# Feladatmegoldási stratégiák

## Algoritmusok – kumulatív összegzés

### Kumulatív összegzés

Adott egy N elemű számsorozat, adjuk meg a sorozat azon [a,b] intervallumát, ahol az elemek összege maximális!

Bemenet: $N \in \mathcal{N}$, $X \in \mathcal{N}^*$

Kimenet: $a,b \in \mathcal{H}^*$

$$1 \le a \le b \le N \quad \text{és} \quad \forall p,q(1 \le p \le q \le N): \sum_{i=a}^{b} X_i \ge \sum_{i=p}^{q} X_i$$

Feladatmegoldási stratégiák 2015.05.21. 7:50 2/73

## Algoritmusok – kumulatív összegzés

```
Alapmegoldás:
  MaxÉrt:=-∞
  Ciklus i=1-től N-ig
    Ciklus j=i-től N-ig
      s:=összeg(i,j)
      Ha s>Maxért akkor MaxÉrt:=s; a:=i; b:=j
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

```
összeg(i,j):
  S:=0
  Ciklus k=i-től j-ig
    S:=S+X(k)
  Ciklus vége
  összeg:=S
Függvény vége.
```



## Algoritmusok – kumulatív összegzés

```
Kumulatív összegzés:
  s(0):=0; MaxÉrt:=-∞
  Ciklus i=1-től N-ig
    s(i):=s(i-1)+X(i)
  Ciklus vége
  ...
```

$$\forall i(0 \le i \le N): \quad s(i) = \sum_{j=1}^{i} x_j$$

$$\forall i(1 \le i \le N) \quad s(i) = s(i-1) + x(i), \quad s(0) = 0$$

# Algoritmusok – kumulatív összegzés

$$\sum_{i=a}^{b} x_i = s(b) - s(a-1)$$

```
 ...
  Ciklus i=1-től N-ig
    Ciklus j=i-től N-ig
      Ha s(j)-s(i-1)>Maxért
         akkor MaxÉrt:=s(j)-s(i-1); a:=i; b:=j
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



# Algoritmusok – kumulatív összegzés

## A megoldások összehasonlítása

Alapmegoldás: 3 egymásba ágyazott ciklus – O(N$^3$)

Kumulatív összegzés: 2 egymásba ágyazott ciklus – O(N$^2$)

## Meggondolandók

- ez az elv milyen struktúrákra alkalmazható?
- ez az elv milyen feladattípusokra alkalmazható?

**Feladat:** Egy mátrix legnagyobb üres téglalapja megadása.

# Rekurzió

## Klasszikus példák

- Faktoriális

$$n! = \begin{cases} n*(n-1)! & ha \ \ n>0 \\ 1 & ha \ \ n=0 \end{cases}$$

- Fibonacci-számok

$$Fib(n) = \begin{cases} 0 & ha \ \ n=0 \\ 1 & ha \ \ n=1 \\ Fib(n-1)+Fib(n-2) & ha \ \ n>1 \end{cases}$$

A rekurzió lényege: önhivatkozás



# Rekurzív specifikáció és algoritmus

## Fibonacci számok:

$$Fib(n) = \begin{cases} 0 & ha \ \ n=0 \\ 1 & ha \ \ n=1 \\ Fib(n-1)+Fib(n-2) & ha \ \ n>1 \end{cases}$$

```
Fib(n):
  Ha n=0 akkor Fib:=0
  különben ha n=1 akkor Fib:=1
  különben Fib:=Fib(n-1)+Fib(n-2)
Eljárás vége.
```

Lame számok: Lame(n)=Lame(n-1)+Lame(n-3)

Q számok: Q(n)=Q(n-Q(n-1))+Q(n-Q(n-2))

# Problémák a rekurzióval

## Bajok a rekurzióval

Hely: nagyra dagadt veremméret.

Idő:a vermelés adminisztrációs többletterhe, a többszörösen ismétlődő hívások. Pl. Fibonacci-számoknál:

r(N):=az N. Fibonacci-szám kiszámításához szükséges hívások száma

r(0):=1, r(1):=1, r(i):=r(i-1)+r(i-2)+1

***Állítás:***

*a) r(i)=F(i+1)+F(i)+F(i-1)-1 i>1*

*b) r(i)=2*F(i+1)-1, ahol F(i)=az i. Fibonacci-szám.*



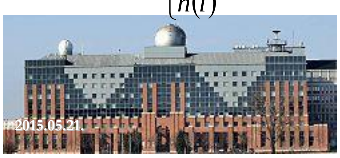
# Korlátos memóriájú függvények

## Korlátos memóriájú függvények

Ha egy rekurzív függvény minden értéke valamely korábban kiszámolható értékből számolható, akkor némi memória-felhasználással elkészíthető a rekurzió mentes változat, amelyben az egyes függvényértékeknek megfeleltetünk egy F(N) vektort.

