## Gyakorlatok

**Din 1** Jelölje $P(n)$ azt a számot, ahányféleképpen mehetünk le egy $n$ lépcsőfokból álló lépcsőn a következő mozgáselemek egy sorozatával (zárójelben, hogy mennyit mozgunk az adott elemmel): lépés (1), hosszú lépés (2), ugrás (2), hosszú ugrás (3). Adjunk egy dinamikus programozási algoritmust, ami kiszámolja $P(n)$-t.

*Megoldás:* A $P(n)$ értékekre a következő összefüggések állnak fenn:

- $P(1) = 1$ (ha egy lépcső van egyféleképpen mehetünk)
- $P(2) = 3$ (ha két lépcső van, $l + l$-ben, $u$-ban vagy $hl$-ben mehetünk)
- $P(3) = 6$ (ha három lépcső van, akkor $l + l + l$-ben $u + l$-ben $hl + l$-ben, $l + hl$-ben, $l + u$-ban vagy $hu$-ban mehetünk)
- $P(n) = P(n-1) + 2P(n-2) + P(n-3)$ ha $n \geq 4$, (utolsó lépésként $l, hl, u, hu$-t léphetünk).

Az összefüggések alapján egy tömböt kell kitöltenünk a lehetséges értékkel, ahol $T[i] = P[i]$.

```
Function P(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN] Of Longint;
     i: Word;
  Begin
     T[1]:=1;
     T[2]:=3;
     T[3]:=6;
     for i:=4 to N Do
        T[i]:=T[i-1]+2T[i-2]+T[i-3];
     P:=T[n];
  End;
```

**Din2** Adottak építőkockák, van 1 egység magas zöld, van 2 egység magas kék, van 3 egység magas piros, és van 3 egység magas sárga. (Mind a négy fajta kockából tetszőlegesen sok van). Adjunk meg egy dinamikus programozási algoritmust, amely kiszámolja, hogy hányféleképpen építhető fel egy $n$ magasságú torony.

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

- $F(1) = 1$ (egy egység magas kocka)
- $F(2) = 2$ (két db egy egység magas, vagy egy db két egység magas)
- $F(3) = 5$ (egyszerű esetszétválasztás)
- $F(n) = F(n-1) + F(n-2) + 2F(n-3)$ ha $n \geq 4$, (legfelül a négy féle kocka valamelyike lehet).

Az előző feladathoz teljesen hasonlóan implementálható a dinamikus proramozási megoldás.

```
Function F(n:Word) :Longint;
  Const
     MaxN=5000;
  Var
     T: Array[1..MaxN] Of Longint;
     i: Word;
  Begin
     T[1]:=1;
     T[2]:=3;
     T[3]:=5;
     for i:=4 to N Do
        T[i]:=T[i-1]+T[i-2]+2T[i-3];
     F:=T[n];
  End;
```

**Din 3** 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 dinamikus programozási eljárást az $R(n,k)$ érték kiszámítására! Igazoljuk, hogy $R(n,k) = \binom{n+k-2}{k-1}$.

*Megoldás* Az $R(n,k)$ 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 $\binom{n+k-2}{k-1}$.

