.2. Feladat Igazoljuk a helyettesítő módszerrel, hogy a $T(n) = 2T(n/2)+n$ rekurzióval megadott függvényre $T(n) = O(n \log n)$. Az egészre kerekítésektől eltekinthetünk.

2.3. Feladat Igazoljuk új változó bevezetésének a segítségével, hogy a $T(n) = 2T(\sqrt{n}) + 1$ rekurzióval megadott függvényre $T(n) = O(\log n)$. Az egészre kerekítésektől eltekinthetünk.

2.4. Feladat A rekurziós fa alapján határozzuk meg a $T(n) = 4T(n/2) + n$ rekurzióval megadott függvény nagyságrendjét! Az egészre kerekítésektől eltekinthetünk.

2.5. Feladat A rekurziós fa módszerével igazoljuk, hogy a $T(n) = T(\alpha n) + T((1 - \alpha)n) + n$ rekurzióval megadott függvényre $T(n) = O(n \log_{1/\alpha} n)$, ha $1/2 \leq \alpha < 1$. Az egészre kerekítésektől eltekinthetünk.

2.6. Feladat Adjuk meg a mester tétel alapján az alábbi rekurzív módon megadott függvények nagyságrendjét!

- $P(n) = 8P(n/2) + n^2$
- $Q(n) = 4Q(n/2) + n^2$
- $R(n) = 2R(n/2) + n^2$

2.7. Feladat Adjunk példát tetszőleges $a \geq 1$ és $b > 1$ konstansok mellett olyan $T(n) = aT(n/b) + f(n)$ rekurzív egyenletre, ahol a mester tételben adott aszimptotikus korlátok egyike sem használható!

3. Legjobb, legrosszabb átlagos korlátok

3.1. Feladat Anna gondolt egy számot 1 és $2^n - 1$ között, és Beának ki kell találni. Minden tippjére megkapja a választ, eltalálta -e a számot, és amennyiben nem talált, megtudja, hogy a gondolt szám nagyobb vagy kisebb a tippjénél. Bea a következő eljárást használja (a és f korlátok, a gondolt szám mindig az (a, f) intervallumban lesz, g pedig a tipp):

Kezdetben $a = 0$, $f = 2^n$ a tipp pedig $g = 2^{n-1}$. Utána amíg nem talált:

- Ha a gondolt szám nagyobb, akkor
  $a := g$, $f := f$, $g := (a + f)/2$
- Ha a gondolt szám kisebb, akkor
  $a := a$, $f := g$, $g := (a + f)/2$

Határozzuk meg az algoritmus által feltett kérdések számának a nagyságrendjét a Θ jelölés segítségével a legjobb, a legrosszabb és az átlagos esetben!

3.2. Feladat Egy professzor elfelejti, hogy az egyetem parkolójában hol hagyta a kocsit. A parkolóban n kocsihely van, egy sorban helyezkednek el, az egyetem kijárata az első helynél helyezkedik el. A professzor nem ismeri fel a saját kocsiját, csak amikor másodjára alaposan megnézi. A professzor a következő stratégiát használja: kezdetben elmegy a parkoló másik széléig minden parkolóhelyre benéz, nem találja meg az autót, ezért újra megnézi az utolsó helyet, majd visszamegy a parkoló elejéig és így vizsgálja át az egész parkolót. Hányszor néz meg a professzor kocsihelyet a legjobb, a legrosszabb és az átlagos esetben?

3.3. Feladat Adott pozitív számok egy sorozata $c_1, \ldots, c_n$. A maximális elemet akarjuk, a következő egyszerű eljárással megtalálni. Az eljárás egy M (kezdetben 0) értéket használ, végigmegy a számokon, mindig összehasonlítja az M értéket az aktuális számmal, és ha nagyobb számot talált felülírja M értékét. Hány összehasonlítást végez az eljárás a legjobb, a legrosszabb és az átlagos esetben?

3.4. Feladat Adott az $1, 2, \ldots, n$ számoknak egy $(c_1 \ldots c_n)$ permutációja, a permutációban az 1 számot keressük. Az eljárás sorban összehasonlítja a számokat az 1 értékkel, amíg az 1 számot meg nem találja. Hány összehasonlítást végez az eljárás a legjobb, a legrosszabb és az átlagos esetben?

3.5. Feladat Egy hosszú országúton n benzinkút helyezkedik el, az i-edik kút a 2^i-es kilométerkőnél. Keressük egy benzinkutas barátunk, akiről nem tudjuk melyik kútnál dolgozik. A 0 kilométerkőnél indulunk és egyenesen megyünk

végig az országúton minden benzinkútnál megállva. Hány kilométer után találjuk meg a barátunk a legjobb, a legrosszabb és az átlagos esetben? Mi a válasz, ha a 2^n-es kilométerkőnél indulunk? Mindkét esetben hasonlítsuk össze a legrosszabb és az átlagos eset nagyságrendjét!

3.6. Feladat Egy 2^n kilométer hosszú országúton n benzinkút helyezkedik el, az i-edik kút a 2^i-es kilométerkőnél. Keressük egy benzinkutas barátunk, akiről nem tudjuk melyik kútnál dolgozik. A 2^{n-1} kilométerkőnél indulunk, először balra a legszélső (2 kilométernél) levő kútig, majd ha nem találtuk meg visszamegyünk jobbra az n-edik kútig. Hány kilométer után találjuk meg a barátunk a legjobb, a legrosszabb és az átlagos esetben? Hasonlítsuk össze a legrosszabb és az átlagos eset nagyságrendjét!

3.7. Feladat Egy 2^n kilométer hosszú országúton n benzinkút helyezkedik el, az i-edik kút a 2^i-es kilométerkőnél. Keressük egy benzinkutas barátunk, akiről nem tudjuk melyik kútnál dolgozik. A 2^{n-1} kilométerkőnél indulunk, először jobbra a legszélső (2^n kilométernél levő) kútig, majd ha nem találtuk meg visszamegyünk balra az első (2-es kilométer) kútig. Hány kilométer után találjuk meg a barátunk a legjobb, a legrosszabb és az átlagos esetben? Hasonlítsuk össze a legrosszabb és az átlagos eset nagyságrendjét!

4. Rekurzív algoritmusok

4.1. Feladat Jelölje $P(n)$ azt a számot, ahányféleképpen mehetünk fel egy n lépcsőfokból álló lépcsőn, ha egyszerre csak 1 vagy 2 lépcsőfokot léphetünk! Adjunk meg egy rekurzív algoritmust, amelyben az eljáráshívások száma $\Theta(P(n))$ és amely meghatározza a $P(n)$ értéket! Igazoljuk, hogy

$$P(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right).$$

4.2. Feladat Jelölje $R(n,k)$ azt a számot ahányféleképpen eljuthatunk egy $n \times k$ méretű sakktábla bal alsó sarkából a jobb felső sarkába, ha csak a jobbra, vagy a felfelé szomszédos mezőre léphetünk! Adjunk meg egy rekurzív algoritmust, amelyben az eljáráshívások száma $\Theta(R(n,k))$ és amely meghatározza $R(n,k)$ értékét! Igazoljuk, hogy $R(n,k) = \binom{n+k-2}{k-1}$.

4.3. Feladat n forintunk van. Minden nap veszünk pontosan egy dolgot a következők közül (zárójelben az ár szerepel) perec (1Ft), fagylalt (2Ft), csoki (2Ft). Adjunk meg egy rekurzív algoritmust, amely meghatározza, hogy hányféleképpen költhetjük el a pénzünket.

4.4. Feladat Egy n egységnyi súlyt elbíró teherautóba egy futószalagról tárgyakat pakolunk. Három féle doboz jöhet, zárójelben a súlyuk egységben mérve: A típus (1 egység), B típus (2 egység), C típus (2 egység). Adjunk meg egy rekurzív algoritmust, amely meghatározza, hogy hányféle különböző dobozsorozattal tölthetjük meg teljesen a teherautót.

4.5. Feladat Adottak építőkockák, van 1 egység magas, van 2 egység magas kék, van 2 egység magas piros, és van 3 egység magas. (Mind a négy fajta kockából tetszőlegesen sok van). Adjunk meg egy dinamikus programozási algoritmust, amely kiszámolja, hogy hányféleképpen építhető fel egy n magasságú torony.

4.6. Feladat Adjunk meg egy rekurzív algoritmust, amely meghatározza az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok páronként különbözőek.

4.7. Feladat Adjunk meg egy rekurzív algoritmust, amely meghatározza, az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok mindegyike páratlan.

4.8. Feladat Adjunk meg egy rekurzív algoritmust, amely meghatározza, az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok mindegyike legalább 2.

4.9. Feladat Adjunk meg egy rekurzív algoritmust, amely meghatározza, az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok mindegyike páratlan és a számok páronként különbözőek.

4.10. Feladat A hanoi torony probléma: Három pálca egyikén n korong van a többi üres. A korongok nagyság szerinti sorrendben helyezkednek el, alul van a legnagyobb. Át akarjuk helyezni a korongokat egy másik pálcára a következő szabályok alapján. Egyszerre csak egy korong mozgatható. A korong vagy üres pálcára vagy egy nála nagyobb korongra helyezhető. Oldjuk meg a feladatot egy rekurzív algoritmussal! Határozzuk meg a korongmozgatások számát!

5. Absztrakt adattípusok, adatszerkezetek

5.1. Feladat Képezzünk az $S_1 = \langle a_1, \ldots, a_n \rangle$ és $S_2 = \langle b_1, \ldots, b_n \rangle$ Sorokból egy $S_3 = \langle a_1, \ldots, a_n, b_1, \ldots, b_n \rangle$ és egy $S_4 = \langle a_1, b_1, \ldots, a_i, b_i, \ldots, a_n, b_n \rangle$ Sort a Sor absztrakt adattípus műveleteinek felhasználásával.

5.2. feladat Adott az $(a_1, \ldots, a_n)$ szám n-es. S-ben Sorként, T-ben Tömbként ábrázolva. Teljesül, az $a_1 \leq a_n$ feltétel. Keressünk egy olyan a_i elemet, amelyre $a_i \leq a_{i+1}$. Igazoljuk, hogy a sor esetén ez legrosszabb esetben $\Omega(n)$ adattípus műveletet igényel! Adjunk meg egy eljárást, amely a Tömb adattípus esetén $O(\log n)$ adattípus művelet segítségével legrosszabb esetben is megtalál egy ilyen indexet!

5.3. Feladat Adott $L_1 = \langle\rangle\langle a_1, a_2, \ldots, a_n \rangle$ és $L_2 = \langle\rangle\langle b_1, b_2, \ldots, b_n \rangle$ Listákból készítsünk egy $L_3 = \langle\rangle\langle a_1, b_1, a_2, b_2, \ldots, a_n, b_n \rangle$ listát a Lista absztrakt adattípus műveleteinek felhasználásával!

5.4. Feladat Adott az $< 1, 4, 5, 6, 2, 7 >$ permutáció. Hány "ADATTIPUSBOL" műveletet követően kaphatjuk meg a 6 elemet, ha a permutáció Veremben, Sorban, Prioritási Sorban van tárolva?

5.5. Feladat Egy V Veremben kezdetben egyetlen szám az 1 van. Utána n-szer egymás után végrehajtjuk a

```
VeremBol(V,x);
y:=2x;
z:=2x+1;
VeremBe(V,y);
VeremBe(V,z);
```

műveletsort. Mi lesz a műveletsorok végrehajtása után a VeremBol(V,x) művelet eredménye? Mi lesz az eredmény, ha Sort használunk Verem helyett? Mi lesz az eredmény ha prioritási sor használunk?

5.6. Feladat Egy V Veremben kezdetben egyetlen szám az 1 van. Utána n-szer egymás után végrehajtjuk a

```
VeremBol(V,x);
y:=2x;
z:=2x+1;
VeremBe(V,z);
VeremBe(V,y);
```

műveletsort. Mi lesz a műveletsorok végrehajtása után a VeremBol(V,x) művelet eredménye? Mi lesz az eredmény, ha Prioritási sort használunk Verem helyett?

5.7. Feladat Egy fát úgy adunk meg, hogy minden cellát azonosítunk a hozzá vezető úttal. A pontok: a gyökér $g = \lambda$, $a = 1, b = 2, c = 3, d = 1.1, e = 1.2, h = 2.1, j = 2.3, k = 3.1, l = 1.1.1, m = 1.1.2, n = 1.1.3$. Rajzoljuk fel a fát! Határozzuk meg a fa magasságát! Adjuk meg a fát azon f függvénnyel, amelyre $f(x)$ x fiait adja meg!

5.8. Feladat Egy fát az algebrai definíciója alapján adunk meg. A fa $F = g(a(d, e(k)), b, c(h, j))$. Rajzoljuk fel a fát! Soroljuk fel a fa leveleit! Adjuk meg a fát elsőfiú-testvér kapcsolattal!

6. Fák, fabejárások

6.1. Feladat Írjunk olyan rekurzív függvényeljárást, amely kiszámítja egy elsőfiú-testvér ábrázolású fa magasságát!

6.2. Feladat Írjunk olyan rekurzív függvényeljárást, amely kiszámítja egy kapcsolati tömb ábrázolású fa magasságát!

6.3. Feladat Írjunk olyan rekurzív függvényeljárást, amely kiszámítja egy kapcsolati lánc ábrázolású fa magasságát!

6.4. Feladat Írjunk olyan nemrekurzív függvényeljárást, amely kiszámítja egy elsőfiú-testvér-apa ábrázolású fa magasságát!

6.5. Feladat Írjunk olyan rekurzív függvényeljárást, amely adott k paraméterre kiszámítja, hogy a fának hány pontja van a k-adik szinten (tehát azon p pontok száma, amelyekre $d(p) = k$)! A fa elsőfiú-testvér ábrázolású.

6.6. Feladat Írjunk olyan rekurzív függvényeljárást, amely adott k paraméterre kiszámítja, hogy a fának hány pontja van a k-adik szinten (tehát azon p pontok száma, amelyekre $d(p) = k$)! A fa kapcsolati tömb ábrázolású.

6.7. Feladat Írjunk olyan rekurzív függvényeljárást, amely adott k paraméterre kiszámítja, hogy a fának hány pontja van a k-adik szinten (tehát azon p pontok száma, amelyekre $d(p) = k$)! A fa kapcsolati lánc ábrázolású.

6.8. Feladat Írjunk olyan rekurzív függvényeljárást, amely kiszámítja a fa leveleinek számát! A fa elsőfiú-testvér ábrázolású.

6.9. FeladatÍrjunk olyan rekurzív függvényeljárást, amely kiszámítja a fa leveleinek számát! A fa kapcsolati tömb ábrázolású.

6.10. Feladat Írjunk olyan rekurzív függvényeljárást, amely kiszámítja a fa leveleinek számát! A fa kapcsolati lánc ábrázolású.

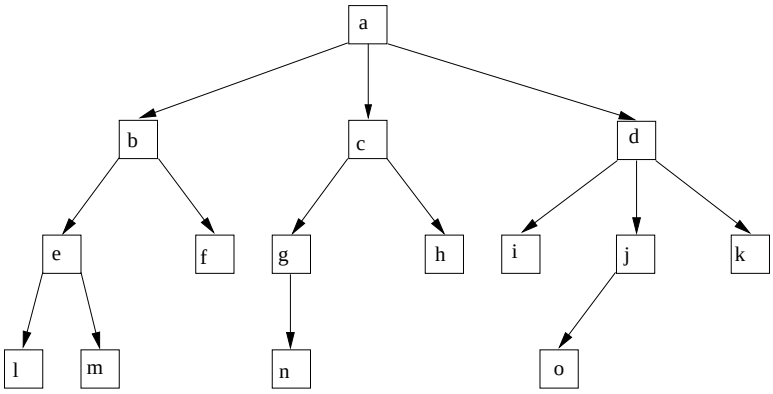
6.11. Feladat Írjunk olyan nemrekurzív függvényeljárást, amely kiszámítja a fa leveleinek számát! A fa elsőfiú-testvér-apa ábrázolású.

6.12. Feladat Írjuk meg a preorder bejárás algoritmusát, ha a fa kapcsolati tömb ábrázolású!

6.13. Feladat Írjuk meg a preorder bejárás algoritmusát, ha a fa kapcsolati lánc ábrázolású!

6.14. Feladat Írjuk meg a szintszerinti bejárás algoritmusát, ha a fa kapcsolati tömb ábrázolású!

6.15. Feladat Írjuk meg a szintszerinti bejárás algoritmusát, ha a fa kapcsolati lánc ábrázolású!



1. ábra. Példa fa

6.16. Feladat Mi a bejárási sorrend a példa fára, ha a rekurzív preorder fabejáró algoritmusban felcseréljük a két rekurzív hívás sorrendjét?

6.17. Feladat Mi a bejárási sorrend a példa fára, ha a szintszerinti bejárás algoritmusában Sor helyett Vermet használunk?

7. Dinamikus programozás

7.1. Feladat Jelölje $P(n)$ azt a számot, ahányféleképpen mehetünk fel egy n lépcsőfokból álló lépcsőn, ha egyszerre csak 1 vagy 2 lépcsőfokot léphetünk! Adjunk meg egy dinamikus programozási eljárást (lineáris táblázatkitöltéssel) a $P(n)$ érték kiszámítására! Igazoljuk, hogy a futási idő lineáris.

7.2. Feladat Jelölje $R(n,k)$ azt a számot ahányféleképpen eljuthatunk egy $n \times k$ méretű sakktábla bal alsó sarkából a jobb felső sarkába, ha csak a jobbra, vagy a felfelé szomszédos mezőre léphetünk! Adjunk meg két dinamikus programozási eljárást (egyet négyzetes, egyet lineáris táblázatkitöltéssel) az $R(n,k)$ érték kiszámítására! Igazoljuk, hogy a futási idő $O(n \cdot k)$.

7.3. Feladat n forintunk van. Minden nap veszünk pontosan egy dolgot a következők közül (zárójelben az ár szerepel) perec (1Ft), fagylalt (2Ft), csoki (2Ft). Adjunk meg egy dinamikus programozási eljárást (lineáris táblázatkitöltéssel) a $P(n)$ érték kiszámítására! Igazoljuk, hogy a futási idő lineáris!

7.4. Feladat Egy n egységnyi súlyt elbíró teherautóba egy futószalagról tárgyakat pakolunk. Három féle doboz jöhet, zárójelben a súlyuk egységben mérve: A típus (1 egység), B típus (2 egység), C típus (2 egység). Adjunk meg egy dinamikus programozási algoritmust, amely meghatározza, hogy hányféle különböző dobozsorozattal tölthetjük meg teljesen a teherautót!

7.5. Feladat Adjunk meg egy dinamikus programozási algoritmust, amely meghatározza az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok páronként különbözőek! Mennyi az algoritmus futási ideje?

7.6. Feladat Adjunk meg egy dinamikus programozási algoritmust, amely meghatározza, az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok mindegyike páratlan! Mennyi az algoritmus futási ideje?

7.7. Feladat Adjunk meg egy dinamikus programozási algoritmust, amely meghatározza, az n szám azon partícióinak a számát, amelyben a partícióban szereplő számok mindegyike legalább 2.

7.8. Feladat Egy társaság hierarchikus rendszerben dolgozik, azaz a főnök beosztott reláció egy fát alkot. A társaság egy fogadást szervez. Minden alkalmazotthoz hozzá van rendelve egy kedélyességi mérték. A parti résztvevőit, úgy akarják kiválasztani, hogy egyetlen alkalmazottnak se legyen ott a közvetlen főnöke és az összkedélyesség a maximális legyen.

- Adjunk megy egy dinamikus programozási algoritmust a meghívandók listájának elkészítésére!

- Miként érhetjük el, hogy a vállalat főnöke biztos kapjon meghívást?

7.9. Feladat Egy adott hátizsákba tárgyakat akarunk pakolni. Minden tárgynak van egy fontossági értéke (f_i), és egy súlya (s_i), a hátizsákba maximum

összesen S súlyt pakolhatunk. Az s_i és S értékek egészek. Szeretnénk úgy választani tárgyakat, hogy az összfontosság maximális legyen. Tehát feladatunk, hogy kiválasszuk a tárgyaknak olyan halmazai közül, amelyekre az összsúly nem haladja meg S-t azt, amelyre maximális az összfontosság. Adjunk meg egy dinamikus programozási eljárást a feladat megoldására!

7.10. Feladat Hányféleképp tudunk n darab, egymást nem metsző, $(x, 0)$ középpontú kört rajzolni úgy, hogy az x tengelyt 1 és $2n$ közti egész koordinátákon metsszék? Adjunk egy dinamikus programozási eljárást ami kiszámolja a választ!

7.11. Feladat Adott egy n lépcsőfokból álló öreg lépcső. Néhány lépcsőfok nagyon elkorhadt, ezekre nem léphetünk, mert leszakad. Egyszerre 1 vagy 2 lépcsőt tudunk lépni. Adjunk meg egy dinamikus programozási eljárást, amely kiszámolja hányféleképp mehetünk fel a lépcsőn.

7.12. Feladat Vegyünk egy konvex n-szöget. Legyenek a csúcsai $A_1, \ldots, A_n$. Minden $A_iA_jA_k$ háromszöghöz adott egy $w(A_iA_jA_k)$ érték. Határozzuk meg az n-szög azon átlók általi háromszögekre bontását, amelyre a felbontásban szereplő háromszögekre vett w értékek összege minimális. Adjunk meg egy dinamikus programozási eljárást!

7.13 Feladat Egy sorozat részsorozatainak azokat a sorozatokat nevezzük, amelyeket a sorozatból bizonyos elemek elhagyásával megkaphatunk. (Minden sorozat részsorozata saját magának). Adjunk meg olyan algoritmust, amely két sorozatra meghatározza a leghosszabb közös részsorozatukat!

7.14. Feladat Határozzuk meg az (a, b, b, a, b, a, b, a) és $(b, a, b, a, a, b, a, a, b)$ sorozatok leghosszabb közös részsorozatát!

7.15. Feladat Vegyük a következő ütemezési problémát. Adott három gép és munkák, minden munkának van egy p_i végrehajtási ideje. Feladatunk a munkákat a gépekhez rendelni úgy, hogy a gépekhez rendelt munkák végrehajtási idejeinek összegének (mind a három gépre kiszámolva ezt az összeget) a maximuma minimális legyen.

7.16. Feladat Egy szót tükörszónak nevezünk, ha előre és visszafele olvasva is ugyanazt a szót kapjuk. Adjunk meg egy dinamikus programozási algoritmust, amely meghatározza, hogy minimálisan hány betűt kell beszúrni egy szóba, hogy tükörszó legyen.

7.17. Feladat Adott egy $k \times n$-es tábla. Minden mezőre meg van adva egy c_{ij} pozitív szám, ami a mezőről begyűjthető érték. Egy játékos a bal alsó sarokból szeretne eljutni a jobb felső sarokba úgy, hogy csak jobbra és felfelé léphet szomszédos mezőre. Az útja során, összegyűjtheti a mezőkről az értékeket. Mi az az útvonal, amivel a maximális értéket tudja összegyűjteni?