A függvény általános formája:

$$f(i)=\begin{cases} g\big(f(i-1), f(i-2), \ldots, f(i-K)\big) & ha \quad i \ge K \\ h(i) & ha \quad 0 \le i < K \end{cases}$$



# Korlátos memóriájú függvények

## Korlátos memóriájú függvények

```
f(N):
  Ha N<K akkor f:=h(N)
  különben f:=g(f(N-1),...,f(N-K))
Függvény vége.
```




# Korlátos memóriájú függvények

## Korlátos memóriájú függvények

Az ennek megfelelő vektoros változat:

```
f(N):
  Ciklus I=0-tól K-1-ig
    F(I):=h(I)
  Ciklus vége
  Ciklus I=K-tól N-ig
    F(I):=g(F(I-1),...,F(I-K))
  Ciklus vége
  f:=F(N)
Függvény vége.
```

# Korlátos memóriájú függvények

## Korlátos memóriájú függvények

Ez így természetesen nem hatékony tárolás, hiszen a rekurzív formulából látszik, hogy minden értékhez csak az őt megelőző K értékre van szükség.

A hatékony megoldásban az alábbi ciklust kell átalakítani:

```
Ciklus I=K-tól N-ig
  F(I):=g(F(I-1),...,F(I-K))
Ciklus vége
```

Lehet pl. `F(I mod K):=g(F(K-1),...,F(0))`, ha a g() függvény kiszámítása nem függ a paraméter sorrendtől.




# Korlátos memóriájú függvények

## Példa: Fibonacci-számok

$$Fib(n) = \begin{cases} 0 & ha \ \ n = 0 \\ 1 & ha \ \ n = 1 \\ Fib(n-1) + Fib(n-2) & ha \ \ n > 1 \end{cases}$$

Helytakarékos megoldás:

```
Fib(n):
  F(0):=0; F(1):=1
  Ciklus i=2-től n-ig
    F(i mod 2):=F(0)+F(1)
  Ciklus vége
  Fib:=F(n mod 2)
Függvény vége.
```

# Rekurzió memorizálással

Megoldási ötlet: amit már kiszámoltunk egyszer, azt ne számoljuk újra! Tároljuk a már kiszámolt értékeket, s ha szükségünk van rájuk, használjuk fel őket!

A megoldásban F(i)≥0 jelenti, ha már kiszámoltuk az i-edik Fibonacci számot.

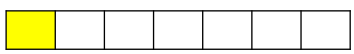
```
Fib(N):
  Ha F(N)<0 akkor ha N<2 akkor F(N):=N
                          különben F(N):=Fib(N-1)+Fib(N-2)
  Fib:=F(N)
Függvény vége.
```

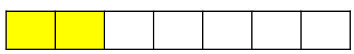


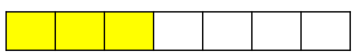

# Közvetett rekurzió - járdakövezés

## Feladat

Számítsuk ki, hogy hányféleképpen lehet egy n egység méretű járdát kikövezni 1x1, 1x2 és 1x3 méretű lapokkal!

Az első helyre tehetünk 1x1-es lapot: 

Az első helyre tehetünk 1x2-es lapot: 

Az első helyre tehetünk 1x3-as lapot: 

Az első esetben n-1, a másodikban n-2-t, a harmadikban pedig n-3 cellát kell még lefednünk. Azaz az n cella lefedéseinek Lefed(n) száma:

Lefed(n-1)+Lefed(n-2)+Lefed(n-3).

# Közvetett rekurzió - járdakövezés

```
Lefed(N):
  Elágazás
    N=0 esetén Lefed:=0
    N=1 esetén Lefed:=1
    N=2 esetén Lefed:=2
    egyéb esetben Lefed:=Lefed(N-1)+
                         Lefed(N-2)+Lefed(N-3)
  Elágazás vége
Függvény vége.
```

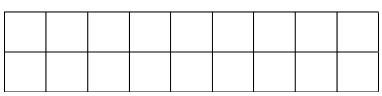
Sokszoros hívás esetén vagy memorizálás, vagy ciklusos megoldás kell!




# Közvetett rekurzió - járdakövezés

## Feladat

Számítsuk ki, hogy hányféleképpen lehet egy 2xn egység méretű járdát kikövezni 1x2 és 1x3 méretű lapokkal!

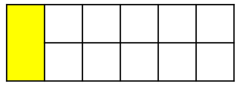


## Megoldás

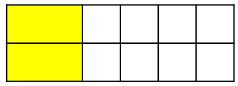
Biztos nincs megoldás, ha n<2!

