Backtrack-es feladatok

1. feladat:

Egy éhes egérnek egy labirintusban elhelyeznek egy darab sajtot. Írjunk programot, amely segít az egérnek megkeresni a sajthoz vezető utat!

2. feladat:

Egy éhes egérnek egy labirintusban elhelyeznek egy darab sajtot. Írjunk programot, amely segít az egérnek megkeresni a sajthoz vezető legrövidebb utat!

3. feladat:

*Helyezzünk el N db vezért az N*x*N-es sakktáblán úgy, hogy ne üssék egymást! Írjunk programot, amely az összes elhelyezést kiírja a képernyőre!*

4. feladat:

*Helyezzünk el N db huszárt az N*x*N-es sakktáblán úgy, hogy ne üssék egymást, továbbá egy sorban, egy oszlopban és a (fő)átlóban is csak egy huszár lehet! Írjunk programot, amely az összes elhelyezést kiírja a képernyőre!*

5. feladat:

A sakktábla egy adott mezejéről indítva keressünk a huszár számára olyan utat, amely során a huszár minden mezőt pontosan egyszer érint!

6. feladat:

Tegyük le az összes dominót úgy, hogy csak az egyik irányba tehetünk, és a dominókon az összes lehetséges párosítás ((0:0),(0:1),...,(0:9),(1:1),...,(9:9)*) előfordul!*

7. feladat:

Adottak az F(1),..., F(M) elvégzendő feladatok és az ezek elvégzésével megbízható V(1),..., V(N) vállalatok, amelyek egyszerre legfeljebb D(1),..., D(N) feladat elvégzésére képesek. Hogy egyáltalán melyeket, azt a E(1), ... , E(N) felsorolások sorozata írja le. Válasszunk ki minimális számú vállalatot úgy, hogy ezek együttvéve, egyidőben valamennyi feladatot el tudják végezni!

8. feladat:

Adottak az F(1),..., F(M) elvégzendő feladatok és az ezek elvégzésével megbízható V(1),..., V(N) vállalatok. Az egyes vállalatok különböző költséggel tudják elvégezni az egyes feladatokat. Ezen adatokat tartalmazza a K(N,M) mátrix. A K(i,j) jelentése: az i. vállalatnak a j. feladatra vonatkozó „költsége". Mely vállalatokat bízzuk meg az egyes feladatok elvégzésére úgy, hogy minimális legyen a költség!

9. feladat:

Lefedhető-e pontosan egy adott hosszúságú szakasz egyszeresen a H(1),..., H(N) hosszúságú kisebb szakaszokkal (minden szakasz csak egyszer használható föl?

10. feladat:

Lefedhető-e pontosan egy adott hosszúságú szakasz egyszeresen a H(1),..., H(N) hosszúságú kisebb szakaszokkal (minden szakasz csak egyszer használható föl? Ha igen, akkor adjuk meg a legkevesebb szakasz fölhasználásával elérhető megoldást!

11. feladat:

Fedjünk le egyszeresen egy adott méretű négyzetet H(1),..., H(N) oldalhosszúságú kisebb négyzetekkel!

Megoldások

1. feladat:

Egy éhes egérnek egy labirintusban elhelyeznek egy darab sajtot. Írjunk programot, amely segít az egérnek megkeresni a sajthoz vezető utat!

Problémák:

1. ***Mi az eredmény****? Válasz: egy „útvonal-megadás" (lépésirányok: Irány=(Bal,Fel,Jobb,Le)), ami egy előre nem látható hosszúságú labirintusbeli hely-, vagy iránysorozat: Hely(0.. MaxHossz: Hol), vagy MerreTovább(0..MaxHossz:Irány) tömb, ahol MaxHossz a labirintusbeli utak összhossza, a 0. elem a kezdő helyzet. (A Hol miben léte eddig még kérdéses, bár világos.)*
2. ***Mik a bemeneti sorozatok****? Válasz: a négy irány halmazából éppen annyi, amennyi lépés kell a megoldáshoz. De természetesen ez nem valódi programbemenet, csak a tétel egyik bemeneti kellékének konkretizációja.*
3. ***Hogyan ábrázolandó a labirintus****? Pl. egy „térképpel" (Labirintus(Hol:Elem) tömbbel[1]), amelyben háromféle érték fordulhat elő: fal, út és sajt (Elem). A Hol „logikusan" lehet síkbeli pontok koordinátapárja: (1..N,1..M). Így tehát: Labirintus(1..N,1..M: Elem) –most már tudjuk, hogy– mátrix. Ekkor viszont az Irány-beli értékek elmozdulás-megfelelői már adódnak: Bal→(1,0), Fel→(0,1), Jobb→(-1,0), Le→(0,-1).*
4. *Ebben az esetben **mi az fk (kielégíthetőségi) és az ft függvény**? Válasz: fk legyen olyan, hogy ha $Hely_j$ az útvonal j. lépése utáni hely, és az i. lépés után meglépendő ir irány esetén teljesüljön: $fk(Hely_0,...,Hely_{i-1},ir)=Igaz$ akkor, ha a $Hely_{i-1}+ir=Hely_i \notin \{Hely_0 ... Hely_{i-1}\}$, azaz itt még nem járt az egér; az ft: $Labirintus(Hely_{i-1}+ir) \neq fal$, azaz az úton marad. (A + művelet a vektor-összeadás.)*
5. ***Meddig tart a keresés****? Válasz: amíg a sajtra nem találtunk, vagy a RajtHely-től indulva már minden lehetséges irányt kipróbáltunk, és nem tudtunk a sajtig jutni. Ekkor a szegény egér éhes marad.*