8. Mohó algoritmusok

8.1. Feladat Adott a valós számegyenesen valós pontoknak egy $x_1, \ldots, x_n$ halmaza. Adjunk meg hatékony algoritmust, amely megad minimális számú

egységnyi hosszú intervallumot, amelyek tartalmazzák a pontokat. Igazoljuk az algoritmus helyességét!

8.2. Feladat Egy hosszú utazást tervezünk autóval. Az autó tankjában n liter benzin fér el. Van egy térképünk, amely ábrázolja az úton levő benzinkutakat. Tervezzünk hatékony algoritmust, amely megadja azon benzinkutakat, amelyeknél megállva, minimális számú megállással megoldható az utazás!

8.3. Feladat Mi az optimális Huffman kódja az alábbi abc-nek, $f(a) = 1, f(b) = 1, f(c) = 2, f(d) = 3, f(e) = 5, f(f) = 8, f(g) = 13$. Általánosítsuk arra az esetre, amikor az ábécé i-edik betűjének a gyakorisága az $i-1$-edik és $i-2$-edik gyakoriságok összege.

8.4. Feladat Bizonyítsuk be, hogy minden optimális prefix kódra igaz, hogy ha a karaktereket gyakoriság szerint szigorúan csökkenő sorrendbe helyezzük, akkor a megfelelő kódszavak hosszai nemcsökkenő sorozatot képeznek.

8.5. Feladat Vegyük a pénzváltási probléma azon változatát, ahol a lehető legkevesebb érmére akarunk felváltani n forintot.

a) Tervezzünk mohó algoritmust, ha a felhasználható érmék 1, 5, 10 és 25 forintok. Igazoljuk, hogy az algoritmus optimális!

b) Tervezzünk mohó algoritmust, ha a felhasználható érmék $c^0, c^1, \ldots c^k$ forintok, valamely $c > 1$, $k > 1$ egészekre. Igazoljuk, hogy az algoritmus optimális!

c) Adjunk meg olyan érmesorozatot, amelyre a mohó algoritmus nem optimális!

8.6. Feladat Határozzuk meg milyen $a < b$ számokra optimális a mohó algoritmus az $(1, a, b)$ pénzérmékre a pénzváltási problémában!

8.7. Feladat Adott zárt intervallumoknak egy $\{[a_1, b_1], \ldots, [a_n, b_n]\}$ halmaza. Keressünk olyan minimális számú ponthalmazt, amelynek minden intervallumba esik legalább egy pontja.

8.8. Feladat Vegyük azt az ütemezési problémát, ahol egy gép van, minden munkának van egy végrehajtási ideje és egy határideje. A cél a maximális késedelem (befejezési időből kivont határidő) minimalizálása. Igazoljuk, hogy a mohó algoritmus, amely mindig a legkisebb határidővel rendelkező még nem végrehajtott munkát hajtja végre optimális.

9. Bactrack és $B\&B$

9.1. Feladat Oldjuk meg backtracking algoritmussal az alábbi optimális pénzváltási problémát (nem bináris fával reprezentáljuk a megoldástért)! A váltópénzek $5, 4, 4, 2, 1$ a felváltandó összeg 8.

9.2. Feladat Adjunk meg egy backtracking algoritmust a hátizsák problémára! Definiáljuk a megoldástér fáját, és a problémaspecifikus URESX, EFIU, APA, TESTVER, VISSZAALLIT, LEHETMEGO, MEGOLDAS műveleteket!

9.3. Feladat Oldjuk meg a fenti algoritmussal az alábbi hátizsák problémát (nem bináris fával reprezentáljuk a megoldástért)! A tárgyak (súly, fontosság) párokban $(4,6)$ $(3,5)$ $(2,3)$ $(2,3)$ a hátizsák kapacitása 8.

9.4. Feladat Oldjuk meg az elágazás-korlátozás módszerrel az alábbi optimális pénzváltási problémát (nem bináris fával reprezentáljuk a megoldástért, az ADAGOLO felső korlát szerinti minimumos prioritási sor)! A váltópénzek $5,4,4,2,1$ a felváltandó összeg 8.

9.5. Feladat Adjunk meg egy elágazás-korlátozás algoritmust a hátizsák problémára! Definiáljuk a megoldástér fáját, és a problémaspecifikus URESX, EFIU, TESTVER, LEHETMEGO, MEGOLDAS műveleteket, az alsókorlát és felsőkorlát függvényeket!

9.6. Feladat Oldjuk meg a fenti algoritmussal az alábbi hátizsák problémát (nem bináris fával reprezentáljuk a megoldástért)! A tárgyak (súly, fontosság) párokban $(4,6)$ $(3,5)$ $(2,3)$ $(2,3)$ a hátizsák kapacitása 8.

10. Gráfok, gráfábrázolások

10.1. Feladat Adott egy társaságban 6 ember. Igazoljuk, hogy vagy van 3 akik közül mindenki ismer mindenkit vagy van három, akik közül senki sem ismer senkit. Reprezentáljuk a problémát gráffal, a pontok az emberek és két pont össze van kötve, ha ismerik egymást. Mit jelent a fenti állítás a 6 ponttal rendelkező gráfokra?

10.2. Feladat Adott n darab munka, és n munkás. Minden munka-munkás párosra adott egy c_{ij} érték, amely megadja, hogy mennyi a költség, ha az i-edik munkát a j-edik munkás hajtja végre. Célunk a munkák elvégzése a legkisebb költséggel.

a) Miként reprezentálható a probléma egy súlyozott gráffal?

b) Miként kaphatunk egy optimális megoldást?

c) Miként változik a feladat a gráf reprezentációban, ha kikötjük, hogy minden munkásnak legalább egy munkát el kell végeznie?

d) Miként változik a feladat a gráf reprezentációban, ha minden munkásra vannak feladatok, amiket nem tud végrehajtani?

10.3. Feladat Adott golyóknak egy halmaza, amely k_1 darab kék p_1 darab piros és z_1 darab zöld golyót tartalmaz. Egy játékos a következő lépéseket teheti:

- Elvehet egyszerre két piros és három zöld golyót.
- Elvehet egyszerre három piros és két kék golyót.
- Elvehet egyszerre három kék és két zöld golyót.

A kérdés az, hogy elérheti -e, hogy néhány lépés után az asztalon k_2 darab kék p_2 darab piros és z_2 darab zöld golyó legyen, ha igen minimálisan hány lépést kell tegyen? Reprezentáljuk a feladatot egy gráffal!

10.4. Feladat Adott egy sziget ahol 42 kaméleon lakik. A kaméleonok három színt vehetnek fel piros, kék és zöld. Ha két különböző színű kaméleon találkozik, akkor megijednek és mindkettő átváltoztatja a színét a harmadik színre. A szigeten a kiindulási időpontban 13,14,15 darab piros zöld és kék kaméleon van. Reprezentáljuk a feladatot gráffal! Igazoljuk, hogy ebből a kiindulási helyzetből nem érhetjük el, hogy minden kaméleonnak ugyanaz legyen a színe.

10.5. Feladat Adott egy sziget ahol fejlettebb kaméleonok élnek, mint azon, amit az előző feladatban néztünk. A kaméleonok négy színt vehetnek fel piros, kék, zöld és fehér. Ha három különböző színű kaméleon találkozik, akkor megijednek és mindhárom átváltoztatja a színét a negyedik színre. Előfordulhat, hogy egyszerre négy kaméleon találkozik, amelyeknek mindegyiküknek különböző a színe, ekkor méregbe gurulnak és mind a négy kaméleon felrobban. Egyéb találkozások esetén semmi nem történik. A szigeten a kiindulási időpontban (a, b, c, d) darab piros, zöld, kék és fehér kaméleon van. A természetvédők szeretnék tudni előfordulhat -e, hogy kihalnak a szigetről a kaméleonok. Adjunk algoritmust, amellyel eldönthetik. Reprezentáljuk a feladatot gráffal!

10.6. Feladat Adott egy $n \times n$-es sakktábla. Az $(1, 1)$ mezőn áll egy huszár. Határozzuk meg eljuthat -e az (u, v) mezőre, ha igen adjunk meg egy legkevesebb lépésből álló utat! Reprezentáljuk a feladatot egy gráffal!

10.7. Feladat Egy G irányított gráfnak a pontjai $V = \{1, \ldots, 6\}$, az élei $E = \{(1,2), (1,4), (1,6), (2,2), (2,3), (2,4), (3,1), (3,5), (4,1), (4,4), (5,2), (5,3), (5,6), (6,1), (6,2), (6,5)\}$. Adjuk meg a gráfot kapcsolatmátrix, éllista, halmazfüggvény reprezentációval!

11. Szélességi, mélységi keresés,

11.1. Feladat Egy G irányított gráfnak a pontjai $V = \{1, \ldots, 6\}$, az élei $E = \{(1,2), (1,4), (1,6), (2,2), (2,3), (2,4), (3,1), (3,5), (4,1), (4,4), (5,2), (5,3)$ $(5,6), (6,1), (6,2), (6,5)\}$. Adjuk meg milyen sorrendben járja be a szélességi keresés a pontokat, az 1 illetve a 5 pontokból kiindulva. A bejárások alapján adjuk meg a $\delta(1,5)$ és $\delta(5,1)$ értékeket!

11.2. Feladat Adott egy labirintus térképe. A labirintust úgy ábrázoljuk, hogy a labirintus egy négyzetrács, ahol minden mező vagy fal vagy üres. Feltehetjük, hogy a labirintus egy $k \times n$-es boolean mátrix által van megadva (1 jelöli az üres mezőt). A feladatunk olyan algoritmus tervezése, amely tetszőleges belső pontból a legrövidebb úton kitalál a labirintusból. (Használjuk a szélességi keresés algoritmusát és a számított gráf reprezentációt!)

11.3. Feladat Tegyük fel, hogy a 11.2. feladatban a labirintust a következő mátrix írja le.

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Hajtsuk végre a fentiekben kidolgozott eljárást a $(4,3)$ illetve a $(2,5)$ pontokból indulva!

11.4. Feladat Adjuk meg miként oldhatóak meg a 10.3., 10.4., 10.5. és 10.6. feladatokban leírt problémák a szélességi keresés segítségével!

11.5. Feladat Tegyük fel, hogy a 10.6. feladatban a huszár az $(1,1)$ mezőn áll. Határozzuk meg a szélességi kereséssel, hogy minimálisan hány lépést kell tennie, hogy az átlósan szomszédos $(2,2)$ mezőre érkezzen!

11.6. Feladat Egy G irányított gráfnak a pontjai $V = \{1, \ldots, 9\}$, az élei $E = \{(1,2),(1,3),(2,3),(2,4),(3,5),(5,2),(5,6),(6,7),(7,5),(8,9),(8,1),(9,8)$. Hajtsuk végre a gráfon a mélységi keresés algoritmusát! Adjuk meg a pontoknak az elérési és elhagyási idejét! Osztályozzuk az éleket (faél, visszaél, előreél, keresztél)!

11.7. Feladat Adott egy irányított gráf, csúcsai $1,2,3,4$. A gráfban adottak az $(1,2),(1,3),(2,4),(3,4)$ irányított élek, van még él az 1 és 4 továbbá a 2 és 3 csúcsok között de ezeknek az éleknek az irányítását nem ismerjük. Végrehajtunk az 1 pontból egy mélységi keresést. Milyen típusba eshetnek a gráf még irányítatlan élei az irányításuktól függően?

11.8. Feladat Tudjuk, hogy egy labirintusban vagyunk. A labirintus alapja négyzetrács, ahol minden mező vagy fal vagy üres. Mi mindig csak a velünk szomszédos négy mezőt látjuk (balra, jobbra, előre, hátra). Van nálunk kréta, amellyel jeleket tehetünk azokra a mezőkre ahol járunk. Tervezzünk eljárást, amellyel kitalálunk a labirintusból. (Használjuk a mélységi keresés algoritmusát és a számított gráf reprezentációt!)

11.9. Feladat Tegyük fel, hogy az előző feladatban a labirintust a következő mátrix írja le. (1 az üres mező, 0 a fal).

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Hajtsuk végre a fentiekben kidolgozott eljárást a $(4,3)$ illetve a $(2,5)$ pontokból indulva!

12. Topologikus rendezés, összefüggőség, Euler kör

12.1. Feladat Egy G irányított gráfnak a pontjai $V = \{1, \ldots, 9\}$, az élei $E = \{(1,2), (1,3), (2,3), (2,4), (3,5), (5,2), (5,6), (6,7), (7,5), (8,9), (8,1), (9,8)\}$. Határozzuk meg a mélységi keresés alapján gráf erősen összefüggő komponenseit!

12.2. Feladat Egy munkásnak az A, B, C, D, E, F, G, H munkákat kell végrehajtania. A munkák között vannak bizonyos relációk $X \to Y$ azt jelenti, hogy a X-et Y előtt kell végrehajtani a munkásnak. A következő precedencia feltételeink vannak $A \to H$, $B \to H$, $D \to E$, $D \to F$, $E \to C$, $E \to G$, $F \to C$, $G \to A$. Reprezentáljuk gráffal a problémát! Adjunk meg egy lehetséges végrehajtási sorrendet (a topologikus rendezési algoritmus alapján)! Mi történik, ha a $H \to E$ relációnak is fenn kell állnia?

12.3. Feladat Egy G irányított gráfnak a pontjai $V = \{1, \ldots, 6\}$, az élei $E = \{(1,2), (1,3), (2,3), (2,4), (3,1), (3,4), (3,5), (4,1), (4,6), (5,3), (5,6), (6,2), (6,5)\}$. Adjunk meg egy Euler körét, ha van! A H irányítatlan gráfnak a pontjai $V = \{1, \ldots, 6\}$, az élei $(1,2), (1,3), (1,4), (1,6), (2,3), (2,4), (3,4), (4,5), (5,6)$. Adjunk meg a gráfnak egy Euler körét ha van! Van -e a gráfnak Euler útja?

12.4. Feladat Adott egy város úthálózata. A városban minden utca egyirányú. Egy kereskedő vállalat olyan raktárhelyeket akar telepíteni, amelyekre minden útkereszteződés elérhető és a kereszteződéshez kiküldött kocsi vissza tud térni a raktárhelyre a szabályok betartásával. Adjunk meg egy eljárást, amely meghatározza minimálisan hány raktárt kell létrehozni!

12.5. Feladat Igazoljuk, hogy ha egy irányított gráfban nincsen kör, akkor a gráfnak van olyan csúcsa, amelynek a befoka 0, ezeket a pontokat forrásnak nevezzük. Ezen észrevétel alapján adjunk meg egy eljárást, amely az aktuális fokszámok alapján, egyenként törölve a csúcsokat megad a körmentes gráfok csúcsaira egy topologikus rendezést.

12.6. Feladat Igazoljuk, hogy ha egy egy darab erősen összefüggő komponensből álló irányított gráfban minden pont fokszáma pontosan kettő, akkor a gráf egy irányított körút.

13. Feszítőfák

13.1. Feladat Egy G irányítatlan gráfnak a pontjai $V = \{a, b, c, d, e, f\}$, az élei a súlyokkal együtt $E = \{c(a,b) = 1, c(a,c) = 2, c(b,c) = 3, c(b,e) = 2, c(c,d) = 3, c(c,e) = 1, c(d,e) = 2, c(d,f) = 1, c(e,f) = 3\}$. Hajtsuk végre a Kruskal és a Prim algoritmusokat a gráfon. Adjuk meg mely éleket és milyen sorrendben választják be a feszítőfába!

13.2. Feladat Egy kommunikációs hálózatot akarunk kiépíteni. Adott n állomás, amelyeket el kell érnünk. Minden állomáspárra adott, hogy mekkora költséggel építhető ki közöttük közvetlen csatorna. Feladatunk meghatározni, hogy mely közvetlen csatornákat építsük ki. Reprezentáljuk a feladatot gráffal!

13.3. Feladat Egy G irányítatlan súlyozott gráf egy feszítőfáját üvegnyakú feszítőfának nevezünk, ha a maximális élsúlya minimális a feszítőfák között.

- Igaz-e, hogy minden minimális feszítőfa egyben üvegnyakú is?
- Igaz -e, hogy minden üvegnyakú feszítőfa egyben minimális is?

13.4. Feladat Tekintsük a következő súlyozott gráfot, a csúcsok a, b, c, d, az élek a súlyokkal $w(a,b) = x, w(a,c) = y, w(a,d) = 2$, $w(b,c) = 3, w(b,d) = 4, w(c,d) = 3$, ahol az x, y értékek pozitív valós számok. Az x, y értékektől függően milyen minimális költségű feszítőfák lehetnek a gráfban?

13.5. Feladat Igaz -e, hogy ha egy összefüggő irányítatlan súlyozott gráfban egy élnek a súlya szigorúan nagyobb, mint a többi él súlya, akkor nem szerepelhet minimális költségű feszítőfában? Igazoljuk, hogy ha van kör, ami tartalmazza ezt a szigorúan legnagyobb élt, akkor igaz az állítás!

13.6. Feladat Adott egy négy csúcsú teljes gráf (bármely két csúcs között vezet él). Az éleknek súlyokat adhatunk az $1, 2, 3, 4, 5, 6$ értékeket oszthatjuk szét a hat él között. Mekkora lehet a minimális súlyú feszítőfa súlya a súlyok szétosztásától függően?

14. Legrövidebb utak

14.1. Feladat Hajtsuk végre az 1 pontból a Dijkstra algoritmust az alábbi G_1 és G_2 gráfokra. Igaz -e, hogy a D értékek a legrövidebb utak hosszát adják vissza? Amennyiben nem, mi az oka? (A mátrixokban a c_{ij} érték az (i,j) él hossza, ∞ ha nincs él.)

$$G_1 = \begin{pmatrix} \infty & 2 & \infty & 1 & 8 & \infty \\ 2 & \infty & 1 & 4 & 1 & 5 \\ 2 & \infty & 2 & 1 & 1 & 7 \\ \infty & 1 & 4 & \infty & 2 & 7 \\ 1 & \infty & 1 & 4 & \infty & 1 \\ \infty & 1 & \infty & 4 & 2 & 1 \end{pmatrix}$$

$$G_2 = \begin{pmatrix} \infty & 2 & \infty & 1 & 8 & \infty \\ 2 & \infty & 1 & -4 & 1 & 5 \\ 2 & \infty & 2 & 1 & 1 & 7 \\ \infty & 1 & 4 & \infty & 2 & 7 \\ 1 & \infty & 1 & 4 & \infty & 1 \\ \infty & 1 & \infty & 4 & 2 & 1 \end{pmatrix}$$

14.2. Feladat Legyen G egy irányított súlyozott körmentes gráf, amelynek a csúcsai $(1, 2, \ldots, n)$ topologikus rendezés szerint vannak megadva (azaz minden él kisebb sorszámú csúcsból nagyobb sorszámú csúcsba vezet). Adjunk meg egy algoritmust, amely kiszámolja az 1 pontból a többi pontba vezető legrövidebb utak hosszát! Miként használhatjuk az algoritmust arra, hogy a gráfban az 1 pontból a többi pontba vezető leghosszabb utak hosszát adjuk meg?

14.3. Feladat Vegyük a következő gráfot, a gráf pontjai $V = \{1, \ldots, 6\}$ topologikusan vannak rendezve. Az élek a súlyokkal a következők. $c(1,2) = -5, c(1,3) = -2, c(1,5) = 3, c(1,6) = 3,\ c(2,4) = 4,\ c(2,5) = 1,\ c(2,6) = -2,\ c(3,4) = -7,\ c(3,6) = 1,\ c(4,6) = 3,\ c(5,6) = 1$. Az előző feladatban kifejlesztett eljárás alapján határozzuk meg az 1 pontból a többi pontba vezető legrövidebb és leghosszabb utakat.

14.4. Feladat Legyen G egy irányított súlyozott gráf, amely nem tartalmaz negatív súlyú kört. Adjunk meg egy algoritmust, amely kiszámolja az 1 pontból a többi pontba vezető legrövidebb utakat! Miként módosítható az algoritmus úgy, hogy felismerje, ha a gráf tartalmaz negatív súlyú kört?

14.5. Feladat Adjuk meg a fenti algoritmus alapján az 1 pontból a többi pontba vezető legrövidebb utakat az alábbi gráfra! (A mátrixban c_{ij} az (i,j) él súlya.)

$$G = \begin{pmatrix} \infty & \infty & \infty & 6 & 8 \\ 2 & \infty & 4 & 4 & 2 \\ 2 & \infty & 1 & -1 & 2 \\ 1 & -3 & \infty & 4 & \infty \\ \infty & 4 & 1 & 2 & \infty \end{pmatrix}$$

14.6. Feladat Adjuk meg a fentiekben megadott algoritmus alapján hogy az alábbi gráf az A paraméter mely értékeire tartalmaz negatív súlyú kört! (A mátrixban c_{ij} az (i,j) él súlya.)

$$G = \begin{pmatrix} \infty & 2 & 3 \\ \infty & 0 & A \\ \infty & 1 & 0 \end{pmatrix}$$

14.7. Feladat Adjuk meg az alábbi gráfra minden pontpárra a legrövidebb utakat, a Floyd Warshall algoritmus alapján! (A mátrixban c_{ij} az (i,j) él súlya.)

$$G = \begin{pmatrix} 0 & 2 & \infty & 1 & 5 \\ 2 & 0 & 3 & 4 & \infty \\ 2 & \infty & 0 & 3 & 1 \\ \infty & 1 & 4 & 0 & 2 \\ 1 & \infty & 1 & 4 & 0 \end{pmatrix}$$

14.8. Feladat Adjuk meg az alábbi irányított gráfra a tranzitív lezártját, a módosított Floyd Warshall algoritmus alapján! (A mátrixban $c_{ij} = 1$ ha létezik (i,j) él.)

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

14.9. Feladat Egy $x_1, \ldots, x_m$ méretű doboz belefér az $y_1, \ldots, y_m$ méretű dobozba, ha $x_i \leq y_i$ teljesül minden i-re. A feladatunk az, hogy dobozoknak egy megadott listájára határozzuk meg leghosszabb $B_1, \ldots, B_i$ dobozsorozatot, amelyre a B_j doboz elfér a B_k dobozokban $k \leq j$ esetén. Reprezentáljuk gráffal a feladatot!

14.10. Feladat Meseországban minden közhivatalnok megvesztegethető. Nem mindegyiküket lehet közvetlenül megvesztegetni a magasabb hivatalban levő embereket csak az vesztegetheti meg, aki rendelkezik megfelelő ajánlólevéllel, amit az ajánlólevél írójának megvesztegetésével lehet megszerezni. Egy jól értesült vállalkozó tudja kit mennyibe kerül megvesztegetni és azt is, hogy az egyes emberek milyen ajánlóleveleket fogadnak el. A miniszterelnököt szeretné megvesztegetni a lehető legkisebb teljes költség megfizetése mellett. Adjunk meg egy eljárást, amellyel meghatározhatja kiket kell megvesztegetnie!