# Közvetett rekurzió - járdakövezés

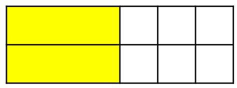
Az első oszlop egyféleképpen fedhető le:



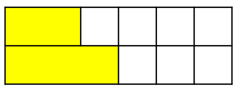
Az első két oszlop további elrendezéssel újra egyféleképpen fedhető le:

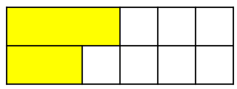


Az első három oszlop újra egyféleképpen:



Sajnos ez is előfordulhat:





# Közvetett rekurzió - járdakövezés

Jelölje A(n) a megoldás értékét 2xn egység méretű járda esetén!

Jelölje B(n) a megoldás értékét 2xn egység méretű járda esetén, ha az egyik jobboldali sarok nincs befestve!

$$A(n)=\begin{cases} 1 & \text{ha} \quad n=1 \\ 2 & \text{ha} \quad n=2 \\ 4 & \text{ha} \quad n=3 \\ A(n-1)+A(n-2)+A(n-3)+2*B(n-2) & \text{ha} \quad n>3 \end{cases}$$

$$B(n)=\begin{cases} 0 & \text{ha} \quad n=1 \\ 0 & \text{ha} \quad n=2 \\ 1 & \text{ha} \quad n=3 \\ A(n-3)+B(n-1)+B(n-3) & \text{ha} \quad n>3 \end{cases}$$

# Közvetett rekurzió - járdakövezés

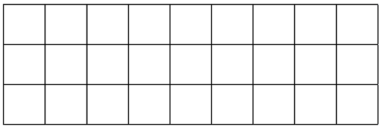
```
A(n):
  Ha n=1 akkor A:=1
  különben ha n=2 akkor A:=2
  különben ha n=3 akkor A:=4
  különben A:=A(n-1)+A(n-2)+A(n-3)+2*B(n-2)
Függvény vége.
B(n):
  Ha n<3 akkor B:=0
  különben ha n=3 akkor B:=1
  különben B:=A(n-3)+B(n-1)+B(n-3)
Függvény vége.
```




# Közvetett rekurzió - járdakövezés

## Feladat

Számítsuk ki, hogy hányféleképpen lehet egy 3xn egység méretű járdát kikövezni 1x2 méretű lapokkal!

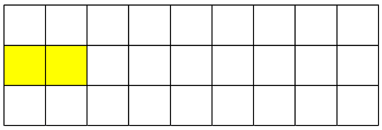


## Megoldás

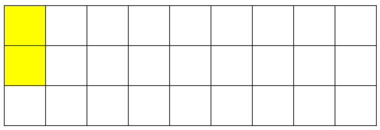
Biztos nincs megoldás, ha n páratlan szám!

# Közvetett rekurzió - járdakövezés

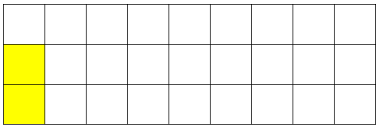
Az első oszlop középső négyzete háromféleképpen fedhető le.



1. eset



2. eset



3. eset




# Közvetett rekurzió - járdakövezés

Az egyes esetek csak az alábbi módon folytathatók:

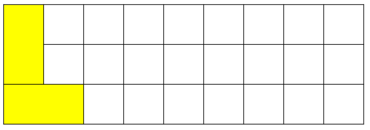
Jelölje A(n) a megoldás értékét 3xn egység méretű járda esetén!

Az 1. eset csak így folytatható

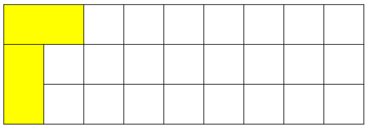
## Közvetett rekurzió - járdakövezés

Jelölje B(n) azt, hogy hányféleképpen fedhető le egy 3xn egység méretű járda, amelynek a bal alsó sarka már le van fedve!

Szimmetria miatt a jobb felső sarok lefedettsége esetén is B(n)-féle lefedés van.



A 2. eset csak így folytatható



A 3. eset csak így folytatható

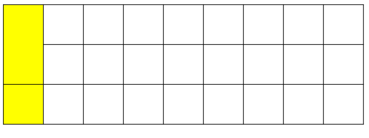
$$A(n)=\begin{cases} 0 & ha \quad n=1 \\ 3 & ha \quad n=2 \\ A(n-2)+2*B(n-1) & ha \quad n>2 \end{cases}$$



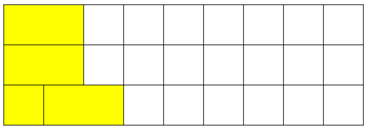

## Közvetett rekurzió - járdakövezés

Az egyes esetek csak az alábbi módon folytathatók:

Jelölje B(n) azt, hogy hányféleképpen fedhető le egy 3xn egység méretű járda, amelynek a bal alsó sarka már le van fedve! B(n) páros n-re mindig 0 értékű!