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 rekurzív hívások alapján a következő dinamikus programozási eljárásokat kaphatjuk meg. Az elsőben egy 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;
```

**Din 4** 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.

*Megoldás* 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

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

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.

**Din 5** 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?

*Megold:* 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 plussz a főnök lesz a módósított 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.

**Végrehajt 1** Egy $G$ irányított gráfnak a pontjai $V = \{1, \ldots, 6, 7\}$, az élei $E = \{(1,2), (1,4), (1,6), (2,2), (2,3), (2,4), (3,1), (3,5), (4,1), (4,4), (5,2), (5,3)$ $(5,6), (6,1), (6,2), (6,5), (7,1)\}$. Adjuk meg milyen sorrendben járja be a szélességi keresés a pontokat az 5 pontból kindulva. A bejárás alapján adjuk meg a $\delta(5,1)$ és $\delta(5,7)$ értékeket!

*Megoldás:* Tegyük fel, hogy a ki iteráció növekvő index szerint veszi a pontokat. Kezdetben az algoritmusban $D(5) = 0$, $Apa(5) = Null$. $Sor = \langle 5 \rangle$. Utána

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

- Kijön 1 és $Sor = \langle\rangle$.

Következésképpen $\delta(5,1) = 2$ és a legrövidebb út $5 = Apa(3), 3 = Apa(1), 1$. Továbbá $\delta(5,7) = \infty$ és nincs köztük út.

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

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

Végrehajtva a gráf transzponáltjára csökkenő $F$ szerint a mélységi keresés algoritmusát a következő sorozatot kapjuk.

$D[1] = 1$, $Apa[8] = 1$, $D[8] = 2$, $Apa[9] = 8$, $D[9] = 3$, $Apa[3] = 9$, $D[3] = 4$, $Apa[2] = 3$, $D[2] = 5$, $F[2] = 6$, $F[3] = 7$, $F[9] = 8$, $F[8] = 9$, $F[1] = 10$

Egy komponens $1, 8, 9, 3, 2$.

Folytatva a mélységi keresést: $D[4] = 11$, $F[4] = 12$, így egy további komponens a 4 pont.

Folytatva a keresést: $D[5] = 13$, $Apa[7] = 5$, $D[7] = 14$, $Apa[6] = 7$, $D[6] = 15$, $F[6] = 16$, $F[6] = 17$, $F[8] = 18$ és megkaptuk, hogy az utolsó komponens $5, 6, 7$.

**GráfVégrehajt 3:** Hajtsuk végre az 1 pontból a Dijkstra algoritmust az alábbi $G_1$ gráfra. (A mátrixokban a $c_{ij}$ érték az $(i,j)$ él hossza, $\infty$ ha nincs él.)

$$G_1 = \begin{pmatrix} \infty & 1 & \infty & 6 & 4 & \infty \\ 2 & \infty & 6 & 4 & 1 & 5 \\ 2 & \infty & 2 & 1 & 1 & 7 \\ \infty & 1 & 4 & \infty & 2 & 7 \\ 1 & \infty & 1 & 4 & \infty & 1 \\ \infty & 1 & \infty & 4 & 2 & 1 \end{pmatrix}$$

Az értékek a következőképpen változnak.
$Kesz = \{1\}, D(2) = 1, Apa(2) = 1, D(4) = 6, Apa(4) = 1, D(5) = 4, Apa(5) = 1,$
$Kesz = \{1,2\}, D(3) = 7, Apa(3) = 2, D(5) = 2, Apa(5) = 2, D(6) = 6, Apa(6) = 2,$
$Kesz = \{1,2,5\}, D(3) = 3, Apa(3) = 5, D(6) = 3, Apa(6) = 5,$
$Kesz = \{1,2,5,3\}, D(4) = 4, Apa(4) = 3,$
$Kesz = \{1,2,5,3,6\},$
$Kesz = \{1,2,5,3,6,4\}$

**Gráf1** Adott egy sziget ahol fejlettebb kaméleonok élnek, mint azon, amit az előadáson néztünk. A kaméleonok négy színt vehetnek fel piros, kék és 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!

*Megoldás:* 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.