14.11. Feladat A háborúban a hadsereg vezetői egy csomagot szeretnének átjuttatni az ellenséges országon, a szövetséges erők főhadiszállására. Ismerik az ország úthálózatát, és minden útra tudják mekkora a valószínűsége annak, hogy a futár át tud jutni rajta. Adjunk algoritmust, amely kiválasztja azt az útvonalat, amelyen a legnagyobb a valószínűsége, hogy a futár célba ér!

15. Rendezés

15.1. Feladat Rendezzük a következő sorozatot kiválasztó rendezéssel: 3, 5, 2, 6, 7, 8, 1, 9.

15.2. Feladat Rendezzük a következő sorozatot beszúró rendezéssel! 3, 5, 6, 8, 9, 1, 3, 4.

15.3. Feladat Rendezzük a következő sorozatot kupacrendezéssel: 3, 5, 6, 9, 1, 3.

15.4. Feladat Rendezzük a következő sorozatot gyorsrendezéssel: 5, 9, 6, 1, 3, 4.

15.5. Feladat Rendezzük a következő sorozatot számláló rendezéssel: 5, 3, 6, 1, 3, 4, 3, 2, 1.

15.6. Feladat Rendezzük a következő sorozatot radix rendezéssel: 236, 423, 654, 145, 311, 452, 334, 215, 178.

15.7. Feladat Rendezzük a következő sorozatot vödrös rendezéssel! 0.53, 0.34, 0.65, 0.12, 0.33, 0.44, 0.32, 0.23, 0.13, 0.82.

15.8. Feladat Mennyi a futási ideje a kupacrendezésnek egy nagyság szerint növekvően rendezett tömbön? Mennyi a futási idő egy nagyság szerint csökkenő tömbön?

15.9. Feladat Adjunk meg egy olyan változatát a gyorsrendezésnek, amely legrosszabb esetben is $O(n \log n)$ időben fut!

16. Medián, külső rendezések

16.1. Feladat Keressük meg az alábbi sorozatban egyidejűleg a minimális és a maximális elemet:
$1, 5, 3, 7, 8, 9, 4, 7, 3$.

16.2. Feladat Találjuk meg az alábbi sorozat negyedik legkisebb elemét kupaccal:
$4, 6, 7, 2, 3, 5, 8$.

16.3. Feladat Találjuk meg az alábbi sorozat ötödik legkisebb elemét ismételt felosztással:
$4, 6, 7, 2, 3, 5, 8$.

16.4. Feladat Rendezzük a következő sorozatot 4 file-os egyenletes összefésülő rendezéssel:
$1, 3, 4, 3, 5, 6, 2, 6, 7, 8, 24, 5, 9, 7, 8$.

16.5. Feladat Adott két n hosszú rendezett tömb, határozzuk meg a két sorozat $2n$ elemének mediánját egy $O(\lg n)$ idejű algoritmussal!

16.6. Feladat Adott n darab $x_1, \ldots, x_n$ szám és ezekhez $w_1, \ldots, w_n$ pozitív súlyok, úgy, hogy $\sum_{i=1}^{n} w_i = 1$. A súlyozott medián az az x_k elem, amelyre teljesül $\sum_{x_i<x_k} w_i < 1/2$ és $\sum_{x_i>x_k} w_i \leq 1/2$.
a) Miként kaphatjuk meg a mediánt súlyozott mediánként?
b) Adjunk egy lineáris algoritmust, amely kiszámolja egy rendezett tömb súlyozott mediánját. Miként adódik ebből $O(n \lg n)$ legrosszabb idejű algoritmus a probléma megoldására?

Megoldásvázlatok, ötletek

1. Aszimptotikus jelölések

1.1. Feladat Lehetséges $f(5) = 5$, hiszen ha $f(n) = n \log n$ minden $n \geq 6$ esetén, a definíció teljesül az $n_0 = 6$ $c_1 = c_2 = 1$ értékekkel. Ellenben tetszőleges c_1 konstansra $n < c_1 n \log n$, ha n olyan prím, amelyre $\log n > 1/c_1$, így $f(n) = n$ minden prímre nem lehetséges.

1.2 Feladat Lehetséges $f(n) = n$ és $f(n) = n^2$ is az $n \leq 100$ értékekre, hiszen ha $f(n) = n \log n$ minden $n \geq 101$ esetén, a definíció teljesül az $n_0 = 101$ $c_1 = 1$ értékekkel. Az $f(n) = n$ feltétel minden páros n esetén is teljesülhet, hiszen $n = O(n \log n)$. Ellenben tetszőleges c konstansra $n^2 > c \cdot n \log n$, ha n olyan páros szám, amelyre $n > c \log n$, így $f(n) = n^2$ minden páros számra nem lehetséges.

1.3. Feladat Legyen $f(n) = n$, ekkor $f(2n) = 2f(n)$ minden n-re, így $f(2n) = O(f(n))$.

1.4. Feladat

- $2^{2n}/2^n = 2^n \to \infty$, így $2^n = o(2^{2n})$, következésképp $2^n = O(2^{2n})$ is teljesül.
- $n \log_{10} n / n \ln n = \log_{10} e$, így $n \log_{10} n = \Theta(n \ln n)$, következésképpen $n \log_{10} n = O(n \ln n)$, $n \log_{10} n = \Omega(n \ln n)$ szintén teljesül.
- Egyrészt $2^n = 4^{n/2}$. Másrészt, ha $n > 4$ akkor $n! > 4^{n-3}$, így $2^n/n! < 4^{3-n/2}$ A felső korlát 0-hoz tart így a vizsgált hányados is. Következésképpen $2^n = o(n!)$, így $2^n = O(n!)$ is teljesül.
- Mivel $n^n/n! \geq 2^{n/2}$, ezért $n^n/n! \geq 2^{n/2}$ tart a végtelenbe, ha n tart a végtelenbe, így $n^n = \omega(n!)$, következésképp $n^n = \Omega(n!)$ is teljesül.
- Mivel $\sqrt{n^7}/n^3 = \sqrt{n}$, amely függvény tart a végtelenbe, ha n tart a végtelenbe, így $\sqrt{n^7} = \omega(n^3)$, következésképp $\sqrt{n^7} = \Omega(n^3)$ is teljesül.

1.5. Feladat

$h(n) = \Theta(p(n))$ teljesül. Legyenek c_1, c_2, n_0 olyan értékek, amelyekre ha $n \geq n_0$, akkor $c_1 g(n) \leq f(n) \leq c_2 g(n)$ (ilyen értékek $f(n) = \Theta(g(n))$ alapján vannak). Ha $n \geq n_0/2$, akkor $c_1 p(n) = c_1 g(2n) \leq f(2n) = h(n) = f(2n) \leq c_2 g(2n) = c_2 p(n)$, így $h(n) = \Theta(p(n))$.

Amennyiben $f(n) = 2^n$, akkor $h(n) = 2^{2n}$, így $h(n)/f(n) = 2^n \to \infty$, következésképp $h(n) = O(f(n))$ nem teljesül. Amennyiben $f(n)$ monoton csökken, akkor a reláció fennáll, hiszen $h(n) = f(2n) \leq f(n)$ minden n esetén.

Amennyiben $f(n) = 2^{-n}$, akkor $h(n) = 2^{-2n}$, így $h(n)/f(n) = 2^{-n} \to 0$, következésképp $h(n) = \Omega(f(n))$ nem teljesül. Amennyiben $f(n)$ monoton növekszik, akkor a reláció fennáll, hiszen $h(n) = f(2n) \geq f(n)$ minden n esetén.

1.6. Feladat Vegyük $f(n) = n^n$ függvényt! Ekkor az $f(n+100)/f(n) > n^{100}$ hányados tart a végtelenbe, így nem adható meg a O jelölésben szükséges konstans. A fenti függvény monoton növekvő, így monoton növekvő függvényekre nem feltétlenül teljesül az $f(n+100) = O(f(n))$ állítás. Másrészt monoton csökkenő függvények esetén $f(n+100) \leq f(n)$, így $f(n+100) = O(f(n))$ a $c = 1$ konstans mellett.

1.7. Feladat

$g(n) + h(n) = \Theta(n)$ teljesül. Ha $n > n_0$ mellett $c_1 \cdot n \leq g(n) \leq c_2 \cdot n$ és $n > n_1$ mellett $d_1 \cdot n \leq h(n) \leq d_2 \cdot n$, akkor $n > \max\{n_0, n_1\}$ mellett $(c_1 + d_1) \cdot n \leq g(n) + h(n) \leq (c_2 + d_2) \cdot n$.

Amennyiben $g(n) = h(n) = n$, akkor $|g(n) - h(n)| + 1 = 1$, így $n/(|g(n) - h(n)| + 1) = n \to \infty$, következésképp $|g(n) - h(n)| + 1 = \Theta(n)$ nem minden esetben teljesül.

Ha $n > n_0$ mellett $g(n) \leq c_2 \cdot n$ és $n > n_1$ mellett $h(n) \leq d_2 \cdot n$, akkor $n > \max\{n_0, n_1\}$ mellett $|g(n) - h(n)| + 1 \leq (\max\{c_2, d_2\} + 1) \cdot n$, így $(|g(n) - h(n)| + 1) = O(n)$ valóban teljesül.

2. Rekurziók megoldásai

2.1. Feladat Igazoljuk indukcióval, hogy a függvényre $T(n) < cn^2 - n$ valamely c konstansra. (A $-n$ tag nélkül az állítás nem következik egyszerű indukciós lépéssel.) A kis n értékekre az állítás nyilvánvalóan teljesül, ha a c értéket elég nagyra választjuk. Most tegyük fel, hogy valamely n-ig teljesül az állítás és igazoljuk a helyettesítő módszerrel n-re is.

$$T(n) = 9T(n/3) + n \leq 9(c(n/3)^2 - n/3) + n = c \cdot n^2 - 2n < c \cdot n^2 - n.$$

2.2. Feladat Teljes indukcióval igazoljuk, hogy a függvényre $T(n) \leq cn \log n$ valamely c konstansra. A kezdeti értékekre az állítás nyilvánvalóan teljesül elegendően nagy c választása mellett. Most tegyük fel, hogy n-ig teljesül. Az indukciós lépés

$$T(n) = 2T(n/2) + n \leq 2(cn/2 \log(n/2)) + n \leq cn \log(n/2) + n \leq cn \log n,$$

ahol az utolsó lépésben kihasználtuk, hogy $\log(n/2) + 1/c \leq \log n$.

2.3. Feladat A számítások egyszerűsítésének érdekében nem foglalkozunk a kerekítésekkel, mivel csak a nagyságrendet akarjuk meghatározni, ez az egyszerűsítés megtehető. Legyen $m = \log n$, a függvényre felírt rekurzió a következő formába kerül: $T(2^m) = 2T(2^{m/2}) + 1$. Legyen $S(m) = T(2^m)$, erre a

függvényre $S(m) = 2S(m/2) + 1$. A függvényre a helyettesítő módszerrel egyszerűen igazolható, hogy $S(m) = \Theta(m)$, a $cm - 1 \leq S(m) \leq dm - 1$ egyenlőtlenségeket használva az indukciós lépésben. Másrészt ha $S(m) = \Theta(m)$, akkor $T(n) = \Theta(\log n)$.

2.4. Feladat A kerekítésektől a számítások egyszerűsítésének érdekében eltekintünk, ez nem változtatja a függvény nagyságrendjét. A rekurziós fa első szintjén az első rekurziós hívásból származó n áll. A második szinten négyszer áll $n/2$. A következő szinten mindegyik $n/2$ alá négy darab $n/4$ kerül, így összesen 16 darab $n/4$ lesz. Teljes indukcióval látható, hogy az i-edik szinten 4^i darab $n/2^i$ áll. Tehát az i-edik szinten az összeg $2^i n$. Összesen $\log_2 n$ szint van, így $T(n) = \sum_{i=0}^{\log n} 2^i n = \Theta(n^2)$.

2.5. Feladat Teljes indukcióval igazolható, hogy a kerekítésektől eltekintve minden szinten n lesz az elemek összege a rekurziós fában. Mivel $\log_\alpha n$ szint van, és igazolható, hogy a kerekítések nem változtatják meg a nagyságrendet, ezért az állítást igazoltuk.

2.6. Feladat Az első esetben a $b = 2$, $a = 8$, $f(n) = n^2$ értékekkel kapjuk a mester tételben használt formát. Ekkor $\log_2 8 = 3$, így a $\varepsilon = 1$ értékkel a mester tétel első esetének feltételei teljesülnek, amiből adódik, hogy $P(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$.

A második esetben a $b = 2$, $a = 4$, $f(n) = n^2$ értékekkel kapjuk a mester tételben használt formát. Ekkor $\log_2 4 = 2$, így $f(n) = n^{\log_b a}$, azaz a mester tétel második esetének feltételei teljesülnek, amiből adódik, hogy $Q(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^2 \log n)$.

A harmadik esetben a $b = 2$, $a = 2$, $f(n) = n^2$ értékekkel kapjuk a mester tételben használt formát. Ekkor $\log_2 2 = 1$, így $n^{\log_b a} = n$, azaz a mester tétel harmadik esetének első feltétele teljesül. Másrészt, $2(n/2)^2 \leq \frac{1}{2}n^2$, így a második feltétel is teljesül. Tehát adódik, hogy $R(n) = \Theta(f(n)) = \Theta(n^2)$.

2.7. Feladat Legyen $f(n) = n^{\log_b a} \log n$. Ekkor $f(n) = \omega(n^{\log_b a})$, így az első két eset nyilvánvalóan nem használható. Másrészt tetszőleges $\varepsilon > 0$ esetén $f(n)/n^{\log_b a + \varepsilon}$ a 0 számhoz konvergál, így a harmadik eset feltételei sem teljesülnek.

3. Legjobb, legrosszabb átlagos eset korlátok

3.1. Feladat Legjobb eset: Amennyiben a gondolt szám 2^{n-1}, első kérdésre kitaláljuk, így a legjobb eset $\Theta(1)$.

Legrosszabb eset: Az i-edik kérdés után a lehetséges gondolt szám kettes számrendszerbeli alakjában az i-edik számjegyet meghatározzuk. Következésképpen az n-edik kérdéssel a legrosszabb esetben is kitaláljuk a keresett számot. Másrészt n kérdés szükséges is lehet (például, ha a gondolt szám 1), így a legrosszabb esetben a kérdések száma $n = \Theta(n)$.

Átlagos eset: Amennyiben a gondolt szám 2^{n-1}, az első kérdéssel kitaláljuk. Amennyiben a szám 2^{n-2} vagy $2^{n-1} + 2^{n-2}$, akkor két kérdésben találjuk ki. Azokat a számokat, amelyek bináris alakjában az i-edik számjegy 1 az utána

következő számjegyek 0-ák, i lépésben találjuk ki. Következésképpen minden i-re, $i = 1, \ldots, n$, 2^{i-1} olyan szám van, amelyet i lépésben találunk ki. Tehát a kérdések átlagos száma

$$S = \frac{\sum_{i=1}^{n} i2^{i-1}}{2^n - 1}$$

Erre az értékre könnyen igazolható, hogy $n/2 \leq S \leq n$, azaz igazoltuk, hogy a kérdések átlagos száma $\Theta(n)$.

3.2. Feladat Legjobb eset: A kocsi a kaputól legtávolabbi helyen áll, így minden kocsihelyet megnéz egyszer, az utolsót kétszer, tehát $n + 1$ kocsihelyet néz meg.

Legrosszabb eset: Ekkor a kocsi az első helyen áll. Tehát kétszer végigmegy és összesen $2n$ helyet néz meg.

Átlagos eset: Ha a kocsi hátulról az i-edik helyen van, akkor elmegy a végére és i helyet jön vissza. Tehát $n + i$ kocsihelyet kell megnézni. Tehát az átlagos eset

$$\frac{\sum_{i=1}^{n}(n+i)}{n} = \frac{n^2 + n(n+1)/2}{n} = \frac{3n+1}{2}.$$

3.3. Feladat Mivel az algoritmusnak végig kell hajtania az összes összehasonlítást a maximális elem megtalálásához, ezért n összehasonlítás van a maximális elem elhelyezkedésétől függetlenül, azaz a legjobb, a legrosszabb és az átlagos esetben is.

3.4. Feladat A legjobb esetben a permutáció első eleme az 1, a legrosszabb esetben az utolsó, így a legjobb esetben 1, a legrosszabb esetben n összehasonlítást hajtunk végre. Az átlagos eset vizsgálatához határozzuk meg minden i-re, hogy $n!$ lehetséges permutációból, hány esetben van az 1 elem az i-edik helyen. Ha az i-edik helyen az 1 le van rögzítve, akkor a többi helyet $(n-1)!$ féleképpen tölthetjük fel, így az ilyen permutációk száma $(n-1)!$. Következésképp az átlagos esetben az összehasonlítások száma

$$\frac{\sum_{i=1}^{n} i \cdot (n-1)!}{n!} = \frac{\sum_{i=1}^{n} i}{n} = \frac{n+1}{2}.$$

3.5. Feladat Ha a 0 kilométerkőnél indulunk, akkor a legjobb esetben az első benzinkúthoz legrosszabb esetben az n-edik benzinkúthoz kell elmennünk, tehát a megtett kilométerek száma a legjobb és legrosszabb esetekben 2 és 2^n. Az átlagos esetben a barátunk n helyen lehet, ha az i-edik benzinkútnál van, akkor 2^i távolságot kell mennünk, így az átlagos esetre a megtett út

$$\frac{\sum_{i=1}^{n} 2^i}{n} = \frac{2^{n+1} - 2}{n} = o(2^n).$$

Ha a 2^n kilométerkőnél indulunk, akkor a legjobb esetben az n-edik benzinkúthoz legrosszabb esetben az első benzinkúthoz kell elmennünk, tehát a megtett kilométerek száma a legjobb és legrosszabb esetekben 0 és $2^n - 2$. Az

átlagos esetben a barátunk n helyen lehet, ha az i-edik benzinkútnál van, akkor $2^n - 2^i$ távolságot kell mennünk, így az átlagos esetre a megtett út

$$\frac{\sum_{i=1}^{n}(2^n - 2^i)}{n} = \frac{(n-2)2^n + 2}{n} = \Theta(2^n).$$

3.6. Feladat A legjobb esetben a 2^{n-1} kilométerkőnél van a benzinkutas, így 0 kilométert kell megtenni. A legrosszabb esetben az első kúthoz majd onnan az n-edik benzinkúthoz kell elmennünk, tehát a megtett kilométerek száma legrosszabb esetben $2^{n-1} + 2^n - 4 = \Theta(2^n)$. Az átlagos esetben a barátunk n helyen lehet, ha az $i < n$-edik benzinkútnál van, akkor $2^{n-1} - 2^i$ távolságot kell mennünk, ha az n-edik kútnál akkor $2^{n-1} + 2^n - 4$ távolságot így az átlagos esetre a megtett út

$$\frac{\sum_{i=1}^{n-1}(2^{n-1} - 2^i) + 2^{n-1} + 2^n - 4}{n} = \frac{n \cdot 2^{n-1} - 2}{n} = \Theta(2^n).$$

3.7. Feladat A legjobb esetben a 2^{n-1} kilométerkőnél van a benzinkutas, így 0 kilométert kell megtenni. A legrosszabb esetben az n-edik kúthoz majd onnan az első benzinkúthoz kell elmennünk, tehát a megtett kilométerek száma legrosszabb esetben $2^{n-1} + 2^n - 2 = \Theta(2^n)$. Az átlagos esetben a barátunk n helyen lehet, ha az $i < n-1$-edik benzinkútnál van, akkor $2^n + 2^{n-1} - 2^i$ távolságot kell mennünk, ha az n-edik kútnál akkor 2^{n-1} távolságot ha az $n-1$-edik kútnál akkor 0 távolságot, így az átlagos esetre a megtett út

$$\frac{\sum_{i=1}^{n-2}(2^n + 2^{n-1} - 2^i) + 2^{n-1}}{n} = \frac{(n-2)(2^n + 2^{n-1}) + 2}{n} = \Theta(2^n).$$

4. Rekurzív algoritmusok

4.1. Feladat: A $P(n)$ értékekre a következő összefüggések állnak fenn:

- $P(1) = 1$ (ha egy lépcső van egyféleképpen mehetünk)
- $P(2) = 2$ (ha két lépcső van, vagy $1+1$-ben vagy egy kettes lépéssel mehetünk)
- $P(n) = P(n-1) + P(n-2)$ ha $n \geq 3$, (utolsó lépésként 1-et vagy 2-t léphetünk).

A következő program számolja ki a $P(n)$ függvényt:

```
Function P(n: Word) : Longint;
Begin
   If n=1 Then
        P:=1
```

```
   Else If n=2 Then
        P:=2
    Else
        P:=P(n-1)+P(n-2)
End;
```

Helyesség: A megállási feltétel $M(n) = (n-1)(n-2)$. A helyesség adódik a megadott összefüggésekből.

A végrehajtott utasítások száma: Igazoljuk teljes indukcióval, hogy $P(n) - 1 \leq E(n) \leq 2P(n)-1$. Ha $n = 1$ vagy $n = 2$, akkor $P(n)-1 \leq E(n) = 2P(n)-1$. Ha az $n \geq 3$ értékre az n-nél kisebb számokra igaz, akkor n-re is, mivel

$P(n) - 1 = P(n-1) + P(n-2) - 1 \leq E(n-1) + E(n-2) + 1 = E(n)$

$E(n) = E(n-1)+E(n-2)+1 \leq 2P(n-1)-1+2P(n-2)-1+1 = 2P(n)-1$

A $P(n)$-re vonatkozó formulát teljes indukcióval igazoljuk. Behelyettesítéssel adódik, hogy

$$P(1) = 1 = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^2 - \left(\frac{1-\sqrt{5}}{2}\right)^2\right),$$

és

$$P(2) = 2 = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^3 - \left(\frac{1-\sqrt{5}}{2}\right)^3\right).$$

Továbbá:

$$P(n+1) = P(n) + P(n-1) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right) +$$

$$\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\left(\frac{1+\sqrt{5}}{2}+1\right) - \left(\frac{1-\sqrt{5}}{2}\right)^n\left(\frac{1-\sqrt{5}}{2}+1\right)\right) =$$

$$\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\left(\frac{1+\sqrt{5}}{2}\right)^2 - \left(\frac{1-\sqrt{5}}{2}\right)^n\left(\frac{1-\sqrt{5}}{2}\right)^2\right) =$$

$$\frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{n+2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+2}\right).$$

4.2. Feladat: A következő összefüggések állnak fenn az $R(n,k)$ értékekre:

- $R(1,k) = 1$ (csak jobbra mehetünk)

- $R(n,1) = 1$ (csak felfelé mehetünk)
- $R(n,k) = R(n,k-1) + R(n-1,k)$ (az utolsó lépés felfelé és jobbra történhet)

A következő program adja meg a $R(n,k)$ függvényt:

```
Function R(n,k: Word) : Longint;
Begin
   If (n=1) Or (k=1) Then
        R:=1
   Else
        R:=R(n-1,k)+R(n, k-1)
End;
```

Helyesség: A megállási feltétel $M(n,k) = (n-1)(k-1)$. A helyesség adódik a megadott összefüggésekből.