A 2. eset csak így folytatható



Az 1. eset csak így folytatható

$$B(n)=\begin{cases} 1 & ha \quad n=1 \\ 0 & ha \quad n=2 \\ A(n-1)+B(n-2) & ha \quad n>2 \end{cases}$$

## Közvetett rekurzió - járdakövezés

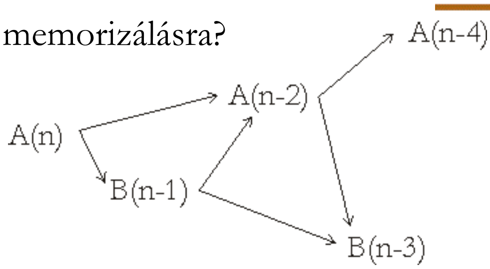
```
A(n):
  Ha n=1 akkor A:=0
  különben ha n=2 akkor A:=3
  különben A:=A(n-2)+2*B(n-1)
Függvény vége.
B(n):
  Ha n=1 akkor B:=1
  különben ha n=2 akkor B:=0
  különben B:=A(n-1)+B(n-2)
Függvény vége.
```

```
Kövezés(n):
 Ha páros(n) akkor Kövezés:=A(n)
  különben Kövezés:=0
Függvény vége.
```




## Közvetett rekurzió - járdakövezés

Szükség van itt memorizálásra?




Igen, B(n-3)-hoz háromféle, A(n-4)-hez ötféle úton juthatunk el (B(n-3)-ból is számoljuk) – Fibonacci számszor!

<u>Megjegyzés</u>: Figyeljük meg, hogy csak minden második A(i) és B(i) értéket számoljuk ki!

# Közvetett rekurzió - járdakövezés

```
A(n):
  Ha TA(n)<0 akkor
     Ha n=1 akkor TA(n):=0
     különben ha n=2 akkor TA(n):=3
     különben TA(n):=A(n-2)+2*B(n-1)
  Elágazás vége
  A:=TA(n)
Függvény vége.
```




# Közvetett rekurzió - járdakövezés

```
B(n):
  Ha TB(n)<0
     Ha n=1 akkor TB(n):=1
     különben ha n=2 akkor TB(n):=0
     különben TB(n):=A(n-1)+B(n-2)
  Elágazás vége
  A:=TB(n)
Függvény vége.
```

# Feladatmegoldási stratégiák

## Oszd meg és uralkodj!

Több részfeladatra bontás, amelyek hasonlóan oldhatók meg, lépései:

- a triviális eset (amikor nincs rekurzív hívás)
- felosztás (megadjuk a részfeladatokat, amikre a feladat lebontható)
- uralkodás (rekurzívan megoldjuk az egyes részfeladatokat)
- összevonás (az egyes részfeladatok megoldásából előállítjuk az eredeti feladat megoldását)



# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

Ezek alapján a következőképpen fogunk gondolkodni:

- Mi a leállás (triviális eset) feltétele? Hogyan oldható meg ilyenkor a feladat?
- Mi az általános feladat alakja? Mik a paraméterei? Ebből kapjuk meg a rekurzív eljárásunk specifikációját.
- Milyen paraméterértékekre kapjuk a konkrét feladatot? Ezekre fogjuk meghívni kezdetben az eljárást!

## Feladatmegoldási stratégiák
## Oszd meg és uralkodj

Ezek alapján a következőképpen fogunk gondolkodni:

- Hogyan vezethető vissza a feladat hasonló, de egyszerűbb részfeladatokra? Hány részfeladatra vezethető vissza?
- Melyek ilyenkor az általános feladat részfeladatainak a paraméterei? Ezekkel kell majd meghívni a rekurzív eljárást!
- Hogyan építhető fel a részfeladatok megoldásaiból az általános feladat megoldása?



## Feladatmegoldási stratégiák
## Oszd meg és uralkodj

**Gyorsrendezés (quicksort):**

- felbontás: $\boxed{X_1, \dots, X_{k-1}}\ X_k\ \boxed{X_{k+1}, \dots, X_n}$ szétválogatás
  ahol $\forall i,j\ (1 \le i < k;\ k < j \le n)$: $X_i \le X_j$
- uralkodás: mindkét részt ugyanazzal a módszerrel felbontjuk két részre, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt
- triviális eset: $n \le 1$

# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## Gyorsrendezés (quicksort):

```
Quick(E,U):
  Szétválogatás(E,U,K)
  Ha E<K-1 akkor Quick(E,K-1)
  Ha k+1<U akkor Quick(K+1,U)
Eljárás vége.
```



# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## Gyorsrendezés (quicksort):

```
Szétválogatás(N,X,K):
  E:=1; U:=N; segéd:=X(E)
  Ciklus amíg E<U
    Ciklus amíg E<U és segéd≤X(U)
      U:=U-1
    Ciklus vége
    ...
```

# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## Gyorsrendezés (quicksort):

```
   ...
   Ha E<U akkor X(E):=X(U); E:=E+1
                Ciklus amíg E<U és segéd≥X(E)
                  E:=E+1
                Ciklus vége
                Ha E<U akkor X(U):=X(E); U:=U-1
   Elágazás vége
  Ciklus vége
  X(E):=segéd; K:=E
Eljárás vége.
```



# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## Összefésüléses rendezés (mergesort):

- felbontás: a sorozat két részsorozatra bontása (középen)

| | |
|---|---|
| $X_1, \ldots, X_k$ | $X_{k+1}, \ldots, X_n$ |

- uralkodás: a két részsorozat rendezése (rekurzívan)
- összevonás: a két rendezett részsorozat összefésülése
- triviális eset: $n \leq 1$

# Feladatmegoldási stratégiák
## Oszd meg és uralkodj

### Összefésüléses rendezés (mergesort):

```
Rendez(E,U):
  Ha E<U akkor K:=(E+U)/2
               Rendez(E,K); Rendez(K+1,U)
               Összefésül(E,K,U)
Eljárás vége.
```



# Feladatmegoldási stratégiák
## Oszd meg és uralkodj

### Összefésüléses rendezés (mergesort):

```
Összefésül(X,E,K,U):
  i:=1; j:=1; DB:=E-1; Y():=X(E..K)
  Z():=X(K+1..U); Y(K-E+2):=+∞; Z(U-K+1):=+∞
  Ciklus amíg i<K-E+2 vagy j<U-K+1
    DB:=DB+1
    Ha Y(i)<Z(j) akkor  X(DB):=Y(i); i:=i+1
               különben X(DB):=Z(j); j:=j+1
  Ciklus vége
Eljárás vége.
```

# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## i-edik legkisebb kiválasztása:

- felbontás: $\boxed{X_1, \dots , X_{k-1}}$ $X_k$ $\boxed{X_{k+1}, \dots , X_n}$ szétválogatás (ahol $\forall i,j$ $(1 \le i \le k;\ k \le j \le n)$: $X_i \le X_j$)
- uralkodás: i<K esetén az első, i>K esetén a második részben keresünk tovább, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt
- triviális eset: i=k




# Feladatmegoldási stratégiák
# Oszd meg és uralkodj


## i-edik legkisebb kiválasztása:

```
Kiválasztás(E,U,i,Y):
  Szétválogatás(E,U,K)
  Ha i=K akkor Y:=X(K)
  különben ha i<K akkor Kiválasztás(E,K-1,i,Y)
  különben Kiválasztás(K+1,U,i-K,Y)
Eljárás vége.
```

# Feladatmegoldási stratégiák
# Oszd meg és uralkodj

## További ilyen feladatok:

- Hanoi tornyai: N korong mozgatása visszavezetése két N-1 korong mozgatása feladatra.
- Logaritmikus keresés: N elemű sorozatban keresés vissza-vezetése N/2 elemű sorozatban keresésre.
- Egy téglalapban levő N ponthoz a legnagyobb résztéglalap keresése, amely egyetlen pontot sem tartalmaz – egy meg-vizsgálandó téglalapot minden belső pont négy megvizsgá-landó részre vág.



# Feladatmegoldási stratégiák
# Visszalépéses keresés

## Visszalépéses keresés (backtrack)

A visszalépéses keresés lényege a feladat megoldásának előállítása rendszeres próbálgatással.

Adott N sorozat, amelyek rendre M(1), M(2), ... M(N) elemszámúak. Ki kell választani mindegyikből egy-egy elemet úgy, hogy az egyes sorozatokból való választások másokat befolyásolnak. Másképp fogalmazva: egy adott tulajdonsággal rendelkező szám N-est kell megadni úgy, hogy ne kelljen az összes lehetőséget végignézni!

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## N vezér elhelyezése egy NxN-es sakktáblán

Helyezzünk el egy NxN-es sakktáblán N vezért úgy, hogy ne üssék egymást!

Egy lehetséges megoldás N=5-re és N=4-re:

| | | | | |
|---|---|---|---|---|
| | | V | | |
| | | | | V |
| | V | | | |
| | | | V | |
| V | | | | |

| | | | |
|---|---|---|---|
| | V | | |
| | | | V |
| V | | | |
| | | V | |




# Feladatmegoldási stratégiák
# Visszalépéses keresés


## Munkásfelvétel: N állás – N jelentkező

Egy vállalkozás N különböző állásra keres munkásokat. Pontosan N jelentkező érkezett, ahol minden jelentkező megmondta, hogy mely munkákhoz ért. A vállalkozás vezetője azt szeretné, ha az összes jelentkezőt fel tudná venni és minden munkát el tudna végeztetni.

| | Darab | Állások: 1. | 2. | 3. |
|---|---|---|---|---|
| 1. jelentkező: | 2 | 1 | **4** | |
| 2. jelentkező: | 1 | **2** | | |
| 3. jelentkező: | 2 | **1** | 2 | |
| 4. jelentkező: | 1 | **3** | | |
| 5. jelentkező: | 3 | 1 | 3 | **5** |

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## A visszalépéses keresés megoldási elve

E feladatok közös jellemzője, hogy eredményük egy sorozat.