Algoritmus:

A megvalósítás során tárolnunk célszerű (miért is?) az utat (Hely tömb), a(z ál)bemeneti sorozatokból (Irány sorozatokból) választott indexeket (LépésIrányIndex tömb). A konstans Irány tömb indexkonvenciója: 1 a Balhoz, azaz az (1,0)-hoz, a 2 a Felhez, azaz a (0,1)-hez éít., a 0 szokás szerint a még „ki nem választottat" jelenti. (Vagyis: Irány:Tömb(0..4:Hol).) A megoldás tehát a LépésIrányIndex tömb és a tényleges hossza, a LépésSzám tartalmazza, és persze a megoldhatóságot kifejező VanÚt logikai változó.

A program valódi paraméterei: Labirintus, RajtHely. (A továbbiakban nem foglalkozunk a deklarációkkal.)

```
Program SajtKeresés:
   …
   Hely(0):=RajtHely [ahonnan indul az egér, pl. (1,1)]
   i:=1; LépésIrányIndex(1..MaxHossz):=0
   Ciklus amíg i≥1 és Labirintus(Hely(i-1))≠Sajt
      VanJóEset(i, van, j)
      Ha van akkor   LépésIrányIndex(i):=j
                     Hely(i):=Hely(i-1)+Irány(j) [+: vektorösszeadás]
                     i:+1
            különben LépésIrányIndex(i):=0
                     i:-1
      Elágazás vége
   Ciklus vége
   VanÚt:=i≥1
   Ha VanÚt akkor LépésSzám:=i-1
Eljárás vége.
```

[1] *Ne akadjunk fel azon, ha egyik-másik programozási nyelv ilyen típust indexként nem fogad el, hanem értsük: mit akar kifejezni!*

```
Eljárás VanJóEset(Konstans i:Egész, Változó van:Logikai, j:Egész):
   j:=LépésIrányIndex(i)+1
   Ciklus amíg j≤4 és (RosszEset(i,j) vagy
               Labirintus(Hely(i-1)+Irány(j))=Fal
               [a Labirintusmátrix Hely(i-1)+Irány(j)
                koordinátájú pontja Fal értékű-e]
      j:+1
   Ciklus vége
   van:=j≤4
Eljárás vége.

Függvény RosszEset(Konstans i,j:Egész): Logikai
   k:=0
   Ciklus amíg k<i és Hely(k)≠Hely(i-1)+Irány(j)
      k:+1
   Ciklus vége
   RosszEset:=k<i
Függvény vége.
```

Kérdések:

1. *Ha a Labirintus-beli értékként megengedünk egy „JártÚt” értéket, annak jelölésére, hogy arra már járt az egér, akkor lényegesen egyszerűsíthető az algoritmus. Hogyan? (A RosszEset-ben nincs szükség az eldöntés tétel alkalmazására.)*
2. *Milyen tulajdonsággal rendelkezik az egér útja, ha csak 1-szélességű folyosók vannak? (Nem biztos, hogy a legrövidebb, de fölösleges „hurkokat” nem tartalmaz.)*
3. *... és ha lehetnek 2-, vagy több szélességű folyosók? Mi lehet a szerepe a „szűk” folyosóknak? (Tartalmazhat az egér útja olyan szakaszokat, amelyeken oda is és vissza is megy a fal mellett, s nem veszi észre, hogy járt már e folyosón. Az 1-szélességűnél képes levágni e hurkokat.)*
4. *Mit lehetne tenni a 3.-beli probléma intelligens megoldására? (A folyosó teljes szélességében figyelni, hogy volt-e már ott; s ha igen, akkor levágni a hurkot; és természetesen ugyanígy figyeli a sajtot is.)*

6. feladat:

Tegyük le az összes dominót úgy, hogy csak az egyik irányba tehetünk, és a dominókon az összes lehetséges párosítás ((0:0),(0:1),...,(0:9),(1:1),...,(9:9)*) előfordul!*

Problémák:

1. *Eredmény...: dominósorozat (fordított dominót nem tároljuk ? darabszám=10+9+...+1=55).*
2. *Bemeneti sorozatok: dominók „halmaza”.*
3. *Ábrázolás...: a dominók = Dominó(1..55[melyik],1..2[oldal]:0..9[hány pötty]).*
4. *fk,ft...: eddig még nem tettük le; s az (eddigi) utolsóhoz illeszthető-e. (Szorosan véve az ft értelmezését –„csak önmagában, saját maga miatt, az eddigi választásainktól függetlenül megfelelő”– „gyenge” kapcsolatot vehetünk észre az utolsóként választottal!)*
5. *Meddig...: amíg az összest le nem tettük.*