A végrehajtott utasítások száma: Igazoljuk teljes indukcióval, hogy $R(n,k) \leq E(n,k) \leq 2R(n,k) - 1$. Ha $n = 1$ vagy $k = 1$, akkor $1 = R(n,k) = E(n,k) = 2R(n,k) - 1$. Ha az összes $(n',k') \neq (n,k)$ pár esetén, amelyre $n' \leq n$ és $k' \leq k$ teljesül igaz az állítás, akkor az (n,k) pár esetén is:

$R(n,k) = R(n,k-1) + R(n-1,k) \leq E(n,k-1) + E(n-1,k) = E(n,k) - 1$

$E(n,k) = E(n,k-1) + E(n-1,k) + 1 \leq 2R(n,k-1) - 1 + 2R(n-1,k) - 1 + 1 = 2R(n,k) - 1$

$R(n,k)$ értéke: Az út a felső sarokba $n+k-2$ lépésből áll, ebből $k-1$ -et lépünk jobbra. Ez a $k-1$ lépés bármely $k-1$ lépés lehet és a jobbra lépések meghatározzák az utat, következésképp az utak száma megegyezik azzal a számmal, ahányféleképp kiválaszthatjuk az $n+k-2$ lépésből a $k-1$ jobbra lépést, és ez $\binom{n+k-2}{k-1}$.

4.3. Feladat: Jelölje $Q(n)$ az n forint lehetséges elköltéseinek a számát! A következő összefüggések állnak fenn:

- $Q(1) = 1$ (ha egy forint van, akkor egy perecet vehetünk)
- $Q(2) = 3$ (két forintból, vagy két perec vagy egy csoki vagy egy fagylalt vásárolható)
- $Q(n) = Q(n-1) + 2Q(n-2)$ ha $n \geq 3$, (utolsó alkalommal perecet, csokit vagy fagylaltot vehetünk).

A következő algoritmus adja meg a $Q(n)$ függvényt:

```
Function Q(n: Word) : Longint;
Begin
   If n=1 Then
        Q:=1
   Else If n=2 Then
        Q:=3
    Else
        Q:=Q(n-1)+2Q(n-2)
End;
```

Helyesség: A megállási feltétel $M(n) = (n-1)(n-2)$. A helyesség adódik a megadott összefüggésekből.

4.4. Feladat: A feladat ekvivalens az előző feladattal, a perec az A típusnak a csoki és a fagylalt a B és C típusoknak felel meg. Következésképpen ugyanazzal az eljárással oldható meg a feladat.

4.5. Feladat: Jelölje $F(n)$ az n magas torony lehetséges felépítéseinek a számát! A következő összefüggések állnak fenn:

- $F(1) = 1$ (egy egység magas kocka)
- $F(2) = 3$ (két egység magas, vagy két egység magas kék vagy két egység magas piros)
- $F(3) = 6$ (egyszerű esetszétválasztás)
- $F(n) = F(n-1) + 2F(n-2) + F(n-3)$ ha $n \geq 4$, (legfelül a négy féle kocka valamelyike lehet).

A következő algoritmus adja meg a $F(n)$ függvényt:

```
Function F(n: Word) : Longint;
Begin
   If n=1 Then
        F:=1
   Else If n=2 Then
        F:=3
   Else If n=3 Then
        F:=6
   Else
        F:=F(n-1)+2F(n-2)+F(n-3)
End;
```

Helyesség: A megállási feltétel $M(n) = (n-1)(n-2)(n-3)$. A helyesség adódik a megadott összefüggésekből.

4.6. Feladat Legyen $L(n)$ n azon partícióinak a száma, amelyben minden szám különböző. Jelölje $L2(n,k)$ ezek közül azok számát, amelyekben minden rész legfeljebb k. A következő összefüggések állnak fenn:

- 1. $L2(1,k) = 1$, (az 1 szám partíciói)
- 2. $L2(n,1) = 0$, ha $n > 1$ (nem lehetnek különbözőek a részek)
- 3. $L2(n,n) = 1 + L2(n,n-1)$ (az n-ből álló partíció és a többi)
- 4. $L2(n,k) = L2(n,n)$, ha $n < k$ (nincs n-nél nagyobb rész)
- 5. $L2(n,k) = L2(n,k-1) + L2(n-k,k-1)$ (vagy szerepel k vagy nem)

A következő algoritmus adja meg a $L(n)$ függvényt:

```
Function L(n: Word) : Longint;
     Function L2(n,k: Word) : Longint;
     Begin
          If n=1 Then
                 L2:=1
          Else If k=1 Then
                 L2:=0
          Else If n<=k Then
                 L2:=1+L2(n,n-1)
          Else
                 L2:=L2(n,k-1)+L2(n-k,k-1)
      End{L2};
Begin
       L:=L2(n,n);
End{L};
```

Helyesség: A megállási feltétel $M(n,k) = (n-1)(k-1)$. A helyesség adódik a megadott összefüggésekből.

4.7. Feladat Legyen $S(n)$ n azon partícióinak a száma, amelyben minden szám páratlan. Jelölje $S2(n,k)$ ezek közül azok számát, amelyekben minden rész legfeljebb k. Hasonlóan, mint az előző feladatban kapjuk, hogy a következő összefüggések állnak fenn:

- 1. $S2(1,k) = 1$, $S2(n,1) = 1$
- 2. $S2(n,n) = 1 + S2(n,n-1)$, ha n páratlan, $S2(n,n) = S2(n,n-1)$, ha n páros

- 3. $S2(n,k) = S2(n,n)$, ha $n < k$
- 4. $S2(n,k) = S2(n,k-1) + S2(n-k,k)$, ha k páratlan, és $S2(n,k) = S2(n,k-1)$ ha k páros.

A következő algoritmus adja meg a $S(n)$ függvényt:

```
Function S(n: Word) : Longint;
     Function S2(n,k: Word) : Longint;
     Begin
          If (n=1) Or (k=1) Then
                 S2:=1
          Else If (n<=k) and (not odd(n)) Then
                 S2:=S2(n,n-1)
          Else If (n<=k) and (odd(n)) Then
                 S2:=1+S2(n,n-1)
          Else If not odd(k) Then
                 S2 :=S2(n,k-1)
          Else
                 S2:=S2(n,k-1)+S2(n-k,k)
      End{S2};
Begin
       S:=S2(n,n);
End{S};
```

Helyesség: A megállási feltétel $M(n,k) = (n-1)(k-1)$. A helyesség adódik a megadott összefüggésekből.

4.8. Feladat Legyen $R(n)$ n azon partícióinak a száma, amelyben minden szám legalább 2. Jelölje $R2(n,k)$ ezek közül azok számát, amelyekben minden rész legfeljebb k. Ekkor a következő összefüggések állnak fenn:

- 1. $R2(1,k) = 0$, $R2(n,1) = 0$,
- 2. $R2(2,k) = 1$, $R2(n,2) = 1$, ha n páros, $R2(n,2) = 0$, ha n páratlan
- 3. $R2(n,n) = 1 + R2(n,n-1)$, ha $n \geq 3$
- 4. $R2(n,k) = R2(n,n)$, ha $n < k$
- 5. $R2(n,k) = R2(n,k-1) + R2(n-k,k)$, ha $n > k$ és $k \geq 2$ és $n \geq 3$

A következő algoritmus adja meg a $R(n)$ függvényt:

```
Function R(n: Word) : Longint;
     Function R2(n,k: Word) : Longint;
```

```
     Begin
          If (n=1) Or (k=1) Then
                R2:=0
           Else If n=2 Then
                R2:=1
           Else If  (k=2) and (odd(n))
                R2:=0
           Else If k=2
                R2:=1
           Else If n<=k  Then
                R2:=1+R2(n,n-1)
           Else
                R2:=R2(n,k-1)+R2(n-k,k)
     End{R2};
Begin
     R:=R2(n,n);
End{R};
```

Helyesség: A megállási feltétel $M(n,k) = (n-1)(n-2)(k-1)(k-2)$. A helyesség adódik a megadott összefüggésekből.

4.9. Feladat Legyen $H(n)$ n azon partícióinak a száma, amelyben minden szám páratlan és a számok páronként különbözőek. Jelölje $H2(n,k)$ ezek közül azok számát, amelyekben minden rész legfeljebb k. Hasonlóan, mint az előző feladatokban kapjuk, hogy a következő összefüggések állnak fenn:

- 1. $H2(1,k) = 1$,
- 2. $H2(n,1) = 0$, ha $n > 1$
- 3. $H2(n,n) = 1 + H2(n,n-1)$, ha n páratlan, $H2(n,n) = H2(n,n-1)$, ha n páros
- 4. $H2(n,k) = H2(n,n)$, ha $n < k$
- 5. $H2(n,k) = H2(n,k-1) + H2(n-k,k-1)$, ha k páratlan, és $H2(n,k) = H2(n,k-1)$ ha k páros.

A következő algoritmus adja meg a $H(n)$ függvényt:

```
Function H(n: Word) : Longint;
     Function H2(n,k: Word) : Longint;
     Begin
          If n=1  Then
                 H2:=1
          Else If k=1 Then
```

```
                H2:=0
          Else If (n<=k) and (not odd(n)) Then
                H2:=H2(n,n-1)
          Else If (n<=k) and (odd(n)) Then
                H2:=1+H2(n,n-1)
          Else If not odd(k) Then
                H2 :=H2(n,k-1)
          Else
                H2:=H2(n,k-1)+H2(n-k,k-1)
      End{H2};
Begin
       H:=H2(n,n);
End{H};
```

Helyesség: A megállási feltétel $M(n,k) = (n-1)(k-1)$. A helyesség adódik a megadott összefüggésekből.

4.10. Feladat Legyen a hanoi eljárás egy olyan algoritmus, amelynek három argumentuma van, az első argumentum azt adja meg, hogy hány korongot helyezünk át, a második megadja, hogy melyik toronyról a harmadik, hogy melyik toronyra. Ekkor az eljárás az $(n, 1, 2)$ argumentummal megoldja a feladatot. Amennyiben $i-1$ korongot már át tudunk helyezni, i korongot a következőképpen helyezhetünk át. Elsőként $i-1$ korongot áthelyezünk az oszlopról egy másik oszlopra. Utána az i-edik korongot rárakjuk a kimaradó üres oszlopra. Végül ezen korong tetejére felrakjuk az $i-1$ korongot. Ezt a rekurziót írja le a következő eljárás (a megengedett lépést a mozgat függvény írja le, az argumentumai, hogy honnan hova):

```
procedure hanoi(n: integer; rol: integer; ra: integer);
Begin
    If n=1 Then
         mozgat(rol, ra)
    Else begin
         hanoi(n-1, rol, 6-rol-ra);
         mozgat(rol, ra);
         hanoi(n-1, 6-rol-ra, ra);
     End
end;
```

Lépések száma: Az i-edik korong átrakásához kétszer kell $i-1$ korongot áthelyezni és egy további mozgatás szükséges. Tehát $T(i)$-vel jelölve az i korong átrakásához szükséges mozgatások számát a $T(i) = 2T(i-1)+1$ rekurzív összefüggés áll fenn. $T(1) = 1$, így $T(2) = 3$, $T(3) = 7$. Azt sejthetjük, hogy $T(i) = 2^i - 1$, amely egyenlőség teljes indukcióval egyszerűen igazolható.

5. Absztrakt adattípusok, adatszerkezetek

5.1. Feladat A sorok képzése:

```
Letesit(S3);
n:=Elemszam(S1);
i:=1;
while(i<=n) do begin
  Sorbol(S1,x);
  Sorba(S3,x);
  i:=i+1;
end;
j:=1;
while(j<=n) do begin
  Sorbol(S2,x);
  Sorba(S3,x);
  i:=i+1;
end;

Letesit(S4);
n:=Elemszam(S1);
i:=1;

while(i<=n) do begin
  Sorbol(S1,x);
  Sorba(S4,x);
  Sorbol(S2,x);
  Sorba(S4,x)
  i:=i+1;
end;
```

5.2. Feladat Az $a_1 \geq a_2 \geq \ldots \geq a_{n-1} \leq a_n$ és az $a_1 \geq \ldots \geq a_{n-2} \leq a_{n-1} \geq a_n$ esetek megkülönböztetéséhez meg kell vizsgálnunk az a_{n-2}, a_{n-1}, a_n értékeket. Másrészt ez sor adattípus esetén csak $\Omega(n)$ művelettel lehetséges. Vizsgáljuk a Tömb adattípus esetét! A következő eljárást használhatjuk:

Kezdetben $a = 1$, $f = n$, ezek az értékek mindig úgy változnak, hogy $T_a \leq T_f$ teljesül, amint $f = a + 1$ az eljárás véget ér.

while($f \neq a + 1$)

- Legyen $c = \lceil (a + f)/2 \rceil$
- Ha $T_c \leq T_f$, akkor
  $a := c$, $f := f$,
- Else
  $a := a$, $f := c$

Az eljárás ismertetése előtti megjegyzés alapján egyből adódik az eljárás helyessége. Az $(f-a)$ érték minden lépésben a felére illetve legfeljebb a felének felső egészrészére csökken. Kezdetben ez n, ezért legrosszabb esetben is legfeljebb $O(\log n)$ lépés után találunk egy megfelelő elemet.

5.3. Feladat A Lista képzése:

```
Letesit(L3); while not Vegen(L1) do begin
  Kiolvas(L1,x);
  Tovabb(L1);
  Bovit(L3,x);
  Tovabb(L3);
  Kiolvas(L2,x);
  Tovabb(L2);
  Bovit(L3,x);
  Tovabb(L3);
end;

Elejere(L3);
```

5.4. Feladat A VEREMBŐL művelet a legfelső elemeket veszi ki, így a kivett elemek sorrendje $7, 2, 6$. A SORBOL művelet a legfelső elemeket veszi ki, így a kivett elemek sorrendje $1, 4, 5, 6$. A prioritási sor mindig a minimális elemet veszi ki, így a sorrend $1, 2, 4, 5, 6$.

5.5. Feladat Verem esetén, az első lépést követően $V =< 2, 3 >$. Utána a 3 kerül ki, és az új verem $V =< 2, 6, 7 >$. Ebből adódik a sejtés, hogy az n-edik lépés után $2^{n+1}-1$ kerül ki, és ez teljes indukcióval könnyen igazolható, hiszen, ha egy lépésben az x elemet használtuk, akkor a verem tetejére a $2x+1$ elem kerül és $2(2^{n+1}-1)+1 = 2^{n+2}-1$.

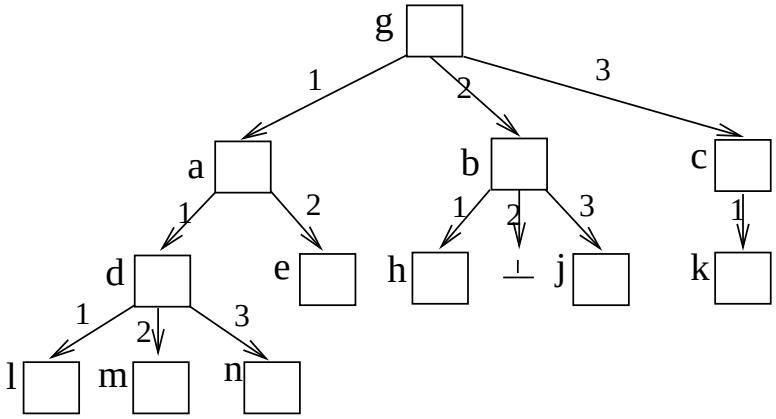
Sor esetén, az első lépést követően $S =< 2, 3 >$. Utána a 2 kerül ki, és az új sor $S =< 3, 4, 5 >$. Majd a 3 kerül ki, és az új sor $S =< 4, 5, 6, 7 >$ Ebből adódik a sejtés, hogy az n-edik lépés után a sor $S =< n+1, n+2, \ldots, , 2n+1 >$, és ez teljes indukcióval könnyen igazolható, hiszen, ha egy lépésben az x elemet használtuk, akkor a sor végére a $2x$ és $2x+1$ elemek kerülnek. Tehát az n-edik lépés után az $n+1$ elemet kapjuk.

Mivel a fenti Sor esetén mindig a minimális elemet vettük ki, ezért a Sor esettel teljesen megegyezik a prioritási sor esete is.

5.6. Feladat Verem esetén, az első lépést követően $V =< 3, 2 >$. Utána a 2 kerül ki, és az új verem $V =< 3, 5, 4 >$. Ebből adódik a sejtés, hogy az n-edik lépés után 2^n kerül ki, és ez teljes indukcióval könnyen igazolható, hiszen, ha egy lépésben az x elemet használtuk, akkor a verem tetejére a $2x$ elem kerül és $2(2^n) = 2^{n+1}$.

A prioritási sor esetén a feladat ekvivalens az előző feladattal, hiszen ugyanazokat az elemeket képezzük csak más sorrendben kerülnek be a prioritási sorba. Viszont prioritási sor esetén nem érdekes milyen sorrendben érkeztek az elemek,

így a műveletsorozatok végrehajtása mellett ugyanazok lesznek a prioritási sorok, mint az előző feladatban.



2. ábra. Az 5.7 feladat fája

5.7. Feladat A fa gyökerét g-vel jelölve a fa az alábbi ábrán látható. A fa magassága a leghosszabb út 3. Az f függvény megadása:

$f(g) = \langle a, b, c\rangle$, $f(a) = \langle d, e\rangle$, $f(b) = \langle h, \bot, j\rangle$, $f(c) = \langle k\rangle$, $f(d) = \langle l, m, n\rangle$, $f(e) = \langle\bot\rangle$, $f(h) = \langle\bot\rangle$, $f(j) = \langle\bot\rangle$, $f(k) = \langle\bot\rangle$, $f(l) = \langle\bot\rangle$, $f(m) = \langle\bot\rangle$, $f(n) = \langle\bot\rangle$.

5.8. Feladat A fa gyökerét g-vel jelölve a fa az alábbi ábrán látható. A fa levelei: d, k, b, h, j. Elsőfiú, testvér kapcsolattal az elsőfiút leíró E függvény:

$E(g) = a$, $E(a) = d$, $E(b) = \bot$, $E(c) = h$, $E(d) = \bot$, $E(e) = k$, $E(h) = \bot$, $E(j) = \bot$, $E(k) = \bot$.

A jobbtestvért leíró T függvény:

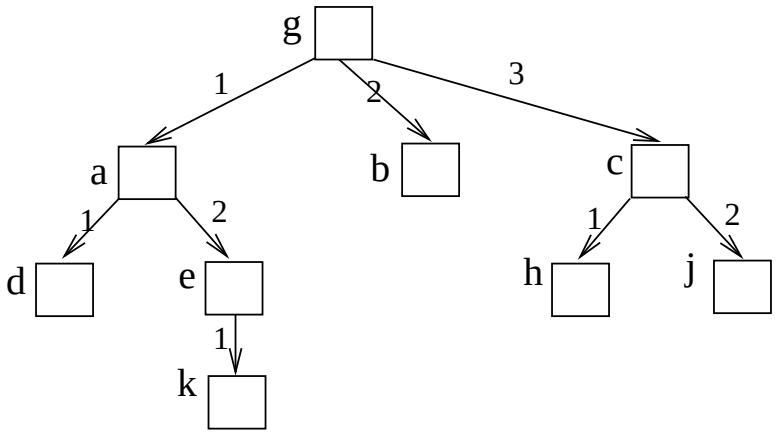
$T(g) = \bot$, $T(a) = b$, $T(b) = c$, $T(c) = \bot$, $T(d) = e$, $T(e) = \bot$, $T(h) = j$, $T(j) = \bot$, $T(k) = \bot$.

6. Fák, fabejárások

6.1. Feladat Ugyanazt az eljárást használhatjuk, mint a részletezett 6.2. feladatban, csak a while ciklusban számolt maximum tagjain az elsőfiú-testvér ábrázolás alapján kell végigmenni.

6.2. Feladat A magasságot kiszámító függvény:

```
Function MagasKT(F:Fa) :Word;
  Var i,m: Word;
Begin
  i:=1;
```



3. ábra. Az 5.8. feladat fája

```
  m:=0;
  If  F^.Fok=0 Then
     MagasKT:=0
  Else  Begin
    while(i<=F^.Fok) Do Begin
      n:=MagasKT(F^.Fiuk[i])
      If m<=n Then
         m:=n;
      i:=i+1;
    End
    MagasKT:=m+1
  End;
End;
```

6.3. Feladat Ugyanazt az eljárást használhatjuk, mint a részletezett 6.2. feladatban, csak a while ciklusban számolt maximum tagjain a kapcsolati lánc ábrázolás alapján kell végigmenni.

6.4. Feladat Használhatjuk a preorder bejárás módosítását (a részletezett 6.11. feladathoz hasonlóan). Annyi módosítás kell, hogy mindig számon tartjuk az aktuális pont mélységét, és ezen értékek maximuma lesz a magasság.

6.5. Feladat Ugyanazt az eljárást használhatjuk, mint a részletezett 6.6. feladatban, csak a for ciklusban számolt összeg tagjain az elsőfiú-testvér ábrázolás alapján kell végigmenni.

6.6. Feladat A szinteken levő pontszámokat megadó függvény:

```
Function SzintKT(F:Fa, k:Word) :Word;
```

```
  Var i,m,: Word;
Begin
  m:=0;
  If k=0 Then
    SzintKT:=1
  Else If F^.Fok=0
    SzintKT:=0
  Else Begin
    for i:=1 to F^.Fok Do Begin
        m:=m + SzintKT(F^.Fiuk[i], k-1);
    End
     SzintKT:=m
  End;
End;
```

6.7. Feladat Ugyanazt az eljárást használhatjuk, mint a részletezett 6.6. feladatban, csak a for ciklusban számolt összeg tagjain a kapcsolati lánc ábrázolás alapján kell végigmenni.

6.8. Feladat A feladat hasonló a 6.5. feladathoz, rekurzív hívásban itt is a fa fiaira vett függvényértékeket kell összeadni.

6.9. Feladat A feladat hasonló a 6.6. feladathoz, rekurzív hívásban itt is a fa fiaira vett függvényértékeket kell összeadni.

6.10. Feladat A feladat hasonló a 6.7. feladathoz, rekurzív hívásban itt is a fa fiaira vett függvényértékeket kell összeadni.

6.11. Feladat A levélszámot megadó függvény:

```
Function Levelek(F:Fa) :Word;
  Var i:Word;
begin i:=0;
  while(F<>Nil) do begin
    while(F^Efiu<>Nil) do begin
      F:=F^.Efiu;
    end
    i:=i+1;
    while(F<>Nil) and (F.Testver=Nil) do
     F :=F^.Apa;
    If F<>Nil Then
     F :=F^.Testver;
  end{while}
Levelek:=i; end
```

6.12. Feladat Preorder bejárás kapcsolati tömb ábrázolással.

```
Procedure Bejar(F:Fa, M:MuvelTip}
  Var i: Word;
Begin
  M(F^.Adat)
  for i:=1 to F^.Fok Do Begin
    Bejar(F^.Fiuk[i], M:MuvelTip);
  End;
End;
```

6.13. Feladat A feladat megoldása hasonló a részletezett 6.12. feladatéhoz, a for ciklust a kapcsolati lánc ábrázolás alapján kell megvalósítani.

