A Verem absztrakt adattípus

Értékhalmaz: E $Verem = [a_1,\ldots,a_n : a_i \in E, i = 1,\ldots,n,]$ Műveletek: V : Verem, x : E

- $\{Igaz\}$ Letesit(V) $\{V = []\}$
- $\{V = V\}$ Megszuntet(V) $\{Igaz\}$
- $\{V = V\}$ Uresit(V) $\{V = []\}$
- $\{V = [a_1,\ldots,a_n]\}$ VeremBe(V,x) $\{V = [a_1,\ldots,a_n,x]\}$
- $\{V = [a_1,\ldots,a_n], n > 0\}$ VeremBol(V,x) $\{V = [a_1,\ldots,a_{n-1}], x = a_n\}$
- $\{V = V\}$ NemUres(V) $\{NemUres = (Pre(V) \neq [])\}$
- $\{V = [a_1,\ldots,a_n], n > 0\}$ Teteje(V,x) $\{x = a_n, V = Pre(V)\}$
- $\{V = [a_1,\ldots,a_n], n > 0\}$ Torol(V) $\{V = [a_1,\ldots,a_{n-1}]\}$

Megvalósítás: tömb, lánc, kombinált.

Megvalósítás tömbbel

A tömb az első elemétől kezdve tartalmazza a veremben szereplő értékeket, adott egy teto változó, amely megadja a verem tetején levő tömbelem indexét, továbbá egy *meret* érték, ami megadja a tömb méretét. A verem műveletek hatékonyan megvalósíthatóak.

Hátránya: előre rögzített méretű memóriát kell foglalni. Tegyük fel, hogy a vermet egy T tömbbel valósítjuk meg. Ekkor

```
Verembol(V,x)
  If teto=0
     then {print "Ures verem"
           Return False}
     Else
         {x:=T[teto]
         teto:=teto-1
         return True}


Verembe(V,x)
  If teto=meret
     Then {Print "Tele a verem"
           Return T
     Else
         {teto:=teto+1
         T[teto]:=x
         return T}
```

Ha nem akarjuk korlátozni a méret változóval a tömb méretét, akkor lehet dinamikusan növelni ezt az értéket.

```
Verembe(V,x)
  If teto=meret
      Then {Letesit(T2:Tomb)
           for i=1 to meret
               T2[i]:=T[i]
           teto:=teto+1
           T2[teto]:=x
           return T2}
      Else
           {teto:=teto+1
           T[teto]:=x
           return T}
```

Megvalósítás lánccal

A láncban az elemek egy elem.csat mutatót tartalmaznak a következő elemre és elem.adat tartalmazza a verembe rakott adatot. A láncot a legelső cellára mutató fej értékkel adjuk meg. A lánccal történő megvalósítás esetén nem kell előre lefoglalni az egész veremnek a memóriát.

```
Verembe(V,x)
   Letesit(E: lancelem)
   E.adat:=x
   E.csat:=fej
   fej:=E


Verembol(V,x)
   If fej=Nil
       Then Print "Ures Verem"
       Else
         {x:=fej.adat
         fej:=fej.csat}
```

Megvalósítás Kombinált adatszerkezettel

A kombinált adatszerkezetben az elemek tárolására adott méretű tömböket használunk, és ezen tömböket egy láncba fűzve tároljuk el.

Sor

Értékhalmaz: E $Sor = [a_1, \ldots, a_n : a_i \in E, i = 1, \ldots, n,]$ Műveletek: S: Sor, x : E

- $\{Igaz\}$ Letesit(S) $\{S = []\}$
- $\{S = S\}$ Megszuntet(S) $\{Igaz\}$
- $\{S = S\}$ Uresit(S) $\{S = []\}$
- $\{S = [a_1, \ldots, a_n]$ SorBa(S,x) $\{S = [a_1, \ldots, a_n, x]\}$
- $\{S = [a_1, \ldots, a_n], n > 0\}$ SorBol(S,x) $\{S = [a_2, \ldots, a_n], x = a_1\}$
- $\{S = [a_1, \ldots, a_n]\}$ Elemszam(S) $\{Elemszam = n\}$

- $\{S = [a_1, \ldots, a_n], n > 0\}$ Elso(S,x) $\{x = a_1, S = Pre(S)\}$
- $\{S = [a_1, \ldots, a_n], n > 0\}$ Torol(S) $\{S = [a_2, \ldots, a_n]\}$

Megvalósítás: tömb, lánc kombinált.

