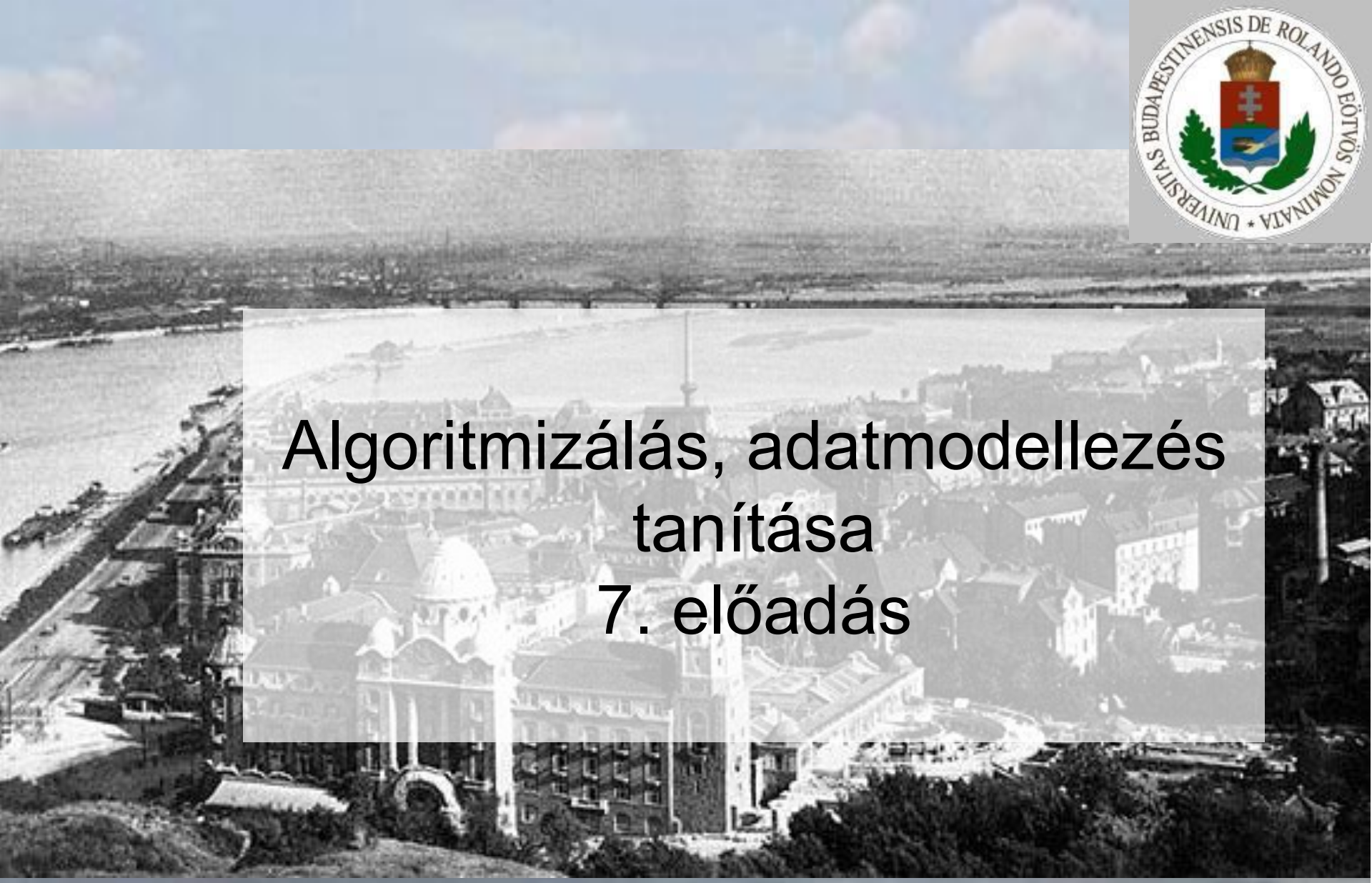
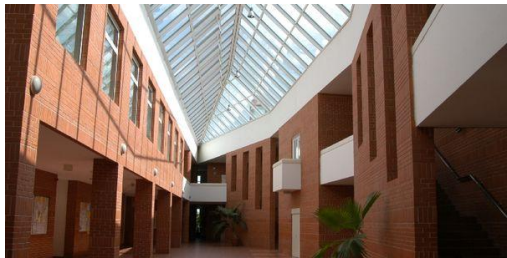


Algoritmizálás, adatmodellezés tanítása
7. előadás

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)