6.14. Feladat A feladat megoldása hasonló a részletezett 6.15. feladatéhoz, a while ciklust a kapcsolati tömb ábrázolással kell megvalósítani.

6.15. Feladat Szintszerinti bejárás kapcsolati lánc ábrázolással.

```
Procedure Szintjar(F:Fa,  M:MuvelTip)
   Var S: Sor;
   Var FL: Falanc;
Begin
   Letesit(S);
   Sorba(S,F);
   While Elemszam(S)>0 Do Begin
      SorBol(S,F);
      M(F^.Adat);
      FL:=F^.Fiuk;
      While(FL<>Nil) Do Begin
         Sorba(S,FL^.Fiu);
         FL:=FL^.Csat;
      End;
    End;
    Megszuntet(S);
End
```

6.16. Feladat A pontok bejárási sorrendje: $a, b, c, d, i, j, k, o, g, h, n, e, f, l, m$.

6.17. Feladat A pontok bejárási sorrendje: $a, d, k, j, o, i, c, h, g, n, b, f, e, m, l$.

7. Dinamikus programozás

7.1. Feladat A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. A rekurzív hívások alapján a következő dinamikus programozási eljárást kaphatjuk meg. Az eljárásban egy T tömböt töltünk ki $T[i]$ a $P(i)$ értékét tartalmazza:

```
Function P(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN] Of Longint;
     i: Word;
  Begin
     T[1]:=1;
     T[2]:=2;
     for i:=3 to N Do
        T[i]:=T[i-2]+T[i-1];
     P:=T[n];
  End;
```

A táblázat minden elemének kitöltése egy műveletet igényel, így a futási idő valóban lineáris.

7.2. Feladat A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. A rekurzív hívások alapján a következő dinamikus programozási eljárásokat kaphatjuk meg. Az elsőben egy T négyzetes táblázatot töltünk ki $T[i,j]$ az $R(i,j)$ értékét tartalmazza:

```
Function R(n,k:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN,1..MaxN] Of Longint;
     i,j: Word;
  Begin
     for i:=1 to k do T[i,1]:=1;
     for j:=2 to n do begin
        T[1,j]:=1;
        for i:=2 to k do T[i,j]:=T[i-1,j]+T[i,j-1];
     end{for j};
     R:=T[k,n];
   End;
```

Miként klasszikus a partíció problémánál, itt is elég az utolsó két sort tárolni, így a feladat megoldható egyetlen sor k szerinti kitöltésével:

```
Function R2(n,k:Word) :Longint;
  Const
     MaxN=5000;
```

```
  Var
     T: Array[1..MaxN] Of Longint;
     i,j: Word;
  Begin
     for i:=1 to k do T[i]:=1;
     for j:=2 to n do begin
        for i:=2 to k do T[i]:=T[i-1]+T[i];
     end{for j};
     R:=T[k];
   End;
```

7.3. Feladat A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. A rekurzív hívások alapján a következő dinamikus programozási eljárást kaphatjuk meg. Az eljárásban egy T tömböt töltünk ki $T[i]$ a $Q(i)$ értékét tartalmazza:

```
Function Q(n:Word):Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN] Of Longint;
     i: Word;
  Begin
     T[1]:=1;
     T[2]:=3;
     for i:=3 to N Do
        T[i]:=2T[i-2]+T[i-1];
     Q:=T[n];
  End;
```

A táblázat minden elemének kitöltése egy műveletet igényel, így a futási idő valóban lineáris.

7.4. Feladat A feladat ekvivalens az előző problémával.

7.5. Feladat A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. Láttuk miben különbözik az eredeti partíció problémától a vizsgált feladat. A rekurzív hívások alapján a következő dinamikus programozási eljárást kaphatjuk meg, amely a partíció problémára adott eljárás egy változata (négyzetes táblázatkitöltést használunk):

```
Function P(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
```

```
    T: Array[1..MaxN,1..MaxN] Of Longint;
    i,j,p: Word;
  Begin
    T[1,1]:=1;
    for i:=2 to N do T[i,1]:=0;
    for j:=2 to N do begin
       T[j,j]:=T[j,j-1]+1;
       for i:=j+1 to N do begin
          p:=j-1;
          if i-j<=j-1 Then p:=i-j;
          T[i,j]:=T[i,j-1]+T[i-j,p];
    end{for j};
    R:=T[n,n];
   End;
```

Az algoritmus futási ideje $O(n^2)$.

7.6. Feladat: A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. Láttuk miben különbözik az eredeti partíció problémától a vizsgált feladat. A rekurzív hívások alapján a következő dinamikus programozási eljárást kaphatjuk meg, amely a partíció problémára adott eljárás egy változata (lineáris táblázatkitöltést használunk):

```
Function P(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN] Of Longint;
     i,j: Word;
  Begin
     for i:=1 to k do T[i]:=1;
         j:=3;
        while(j<=n) do begin
              T[j]:=T[j]+1;
              for i:=j+1 to k do
                T[i]:=T[i]+T[i-j];
              j:=j+2
     end{for j};
  P:=T[n];
```

Az algoritmus futási ideje $O(n^2)$.

7.7. Feladat: A rekurzív algoritmusok között megadtunk egy rekurzív algoritmust a probléma megoldására. Láttuk miben különbözik az eredeti partíció problémától a vizsgált feladat. A rekurzív hívások alapján a következő dinamikus programozási eljárást kaphatjuk meg, amely a partíció problémára adott eljárás egy változata:

```
Function P(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN,1..MaxN] Of Longint;
     i,j,p: Word;
  Begin
     for i:=1 to n do T[i,1]:=0;
     for i:=1 to n do T[1,i]:=0;
     for i:=2 to n do T[2,i]:=1;
     for i:=2 to n do begin
          If odd(i) then
              T[i,2]:=0
          Else  T[i,2]:=1
     end;
     for j:=3 to n do begin
        T[j,j]:=T[j,j-1]+1;
        for i:=j+1 to n do begin
           p:=j;
           if i-j<=j then p:=i-j;
           T[i,j]:=T[i,j-1]+T[i-j,p];
     end{for j};
     P:=T[n,n];
   End;
```

Az algoritmus futási ideje $O(n^2)$.

7.8. Feladat Legyen $k(P)$ a P dolgozóhoz rendelt kedélyességi érték! Bontsuk részproblémákra a feladatot. Minden P pontra a hierarchia fájában vegyük ugyanezt a problémát. Jelölje $F(P)$ azt a maximális összkedélyességet, ami elérhető abban a partiban, amit a P gyökerű részfában szereplő pontokból állítunk össze. (Azaz a P által vezetett részlegből.) Ekkor ez a függvény rekurzívan a következőképpen állítható elő:

A levelekben $F(P) = k(P)$. Amennyiben egy pont nem levél, akkor két lehetőséget kell megvizsgálnunk.

Ha P-t nem hívjuk meg a partira, akkor a P fiaiból induló részfákra nincs semmi kikötésünk. Adódik, hogy ekkor $F(P) = \sum_{Q \text{ fia } P-\text{nek}} F(Q)$.

Ha P-t meghívjuk a partira, akkor P fiai nem jöhetnek, de fiúk fiaiból (unokákból) induló részfákra nincs semmi kikötésünk, egyszerűen adódik, hogy ekkor $F(P) = k(p) + \sum_{Q \text{ unoka } P-\text{nek}} F(Q)$.

Tehát a rekurzió:

$F(P) = \max\{\sum_{Q \text{ fia}P-\text{nek}} F(Q), k(p) + \sum_{Q \text{ unoka } P-\text{nek}} F(Q)\}$.

A rekurzió alapján a pontokhoz tartozó értékek megfelelő sorrendben történő kitöltésével megkapjuk egy dinamikus programozási eljárással az optimális értéket. Azok a megfelelő sorrendek, amelyekben mindenki megelőzi az apját. Ilyen sorrendet kaphatunk egy szint szerinti bejárás sorrendjének a megfordításával.

Az optimális megoldás visszafejthető, ha az eljárás során mindig megjegyezzük, hogy az optimális megoldás tartalmazza -e a gyökeret.

Két módszer is megadható annak biztosítására, hogy a vállalat főnöke biztos meghívást nyerjen. Megtehetjük, hogy a feladatot, csak a főnök unokáira oldjuk meg, és a kapott megoldások uniója plusz a főnök lesz a módosított probléma megoldása. Egy másik lehetőség, hogy a főnök kedélyességét átállítjuk egy nagyon nagy értékre ezzel biztosítva, hogy felkerül a feladat optimális megoldásában a meghívottak listájára.

7.9. Feladat Jelölje n a tárgyak számát! Definiáljuk az $F(i, W)$ függvényt, minden $i = 1, \ldots, n$, $W = 1, \ldots, S$ értékre. Ez a függvény azon hátizsák probléma optimális függvényértékét adja meg, amelyben a tárgyak listája az első i tárgyat tartalmazza, és a hátizsák mérete W. Amennyiben W értéke negatív a függvény értéke $-\infty$.

Ekkor a kezdeti értékekre $F(1, W) = f_1$, ha $s_1 \leq W$ és 0 különben. Másrészt a következő rekurzió teljesül:

$$F(i+1, W) = \max\{F(i, W), f_{i+1} + F(i, W - s_{i+1})\}.$$

A rekurzió valóban fennáll. A részprobléma optimális megoldásában vagy szerepel az $i+1$-edik tárgy vagy nem, és ezen két eset maximuma adja az optimális célfüggvényértéket. Amennyiben $W - s_{i+1}$ negatív a függvényérték $F(i, W)$ lesz.

A fentiek alapján egy négyzetes táblázatkitöltéssel megkaphatjuk az optimális célfüggvényértéket. Egy optimális megoldást megkapunk visszafejtéssel, ha egy $V(i, W)$ táblázatban számon tartjuk, hogy az $F(i, W)$ érték kiszámításakor az i tárgy szerepelt -e az optimum esetén.

7.10. Feladat Legyen $P(n)$ a lehetséges felrajzolások száma. Ekkor $P(1) = 1$. Tekintsük az $n \geq 1$ esetet. Vegyük a $2n$ ponton áthaladó kört. Jelölje i azon körök számát, amelyeket tartalmaz ez a kör. Ekkor a kör másik metszéspontja az x tengellyel a $(2(n-i)-1, 0)$ pont. A körön belül i további kört kell elhelyeznünk, ezt $P(i)$ féleképpen tehetjük meg. A többi kör az $1, \ldots, 2(n-i-1)$ pontokon kell átmenjen, ezeket a köröket $P(n-i-1)$ féleképpen rajzolhatjuk fel. Tehát az összes ilyen felrajzolások száma $P(n-i-1)P(i)$. Másrészt i bármely értéket felveheti 0 és $n-1$ között, így a $P(n)$ értékre a következő rekurzió áll fenn:

$P(n) = \sum_{i=0}^{n-1} P(n-i-1)P(i)$

A formulában az $i = n-1$ esetben és $i = 0$ esetben a szorzó $P(0)$. Definiáljuk $P(0)$ értékét 1-nek. A rekurzió alapján egyből adódik a dinamikus programozási algoritmus lineáris táblázatkitöltéssel.

7.11. Feladat Tegyük fel, hogy a lépcsők állapota egy K tömbben van megadva, az i-edik hely értéke 1 ha lehet az i-edik lépcsőre lépni 0 egyébként. Az eljárás hasonló a 7.1. feladatban használt eljáráshoz, annyi a különbség, hogy $T[i] = 0$, ha az i-edik lépcső korhadt.

```
Function P(n:Word) :Longint;
```

```
Const
   MaxN=5000;
Var
   T: Array[1..MaxN] Of Longint;
   i: Word;
Begin
   If K[1]=0 Then T[1]:=0 Else T[1]:=1;
   If K[2]=0 Then
     T[2]:=0
   Else If K[1]=0 Then
     T[2]:=1
   Else
      T[2]:=2;
   for i:=3 to n Do Begin
     If K[i]=0 Then
       T[i]:=0
     Else
      T[i]:=T[i-2]+T[i-1];
   end
   P:=T[n];
End;
```

7.12. Feladat Definiáljuk a következő részproblémákat. Legyen $t[i,j]$ az $A_i, \ldots, A_j$ sokszögre az optimális felbontásnak a költsége! Az elfajuló esetekre legyen t értéke 0. Ekkor az optimális felbontás költségét a $t[1,n]$ érték adja. Az értékekre a következő rekurzió áll fenn (azon döntés alapján, hogy milyen háromszög tartalmazza A_iA_j oldalt).

$$t[i,j] = \begin{cases} 0, & \text{ha } i = j \text{ vagy } i = j-1, \\ \min_{i<k<j}\{t[i,k] + t[k,j] + w(A_iA_kA_j)\}, & \text{különben} \end{cases}$$

A $t[i,j]$ értékeket olyan sorrendben kaphatjuk meg, ahogy $j-i$ növekszik. A $t[1,n]$ érték megadja az optimális célfüggvényértékét. A megoldást visszafejtéssel megkaphatjuk, ha minden (i,j) párra megjegyezzük, mely k esetén kaptuk az optimális értéket.

7.13. Feladat Vegyünk két sorozatot, $X = (x_1, \ldots, x_m)$-t és $Y = (y_1, \ldots, y_n)$-t. Legyen a leghosszabb közös részsorozat (a továbbiakban az LKR rövidítést használjuk) $Z = (z_1, \ldots, z_k)$. Jelölje X_i $i = 0, \ldots, m$ és Y_i $i = 0, \ldots, n$ és Z_i $i = 0, \ldots, k$ rendre az X, Y és Z sorozatok első i elemeiből álló prefixeit. Ekkor a következő rekurzió áll fenn:

- Ha $x_m = y_n$ akkor $z_k = x_m = y_n$ és Z_{k-1}, az X_{m-1} és Y_{n-1} sorozatok egy LKR-je.
- Ha $x_m \neq y_n$ és $z_k \neq x_m$, akkor Z, az X_{m-1} és Y sorozatok egy LKR-je.

- Ha $x_m \neq y_n$ és $z_k \neq y_n$, akkor Z, az X és Y_{n-1} sorozatok egy LKR-je.

Most legyen $c[i,j]$ az X_i és Y_j sorozatok LKR-jének a hossza. Ekkor ezen értékekre a következő rekurzió teljesül.

$$c[i,j] = \begin{cases} 0, & \text{ha } i = 0 \text{ vagy } j = 0, \\ c[i-1,j-1]+1, & \text{ha } i,j > 0 \text{ és } x_i = y_j, \\ \max\{c[i,j-1], c[i-1,j]\}, & \text{ha } i,j > 0 \text{ és } x_i \neq y_j. \end{cases}$$

Ekkor egy négyzetes táblázatkitöltéssel megkaphatjuk a $c[i,j]$ értékeket és $c[m,n]$ az LKR hosszát adja vissza. Egy LKR-t kapunk visszafejtéssel ha a c tömb kitöltése mellett egy b tömbben mindig megjegyezzük, hogy a rekurzióban mit használtunk c értékének megadásához. Az algoritmusok részletes leírása szerepel a Cormen, Leiserson, Rivest könyvben.

7.14. feladat A $c[ij]$ és $b[i,j]$ értékek: (A második táblázatban A jelöli, ha a végek megegyeztek azaz az átlósan lefele eső értéket használtuk, L és B azt ha a lefele vagy a balra eső értéket. Ha a maximumban mindkét rész egyenlő B-t választottuk.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 5 & 6 & 6 & 6 \\ 1 & 2 & 3 & 4 & 4 & 5 & 5 & 5 & 6 \\ 1 & 2 & 3 & 4 & 4 & 4 & 5 & 5 & 5 \\ 1 & 2 & 3 & 3 & 3 & 4 & 4 & 4 & 5 \\ 1 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} L & A & L & A & A & B & A & A & B \\ A & L & A & L & B & A & B & B & A \\ L & A & L & A & A & B & A & A & B \\ A & L & A & L & B & A & B & B & A \\ L & A & B & A & A & B & A & A & B \\ A & B & A & B & B & A & B & B & A \\ A & B & A & B & B & A & B & B & A \\ B & A & B & A & A & B & A & A & B \end{pmatrix}$$

Tehát az LKR hossza 6, az LKR-t megkapjuk, ha végigmegyünk a B táblázaton a felső sarokból az irányjelzések (balra, le, átlósan) alapján, és ebben az útban az átlós lépéseknek megfelelő betűket vesszük (az i-edik sor j-edik elemének, az X i-edik és az Y j-edik betűje felel meg átlós bejegyzésnél ezek megegyeznek). Következésképp egy LKR (b,b,a,a,b,a).

7.15. Feladat A dinamikus programozási eljáráshoz jelölje $F_i(x,y)$ $(x,y \leq \sum p_i)$ azt a minimális összeget, amelyet elérhetünk az első i munka ütemezése után a harmadik gépen ha az első két gépen az összeg legfeljebb x és y.

Az $F_1(x,y)$ értéke 0, ha az első munka kisebb, mint $\max\{x,y\}$ és p_1 egyébként.

A további értékek az $F_{i+1}(x,y) = \min\{F_i(x-p_{i+1},y), F_i(x,y-p_{i+1}), F_i(x,y)+p_{i+1}\}$ rekurzióval határozhatóak meg. A rekurziónál használt szétválasztás alapja az, hogy az $i+1$-edik munkát melyik gépre ütemezzük, ha az első gépre az első értékhez jutunk (ami egy már megoldott probléma optimuma), ha a másodikra a másodikhoz, ha a harmadikra a harmadikhoz. Így ezek minimuma lesz az F_{i+1} értéke.

Az optimális megoldás célfüggvényértékét az utolsó F_n függvény alapján találhatjuk meg, venni kell minden x,y-ra a $\max\{x,y,F_n(x,y)\}$ értéket és ezek minimuma lesz a célfüggvényérték.

Az optimális megoldást visszafejtéssel megtalálhatjuk, ha minden $F_i(x,y)$ érték esetén megjegyezzük, melyik választás adta a rekurzióban a minimumot.

Megjegyzés: A módszer általánosítható, több gép esetére, illetve más célfüggvényekre is.

7.16. Feladat Legyen $T[i,j]$ azon részprobléma megoldása, amelyben tükörszóvá kell tenni az i-edik karaktertől a j-edik karakterig tartó szót. Ekkor $T[i,i] = 0$, hiszen az egy betűből álló szó az tükörszó. Továbbá $T[i,i+1] = 0$, ha az i-edik és $i+1$-edik betűk megegyeznek, és $T[i,i+1] = 1$ egyébként. A további értékekre a függvényérték rekurzívan számolható. Ha az i-edik karakter megegyezik a j-edikkel, akkor $T[i,j] = T[i+1,j-1]$. Amennyiben nem egyeznek meg, akkor valamelyik szélső karaktert fel kell venni a másik oldalra és ezzel a beszúrással egy további részproblémára vezetjük vissza a feladatot. Ebben az esetben két lehetőség közül választjuk a jobbikat, így ekkor

$$T[i,j] = \min\{T[i+1,j]+1, T[i,j-1]+1\}$$

A rekurzió alapján kiszámolhatóak a $T[i,j]$ értékek, és a megoldás a $T[1,n]$ függvényérték. Arra kell ügyelnünk, hogy mindig egy már megoldott problémára vezessük vissza a feladatot, ez teljesül, ha az értékeket növekvő $j-i$ érték alapján számoljuk ki.

7.17. Feladat Legyen $F(x,y)$ a maximális érték, amit az (x,y) mezőig össze tudunk gyűjteni. Ekkor a kezdeti értékek $F(1,y) = \sum_{j=1}^{y} c_{1j}$ és $F(x,1) = \sum_{i=1}^{x} c_{i1}$, hiszen ezekben az esetekben csak felfele, illetve csak jobbra mehet a játékos. A közbenső mezőkre alulról vagy felülről léphetünk, így

$$F(x,y) = \max\{F(x-1,y)+c_{xy}, F(x,y-1)+c_{xy}\}$$

Kiszámolva ezeket az értékeket az $F(k,n)$ érték adja meg, hogy mit gyűjthetünk össze maximálisan.

Az optimális megoldást visszafejtéssel megtalálhatjuk, ha minden $F(x,y)$ érték esetén megjegyezzük, melyik választás adta a rekurzióban a maximumot.

8. Mohó algoritmusok

8.1. Feladat Legyen a P_i részprobléma az a feladat, hogy az $x_i,\ldots,x_n$ pontokat fedjük le minimális számú egységnyi intervallummal. A P_i részprobléma esetén a mohó választás az, hogy a lehető legnagyobb kezdőponttal rendelkező

intervallumot vesszük, ami tartalmazza az x_i pontot, azaz az $[x_i, x_i + 1]$ intervallumot. Tegyük fel, hogy ez az intervallum az $x_i, \ldots, x_{j-1}$ pontokat fedi le, és a maradék pontokra az optimális megoldást vesszük. Ekkor a P_i probléma így kapott megoldásának a költsége $1 + OPT(P_j)$. Másrészt ez valóban $OPT(P_i)$ hiszen P_i minden megoldásában le kell fedni az x_i pontot, és bárhogy választunk lefedő intervallumot az $x_j, \ldots, x_n$ pontok szabadon maradnak. Tehát a megoldó mohó algoritmus a következőképpen működik

```
M:=[x_1,x_1+1]
x:=x_1+1
for i:=2 to n do
    if x_i>x then{
       M:=M U {[x_i,x_i+1]}
       x=x_i+1
      }
```

8.2. Feladat Legyenek a $K_{n-1}, \ldots, K_0$ a benzinkutak az út során elérhető sorrendben, és K_n a kiindulási pontunk. A P_i részprobléma az a feladat, hogy az K_i pontból, ahol tele a tank jussunk el a célba minimális számú megállással. A P_i részprobléma esetén a mohó választás az, hogy a lehető legutolsó benzinkútnál állunk meg, azaz az utolsó olyan kútnál amely nincs messzebb, mint n. Tegyük fel, hogy ez az út a $K_i, \ldots, K_j$ pontokat fedi le, és a maradék útra az optimális megoldást vesszük. Ekkor a P_i probléma így kapott megoldásának a költsége $1 + OPT(P_j)$. Másrészt ez valóban $OPT(P_i)$ hiszen P_i minden megoldásában tankolni kell a $K_{i+1}, \ldots, K_j$ kutak valamelyikénél, és bárhogy választunk kutat legalább a K_j-től kezdődő utat meg kell még tennünk. Tehát az algoritmus mindig az utolsó kútnál áll meg, ahol még nem fogy el a benzin.