Megvalósítás cirkuláris tömbbel

Az elemeket egy *meret* méretű tömbben tároljuk, a sor első elemének és utolsó elemének indexét megjegyezve. A létrehozott sor esetén az eleje érték 0, a vége érték 0. Az elemek a két érték között helyezkednek el (a tömb vége a tömb elejénél folytatódik).

```
Elemszam(S)
  if (elso<=utolso)
     then return utolso-elso;
   else
     return meret-(elso-utolso)

SorBa(S,x)
   if (Elemszam(S)=meret)
      Then Print "Megtelt"
      Else {utolso:=utolso+1
           If utolso>meret
              Then utolso:=utolso-meret
           T[utolso]:=x}
```

Megvalósítás lánccal

A láncban az elemek egy elem.csat mutatót tartalmaznak a következő elemre és elem.adat tartalmazza a verembe rakott adatot. A láncot a legelső cellára mutató fej értékkel és az utolsó cellára mutató vege értékkel adjuk meg. Továbbá számon kell tartani egy változóban az elmszám értéket. A lánccal történő megvalósítás esetén nem kell előre lefoglalni az egész veremnek a memóriát.

```
Sorba(S,x)
   Letesit(E: lancelem)
   E.adat:=x
   E.csat:=Nil
   vege.csat:=E
   vege:=E
   eszam:=eszam+1
```

Megvalósítás Kombinált adatszerkezettel

A kombinált adatszerkezetben az elemek tárolására adott méretű tömböket használunk, és ezen tömböket egy láncba fűzve tároljuk el. Az első tömbhöz tároljuk a sor első elemének indexét, az utolsóhoz a tömb utolsó elemének indexét.

Sorozat adattípus

Értékhalmaz: E $Sor = [a_1, \ldots, a_n : a_i \in E, i = 1, \ldots, n,]$ Műveletek: S: Sor, x : E

- $\{Igaz\}$ Letesit(S) $\{S = []\}$
- $\{S = S\}$ Megszuntet(S) $\{Igaz\}$

- $\{S = S\}$ Uresit(S) $\{S = []\}$
- $\{S = [a_1, \ldots, a_n]\}$ Elemszam(S) $\{Elemszam = n\}$
- $\{S = [a_1, \ldots, a_n] \wedge 1 \leq i \leq n\}$ Kiolvas(S, i,x) $\{x = a_i \wedge S = Pre(S)\}$
- $\{S = [a_1, \ldots, a_i, \ldots, a_n] \wedge 1 \leq i \leq n\}$ Modosit(S, i,x) $\{S = [a_1, \ldots, x, \ldots, a_n]\}$
- $\{S = [a_1, \ldots, a_i, a_{i+1}, \ldots, a_n] \wedge 0 \leq i \leq n\}$ Bovit(S, i,x) $\{S = [a_1, \ldots, a_i, x, a_{i+1}, \ldots, a_n]\}$
- $\{S = [a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n] \wedge 1 \leq i \leq n\}$ Torol(S, i) $\{S = [a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n]\}$
- $\{S = S\}$ IterKezd(S, I) $\{\}$
- $\{I = I\}$ IterAd(I,x) $\{\}$
- $\{I = I\}$ IterVege(I) $\{\}$

Megvalósítás: kiegyensúlyozott bináris keresőfával.

Tároljuk az $S = \{a_1, \ldots, a_n\}$ sorozatot egy F bináris fában úgy, hogy F inorder bejárása az S sorozatot adja. Ekkor F egy bináris keresőfa lesz a sorozat indexeire nézve.

A fa minden p pontjában tároljuk még extra információként a bal-részfájában levő pontok számánál 1-el nagyobb értéket $s(p)$. Ekkor $s(p)$ a p pontra végrehajtott inorder bejárás során p sorszáma. Ezen $s(p)$ értékek alapján a fa gyökeréből indulva megtalálható minden i-re az i-edik elem.