Algoritmus:

```
Program Dominó:
   … [most N=55]
   i:=1; DominóSor(1..N):=0
   Ciklus amíg i≥1 és i≤N
      VanJóEset(i, van, j)
      Ha van akkor   DominóSor(i):=j; i:+1
            különben DominóSor(i):=0; i:-1
   Ciklus vége
   Letehető:=i>N
   Ha Letehető akkor … DominóSor(1..N) a letevés …
   …
Program vége.
```

```
Eljárás VanJóEset(Konstans i:Egész, Változó van:Logikai, j:Egész):
   j:=DominóSor(i)+1
   Ciklus amíg j≤N és ( RosszEset(i,j) vagy
                        nem UtolsóhozIllik(i,j) )
      j:+1
   Ciklus vége
   van:=j≤N
   Ha van és Dominó(DominóSor(i-1),2)≠Dominó(j,1)
         akkor Csere(Dominó(j,2),Dominó(j,1))
Eljárás vége.

Függvény RosszEset(Konstans i,j:Egész): Logikai
   k:=1
   Ciklus amíg k<i és DominóSor(k)≠j
      k:+1
   Ciklus vége
   RosszEset:=k<i
Függvény vége.

Függvény UtolsóhozIllik(Konstans i,j:Egész): Logikai
  UtolsóhozIllik:=i=1 vagy
                  Dominó(DominóSor(i-1),2)=Dominó(j,1) vagy
                  Dominó(DominóSor(i-1),2)=Dominó(j,2)
Függvény vége.
```

9. feladat:

Lefedhető-e egy adott hosszúságú szakasz egyszeresen a H(1),..., H(N) hosszúságú kisebb szakaszokkal?

Problémák:

1. *Eredmény…: szakasz-sorozat (**legfeljebb** N darab), pontosabban szakaszindex (SzI(1..???)).*
2. *Bemeneti sorozatok: a szakaszok „halmaza”.*
3. *Ábrázolás…: szakaszok hossza (H(1..N)).*
4. *fk,ft…: az eddig még nem kiválasztott szakasz; az eddig lefedett szakasz összhossza (=pillanatnyi hossz) ≤ lefedendő szakasz hossza (=szakaszHossz); itt is „implicit” kapcsolat van a korábbiakkal.*
5. *Meddig…: a lefedett összhossz (=pillanatnyi hossz) ≠ lefedendő szakasz hossza (szakaszHossz); és persze még nem használtuk föl az összes szakaszt.*
   Utóbbi csak akkor következhet be, ha a szakaszok összhossza kisebb, mint a lefedendő szakaszé. Ez könnyen ellenőrizhető, így elkerülhető a backtrack-es algoritmus effajta „elvi” kudarca.

Algoritmus:

Tegyük föl, hogy elvileg akár létezhet megoldás, azaz $\sum_{i=1}^{N} H_i \geq H$.

Érdemes észre venni, hogy amikor majd az fk kiszámolást végző VanJóEset eljárást írjuk minduntalan ki kell számolnunk az eddig letett szakaszok összhosszát, amit könnyen elkerülhetnénk úgy, hogy „göngyölítjük” azt egy segédváltozóban. Ez legyen a pillHossz.

*Ha így teszünk, akkor ennek módosulását nyomon kell követni a legfelsőbb szinten: ha „VanJóEset”, akkor pillHossz-növelés, ha „NincsJóEset”, akkor levonni a pillHossz-ból az utolsó, elfogadott választást. Ez egy problémás helyzet, hiszen ha egyáltalán nincs megoldás, visszalépünk – mint jól ismert– a „nem létező” 0. választáshoz, ami azt jelenti, hogy a hozzátartozó „hosszt” akarjuk a pillHossz-ból levonni. ... „Legkönnyedebb” megoldás: H(**0**..N:Valós) és SzI(**0**..N:Egész).*

```
Program Lefedés:
   pillHossz:=0; H(0):=0; SzI(0):=0
   i:=1; SzI(1..N):=0
   Ciklus amíg i≥1 és pillHossz≠szakaszHossz
      VanJóEset(i, van, j)
      Ha van akkor  SzI(i):=j; pillHossz:+H(j);      i:+1
           különben SzI(i):=0; pillHossz:-H(SzI(i-1)); i:-1
   Ciklus vége
   Lefedhető:=pillHossz=szakaszHossz
Program vége.

Eljárás VanJóEset(Konstans i:Egész, Változó van:Logikai, j:Egész):
   j:=SzI(i)+1
   Ciklus amíg j≤N és (RosszEset(i,j) vagy
                       pillHossz+H(j)>szakaszHossz)
      j:+1
   Ciklus vége
   van:=j≤N
Eljárás vége.

Függvény RosszEset(Konstans i,j:Egész): Logikai
   k:=1
   Ciklus amíg k<i és SzI(k)≠j
      k:+1
   Ciklus vége
   RosszEset:=k<i
Függvény vége.
```