8.3. Feladat Felrajzolva a Huffman kód kiszámítását leíró fákat, kapjuk, hogy $g = 0$, $f = 10$, $e = 110$, $d = 1110$, $c = 11110$, $b = 111110$, $a = 111111$. A megoldás ugyanígy általánosítható. Teljes indukcióval egyszerűen igazolható, hogy az első i gyakoriság összege pontosan az $i + 2$-edik gyakoriság minusz 1, következésképpen amikor a faépítés során az elemeket összevonjuk, akkor a belső pontot párosítjuk az $i + 1$-edik elemmel.

8.4. Feladat Az állítást indirekt igazoljuk. Tegyük fel, hogy nem igaz. Ekkor van olyan optimális prefix kód, amelyre ha a karaktereket gyakoriság szerint szigorúan csökkenő sorrendbe helyezzük, akkor a megfelelő kódszavak hosszai nem alkotnak nemcsökkenő sorozatot. Ez azt jelenti, hogy van két karakter, legyenek a és b, amelyekre $f(a) > f(b)$ és $d_T(a) > d_T(b)$ (d_T a kód hossza). Vegyük azt a kódot, amelyben a és b kódjait felcseréljük. Ez is egy prefix kód lesz. Legyen C azon karakterek halmaza, amelyek nem egyeznek meg a-val és b-vel. Ekkor az eredeti kódolás költsége

$$\sum_{i \in C} f(i)d_T(i) + f(a)d_T(a) + f(b)d_T(b)$$

az új kódunk költsége

$$\sum_{i \in C} f(i)d_T(i) + f(a)d_T(b) + f(b)d_T(a).$$

A különbség

$$f(a)d_T(a)+f(b)d_T(b)-(f(a)d_T(b)+f(b)d_T(a)) = (f(a)-f(b))(d_T(a)-d_T(b)) > 0.$$

Következésképp az eredetinél egy kisebb költségű kódot adtunk meg, azaz ellentmondáshoz jutottunk, amivel igazoltuk az állítást.

8.5. Feladat a) Ebben az esetben a $P(i)$ részprobléma i forint optimális felváltása. A mohó stratégia az, hogy mindig a lehető legnagyobb pénzérmét használjuk a váltáshoz, amely még nem nagyobb az aktuális felváltandó összegnél. Az algoritmus optimalitásának igazolásához azt kell igazolnunk, hogy minden felváltandó összegre van olyan optimális felváltás, amely használja a legnagyobb a felváltandónál nem nagyobb váltópénzt. Elsőként tegyük fel, hogy $i \geq 25$, és vegyünk egy optimális felváltást, ami nem tartalmaz 25-ös pénzérmét. A következő eseteket különböztessük meg!

- A felváltás tartalmaz legalább 3 darab 10-est: Ekkor a három tízest kicseréljük $25 + 5$-re, és így egy jobb megoldást kapunk.
- A felváltás 3-nál kevesebb 10-est tartalmaz. Ekkor egyszerű esetszétválasztással adódik, hogy a felváltásnál használt pénzeknek van olyan részhalmaza, amely pontosan 25 összértékű, és ezt az érmehalmazt kicserélve egy 25-ös érmével, jobb megoldást kapunk.

Azok az esetek, amelyekben $25 > i \geq 10$ vagy $10 > i \geq 5$ teljesen hasonlóan igazolhatóak. Mindig lesz az érméknek olyan részhalmaza, amely pontosan 10 vagy pontosan 5.

b) A feladat teljesen hasonló az a) esethez, azt kell igazolnunk, hogy minden felváltandó összegre van olyan optimális felváltás, amely használja a legnagyobb a felváltandónál nem nagyobb váltópénzt. Az állításnál többet igazolhatunk. Az optimális felváltás, a szám c alapú számrendszerbe felírt formája, ahol a c^k-nál nagyobb helyiértéken levő számokat c^k-s értékekkel fejezzük ki. Ez az állítás úgy igazolható, hogy tetszőleges felbontásból kiindulunk, és az egyesektől kezdve minden c^k-nál kisebb érmére, amennyiben c-nél több van helyettesítjük a c-eseket az eggyel értékesebb érmével, ezzel rendre jobb megoldáshoz jutva.

c) Legyenek az érmék $1, 4, 5$. Ekkor $n = 8$-ra a mohó eredmény $8 = 5 + 1 + 1 + 1$, az optimális $8 = 4 + 4$.

8.6. Feladat Ha a osztója b-nek egyből adódik, hogy a mohó algoritmus optimális, tetszőleges b-nél nem kisebb szám felváltása esetén, lesz az érméknek olyan halmaza, amelyben az összeg b, és ezt kicserélve b-vel egy nem rosszabb megoldást kapunk. Ha a nem osztója b-nek, akkor írjuk b-t fel $k \cdot a + m$ alakban,

ahol $0 < m < a$. Ekkor amennyiben a $(k+1)a$ szám mohó felírása (b és $a-m$ darab 1) több érmét tartalmaz, mint $k+1$, akkor a mohó nem optimális. Ellenkező esetben a mohó algoritmus optimális, mert a $k+1$ darab a-t ki tudjuk cserélni egy b-re és $a-m$ darab 1-re.

8.7. Feladat Tegyük fel, hogy az intervallumokra teljesül, $a_1 \leq a_2 \leq \ldots \leq a_n$, amennyiben ez nem teljesül az intervallumokat egy előkészítő részben így rendezzük. Legyen a P_i részprobléma az a feladat, hogy az $[a_i, b_i], \ldots, [a_n, b_n]$ intervallumokra jelöljünk ki minimális elemszámú ponthalmazt, amelynek minden intervallumban van pontja. A P_i részprobléma esetén a mohó választás az, hogy a lehető legnagyobb számot választjuk, amely nem nagyobb egyetlen b_j végpontnál sem. Ez valójában a minimális b_j érték. Tegyük fel, hogy $k-1$ a maximális érték, amelyre $a_{k-1} \leq b_j$. Ekkor a P_i probléma így kapott megoldásának a költsége $1 + OPT(P_k)$ lesz. Másrészt ez valóban $OPT(P_i)$ hiszen P_i minden megoldásában kell, hogy legyen pont az $[a_j, b_j]$ intervallumban, és ezen intervallumnak nincs közös pontja az $[a_k, b_k], \ldots, [a_n, b_n]$ intervallumok egyikével sem.

Tehát az algoritmus minden iterációs lépésben kiválasztja a minimális b_j elemet az aktuális intervallumok halmazából, beveszi a megoldásba, majd elhagyja azokat az intervallumokat, amelyeknek a kezdőpontja kisebb b_j-nél.

8.8. Feladat Vegyünk egy részproblémát, ami ez esetben a hátralevő munkák ütemezése. Tekintsünk egy olyan ütemezést (jelölje S), amely nem egy legkisebb határidővel rendelkező munkát ütemez elsőnek. Legyen a legkisebb határidővel rendelkező munka j. Tegyük fel, hogy S valamilyen $i_1, i_2, \ldots, i_k, j$ munkasorrenddel kezdődik. Vegyük S'-t, amelyet úgy kapunk S-ből, hogy a munkák ezen kezdeti sorrendjét kicseréljük a $j, i_1, i_2, \ldots, i_k$ sorrendre. Az új ütemezésre az $i_1, \ldots, i_k$ munkák később fejeződnek be, viszont a befejezési idejük nem lesz nagyobb, mint j-nek az S ütemezésben, ezért a késedelmük sem, hiszen j határideje minimális volt. Következésképpen az így kapott megoldás is optimális lesz, amivel igazoltuk, hogy a mohó algoritmus optimális.

9. Bactrack és $B\&B$

9.1. Feladat A megoldásokat $i_1 < i_2 < \ldots < i_m$ formában keressük, a megoldástér pontjait a kiválasztott pénzérmék vektoraként írjuk le, a két db 4 forintost megkülönböztetjük, az egyiket, amelyiket a sorban előre vesszük $4'$ jelöli.

- Az (5) megoldáskezdeménnyel indulunk.
- Utána az $(5, 4')$ megoldáskezdemény nem LEHETMEGO (kizárt)
- Utána az $(5, 4)$ megoldáskezdemény nem LEHETMEGO (kizárt)
- Utána az $(5, 2)$ megoldáskezdemény esetén továbbmegyünk a fiára
- Az $(5, 2, 1)$ megoldás a célfüggvényt felülírjuk 3-ra, a megoldást megjegyezzük.

- Utána az $(5,1)$ megoldáskezdemény nem LEHETMEGO (kizárt)
- A $(4')$ megoldáskezdeménnyel folytatjuk.
- A $(4',4)$ megoldás, a célfüggvényt felülírjuk 2-re a megoldást megjegyezzük.
- Utána az $(4',2)$ nem LEHETMEGO (kizárt)
- A (4) nem LEHETMEGO (kizárt)
- A (2) nem LEHETMEGO (kizárt)
- A (1) nem LEHETMEGO (kizárt)

9.2. Feladat A megoldástért hasonlóan építhetjük fel, miként a pénzváltási problémában. A megoldásokat $i_1 < i_2 < \ldots < i_m$ formában keressük, ami azt jelenti, hogy az $i_1, i_2, \ldots, i_m$-edik tárgy van berakva a megoldáskezdeményben a hátizsákba. Ugyanolyan struktúrájú fa a megoldástér, mint a pénzváltási problémában a nem bináris esetben. Definiáljuk a problémaspecifikus műveleteket.
Az URESX műveletben egy üres megoldáskezdményt kell létrehozni.
Az EFIU művelet az $\langle i_1, \ldots, i_m \rangle$ megoldáskezdemény helyett az $\langle i_1, \ldots, i_m, i_m + 1 \rangle$ megoldáskezdeményt adja, ha $i_m < n$, egyébként hamis.
Az APA művelet az $\langle i_1, \ldots, i_m \rangle$ megoldáskezdemény helyett az $\langle i_1, \ldots, i_{m-1} \rangle$ megoldáskezdeményt adja.
Az TESTVER művelet az $\langle i_1, \ldots, i_m \rangle$ megoldáskezdemény helyett az $\langle i_1, \ldots, i_{m-1}, i_m + 1 \rangle$ megoldáskezdeményt adja, ha $i_m < n$, egyébként hamis.
A VISSZAALLIT műveletet, arra használjuk, hogy a bejegyzett segédinformációkat töröljük, ebben az esetben a $Resz = \sum_{j=1}^{m} s_{i_j}$ és az érték $ertek = \sum_{i=1}^{m} f_{i_j}$ mezőket tároljuk, a visszaállítás során ezeket kell visszaállítani.
A LEHETMEGO művelet akkor adjon igaz értéket, ha $\sum_{j=1}^{m} s_{i_j} \leq S$, azaz ha a kijelölt tárgyaink elférnek a hátizsákban.
A MEGOLDAS művelet szintén akkor adjon igaz értéket, ha $\sum_{j=1}^{m} s_{i_j} \leq S$, azaz ha a kijelölt tárgyaink elférnek a hátizsákban.

Az eljárást minimalizálási problémára adták meg. A célfüggvényt -1-el szorozva a maximum problémáról áttérhetünk minimum problémára. Másik lehetőség az eljárás egyszerű módosítása, miszerint az aktuális optimális megoldást akkor írjuk felül, ha a kapott célfüggvényünk nagyobb. A következő példában ezt a módszert használjuk.

9.3. Feladat Jelölje a tárgyakat $A = (4,6)$ $B = (3,5)$ $C = (2,3)$ $D = (2,3)$!

- Az (A) megoldáskezdeménnyel indulunk, ez MEGOLDAS, megjegyezzük az aktuális legjobb célfüggvény $CEL = 6$.
- Utána az (A,B) MEGOLDAS, jobb a célfüggvény $11 > 6$, így $CEL = 11$. (A,B)-vel felülírjuk (A)-t.
- Utána az (A,B,C) megoldáskezdemény nem LEHETMEGO (kizárt).

- Utána az (A, B, D) megoldáskezdemény nem LEHETMEGO (kizárt).
- Utána az (A, C) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (A, C, D) MEGOLDAS, jobb a célfüggvény $12 > 11$, így $CEL = 12$. (A, C, D)-vel felülírjuk (A, B)-t.
- Utána az (A, D) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (B) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (B, C) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (B, C, D) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (B, D) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (C) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (C, D) MEGOLDAS, a célfüggvény nem jobb mint CEL.
- Utána az (D) MEGOLDAS, a célfüggvény nem jobb mint CEL.

9.4. Feladat A megoldásokat $i_1 < i_2 < \ldots < i_m$ formában keressük, a megoldástér pontjait a kiválasztott pénzérmék vektoraként írjuk le, a két db 4 forintost megkülönböztetjük, az egyiket, amelyiket a sorban előre vesszük $4'$ jelöli.

- Az ADAGOLO-ba az (5), (4') megoldáskezdemények kerülnek be. Ekkor a GFK (legjobb ismert megoldás) értéke végtelen. Az ADAGOLO-ban a függvényértékek $AK((5)) = 1 + \lceil (8-5)/4 \rceil = 2$, $FK((5)) = 1 + \lceil (8-5)/1 \rceil = 4$ $AK((4')) = 1 + \lceil (8-4)/4 \rceil = 2$, $FK((4')) = 1 + \lceil (8-4)/1 \rceil = 5$.
- Az FK szerinti minimumos sor az (5) kezdeményt adja ki.
- Az ADAGOLO-ba az $(5, 2)$ megoldáskezdemény kerül be. Erre $AK((5, 2)) = 2 + \lceil (8-7)/1 \rceil = 3$, $FK((5, 2)) = 1 + \lceil (8-7)/1 \rceil = 3$.
- Az FK szerinti minimumos sor az $(5, 2)$ kezdeményt adja ki.
- Az ADAGOLO-ba egyetlen megoldáskezdemény kerül be az $(5, 2, 1)$, erre $AK(5, 2, 1) = FK(5, 2, 1) = 3$, és találtunk egy megoldást. Az aktuális optimális megoldás $(5, 2, 1)$ a hozzá tartozó GFK érték 3.
- Az FK szerinti minimumos sor az $(5, 2, 1)$ kezdeményt adja ki, nem kerül új elem az ADAGOLO-ba.
- Az FK szerinti minimumos sor a $(4')$ kezdeményt adja ki.
- Az ADAGOLO-ba egyetlen megoldáskezdemény kerül be $(4'4)$, találunk egy megoldást a $(4', 4)$ megoldást. Ezt megjegyezzük, és $GFK = 2$.

- Az FK szerinti minimumos sor az $(4', 4)$ kezdeményt adja ki, nem kerül új elem az ADAGOLO-ba.
- Az ADAGOLO üres, az eljárás véget ér.

9.5. Feladat Mind a megoldástér, mind a problémaspecifikus függvények definíciója megegyezik a 9.2. feladatban megadottal. Itt a korlátozó függvényeket kell definiálnunk. Az alsó korlátra nem adható jobb függvény, mint a már hátizsákban levő tárgyak összhasznossága. Azaz

$$AK((i_1, \ldots, i_m)) = \sum_{j=1}^{m} f_{i_j}.$$

Felső korlátot jobbat adhatunk, mint a triviálisan adódó összes a hátizsákból nem kizárt tárgy összhasznossága. Tekintsük azt a változatot (töredékes hátizsák probléma), amelyben a tárgyak részei berakhatóak a hátizsákba és a tárgy x-ed részének a hasznossága az eredeti hasznosság szorozva x-el. Nyilván ezen probléma optimális megoldása nem ad kisebb hasznosságot, mint az eredeti problémáé. Legyen ezen probléma megoldása az I tárgyhalmazon, egy W kapacitású hátizsákkal $OPT(I, W)$. Ekkor

$$FK((i_1, \ldots, i_m)) = \sum_{j=1}^{m} f_{i_j} + OPT(\{i_m + 1, \ldots, n\}, S - \sum_{j=1}^{m} s_{i_j}).$$

Ahhoz, hogy az eljárásunk hatékony legyen szükséges az $OPT(I, W)$ értékeket hatékonyan meghatározni. Ezen problémára a mohó eljárás optimális megoldást ad (a tárgyakat sorba rakjuk csökkenő relatív hasznosság szerint (f_i/s_i), és így töltjük a hátizsákot, amíg be nem telik).

A kereteljárást minimalizálási problémára adták meg. A célfüggvényt -1-el szorozva a maximum problémáról áttérhetünk minimum problémára. Másik lehetőség az eljárás egyszerű módosítása, miszerint az aktuális optimális megoldást akkor írjuk felül, ha a kapott célfüggvényünk nagyobb. Továbbá, akkor tudjuk, hogy egy megoldáskezdeménynek nincs folytatása, ha a felső korlát kisebb, mint az ismert legjobb megoldás.

A következő példában ezt a módszert használjuk.

9.6. Feladat Jelölje a tárgyakat $A = (4, 6)$ $B = (3, 5)$ $C = (2, 3)$ $D = (2, 3)$! Felső korlát szerinti maximum prioritási sort használunk.

- Az ADAGOLO-ba az (A), (B) megoldáskezdemények, akik egyben megoldások kerülnek be. Ekkor a GFK (legjobb ismert megoldás) 6. Amikor a (C) és (D) megoldáskezdeményeket vizsgáltuk $GFK \geq FK(C)$ és $GFK \geq FK(D)$ állt fenn, így nem kerültek be. Az ADAGOLO-ban a korlátozó függvények

  $AK((A)) = 6$, $FK((A)) = 6 + 5 + 3/2 = 25/2$ $AK((B)) = 5$, $FK((B)) = 5 + 3 + 3 = 11$.

- Az FK szerinti maximumos sor az (A) kezdeményt adja ki.
- Az (A, B) megoldás, így $GFK = 6 + 5 = 11$. Az aktuális optimális megoldást felülírjuk. Bekerül az ADAGOLO-ba. A korlátok $AK((A, B)) = 11$, $FK((A, B)) = 6 + 5 + 3/2 = 25/2$.
- Az (A, C) megoldás, és $FK(A, C) = 6+3+3 \geq GFK = 6+5$, így bekerül az ADAGOLO-ba. A korlátok $AK((A, C)) = 9$, $FK((A, C)) = 12$.
- Az FK szerinti maximumos sor az (A, B) kezdeményt adja ki.
- A fiúkra (A, B, C) és (A, B, D) nem LEHETMEGO, így nem kerülnek be az ADAGOLO-ba.
- Az FK szerinti maximumos sor az (A, C) kezdeményt adja ki.
- A fia (A, C, D) megoldás, így $GFK = 6+3+3 = 12$. Az aktuális optimális megoldást felülírjuk, a megoldás bekerül az ADAGOLO-ba.
- Az FK szerinti maximumos sor az (A, C, D) majd a (B) kezdeményeket adja ki, mindkettőre az FK érték nem nagyobb, mint GFK.
- Az ADAGOLO üres, az eljárás véget ér.

10. Gráfok és gráfábrázolások

10.1. Feladat Vegyünk egy embert A-t! Ekkor A-ra teljesül, hogy vagy van három ember, hogy mindegyiket ismeri vagy van három, hogy egyiket sem. Ha van három ember, hogy mindet ismeri, legyenek ők B, C, D. Ha van B, C, D között kettő, aki ismeri egymást, akkor ezen két ember és A megad egy keresett hármast. Ha B, C, D közül semelyik kettő nem ismeri egymást, akkor a B, C, D hármas rendelkezik a kívánt tulajdonsággal. Az az eset, ha A-ra van három ember, akiket nem ismer hasonlóan kezelhető. A gráf esetén az állítás azt mondja ki, hogy vagy van a gráfban három páronként összekötött pont (háromszög) vagy van három pont, amelyek közül semelyik kettő nincs összekötve (háromszög a komplementer gráfban).

10.2. Feladat a) Legyenek a gráf pontjai a munkások és a munkák. Legyen egy él minden munka és minden munkás között, az él súlya a c_{ij} érték. Feladatunk kiválasztani élek egy olyan halmazát, amely minimális összsúlyú és tartalmazza az összes munkát.

b) Egy optimális megoldást mohó módszerrel kaphatunk. Minden munkára kiválasztjuk azt az élt, a munkát tartalmazó élek közül amire minimális a súly. Ez optimális megoldást ad, mivel minden él pontosan egy munkát tartalmaz.

c) A feladatot ugyanazzal a gráffal reprezentálhatjuk, de a cél olyan élhalmazok közül a minimális összsúlyú halmaz kiválasztása, amelyek minden pontot a gráfban pontosan egyszer tartalmaznak (minimális súlyú párosítás).

d) Ekkor a gráf is változik. Töröljük azokat az éleket, amik olyan munkát és munkást kötnek össze, amelyekre a munkás nem végezheti el a munkát.

10.3. Feladat A gráf pontjai az (x, y, z) egész számhármasok, amelyekre $k_1 \geq x \geq k_2$, $p_1 \geq y \geq p_2$, $z_1 \geq z \geq z_2$. Egy ilyen pont azt az asztalt reprezentálja, amelyen x kék, y piros és z zöld pont van. Az élek azt adják meg ha egy pontnak megfelelő asztalból megkapható egy másik, így az (x, y, z) pont szomszédjai $(x, y-2, z-3)$, $(x-2, y-3, z)$ és $(x-3, y, z-2)$. A feladat, hogy megállapítsuk van -e a (k_1, p_1, z_1) pontból a (k_2, p_2, z_2) pontba út és ha van, akkor határozzuk meg a minimális hosszút.

Fontos megjegyeznünk, hogy ebben az esetben minden lehetséges út hossza ugyanakkora, mivel az asztalon levő golyók száma minden lépésben 5-tel csökken.

10.4. Feladat A gráf pontjai az (x, y, z) számhármasok, ahol minden komponens nemnegatív egész és $x+y+z = 42$. Az élek a lehetséges átváltozásoknak felelnek meg. Tehát (x, y, z)-nek három szomszédja lesz $(x-1, y-1, z+2)$, $(x-1, y+2, z-1)$, $(x+2, y-1, z-1)$. A feladat, hogy meghatározzuk van -e a gráfban a $(13, 14, 15)$ pontból a $(42, 0, 0)$, $(0, 42, 0)$, $(0, 0, 42)$ pontok valamelyikébe út, és ha vannak ilyen utak adjunk meg egy legrövidebbet.

Az invariáns keresés módszerével igazolható az állítás. Vegyük észre, hogy a megengedett lépések során az $(x-y, y-z, x-z)$ számok hárommal való osztásmaradéka nem változik. (Azaz ez a tulajdonság a megengedett lépésekre nézve invariáns.) Kezdetben, ezen osztásmaradékok $(2, 2, 1)$, és minden kívánt végállapotban $(0, 0, 0)$, így a végállapotok egyike sem érhető el.

10.5. Feladat A gráf pontjai az (x, y, z, w) számnégyesek, ahol minden komponens nemnegatív egész és $x + y + z + w \leq a + b + c + d$. Az élek a lehetséges átváltozásoknak felelnek meg. Tehát (x, y, z, w)-nek öt szomszédja lesz $(x-1, y-1, z-1, w+3)$, $(x-1, y-1, z+3, w-1)$, $(x-1, y+3, z-1, w-1)$, $(x+3, y-1, z-1, w-1)$, $(x-1, y-1, z-1, w-1)$. A feladat, hogy meghatározzuk van -e a gráfban az (a, b, c, d) pontból a $(0, 0, 0, 0)$ pontba út.