Bővítés esetén ha a keresőút balra halad egy p ponttól, akkor $s(p)$ értékéhez egyet kell adni. Törlés esetén, ha a keresőút balra halad egy p ponttól, akkor $s(p)$ értékéből egyet le kell vonni.

Interaktív medián keresése

Adott n tárgy, melyeket 1-től n-ig számozunk, ahol n páratlan. Minden tárgy különböző súlyú (természetes számok), de magukat a súlyokat nem ismerjük. Minden y súlyra igaz, hogy $1 \leq y \leq n$. Mediánnak nevezzük azt a tárgyat, amelyiknél ugyanannyi könnyebb, mint nehezebb tárgy van. Írjunk programot, amely meghatározza a mediánt! A tárgyak súlyát egy olyan eszközzel hasonlíthatjuk össze, amely három tárgy közül megadja a mediánt.

Hagymahámozó algoritmus: Határozzuk meg a legnagyobb és legkisebb elemeket, hagyjuk el őket a maradék halmaz mediánja ugyanaz, mint az eredeti halmazé.

Egy elem akkor és csak akkor szélső elem, ha nem mediánja egyetlen hármasnak sem. Így minden lekérdezéssel ki tudunk zárni egy elemet, tehát $n-2$ kérdéssel meghatározhatjuk a szélső elemeket. Következésképpen a halmaz mediánja $(n-2)+(n-4)+(n-6)+\cdots+1 = O(n^2)$ kérdéssel meghatározható a hagymahámozó algoritmussal.

Számjáték: Egy játéktábla pozitív egész számok sorozata, ami 2n számot tartalmaz. Két játékos felváltva lép, egy lépésnem a játékos a számsorozat bal vagy jobb végéről elvesz egy számot. A játék akkor ér véget, ha a számok elfogytak. Az a játékos nyer, akinél az elvett számok összege több. Adjunk egy eljárást, amivel az első játékos legalább döntetlen elér. A második játékos lépéseit egy már adott számítógépes program szolgáltatja.

Megoldás Az első játékos kényszerítheti a másodikat, hogy csak a páratlan vagy csak a páros indexű számokat vegye el, így könnyen elérheti, hoyg ne veszítsen.

Maxszámjáték A játék csak a céljaiban különbözik az előző játéktól. Most a játékos célja a lehető legnagyobb érték összegyűjtése.

Az optimális stratégia dinamikus programozással meghatározható. Legyen minden $i \leq j$-re $OPT(i, j)$ az $a_i, \ldots, a_j$ számsorozatból az első játékos által összegyűjthető összeg.

- Ha $i = j$, akkor ez az érték 0 mivel az utolsó elemet a második játékos veszi el.

- Ha $i+j$ páratlan, akkor az első játékos jön, és a két szélső elem közül a neki jobbat választja, tehát ekkor $OPT(i,j) = \max\{a_i + OPT(i+1,j), a_j + OPT(i,j-1)\}$.
- Ha $i+j$ páros, akkor a második játékos jön, ő azt választja, ami az elsőnek rosszabb, azaz ekkor az érték $OPT(i,j) = \min\{OPT(i+1,j), OPT(i,j-1)\}$.

Ezen összefüggésekkel dinamikus programozással minden állásra kiszámítható a maximálisan nyerhető összeg és az ehhez szükséges lépés.

Adatszerkezet meghatározása Adott egy adatszerkezet, amelyről nem tudjuk, hogy lista vagy nyalóka (egy lista végére illesztve egy körlánc). Van két bélyeg. Az adatszerkezettel a következő lépéseket tehetjük.

- egy bélyeget a helyéről felvehetünk
-egy bélyeget a helyéről a rákövetkező helyre továbbmozdíthatunk
-egy bélyeg helyére mellé a másikat is odatehetjük
-a legelső elemre bélyeget tehetünk.

Határozzuk meg a méretben lineáris időben, melyik adatszerkezetről van szó!