E sorozat minden egyes tagját valamilyen sorozatból kell kikeresni (vezért egy oszlop valamely helyére, egy munkásnak a vállalt munkák közül valamelyiket), de az egyes keresések összefüggenek egymással (vezért nem lehet oda tenni, ahol egy korábban letett vezér ütné; egy munkát nem lehet két munkásnak adni).

Feladatmegoldási stratégiák 2015.05.21. 7:50 47/73


# Feladatmegoldási stratégiák
# Visszalépéses keresés

## A visszalépéses keresés megoldási elve


- A visszalépéses keresés olyan esetekben használható, amikor a keresési tér fastruktúraként képzelhető el, amiben a gyökérből kiindulva egy csúcsot keresünk.
- Az algoritmus lényege, hogy a kezdőpontból kiindulva megtesz egy utat a feladatot részproblémákra bontva, és ha valahol az derül ki, hogy már nem juthat el a célig, akkor visszalép egy korábbi döntési ponthoz, és ott más utat – más részproblémát választ.

Feladatmegoldási stratégiák 2015.05.21. 7:50 48/73

## Feladatmegoldási stratégiák
## Visszalépéses keresés

Először megpróbálunk az első sorozatból kiválasztani egy elemet, ezután a következőből, ...

Ha nincs jó választás, akkor visszalépünk az előző sorozathoz, s megpróbálunk abból egy másik elemet választani.

Visszalépésnél törölni kell a választást abból a sorozatból, amelyikből visszalépünk. Az eljárás akkor ér véget, ha minden sorozatból sikerült választani, vagy pedig a visszalépések sokasága után már az első sorozatból sem lehet újabb elemet választani (ekkor a feladatnak nincs megoldása).

Feladatmegoldási stratégiák 2015.05.21. 7:50 49/73

## Feladatmegoldási stratégiák
## Visszalépéses keresés

### Visszalépéses keresés algoritmus:

```
Keresés(N,Van,Y):
  i:=1; Y:=(0,...,0)
  Ciklus amíg i≥1 és i≤N {lehet még és nincs még kész}
    Jóesetkeresés(i,Van,j)
    Ha Van akkor Y(i):=j; i:=i+1   {előrelépés}
        különben Y(i):=0; i:=i-1   {visszalépés}
  Ciklus vége
  Van:=(i>N)
Eljárás vége.
```

A megoldás legfelső szintjén keressünk az i. sorozatból megfelelő elemet! Ha ez sikerült, akkor lépjünk tovább az i+1. sorozatra, különben lépjünk vissza az i-1.-re, s keressünk abban újabb elemet!

Feladatmegoldási stratégiák 2015.05.21. 7:50 50/73

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## Visszalépéses keresés algoritmus:

```
Jóesetkeresés(i,Van,j):
  j:=Y(i)+1
  Ciklus amíg j≤M(i) és
                     (rossz(i,j) vagy tilos(j))
    j:=j+1
  Ciklus vége
  Van:=(j≤M(i))
Eljárás vége.
```

Megjegyzés: az i-edik lépésben a j-edik döntési út nem választható, ha az előzőek miatt rossz, vagy ha önmagában rossz.

Feladatmegoldási stratégiák 2015.05.21. 7:50 51/73

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## Visszalépéses keresés algoritmus:

```
rossz(i,j):                    {1. változat}
  k:=1
  Ciklus amíg k<i és szabad(i,j,k,Y(k))
    k:=k+1
  Ciklus vége
  rossz:=(k<i)
Eljárás vége.
```

Megjegyzés: Rossz egy választás, ha valamelyik korábbi választás miatt nem szabad – eldöntés.

Feladatmegoldási stratégiák 2015.05.21. 7:50 52/73

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## Visszalépéses keresés algoritmus:

```
rossz(i,j):                    {2. változat}
  s:=F0
  Ciklus k=1-től i-1-ig
    s:=f(s,k,Y(k))
  Ciklus vége
  rossz:=nem szabad(s,i,j)
Eljárás vége.
```

Megjegyzés: Rossz egy választás, ha a korábbiak összessége miatt nem szabad – sorozatszámítás, megszámolás, …




# Feladatmegoldási stratégiák
# Visszalépéses keresés


## Visszalépéses keresés algoritmus:

```
rossz(i,j):                    {3. változat}
  rossz:=i>1 és nem szabad(i,j,i-1,Y(i-1))
Eljárás vége.
```

Megjegyzés: Rossz egy választás, ha az előző választás miatt nem szabad – feltétel vizsgálat.