10.6. Feladat Konstruáljunk a feladathoz egy gráfot. A gráf csúcsai a sakktábla mezői, két n-nél nem nagyobb pozitív koordinátából álló vektorok. Két csúcs össze van kötve, ha a huszár átléphet egyikből a másikba. Tehát az (x, y) szomszédjai az $(x-1, y+2)$, $(x-1, y-2)$, $(x+1, y+2)$, $(x+1, y-2)$, $(x+2, y-1)$, $(x+2, y+1)$, $(x-2, y-1)$, $(x-2, y+1)$ párosok közül azok, amelyekben mindkét koordináta pozitív és nem nagyobb, mint n. A feladat megtalálni a legrövidebb utat az $(1, 1)$ pontból az (u, v) pontba.

Megjegyzés A 10.3-10.6 feladatoknál használt módszer minden olyan esetben használható, ahol azt kell eldönteni, hogy valamilyen kiindulási konfigurációból eljuthatunk -e végkonfigurációk valamelyikébe valamely lépések alapján. Ekkor a gráfban a csúcsok a lehetséges konfigurációk, él megy két csúcs között ha közvetlenül egy lépéssel át lehet lépni az egyik konfigurációból a másikba.

10.7. Feladat A kapcsolatmátrix a következő.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Az éllista:

	1	2	3	4	5	6
1	2	4	6	0		
2	2	3	4	0		
3	1	5	0			
4	1	4	0			
5	2	3	6	0		
6	1	2	5	0		

A halmazfüggvény:
$r_E(1) = \{2,4,6\}$, $r_E(2) = \{2,3,4\}$, $r_E(3) = \{1,5\}$, $r_E(4) = \{1,4\}$, $r_E(5) = \{2,3,6\}$, $r_E(6) = \{1,2,5\}$

11. Szélességi, mélységi keresés,

11.1. Feladat Tegyük fel, hogy a ki iteráció növekvő index szerint veszi a pontokat. Kezdetben az algoritmusban $D(1) = 0$, $Apa(1) = Null$. $Sor = \langle 1 \rangle$. Utána

- Kijön 1 majd $D(2) = 1$, $Apa(2) = 1$, $D(4) = 1$, $Apa(4) = 1$, $D(6) = 1$, $Apa(6) = 1$ és $Sor = \langle 2,4,6 \rangle$.
- Kijön 2 majd $D(3) = 2$, $Apa(3) = 2$ és $Sor = \langle 4,6,3 \rangle$.
- Kijön 4 és $Sor = \langle 6,3 \rangle$.
- Kijön 6 majd $D(5) = 2$, $Apa(5) = 6$ és $Sor = \langle 3,5 \rangle$.
- Kijön 3 és $Sor = \langle 5 \rangle$.
- Kijön 5 és $Sor = \langle \rangle$.

Következésképpen $\delta(1,5) = 2$.
Amennyiben az 5 pontból indulunk, akkor teljesen hasonlóan kapjuk, hogy a sorrend: $5,2,3,6,4,1$ és $\delta(5,1) = 2$.

11.2. Feladat A labirintust gráfként reprezentáljuk. Az üres mezők lesznek a gráf pontjai, két pont között él vezet, ha egyikből át lehet lépni a másikba, azaz ha szomszédosak. Ezen gráfra végrehajtunk egy szélességi keresést, pontosabban

nem kell végrehajtani egy teljes szélességi keresést az algoritmus megáll az első olyan pontnál, amely a labirintus szélén van. Az alábbiakban azt az algoritmust prezentáljuk, amelyik eldönti ki lehet-e jutni a labirintusból, és egy a kiindulási ponttól minimális távolságra levő pontnál áll meg, az algoritmusban szereplő segédinformációk alapján, könnyen megkapható a legrövidebb út is.

```
Const
  MaxN=1000;
  Lep:Array[1..4] of           {a négy lehetséges lépés}
       Record dx,dy:-1..1 End= {relatív koordinátái}
      ((dx:-1;dy:0), (dx:0;dy:1), (dx:1;dy:0), (dx:0;dy:-1) );
Type
  PontTip=Record
            Sorszam:Word;
            Oszlopszam:Word
          End;
  Fuggveny1=Array[1...MaxN,1...MaxN] of PontTip;
  Fuggveny2=Array[1...MaxN,1...MaxN] of Word;
  Labirintus=Array[1...MaxN,1...MaxN] of Boolean;

Procedure Kijut (Const L:Labirintus; {a labirintus}
                     K,N:Word;       {a labirintus mérete}
                       S:Ponttip;    {az indulási mező}
                 Var Van: Boolean;
                 Var Apa:Fuggveny1;
                 Var   D:Fuggveny2);
Const
  Null=0;
  Inf=MaxN*MaxN;
Var
  Q:Sor;
  u,v:PontTip;
  i,j:Word;
Begin{Kijut}
   Letesit(Q);
   For i:=1 to K Do
     For j:=1 to N Do Begin
       Apa[i,j].Sorszam:=Null; Apa[i,j].Oszlopszam:=Null;
       D[i,j]:=Inf;
     End{for j};

    D[S.Sorszam, S.Oszlopszam]:=0;
    Sorba(Q,S);

    While Elemszam(Q)<>0 Do Begin
        Sorbol(Q,u);
```

```
      If ((u.Sorszam=1) or (u.Sorszam=K) or
         (u.Oszlopszam=1) or (u.Oszlopszam=N)) Then Begin
          Van:=True;
          Exit;
      End Else Begin
        For i:=1 To 4 Do Begin {lépés a négy lehetséges irányban}
          v.Sorszam:=u.Sorszam+Lep[i].dx;
          v.Oszlopszam:=u.Oszlopszam+Lep[i].dy;
          If (1<=v.Sorszam)And(v.Sorszam<=N) And
             L[v.Sorszam, v.Oszlopszam] And
             D[v.Sorszam, v.Oszlopszam]=Inf Then Begin
               D[v.Sorszam, v.Oszlopszam]:=D[u.Sorszam, u.Oszlopszam]+1;
               Apa[v.Sorszam, v.Oszlopszam]:=u;
               Sorba(Q,v)
          End
        End{for i};
      End;
   End{while}
   Van:=False;
End{Kijut};
```

11.3. Feladat Megadjuk a D táblázatot a $(4,3)$-ból indulva.

$$\begin{pmatrix} Inf & 4 & Inf & Inf & Inf & Inf \\ 4 & 3 & 4 & Inf & Inf & Inf \\ Inf & 2 & Inf & Inf & Inf & Inf \\ Inf & 1 & 0 & Inf & Inf & Inf \\ Inf & Inf & 1 & Inf & Inf & Inf \\ Inf & Inf & Inf & Inf & Inf & Inf \end{pmatrix}$$

Az Apa táblázat:

$$\begin{pmatrix} (0,0) & (2,2) & (0,0) & (0,0) & (0,0) & (0,0) \\ (2,2) & (3,2) & (2,2) & (0,0) & (0,0) & (0,0) \\ (0,0) & (4,2) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (4,3) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (4,3) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \end{pmatrix}$$

Tehát ki lehet jutni, egy legrövidebb út a $(4,3),(4,2),(3,2),(2,2),(2,1)$.
Megadjuk a D táblázatot a $(2,5)$-ből indulva.

$$\begin{pmatrix} Inf & Inf & inf & Inf & Inf & Inf \\ Inf & Inf & Inf & Inf & 0 & Inf \\ Inf & Inf & Inf & 2 & 1 & Inf \\ Inf & Inf & Inf & Inf & 2 & Inf \\ Inf & Inf & Inf & Inf & Inf & Inf \\ Inf & Inf & Inf & Inf & Inf & Inf \end{pmatrix}$$

Az Apa táblázat:

$$\begin{pmatrix} (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (3,5) & (2,5) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (3,5) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \\ (0,0) & (0,0) & (0,0) & (0,0) & (0,0) & (0,0) \end{pmatrix}$$

Tehát nem lehet kijutni.

11.4. Feladat Mindegyik feladatban az a kérdés, hogy egy gráf valamely pontjából el lehet -e jutni egy másik pontjába, illetve néhány feladatban a legrövidebb utat is meg kell adni. Ez pontosan az a feladat, ami megoldható a szélességi kereséssel. Ezeknél a feladatoknál is a számított gráf reprezentációt érdemes használni, nem szükséges előre meghatározni az egész gráfot, elegendő mindig az aktuális pont szomszédait kiszámolni.

11.5. Feladat Elsőként az $(1,1)$ pont szomszédai kerülnek be a sorba, ezekre az Apa függvény $(1,1)$ lesz, a D függvény 1, tehát a használt sor, $S = \langle(3,2),(2,3)\rangle$. Utána kivesszük a $(3,2)$ pontot, ezen pont még nem vizsgált szomszédai kerülnek be a sorba, ezekre az Apa függvény $(3,2)$ lesz, a D függvény 2, tehát $S = \langle(2,3),(1,3),(5,1),(5,3),(4,4),(2,4)\rangle$. Utána kivesszük a $(2,3)$ pontot, ezen pont még nem vizsgált szomszédai kerülnek be a sorba, majd a szélességi keresés eljárása alapján tovább folytatjuk a keresést. A talált legrövidebb út $(1,1),(3,2),(1,3),(3,4),(2,2)$.

11.6. Feladat Az elérési és az elhagyási idők és az Apa értékek a végrehajtás sorrendjében: $D[1] = 1$, $Apa[2] = 1$, $D[2] = 2$, $Apa[3] = 2$, $D[3] = 3$, $Apa[5] = 3$, $D[5] = 4$, $Apa[6] = 5$, $D[6] = 5$, $Apa[7] = 6$, $D[7] = 6$, $F[7] = 7$, $F[6] = 8$, $F[5] = 9$, $F[3] = 10$, $Apa[4] = 2$, $D[4] = 11$, $F[4] = 12$, $F[2] = 13$, $F[1] = 14$, $D[8] = 15$, $Apa[9] = 8$, $D[9] = 16$, $F[9] = 17$, $F[8] = 18$.

Az élek osztályozása:

Faélek: $(1,2),(2,3),(3,5),(5,6),(6,7),(2,4),(8,9)$

Visszaélek: $(5,2),(7,5),(9,8)$

Előreélek: $(1,3)$

Keresztélek: $(8,1)$.

11.7. Feladat Ha a 2 és 3 pontok közötti él irányítása $(2,3)$, akkor a mélységi bejárásban az érkezési és elhagyási idők: $D(1) = 1, D(2) = 2, D(3) = 3, D(4) = 4$, $F(4) = 5, F(3) = 6$, $F(2) = 7$, $F(1) = 8$. Ekkor $(2,3)$ faél, az 1 és 4 közötti él pedig előreél (ha $(1,4)$) vagy visszaél (ha $(4,1)$). Ha a 2 és 3 pontok közötti él irányítása $(3,2)$, akkor a mélységi bejárásban az érkezési és elhagyási idők: $D(1) = 1, D(2) = 2, D(4) = 3, F(4) = 4$, $F(2) = 5, D(3) = 6$, $F(3) = 7$, $F(1) = 8$. Ekkor $(3,2)$ keresztél, az 1 és 4 közötti él pedig előreél (ha $(1,4)$) vagy visszaél (ha $(4,1)$).

11.8. Feladat Megadjuk miként használhatjuk a mélységi keresés algoritmusát. Pontosabban megadjuk azokat a részeket ahol változtatnunk kell.

- A gráfunk pontjai a labirintus üres négyzetrácsai. Egy pont azokkal a szomszédos négyzetrácsokkal van összekötve, amelyek szintén üresek. Fontos megjegyeznünk, hogy nem ismerjük a gráfot, nem is férhetünk hozzá a gráf pontjaihoz.

- Nem tudjuk a teljes szélességi keresést végrehajtani, csak olyan pontokban hívhatjuk meg a MelyBejar eljárást, amit elértünk a keresésünk során. Tehát lényegében a kiindulási pontunkra hívjuk meg a MelyBejar algoritmust, és az eljárás vagy úgy ér véget, hogy kitaláltunk, ez egy kilépési feltétel, vagy úgy, hogy az eljárás befejeződik, ekkor nem lehet kijutni a labirintusból.

- Az Apa függvényt csak úgy tarthatjuk számon, hogy felírjuk milyen irányból léptünk a cellába, ahol a eljárást rekurzívan hívjuk.

11.9. Feladat Az eljárás elején azt látjuk, hogy előre, jobbra fal van, tehát két ponttal van összekötve a pontunk a hátra és a balra levővel.

Megjelöljük, hogy a mező, ahol állunk szürke, meghívjuk az eljárást a hátra eső ponttal.

Átlépünk. És felírjuk rá, hogy elölről érkeztünk (Apa függvény). Megjelöljük szürkére. Látjuk, hogy nincs érintetlen szomszédja, elöl szürke van, a többi szomszédos cella fal, a rekurzív hívás véget ér. Megjelöljük feketére, visszalépünk az apára.

Átlépünk a balra esőre. Ellenőrizzük, hogy érintettük-e már. Mivel nem felírjuk rá, hogy jobbról érkeztünk (Apa függvény). Megjelöljük szürkére. Látjuk, hogy egyetlen érintetlen szomszédja van, (hátra és balra fal van, azaz nincs szomszédos pont, jobbról érkeztünk az a cella már nem érintetlen). Meghívjuk az eljárást az előttünk levő cellára.

Átlépünk a előre eső cellára. Ellenőrizzük, hogy érintettük-e már. Mivel nem felírjuk rá, hogy hátulról érkeztünk (Apa függvény). Megjelöljük szürkére. Körbenézve látjuk, hogy jobbra és balra fal van, azaz nincs szomszédos pont, hátulról érkeztünk az a cella már nem érintetlen. Tehát egy érintetlen cella lehet ami előttünk van. Meghívjuk az eljárást az előttünk levő cellára.

Átlépünk a előre eső cellára. Ellenőrizzük, hogy érintettük-e már. Mivel nem felírjuk rá, hogy hátulról érkeztünk (Apa függvény). Megjelöljük szürkére. Körbenézve látjuk, hogy minden irányba mehetünk, hátulról érkeztünk az a cella már nem érintetlen. Tehát három érintetlen cella lehet ami balra, jobbra és előttünk van. Meghívjuk az eljárást a balra levő cellára.

A cellából kiléphetünk így az eljárás véget ér.

12. Topologikus rendezés, összefüggőség, Euler kör

12.1. Feladat Az elérési és az elhagyási idők és az Apa értékek a végrehajtás sorrendjében: $D[1] = 1$, $Apa[2] = 1$, $D[2] = 2$, $Apa[3] = 2$, $D[3] = 3$, $Apa[5] = 3$, $D[5] = 4$, $Apa[6] = 5$, $D[6] = 5$, $Apa[7] = 6$, $D[7] = 6$, $F[7] = 7$, $F[6] = 8$, $F[5] = 9$, $F[3] = 10$, $Apa[4] = 2$, $D[4] = 11$, $F[4] = 12$, $F[2] = 13$, $F[1] = 14$, $D[8] = 15$, $Apa[9] = 8$, $D[9] = 16$, $F[9] = 17$, $F[8] = 18$.

Végrehajtva a gráf transzponáltjára csökkenő F szerint a mélységi keresés algoritmusát kapjuk, hogy az erősen összefüggő komponensek:
$\{8,9\},\{1\},\{2,5,3,7,6\},\{4\}$.

12.2. Feladat Definiáljunk egy gráfot! Legyenek a pontjai a munkák. Egy pontból akkor vezessen él egy másik pontba, ha a megfelelő munkának meg kell előznie a másik pontnak megfelelő munkát. Végrehajtva a mélységi keresést A-ból kiindulva az elhagyási idők $F(A)=4$, $F(B)=6$, $F(C)=8$, $F(D)=16$, $F(E)=13$, $F(F)=15$, $F(G)=12$, $F(H)=3$. Tehát a munkák egy jó sorrendje D,F,E,G,C,B,A,H. Amennyiben a $H\to E$ relációnak is fenn kell állnia, akkor a mélységi keresésben találunk visszaélt, amiből következik, hogy a előfeltételekben van kör, azaz a munkasorozat nem hajtható végre.

12.3. Feladat A G gráfnak van Euler köre, minden pontnak megegyezik a ki és befoka. A körépítő algoritmus által kapott kör a következő séta $(1,2,3,1,3,5,6,5,3,4,6,2,4,1)$. A H gráf esetén a fokszámok $4,3,3,4,2,2$ és nem teljesül, hogy minden pont páros fokszámú. Tehát nincs Euler kör. Másrészt Euler út van, hiszen két pont van, ami páratlan fokszámú.

12.4. Feladat Definiáljuk a következő gráfot! A gráf pontjai az útkereszteződések. Egy A pontból akkor megy él egy B pontba, ha az AB út a B irányában egyirányú. Ekkor az a feltétel, hogy az útkereszteződések mindegyike elérhető legyen egy raktárból és utána visszamehessünk a raktárba pontosan azt jelenti, hogy minden erősen összefüggő komponensnek tartalmaznia kell legalább egy raktárt. Tehát meg kell határoznunk az így kapott gráf erősen összefüggő komponenseit, majd minden komponensben választanunk kell egy raktárt.

12.5. Feladat Elsőként igazoljuk, hogy mindig van a körmentes gráfokban forrás. Ellenkező esetben minden pontnak lenne legalább egy őse, de akkor bármelyik csúcsból kiindulva tudnánk képezni csúcsok egy sorozatát úgy, hogy a sorozat minden csúcspontjához vennénk az illető csúcs őseinek valamelyikét. Mivel minden csúcsnak van legalább egy őse, ezért a sorozat tagjainak képzése vég nélkül folytatható. Másrészt a csúcsok száma n, így legfeljebb $n-1$ lépés után a sorozatnak egy olyan tagját kapnánk amely már korábban előfordult a sorozatban. Nyilvánvalóan, a sorozat két azonos tagja közötti csúcsokhoz tartozó élek egy körutat alkotnak, ami ellentmondás. Következésképpen a gráf tartalmaz legalább egy forrást.

Most vegyük a gráf egy forrását, jelölje ezt v, és adjuk v-nek az 1 sorszámot. Majd töröljük v-t és a v-ból kiinduló éleket a gráfból, és az tekintsük az előálló részgráfot. Mivel ez a gráf is körmentes irányított gráf, ezért ismét tartalmaz forrást, ezek közül bármelyiknek adhatjuk a 2 sorszámot, majd ismét töröljük a forrást a belőle kivezető élekkel együtt, és a törlés után előálló részgráffal folytatjuk az eljárást. Miután minden lépésben eggyel csökken a csúcspontok száma, $n-1$ lépés után már csak egyetlen csúcsot tartalmazó gráfot, aminek adjuk az n sorszámot.

Mivel minden iterációs lépésben az aktuális részgráfból egy forrást hagyunk el, ezért az illető forrásból csak olyan csúcsokba visznek élek, amelyek később

kapnak sorszámot, így sorszámuk nagyobb lesz, mint a forrás sorszáma, amivel igazoltuk, hogy valóban egy topologikus rendezést kapunk.

12.6. Feladat Elsőként vegyük észre, hogy minden csúcsnak a befoka és a kifoka is 1. Ellenkező esetben a csúcsot nem lehet elérni vagy a csúcsból más pontokat nem lehet elérni, ami ellentmond annak, hogy a gráf egyetlen erősen összefüggő komponensből áll. Másrészt ha minden csúcs befoka és kifoka is egy, akkor a gráf vagy irányított kör vagy irányított körök uniója. Viszont a gráf erősen összefüggő, így csak egyetlen körből állhat.

13. Feszítőfák

13.1. Feladat A Kruskal algoritmus a következő sorrendben választja az éleket: $(a,b),(c,e),(d,f),(a,c),(d,e)$, A Prim algoritmus sorrendje $(a,b),(a,c)$, (c,e), $(d,e)(d,f)$.

13.2. Feladat Legyenek a gráf pontjai a hálózatban szereplő állomások. Egy él súlya legyen a két állomás közötti közvetlen vonal kiépítésének a költsége. Ekkor azon feltétel, hogy bármely pont elérhető legyen azt jelenti, hogy egy összefüggő részgráfot kell kiválasztani, amelyben minden pont szerepel. Mivel a súlyok nemnegatívak, ezért a legkisebb költséggel rendelkező gráf feszítőfa lesz, azaz a feladatunk pontosan egy minimális költségű feszítőfa meghatározása.

13.3. Feladat Elsőként igazoljuk, hogy minden minimális feszítőfa üvegnyakú is. Tegyük fel, hogy ez az állítás nem igaz. Ekkor van egy gráf és abban egy F minimális feszítőfa, amely nem üvegnyakú. Legyen F-ben a maximális súlyú él (a,b). Mivel a F nem üvegnyakú, ezért van olyan U üvegnyakú feszítőfa, amelyben minden él súlya kisebb, mint $c(a,b)$. Hagyjuk el F-ből (a,b)-t, ekkor két összefüggőségi komponensre esik szét. Mivel U feszítőfa, ezért van olyan éle, ami összeköti a két komponenst. Másrészt ennek az élnek kisebb a súlya, mint (a,b)-nek. Következésképp (a,b)-t kicserélhetjük egy nála kisebb súlyú éllel úgy, hogy ismét feszítőfát kapjunk, ami ellentmondás.

A másik irány nem igaz. Vegyünk $n>3$ ponton egy teljes gráfot (bármely két pont össze van kötve), amelyben az egyik él súlya 1 és minden más él súlya 2. Ekkor azok a feszítőfák, amelyek tartalmazzák az 1 súlyú élt minimálisak. Másrészt minden feszítőfa üvegnyakú.

13.4. Feladat Elsőként tegyük fel, hogy $x \leq 3$ és $y \leq 3$, akkor a Kruskal algoritmus által kapott minimális költségű feszítőfa élei $(a,b),(a,c),(a,d)$ és a költsége $2+x+y$. Ha $x>3$ és $y \leq 3$, akkor a Kruskal algoritmus által kapott minimális költségű feszítőfa élei $(a,c),(a,d),(b,c)$ és a költsége $5+y$. Ha $x \leq 3$ és $y>3$, akkor a Kruskal algoritmus által kapott minimális költségű feszítőfa élei $(a,b),(a,d),(b,c)$ és a költsége $5+x$. Végül ha $x>3$ és $y>3$, akkor a Kruskal algoritmus által kapott minimális költségű feszítőfa élei $(a,d),(b,c),(c,d)$ és a költsége 8.

13.5. Feladat Az állítás nem igaz minden esetben, ellenpéldaként felhozható az az eset, amikor a gráf egy fa, ekkor minden élét, így a maximális súlyú

élét is be kell választani a feszítőfába. Ez akkor is igaz, ha a maximális él végpontjai közül valamely csúcsnak a fokszáma egy.