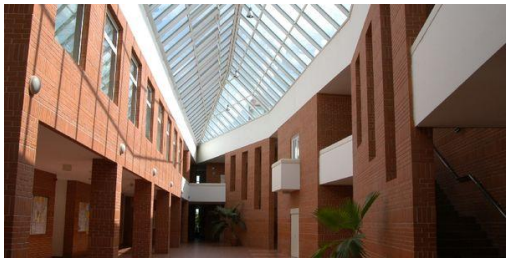


Feladatmegoldási stratégiák
Oszd meg és uralkodj

Gyorsrendezés (quicksort):

- felbontás: $\boxed{X_1, \ldots, X_{k-1}}\, X_k \boxed{X_{k+1}, \ldots, X_n}$ szétválogatás

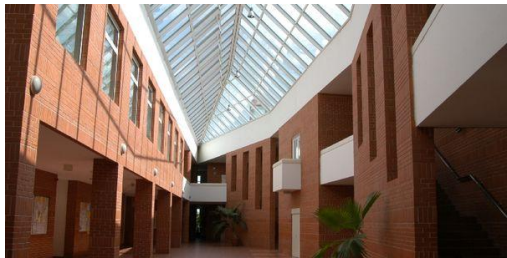
  ahol $\forall i,j\ (1 \le i < k;\ k < j \le n)$: $X_i \le X_j$
- uralkodás: mindkét részt ugyanazzal a módszerrel felbontjuk két részre, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás miatt
- triviális eset: $n \le 1$



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.
```



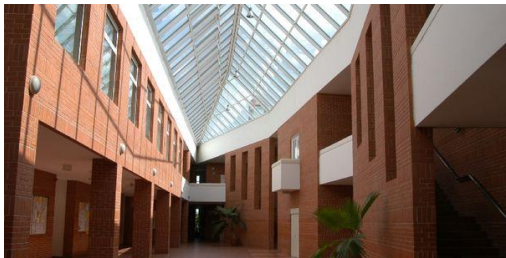
Feladatmegoldási stratégiák
Oszd meg és uralkodj

Összefésüléses rendezés (mergesort):

- felbontás: a sorozat két részsorozatra bontása (középen)

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

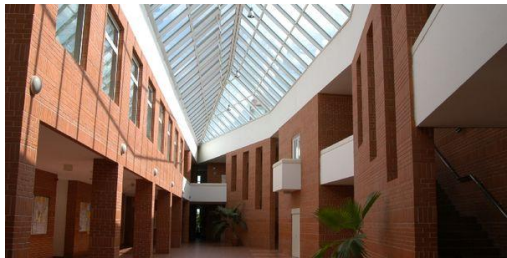
- uralkodás: a két részsorozat rendezése (rekurzívan)
- összevonás: a két rendezett részsorozat összefésülése
- triviális eset: $n \leq 1$



Feladatmegoldási stratégiák
Oszd meg és uralkodj

Összefésüléses rendezés (mergesort):

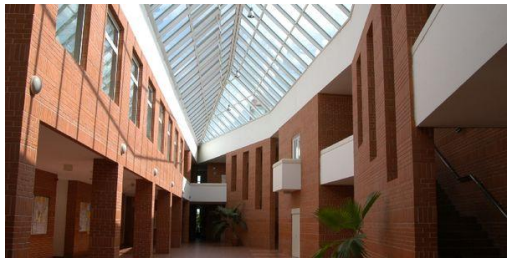
```
Rendez(E,U):
   Ha E<U akkor K:=(E+U)/2
                Rendez(E,K); Rendez(K+1,U)
                Összefésül(E,K,U)
Eljárás vége.
```



Feladatmegoldási stratégiák
Oszd meg és uralkodj

i-edik legkisebb kiválasztása:

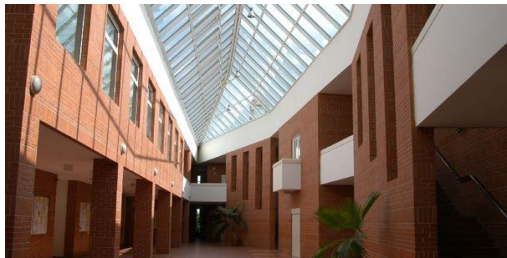
- felbontás: $\boxed{X_1, \dots, X_{k-1}}\ X_k\ \boxed{X_{k+1}, \dots, X_n}$ szétválogatás (ahol $\forall i,j\ (1 \le i \le k;\ k \le j \le n)$: $X_i \le X_j$)
- uralkodás: i<K esetén az első, i>K esetén a második részben keresünk tovább, rekurzívan
- összevonás: automatikusan történik a helyben szétválogatás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



Feladatmegoldási stratégiák
Oszd meg és uralkodj

i-edik legkisebb kiválasztása:

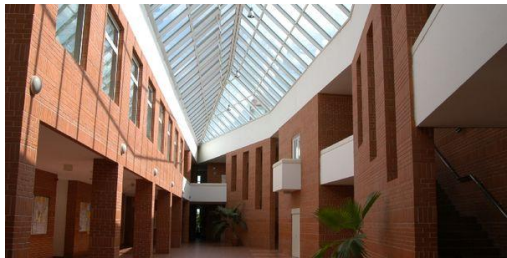
```
Kiválasztás(E,U,i,Y):
   Szétválogatás(E,U,K)
   Ha i=K akkor Y:=X(K)
   különben ha i<K akkor Kiválasztás(E,K-1,i,Y)
   különben Kiválasztás(K+1,U,i-K,Y)
Eljárás vége.
```



Feladatmegoldási stratégiák
Oszd meg és uralkodj

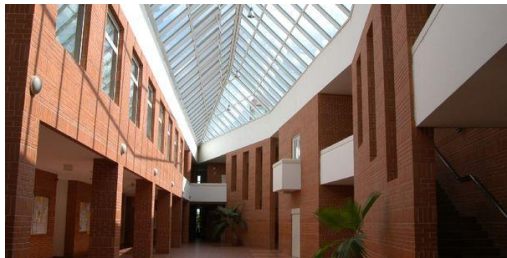
További ilyen feladatok:

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



Feladatmegoldási stratégiák
Visszalépéses keresés

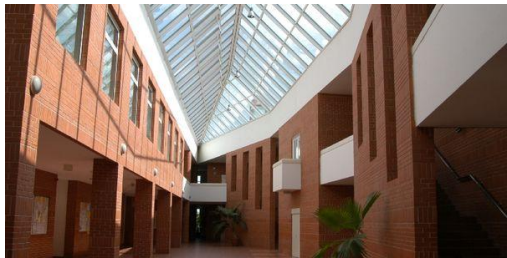
- A visszalépéses keresés olyan esetekben használható, amikor a keresési tér fastruktúraként képzelhető el, amiben a gyökérből kiindulva egy csúcsot keresünk.
- Az algoritmus lényege, hogy a kezdőpontból kiindulva megtesz egy utat a feladatot részproblémákra bontva, és ha valahol az derül ki, hogy már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



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses keresés algoritmus:

```
Keresés(N,Van,Y):
  i:=1
  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.
```

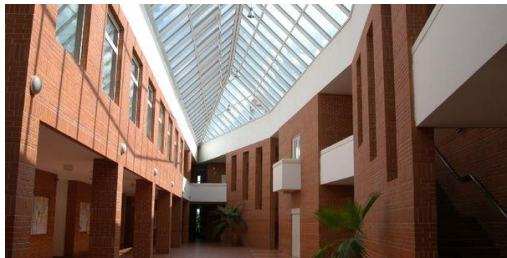


Feladatmegoldási stratégiák
Visszalépéses keresés

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.

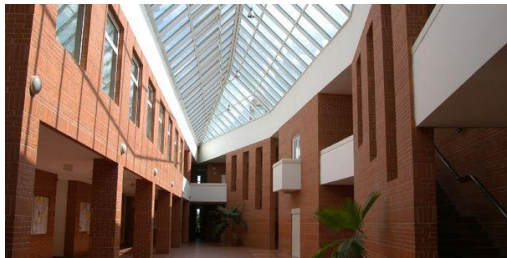


Feladatmegoldási stratégiák
Visszalépéses keresés

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.

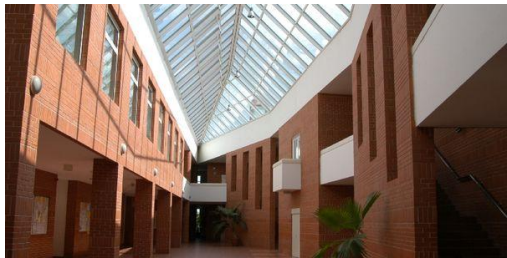


Feladatmegoldási stratégiák
Visszalépéses keresés

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, …

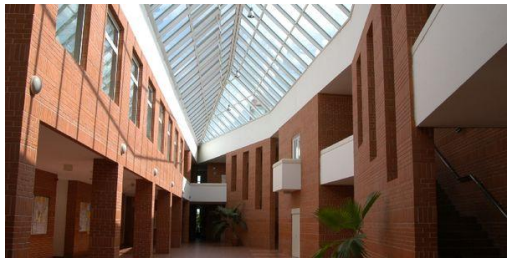


Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses keresés algoritmus:

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

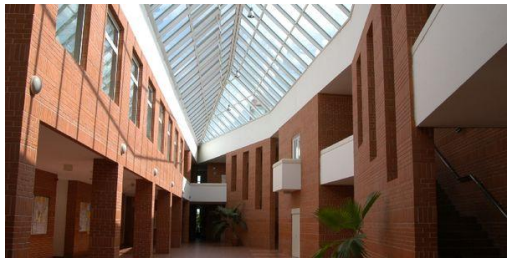
Megjegyzés: Rossz egy választás, ha az előző választás miatt nem szabad – feltétel vizsgálat.



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses kiválogatás rekurzív algoritmus:

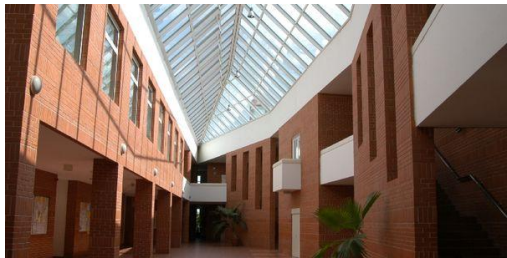
```
Visszalépéses kiválogatás(N,Db,Y):
  Db:=0; X:=(0,…,0); Backtrack(1,N,X,Db,Y)
Eljárás vége.