**Gráf2:** Adott egy 3-szor 3-as tábla, amelynek a mezőin az 1,2,...,9 számok állnak. Minden mezőnek van egy színe, ami piros vagy fekete. Ha megnyomjuk a mezőt, akkor a mezőnek és a vele oldalban szomszédos mezőknek a színe ellenkező színűre változik. Határozzuk meg, hogyan lehet a lehető legkevesebb gombnyomással egyszínűvé tenni a táblát. Reprezentáljuk a feladatot egy gráffal, és adjuk meg, hogy melyik gráfalgoritmussal oldható meg!

*Megoldás:* Vegyünk egy gráfot, amelynek csúcsai a 9 elemből álló bitvektorok, azaz a $[0, 1]^9$ halmaz elemei. Ha az $i$-edik komponens 1 az azt jelenti, hogy az $i$-edik mező piros, ha a komponense a vektornak 0 az azt jelenti, hogy az $i$-edik mező fekete. Tehát a vektorok a színezéseknek felelnek meg. Egy csúcsból akkor megy él egy másikba, ha a vektoroknak megfelelő színezések valamely mező megnyomásával átmennek egymásba. Az így kapot gráfban a kindulási színsorozat vektorából kell egy szélességi kereséssel megkeresni a legrövidebb utat a $(0, 0, \ldots, 0)$ és az $(1, 1, \ldots, 1)$ vektorok valamelyikébe.

**Gráf3** 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!

*Megoldás:* 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.

**Gráf 4:** Adott egy irányítatlan körmentes gráf. határozzuk meg a középpontját, azaz egy olyan pontot, amelyre igaz, hogy a többi pont tőle vett távolságának maximuma minimális.

*Megoldás* Az irányítatlan körmentes gráfok, fák amelyekben vannak olyan pontok (levelek), amelyeknek a fokszáma egy. Vegyük észre, hogy ha egy fából töröljük az összes levelet, akkor a maradék fának és az eredeti fának ugyanazok lesznek a középpontjai. Így a feladat megoldható egy olyan algoritmussal, amely minden lépésben törli az aktuális fa összes levelét, addig amí egyetlen egy vagy két pont nem marad.

**Gráf 5** Egy faluban mindenkire ismert azoknak a halmaza, akiknek továbbmondja az általa megismert pletykát. Határozzunk meg egy minimális elemszámú halmazát az embereknek, amelyre teljesül, hogy minden pletykát megtud valamelyikük.

*Megoldás* Definiáljunk egy irányított gráfot, a pontok az emberek a-ból megy él b-be, ha a továbbmondja a pletykát. A gráfban a feladat minimális számú ponthalmaz keresése, hogy minden pontból vezessen út a ponthalmaz valamely pontjába. A megoldást a következő (elvi) algoritmus szolgáltatja:

- határozzuk meg az erősen összefüggő komponensek komponensgráfját
- vegyük azokat a komponenseket, amelyekbol a komponensgráfban nem vezet ki él
- minden ilyen komponensből válasszunk ki egy pontot.

**Gráf 6** Az előző faluban meg szeretnénk rágalmazni valakit egy pletykával. Ehhez határozzunk meg egy minimális elemszámú halmazát az embereknek, amelyre teljesül, hogy a nekik elmondott pletykát a falu összes embere megtudja.

*Megoldás:* A megoldás nagyon hasonló az előző feladathoz, tulajdonképpen annak komplementeréről van szó:

- határozzuk meg az erősen összefüggő komponensek komponensgráfját
- vegyük azokat a komponenseket, amelyekbe a komponensgráfban nem vezet be él
- minden ilyen komponensből válasszunk ki egy pontot.

**Komb1:** Legyen $S$ az $1, \ldots, n+1$ számokból összeállítható olyan lehetséges szám $n$-esek halmaza, amelyek egyetlen számot sem tartalmaznak kétszer. Vegyük a lexikografikus rendezést ezen a halmazon. A feladat egy adott szám-n-esre a rákövetkező szám $n$-es meghatározása.

*Megoldás* Az előadáson szereplő betűsorok esetéhez hasonlóan meg kell találnunk az utolsó számot a szóban, amelyet kicserélhetünk nagyobbra. Ebben az esetben ez az utolsó olyan szám lesz, amelyre

- vagy mögötte szerepel nagyobb érték
- vagy a szám $n$-esből kimaradó érték nagyobb (egy ilyen kimaradó szám van).