Igazoljuk, hogy a megadott feltétel mellett egy szigorúan legnagyobb súlyú él nem kerülhet be a minimális költségű feszítőfába. Ezt indirekt igazoljuk. Tegyük fel, hogy van egy olyan minimális költségű feszítőfa, amely tartalmazza ezt a szigorúan legnagyobb súlyú e élt. Tekintsük ezt a feszítőfát! Ha elhagyjuk belőle az e élt a fa két részfájára esik szét. Másrészt az e él része egy körnek, így a kapott két részfa összeköthető e felhasználása nélkül is. Ebből adódik, hogy van olyan él, amely a két részfa között megy. Egyszerűen látható, hogy ebben az esetben egy ilyen éllel kicserélve e-t szintén feszítőfát kapunk. Másrészt e volt a szigorúan legnagyobb súlyú él, így a kapott feszítőfa költsége kisebb lesz, amivel ellentmondáshoz jutottunk.

13.6. Feladat A minimális feszítőfa költsége minimum a három legkisebb súly összege, ami 6, ez elő is állhat, ha a $w(A,B)=1$, $w(A,C)=2$, $w(A,D)=3$ súlyokat adjuk, akkor ezek az élek alkotják a minimális költségű feszítőfát. Szintén lehet a költség 7, például a $w(A,B)=1$, $w(A,C)=2$, $w(A,D)=4$, $w(B,C)=3$, $w(B,D)=5$, $w(C,D)=6$ esetben. Másrészt több eset nem lehetséges, ezt Kruskal algoritmusa alapján láthatjuk. Az algoritmus az első két lépésben kiválasztja az 1 és 2 súlyú éleket, a harmadik lépésben kipróbálja a 3 súlyú élet, ha nem alkot kört, akkor beveszi a költség 6, ha kört alkot továbblép a 4 költségű élre, amely már nem alkothat kört, így a feszítőfa költsége 7 lesz.

14. Legrövidebb utak

14.1. Feladat Az értékek G_1-re és G_2-re is a következőképpen változnak.
$Kesz=\{1\}, D(2)=2, Apa(2)=1, D(4)=1, Apa(4)=1, D(5)=8, Apa(5)=1$,
$Kesz=\{1,4\}, D(3)=5, Apa(3)=4, D(5)=3, Apa(5)=4, D(6)=8, Apa(6)=4$,
$Kesz=\{1,4,2\}, D(3)=3, Apa(3)=2, D(6)=6, Apa(6)=2$,
$Kesz=\{1,4,2,3\}$,
$Kesz=\{1,4,2,3,5\}D(6)=4, Apa(6)=5$,

G_1-ben a megoldás a legrövidebb utakat adja, G_2-ben nem, mivel ott van negatív érték is. Ekkor az $1,2,4$ út hossza $-2<1$.

14.2. Feladat Legyen Kozelit(u,v) eljárás, amit a Dijkstra algoritmus is használ

```
If D[v] > D[u] +c(u,v)
   Then D[v]:= D[u] +c(u,v)
        Apa[v]:=u
```

Tekintsük a következő eljárást (a csúcsokat az $1,\ldots,n$ egészeknek feleltettük meg).

```
for i:=1 to n
  For j in Be(G,i) Do
     Kozelit(j,i);
```

Azt állítjuk, hogy fenti eljárás végén a D függvény minden pontra a legrövidebb út hosszát adja meg. Ezt i szerinti teljes indukcióval igazoljuk. Mivel 1-be nem vezet él, ezért az állítás az 1 pontra nyilvánvalóan teljesül. Most legyen $i \leq 1 < n$ és tegyük fel, hogy az állítás az $1, \ldots, i$ pontokra teljesül. Igazoljuk $i+1$-re. Az indukciós feltevés alapján adódik, hogy a kiszámolt $D(i+1)$ érték a $\delta(1,j) + c(j,i+1)$ értékek minimuma. Legyen j a legrövidebb $i+1$-be vezető útban az $i+1$ előtti pont. Ekkor az út hossza nem lehet rövidebb, mint $\delta(1,j) + c(j,i+1)$, amely legalább $D(i+1)$. Másrészt a D értékek mindig felső korlátjaik a legrövidebb utaknak. Tehát igazoltuk az állítást $i+1$-re is.

A leghosszabb utat megkapjuk, ha minden élsúlyt megszorzunk -1-el és így határozzuk meg a legrövidebb utat. Másik lehetőség, hogy a Kozelit algoritmust írjuk át, akkor írjuk felül a D értéket, ha nagyobbat kapunk.

14.3. Feladat A legrövidebb utak számítása során.

$D(1) = 0$, $D(2) = -5$, $Apa(2) = 1$, $D(3) = -2$, $Apa(3) = -2$, $D(4) = -9$, $Apa(4) = 3$, $D(5) = -4$, $Apa(5) = 2$, $D(6) = -7$, $Apa(6) = 2$.

A leghosszabb utak.

$D(1) = 0$, $D(2) = -5$, $Apa(2) = 1$, $D(3) = -2$, $Apa(3) = -2$, $D(4) = -1$, $Apa(4) = 2$, $D(5) = 3$, $Apa(5) = 1$, $D(6) = 4$, $Apa(6) = 5$.

14.4. Feladat Ismét használjuk a 14.2. feladatban is szereplő Kozelit részeljárást. Tekintsük a következő eljárást (a csúcsokat az $1, \ldots, n$ egészeknek feleltettük meg).

```
for k:=1 to n-1
  For (i,j) in E Do
     Kozelit(i,j);
```

Igazoljuk, hogy az eljárás negatív súlyú kört nem tartalmazó gráfok esetén valóban helyes megoldást ad vissza. Tegyük fel, hogy j elérhető a kiindulási 1 pontból. Legyen a legrövidebb út $1 = v_0, v_1, \ldots, v_m = j$. Mivel a gráf nem tartalmaz negatív kört, ezért $m \leq n-1$, azaz legfeljebb $n-1$ él van a útban. Teljes indukcióval igazolható, hogy az i-edik iteráció után a $D(v_i) = \delta(1, v_i)$ és az Apa függvény az oda vezető legrövidebb út utolsó elemére mutat, amiből adódik, hogy az eljárás végén valóban megkapjuk a legrövidebb utakat. Ha egy pont nem elérhető, akkor adódik, hogy a visszaadott érték valóban ∞ lesz. A negatív köröket úgy ismerhetjük fel, hogy még egyszer n-edik alkalommal is végrehajtjuk a külső iterációt. Amennyiben nem változnak az értékek, akkor nem lehet negatív kör, hisz ebben az esetben akárhányszor végrehajtjuk az iterációt

továbbra sem változnak, ami ellentmond annak, hogy a negatív kör miatt tetszőlegesen kicsik lehetnek egyes távolságok. Amennyiben változnak, akkor kell negatív körnek lennie, hiszen a fentiekben beláttuk, hogy ha nincs negatív kör, akkor $n-1$ lépés után, minden D érték a legrövidebb út hosszát adja meg, és ez nem csökkenhet további iterációk során.

14.5. Feladat Az első iteráció után az éleket kiindulási pont sorszáma azon belül végpont sorszáma szerinti növekvő sorrendben vesszük:

$D(1) = 0$, $Apa(1) = 0$, $D(2) = 3$ $Apa(2) = 4$, $D(3) = 9$, $Apa(3) = 5$, $D(4) = 6$, $Apa(4) = 1$, $D(5) = 8$, $Apa(5) = 1$.

A második iteráció után:

$D(1) = 0$, $Apa(1) = 0$, $D(2) = 3$ $Apa(2) = 4$, $D(3) = 7$, $Apa(3) = 2$, $D(4) = 6$, $Apa(4) = 1$, $D(5) = 5$, $Apa(5) = 2$

A következő iteráció újra ezt az eredményt adja következésképp a többi is, így az eljárás véget ért. A legrövidebb utak és úthosszak: $s(1,4,2) = 3, s(1,4,2,3) = 7$, $s(1,4) = 6$, $s(1,5) = 8$.

14.6. Feladat Az eljárás végrehajtásában meghozandó döntések alapján a következő 3 esetet kell megkülönböztetnünk.

- Ha $A \geq 1$, akkor az első iteráció után $D(1) = 0$, $D(2) = 2$, $D(3) = 3$ és ezek az értékek a következő iterációban sem változnak. Következésképp az algoritmus megadja a legrövidebb utat, így a gráfban nincs negatív kör.
- Ha $-1 \leq A < 1$, akkor az első iteráció után $D(1) = 0$, $D(2) = 2$, $D(3) = 2 + A$ és ezek az értékek a következő iterációban sem változnak. Következésképp az algoritmus megadja a legrövidebb utat, így a gráfban nincs negatív kör.
- Ha $A < -1$, akkor az első iteráció után $D(1) = 0$, $D(2) = 3 + A$, $D(3) = 2 + A$. A második iteráció után $D(1) = 0$, $D(2) = 4 + 2A$, $D(3) = 3 + 2A$, harmadik iteráció után $D(1) = 0$, $D(2) = 5 + 3A$, $D(3) = 4 + 3A$. Következésképp az algoritmus nem találta meg $n-1$ iterációban a legrövidebb utakat, azaz van negatív kör.

14.7. Feladat A következő $U[i,j]$ mátrixokat kapjuk. Az első érték a távolság a második az Apa.

$$U_0 = \begin{pmatrix} (0,0) & (2,1) & (\infty,0) & (1,1) & (5,1) \\ (2,2) & (0,0) & (3,2) & (4,2) & (\infty,0) \\ (2,3) & (\infty,0) & (0,0) & (3,3) & (1,3) \\ (\infty,0) & (1,4) & (4,4) & (0,0) & (2,4) \\ (1,5) & (\infty,0) & (1,5) & (4,5) & (0,0) \end{pmatrix}$$

$$U_1 = \begin{pmatrix} (0,0) & (2,1) & (\infty,0) & (1,1) & (5,1) \\ (2,2) & (0,0) & (3,2) & (3,1) & (7,1) \\ (2,3) & (4,1) & (0,0) & (3,3) & (1,3) \\ (\infty,0) & (1,4) & (4,4) & (0,0) & (2,4) \\ (1,5) & (3,1) & (1,5) & (2,5) & (0,0) \end{pmatrix}$$

$$U_2 = \begin{pmatrix} (0,0) & (2,1) & (5,2) & (1,1) & (5,1) \\ (2,2) & (0,0) & (3,2) & (3,1) & (7,1) \\ (2,3) & (4,1) & (0,0) & (3,3) & (1,3) \\ (3,2) & (1,4) & (4,4) & (0,0) & (2,4) \\ (1,5) & (3,1) & (1,5) & (2,5) & (0,0) \end{pmatrix}$$

$$U_3 = \begin{pmatrix} (0,0) & (2,1) & (5,2) & (1,1) & (5,1) \\ (2,2) & (0,0) & (3,2) & (3,1) & (4,3) \\ (2,3) & (4,1) & (0,0) & (3,3) & (1,3) \\ (3,2) & (1,4) & (4,4) & (0,0) & (2,4) \\ (1,5) & (3,1) & (1,5) & (2,5) & (0,0) \end{pmatrix}$$

$$U_4 = \begin{pmatrix} (0,0) & (2,1) & (5,2) & (1,1) & (3,4) \\ (2,2) & (0,0) & (3,2) & (3,1) & (4,3) \\ (2,3) & (4,1) & (0,0) & (3,3) & (1,3) \\ (3,2) & (1,4) & (4,4) & (0,0) & (2,4) \\ (1,5) & (3,1) & (1,5) & (2,5) & (0,0) \end{pmatrix}$$

$$U_5 = \begin{pmatrix} (0,0) & (2,1) & (5,2) & (1,1) & (3,4) \\ (2,2) & (0,0) & (3,2) & (3,1) & (4,3) \\ (2,3) & (4,1) & (0,0) & (3,3) & (1,3) \\ (3,2) & (1,4) & (3,5) & (0,0) & (2,4) \\ (1,5) & (3,1) & (1,5) & (2,5) & (0,0) \end{pmatrix}$$

Az utolsó mátrixban szereplő párok első értékei adják meg a legrövidebb utak hosszait. A második értékek megadják, hogy egy legrövidebb útban melyik a megelőző pont. Így leolvasható az egész út. Például a legrövidebb $2,5$ út hossza 4. A megelőző pont 3. A legrövidebb $2,3$ út hossza 3, itt a megelőző pont 2, tehát a legrövidebb út $2,3,5$.

14.8. Feladat A felhasznált T_i mátrixok (T_0 az eredeti):

$$T_1 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_2 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_3 = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_4 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T_5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

14.9. Feladat A dobozokat feleltessük meg a gráf pontjainak. Az X pontból vezessen Y-ba él, ha Y belefér X-be. Ekkor feladatunk az, hogy ebben a gráfban keressük meg a leghosszabb utat.

14.10. Feladat Vegyünk egy gráfot, amelynek pontjai a vállalkozó és a köztisztviselők. A vállalkozóból él megy azokba a tisztviselőkbe, akiket közvetlenül megvesztegethet, továbbá minden tisztviselőből él megy azokba a tisztviselőkbe, akik elfogadják az ajánlólevelét. A gráfban a csúcsoknak van súlya, a vállalkozóé 0, a tisztviselőké a megvesztegetés összege. A cél a vállalkozóból a miniszterelnökbe egy legrövidebb út megkeresése, ahol az út hossza a benne levő csúcsok súlyainak összege. Ez megoldható a legrövidebb út algoritmusok módosításával is, de visszavezethetjük klasszikus legrövidebb út problémára a feladatot a következőképpen. Minden csúcsot helyettesítsünk egy éllel, amelynek súlya a csúcs súlya. Az él kezdőpontjába mennek azok az élek, amelyek a csúcsba mentek, az él végpontjából mennek azok az élek, amelyek a csúcsból mentek. (Szemléletesen ez azt jelenti, hogy az új csúcsok a megvesztegetések kezdetei és végei, a kezdet előfeltétele az ajánlólevél, és ajánlólevelet az ember csak a vesztegetés végén kap.)

14.11. Feladat A cél olyan út keresése, amelyre a p_e valószínűségek szorzata maximális. Mivel a logaritmus függvény monoton, ezért ez ekvivalens azzal a feladattal, hogy találjuk meg azt az utat, amelyben a szorzat logaritmusa, ami a $\log p_e$ értékek összege maximális. Ez a feladat ekvivalens azzal, hogy találjuk meg azt az utat, amire a benne szereplő élekre a $-\log p_e$ értékek összege minimális. Viszont valószínűségekről van szó, így $\log p_e \leq 0$, tehát $-\log p_e \geq 0$, azaz használhatjuk Dijkstra algoritmusát a fentiek szerint definiált ekvivalens feladatra.

15. Rendezések

15.1. Feladat A rendezés során a tömbök:
1, 5, 2, 6, 7, 8, 3, 9,
1, 2, 5, 6, 7, 8, 3, 9,
1, 2, 3, 6, 7, 8, 5, 9,
1, 2, 3, 5, 7, 8, 6, 9,

$1, 2, 3, 5, 6, 8, 7, 9,$
$1, 2, 3, 5, 6, 7, 8, 9,$
$1, 2, 3, 5, 6, 7, 8, 9.$

15.2. Feladat A rendezés során a tömbök: $3, 5, 6, 8, 9, 1, 3, 4,$
$3, 5, 6, 8, 9, 1, 3, 4,$
$3, 5, 6, 8, 9, 1, 3, 4,$
$3, 5, 6, 8, 9, 1, 3, 4,$
$1, 3, 5, 6, 8, 9, 3, 4,$
$1, 3, 3, 5, 6, 8, 9, 4,$
$1, 3, 3, 4, 5, 6, 8, 9.$

15.3. Feladat Az elemekből felépített kupac: $9, 5, 6, 3, 1, 3$. A 9 lesz az utolsó elem. A maradékban az utolsó elemet felvéve az első helyre majd kupaccá alakítva: $6, 5, 3, 3, 1, 9$. A 6-ot kicseréljük a kupac utolsó elemével, majd a maradék négy elemből álló halmazból kupacot csinálunk: $5, 3, 3, 1, 6, 9$. Majd a sorozatok: $3, 3, 1, 5, 6, 9,\ 3, 1, 3, 5, 6, 9,\ 1, 3, 3, 5, 6, 9$.

15.4. Feladat A gyorsrendezés során 4 a felosztó elem a tömb: $1, 3, 4, 5, 9, 6$. Utána rendezzük rekurzívan az $(1, 3)$ és $(5, 9, 6)$ tömböket. A felosztó elemek 3 és 6 ezt követően a maradék rekurzív hívások az (1), (5), (9) tömbökre vonatkoznak, amik már egy eleműek, így az eljárás véget ér.

15.5. Feladat Az a tömb, mely számolja az egyes elemek előfordulását: $2, 1, 3, 1, 1, 1$. Az S tömb, amely minden j-re megadja a j-nél nem nagyobb elemek számát: $2, 3, 6, 7, 8, 9$. Utána megkapjuk a rendezett halmazt, minden elemet arra a helyre írva, amekkora szám szerepel S-ben az elemnek megfelelő helyen, és csökkentve utána az S-beli értéket. Kezdetben az 5 bekerül az $S(5) =$ 8-adik helyre és $S(5) = 7$, majd a 3 bekerül a 6-odik helyre és $S(3) = 5$, és így tovább.

15.6. Feladat Radix rendezéssel a következő sorozatokat kapjuk:
I. 311, 452, 423, 654, 334, 145, 215, 236, 178.
II: 311, 215, 423, 334, 236, 145, 452, 654, 178
III: 145, 178, 215, 236, 311, 334, 423, 452, 654.

15.7. Feladat A rendezett vödrök $1 = \{0.12, 0.13\}$ $2 = \{0.23\}$, $3 = \{0.32, 0.33, 0.34\}$ $4 = \{0.44\}$, $5 = \{0.53\}$, $6 = \{0.65\}$, $8 = \{0.82\}$. A rendezett vödröket egymás mögé illesztve megkapjuk a rendezést.

15.8. Feladat A futási idő mindkét esetben $O(n \lg n)$. A bizonyítás alapötlete, hogy a rendezés során, mindig az utolsó elemet visszük fel, (legyen ez az i-edik szinten) és utána ezt az elemet süllyesztjük. Ezekben az esetekben süllyesztés során legalább az $i - 1$-edik szintig le kell süllyesztenünk az elemet. Ez azt jelenti, hogy amennyiben az aktuális kupac mélysége i, akkor a süllyesztés során legalább $i - 1$ cserét végrehajtunk. Tehát a cserék száma legalább $\sum_{j=1}^{n} \lceil \log_2 j \rceil - 1 \geq n/2 \log_2 n/2 - n = \Omega(n \log n)$. Másrészt a kupacrendezésre a legrosszabb eset $O(n \log n)$, így állításunk igazoltuk.

15.9. Feladat Azt kell biztosítani, hogy a felosztás során a két tömb mérete kiegyensúlyozott legyen. Azt a módszert használhatjuk, amelyet a lineáris idejű kiválasztási problémában. Az elemekből képezett elemötösök mediánjait határozzuk meg, és az ezen elemekből álló halmaz mediánja lesz a felosztó elem. Ezzel a felosztás továbbra is lineáris idejű marad, de a mindkét kapott résztömb mérete additív konstanstól eltekintve legalább $3/10n$. Ebből adódik, hogy a rekurzív hívások száma $O(\log n)$. Továbbá minden rekurziós szinten a teljes műveletigény lineáris, így az algoritmus időigénye valóban $O(n \log n)$ lesz.

16. Medián, külső rendezések

16.1. Feladat Az első pár után, $i = 1$, $j = 5$, a második párból 3 a kisebb és $3 \geq i$, $7 > j$, így $i = 1$, $j = 7$. A harmadik párból 8 a kisebb és $8 \geq i$, $9 > j$, így $i = 1$, $j = 9$. A negyedik párból 4 a kisebb és $4 \geq i$, $7 \leq j$, így marad $i = 1$, $j = 9$. Végül $3 \geq i$ és $3 \leq j$ így marad $i = 1$, $j = 9$.

16.2. Feladat Az első négy elemből kupacot építünk: $K = (7, 6, 4, 2)$. Ha a következő elem kisebb a gyökérnél kicseréljük és lesüllyesztjük. Mivel $3 < 7$, ezért ekkor $K = (6, 3, 4, 2)$. Utána mivel $5 < 6$ az új kupac $K = (5, 3, 4, 2)$. Végül $8 > 5$, így nem cseréljük ki a gyökérrel. A negyedik legkisebb elem az utolsó kupac gyökere.

16.3. Feladat Ismételt felosztással ugyanúgy mint a gyorsrendezésnél szétosztjuk egy felosztó elemmel a tömböt a nála nem nagyobb és a nála nagyobb elemekre. Az első felosztó elem a 8 a nála kisebb (H_b-beli) elemek száma 6, tehát az új feladat a $H_b = (4, 6, 7, 2, 3, 5)$ tömb ötödik legkisebb elemének megtalálása. A felosztó elem az 5, ekkor $H_b = (4, 2, 3)$ és $H_j = (7, 6)$. Mivel H_b elemszáma kisebb, mint $5-1$ ezért az új feladat az, hogy megtaláljuk a H_j tömb $5-3-1 = 1$-edik legkisebb elemét. A felosztó elem a 6, a H_b halmaz elemszáma $1 - 1$, így a 6 a keresett elem.

16.4. Feladat Kezdetben a futamokat A-ról átrakjuk C-re és D-re. Ekkor $C = 1, 3, 4 - 2, 6, 7, 8, 24 - 7, 8$ $D = 3, 5, 6 - 5, 9$. Utána ezeket összefésüljük, az A-ra és B-re. Ekkor $A = 1, 3, 3, 4, 5, 6 - 7, 8$ $B = 2, 5, 6, 7, 8, 9, 24$. Utána újra összefésülünk, $C = 1, 2, 3, 3, 4, 5, 5, 6, 6, 7, 8, 9, 24$ $D = 7, 8$. Végül
$A = 1, 2, 3, 3, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 24$

16.5. Feladat A következő rekurzív algoritmust használhatjuk. Vegyük mindkét tömbnek a középső elemét. Amennyiben n páros az X tömbnek az $n/2$-edik az Y tömbnek az $n/2 + 1$-edik elemét. Legyenek az elemek x és y. Ha $x \leq y$, akkor az X tömb első felét, az Y tömb második felét hagyjuk el, ha $x \geq y$, akkor az X tömb második felét, az Y tömb első felét hagyjuk el. Könnyen látható, hogy a maradék elemek mediánja, megegyezik az eredeti $2n$ elem mediánjával. Mivel az elemek száma minden lépésben feleződik, ezért az algoritmus legrosszabb esetben $O(\lg n)$ idejű.

16.6. Feladat a) Minden elemnek legyen a súlya $1/n$. Egyszerűen adódik, hogy ekkor a súlyozott medián a mediánnal egyezik meg.

b) Adjuk össze az elemeket a tömb első elemétől kezdve, amíg el nem érjük az 1/2-et, az az elem, amivel legalább 1/2 lesz az összeg, a súlyozott medián. Az $O(n \lg n)$ legrosszabb idejű algoritmus először rendezi az elemeket utána a fenti eljárást használja.