# Feladatmegoldási stratégiák
# Visszalépéses keresés

## Feladat

Helyezzünk el egy NxN-es sakktáblán N vezért úgy, hogy ne üssék egymást!

A vezérek a sorukban, az oszlopukban és az átlójukban álló bábukat üthetik. Tehát úgy kell elhelyezni a vezéreket, hogy minden sorban és minden oszlopban is pontosan 1 vezér legyen, és minden átlóban legfeljebb 1 vezér legyen!



Feladatmegoldási stratégiák
Visszalépéses keresés

## N vezér a sakktáblán:

```
Keresés(N,Van,Y):
  i:=1; Y:=(0,...,0)
  Ciklus amíg i≥1 és i≤N  {lehet még és nincs még kész}
    Jóesetkeresés(i,Van,j)
    Ha Van akkor Y(i):=j; i:=i+1   {előrelépés}
       különben Y(i):=0; i:=i-1   {visszalépés}
  Ciklus vége
  Van:=(i>N)
Eljárás vége.
```

# Feladatmegoldási stratégiák
## Visszalépéses keresés

### N vezér a sakktáblán:

```
Jóesetkeresés(i,Van,j):
  j:=Y(i)+1
  Ciklus amíg j≤N és rossz(i,j)
    j:=j+1
  Ciklus vége
  Van:=(j≤N)
Eljárás vége.
```



# Feladatmegoldási stratégiák
## Visszalépéses keresés

### N vezér a sakktáblán:

```
rossz(i,j):
  k:=1
  Ciklus amíg k<i és Y(k)≠j és i-k≠abs(j-Y(k))
    k:=k+1
  Ciklus vége
  rossz:=(k<i)
Eljárás vége.
```

# Feladatmegoldási stratégiák
## Visszalépéses kiválogatás

Visszalépéses kiválogatás rekurzív algoritmus:

```
Visszalépéses kiválogatás(N,Db,Y):
  Db:=0; X:=(0,…,0); Backtrack(1,N,X,Db,Y)
Eljárás vége.
```



# Feladatmegoldási stratégiák
## Visszalépéses kiválogatás

```
Backtrack(i,N,X,Db,Y):
  Ha i=N+1 akkor Db:=Db+1; Y(Db):=X
  különben Ciklus j=1-től N-ig
             Ha ft(i,j) és nem Rossz(i,j)
                akkor X(i):=j
                      Backtrack(i+1,N,X,Db,Y)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

## Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

Visszalépéses maximumkeresés rekurzív algoritmus:

```
Visszalépéses maximumkeresés(N,Van,Y):
  X:=(0,…,0); Y:=X; Backtrack(1,N,X,Y)
  Van:=Y≠(0,…,0)
Eljárás vége.
```



## Feladatmegoldási stratégiák
## Visszalépéses maimumkeresés

Visszalépéses maximumkeresés rekurzív algoritmus:

```
Backtrack(i,N,X,Y):
  Ha i=N+1 akkor ha nagyobb?(X,Y) akkor Y:=X
  különben Ciklus j=1-től N-ig
             Ha ft(i,j) és nem Rossz(i,j)
                akkor X(i):=j
                     Backtrack(i+1,N,X,Y)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

# Feladatmegoldási stratégiák
# Visszalépéses maximumkeresés

Példa: (1. változat)

Egy vállalkozás N különböző állásra keres munkásokat. Pontosan N jelentkező érkezett, ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne.

Minden munkát el kell végeztetni valakivel, mindenkinek munkát kell adni, de a legolcsóbban!

| Allások: | 1. | 2. | 3. | 4. | 5. |
|---|---|---|---|---|---|
| 1. jelentkező: | 100 | 0 | 0 | **100** | 0 |
| 2. jelentkező: | 0 | **200** | 0 | 0 | 0 |
| 3. jelentkező: | **200** | 100 | 0 | 0 | 0 |
| 4. jelentkező: | 0 | 0 | **200** | 0 | 400 |
| 5. jelentkező: | 500 | 0 | 400 | 0 | **200** |



# Feladatmegoldási stratégiák
# Visszalépéses maximumkeresés

Ha egy megoldás elkészül, akkor a költségét így számíthatjuk ki:

```
költség(X):
  S:=0
  Ciklus i=1-től N-ig
    S:=S+F(i,X(i))
  Ciklus vége
Függvény vége.
```

Kezdetben olyan – fiktív – megoldásból kell kiindulni, aminél minden valódi megoldás jobb.

# Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

```
Legjobb állás(N,i):
  Ha i>N akkor
       Ha költség(X)<költség(Y) akkor Y:=X
  különben
       Ciklus j=1-től N-ig
         Ha nem volt(i,j) és F(i,j)>0
            akkor X(i):=j; Legjobb állás(N,i+1)
       Ciklus vége
  Elágazás vége
Eljárás vége.
```

Ebben a megoldásban feleslegesen sokszor hívjuk a Költség függvényt.

Feladatmegoldási stratégiák 2015.05.21. 7:50 65/73

# Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

```
Legjobb állás(N,i):
  Ha i>N akkor Ykölt:=költség(Y)
               Ha maxkölt<Ykölt
                  akkor Y:=X; maxkölt:=Ykölt
  különben
       Ciklus j=1-től N-ig
         Ha nem volt(i,j) és F(i,j)>0
            akkor X(i):=j; Legjobb állás(N,i+1)
       Ciklus vége
  Elágazás vége
Eljárás vége.
```

Itt feleslegesen nem hívjuk a Költség függvényt, jó maxkölt kezdőérték kell.

Feladatmegoldási stratégiák 2015.05.21. 7:50 66/73

## Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

Már csak egy apróságra gondolhatunk: ha van egy megoldásunk és a most készülő megoldásról látszik, hogy már biztosan rosszabb lesz – többe fog kerülni –, akkor azt már nem érdemes tovább vinni.

Ha lehetséges, adhatunk kezdő felső korlátot is!

Legyen az eljárás paramétere az eddigi költség, s az eljárást csak akkor folytassuk, ha még nem érjük el a korábban kiszámolt maximális költséget. Emiatt nem a megoldások elkészültekor kell számolni költséget, hanem menet közben, folyamatosan.



## Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

```
Legjobb állás(N,i,költ):
  Ha i>N akkor
      Ha költ<maxkölt akkor Y:=X; maxkölt:=költ
  különben Ciklus j=1-től N-ig
             Ha nem volt(i,j) és F(i,j)>0
                és költ+F(i,j)<maxkölt
             akkor X(i):=j
                 Legjobb állás(N,i+1,költ+F(i,j))
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

## Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

Példa: (2. változat)

Egy vállalkozás N különböző állásra keres munkásokat. Pontosan M jelentkező érkezett (M<N), ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne.

Mindenkinek munkát kell adni (csak egyet mindenkinek), de a legolcsóbban!

| Állások: | 1. | 2. | 3. | 4. | 5. |
|---|---|---|---|---|---|
| 1. jelentkező: | 100 | 0 | 0 | **100** | 0 |
| 2. jelentkező: | 0 | **200** | 0 | 0 | 0 |
| 3. jelentkező: | **200** | 100 | 0 | 0 | 0 |
| 4. jelentkező: | 0 | 0 | **200** | 0 | 400 |



## Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

```
Legjobb állás(N,M,i,költ):
  Ha i>M akkor
      Ha költ<maxkölt akkor Y:=X; maxkölt:=költ
  különben Ciklus j=1-től N-ig
             Ha nem volt(i,j) és F(i,j)>0
                és költ+F(i,j)<maxkölt
             akkor X(i):=j
               Legjobb állás(N,M,i+1,költ+F(i,j))
           Ciklus vége
  Elágazás vége
Eljárás vége.
```

# Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

Példa: (3. változat)

Egy vállalkozás N különböző állásra keres munkásokat. Pontosan M jelentkező érkezett (M>N), ahol minden jelentkező megmondta, hogy mely munkákhoz ért, illetve amihez ért, arra mennyi fizetést kérne.

Mindenkinek munkát el kell végezni, egy-egy embernek, de a legolcsóbban!

| Állások: | 1. | 2. | 3. | 4. |
|---|---|---|---|---|
| 1. jelentkező: | 100 | 0 | 0 | **100** |
| 2. jelentkező: | 0 | **200** | 0 | 0 |
| 3. jelentkező: | **200** | 100 | 0 | 0 |
| 4. jelentkező: | 0 | 0 | **200** | 0 |
| 5. jelentkező: | 500 | 0 | 400 | 0 |



# Feladatmegoldási stratégiák
## Visszalépéses maximumkeresés

```
Legjobb állás(N,M,i,költ):
  Ha i>N akkor
      Ha költ<maxkölt akkor Y:=X; maxkölt:=költ
  különben Ciklus j=1-től M-ig
             Ha nem volt(j,i) és F(j,i)>0
                és költ+F(j,i)<maxkölt
             akkor X(i):=j
               Legjobb állás(N,M,i+1,költ+F(j,i))
           Ciklus vége
  Elágazás vége
Eljárás vége.
```


Feladatmegoldási stratégiák