Miután megtaláltuk ezt az elemet, helyére a nála nagyobb elemet rakjuk és utána a maradék elemekből képezhető (itt is kimarad egy elem) lexikografikusan legkisebb megfelelő hosszú számsort rakjuk. Ehhez nem kell rendezni az elemeket, akik a szám után jönnek monoton csökkenőek, ebbe kell beszúrni a kimaradó elemet.

*Példák* $n = 5$.

(65213) Itt az utolsó elem, aki mögött van nagyobb az 1, de a 3-nál is nagyobb a kimaradó 4, így a 3 elemnél növelünk, a rákövetkező (65214)

(65234) Itt az utolsó elem, aki mögött van nagyobb az 2, és a többieknél sem nagyobb a kimaradó 1, így a 2 elemnél növelünk, a rákövetkező (65312), mivel a kimaradó elemet is figyelembe kell vennünk.

**Komb2:** Az előadáson ismertetett ládapakolási feladat (három típusú láda, amely típusok egymásba rakhatók), azon változatára, ahol mindkét irányba mozgathatók a ládák, mi a minimális számú láda, mi elérhető?

*Megoldás* A feladat ebben az esetben sokkal egyszerűbb, mint az előadáson vizsgált példa. Ekkor nem kell visszavezetni rövidebb sorozatokra az inputot. Egyszerűen adódik, hogy a megoldás az egyes típusokba tartozó ládaszámok maximuma.

**Komb3:** Adott négy n elemből álló, egészeket tartalmazó halmaz A,B, C,D. Határozzuk meg $O(n^2)$ időben azon $(a, b, c, d) \in A \times B \times C \times D$ négyesek számát, amelyekre $a + b + c + d = 0$.

*Megoldás* Ha az összes négyesre kiszámoljuk az összeget az $O(n^4)$ időben fut. Ezzel szemben számoljuk ki elsőként a $A \times B$ beli elemekre az összegeket, egy tömbben minden lehetséges értékre írjuk be hányféleképpen jöhet ki.

Utána számoljuk ki a $C \times D$ beli összegeket is, és ha egy ilyen összegnek a negatívja k féleképpen kijött A és B beli számok összegeként, akkor $k$-val növeljük a lehetséges előállítások számát.

**Mohó1** Adott az egyenesen pontoknak egy $x_1 \leq x_2 \leq \ldots x_n$ egy halmaza. A feladat az, hogy fedjük le a pontokat körökkel úgy, hogy a $\sum(1 + d_i)$ érték minimális legyen, ahol $d_i$ az $i$-edik kör átmérője. Adjunk egy lineáris idejű algoritmust, amely megad egy optimális lefedést.

*Megoldás:* Egyszerűen látható, hogy az lesz az optimális megoldás, amelyben az $x_i$ és $x_{i+1}$ pontok akkor kerülnek ugyanabba a körbe ha távolságuk legfeljebb 1. Ez azért igaz, mert ha két ilyen pont külön körbe kerül, akkor a két kört összevonva a körszám csökkenése miatt a célfüggvény 1-el csökken, viszont az átmérő növekedése miatt 1-nél kevesebbel nő. Ez alapján az észrevétel alapján egyből adódik a lineáris idejű algoritmus.

**Mohó2** Adott zárt intervallumoknak egy $\{[a_1, b_1], \ldots, [a_n, b_n]\}$ halmaza (ezek azok az időintervallumok, amikor az egyes vendégek ott vannak a fogadáson). Keressünk olyan minimális számú ponthalmazt (fényképezések időpontjai), amelynek minden intervallumba esik legalább egy pontja (mindenki rajta van legalább egy fényképen).

*Megoldás* 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.

**Rendezés** Adjunk egy algoritmust, amely egy n-elemű halmaznak meghatározza a legkisebb és második legkisebb elemét is $n + \log n$ összehasonlítással.

*Megoldás:* Az alapötlet, hogy a minimális elemet egy kieséses versenynek megfelelően válasszuk ki. Párokat képezünk, mindegyikből a kisebbet tartjuk meg, és ezekből ugyanígy párokat képezünk mindaddig, amíg egyetlen egy elem, a minimális nem marad. Másrészt a második legkisebb elem csak a legkisebb elem ellen veszíthetett, így elegendő azon elemek közül kiválasztanunk a legkisebbet. Viszont ezek annyian vannak, mint egy n elemű majdnem teljes bináris fa magassága, ami $\lceil \log n \rceil$.