Megoldás A nyalóka adatszerkezetet úgy ismerhetjük fel, ha találunk kört. Kört pedig úgy találhatunk, hogy egy bélyeg helyétől elindulva a másik bélyeggel mindig továbblépve újra megtaláljuk a bélyeget. Két módszer lehetséges:

Próbálgatós Minden k-ra megnézzük, hogy van -e a kiindulási ponttól 2^k lépésre levő pontot tartalmazó legfeljebb 2^k hosszú kör. Ezt úgy tehetjük meg, hogy a 2^k lépésnyire levő bélyeg helyéről elindulunk a másik bélyeggel, teszünk 2^k lépést. Ha újra látjuk az első bélyeget van kör, ha nem, akkor áthelyezzük az első bélyeget is a második mellé, és rátérünk a 2^{k+1} hosszú kör ellenőrzésére.

Ötletes Elinditjuk a két bélyegzőt, az egyiket kétszer olyan gyorsan. (Időegységenként 2-t lép, mig a lassabb csak egyet). Ha nyalóka adatszerkezet van, lineáris időben lekörözi a gyors a lassút, és megtaláljuk, hogy van kör, egyébként pedig nyilvánvalóan lineáris időben eljutunk a lista végére.

Tortafelosztás

A feladat n felhasználó között felosztani igazságosan az erőforrásokat. A probléma az, hogy minden résztvevőnek saját mérőszáma van.

Adott a torta (általában a [0,1] intervallum). A résztvevőket felkérhetjük

- valamely rész adott arányban történő felosztására
- két rész összehasonlítására, hogy szerintük melyik a nagyobb.

Igazságos felosztás: mindenki úgy gondolja, hogy megkapta a torta 1/n-ed részét (a saját mércéje szerint).
Irigységmentes felosztás: mindenki úgy gondolja, hogy senki sem kapott nála többet (a saját mércéje szerint).

Mozgókéses eljárás

- Mozgatunk egy kést, és mindenkinek akkor kell szólni, amikor szerinte elérte a torta 1/n-ed részét.
- Mindenkit igazmondásra ösztönöz és egy igazságos felosztást eredményez.
- A probléma, hogy végtelen sok kiértékelést kell a résztvevőknek végrehajtani.

Fink eljárása

- Két ember, egyik felez másik választ.
- Ha 3 ember van, akkor A felez, B kiválasztja a nagyobbikat. Majd A és B is felharmadolja a saját szeletét. C mindkét részből választhat egyet.

- Általánosabban elsőként az első n-1 résztvevő rekurzívan igazságosan elosztja a tortát. Majd minden résztvevőt megkérünk, hogy vágja n részre a saját tortáját. Az n-edik személy mindenkitől választ egy részt.

Banach és Knaster eljárása

- Az első ember levág egy 1/n nagyságú darabot majd továbbadja 2-nek.
- 2 vagy továbbadja 3-nak vagy ha nagyobbnak találja, mint 1/n, akkor levág belőle és úgy adja tovább. 3 is igazíthat rajta, majd továbbadja, és így tovább egészen addig, amíg el nem jut az n-edik személyig.
- Ha n elfogadja elviszi, egyébként annak kell elvinni, aki utoljára belevágott. Utána az n-1 személy újra végrehajtja az eljárást.
- Mindenkit igazmondásra ösztönöz. Nagyobbat nem hagyhat, mert esetleg elvisznek egy 1/n-nél nagyobb szeletet. Kisebbet sem, mert esetleg neki kell elvinni.
- Legfeljebb n(n-1)/2 vágás kell.

Even és Paz eljárása(4 ember)

- Kérjük fel ez első 3 embert, hogy felezzék meg a tortát (feltesszük, hogy a torta a 0,1 intervallum).
- Középső vágó, aki a másik kettő között jelölte be a felét.
- A negyedik ember kiválasztja a jobbik részt a középső vágó által adott két fél közül, és azt osztja tovább azzal az emberrel, akinek a felezőpontja oda esett.

Vizsgaleírás

A vizsgán elérhető maximális pontszám: 100. A vizsgán 10 kérdés lesz a honlapon található kérdéssorból. Lesz olyan kérdés, ami algoritmus végrehajtásáról szól, és egy kérdés lesz, amiben bizonyítás van (pl. futási idő vagy helyesség igazolása).