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 keresés

Visszalépéses maximumkeresés rekurzív algoritmus:

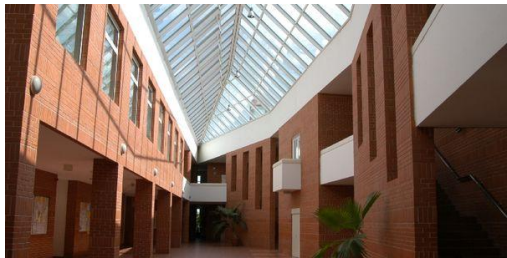
```
Visszalépéses maximumkeresés(N,Van,Y):
   X:=(0,…,0); Y:=X; Backtrack(1,N,X,Y)
   Van:=Y≠(0,…,0)
Eljárás vége.
```



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses maximumkeresés rekurzív algoritmus:

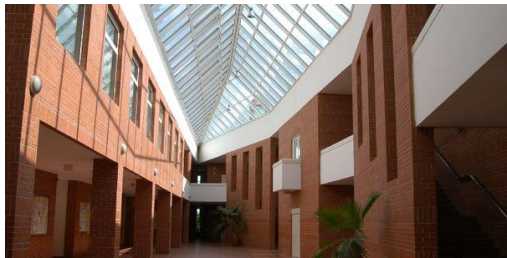
```
Backtrack(i,N,X,Y):
  Ha i=N+1 akkor ha nagyobb?(X,Y) akkor Y:=X
  különben Ciklus j=1-től N-ig
             Ha ft(i,j) és nem Rossz(i,j)
                akkor X(i):=j
                       Backtrack(i+1,N,X,Y)
           Ciklus vége
  Elágazás vége
Eljárás vége.
```



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses keresés feladatok:

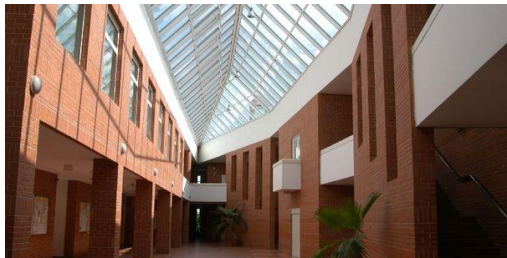
- 8 vezér a sakktáblán
- N munka – N munkás
- N közért – M pékség
- Labirintusban útkeresés
- Permutációk, kombinációk előállítása
- Térképszínezés
- Pénzfelbontás adott címletekre



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses keresés feladatok:

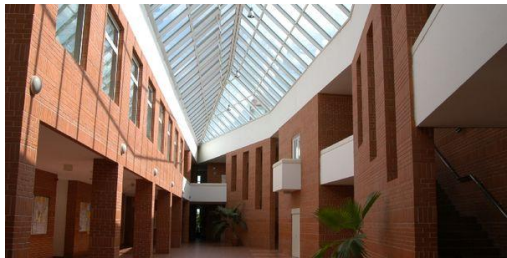
- Úthossz-korlát: Fává egyenesítünk, végtelen fát állítunk elő. Nem engedjük, hogy az aktuális út hossza meghaladja az úthossz-korlátot.
  - Ha túl rövidre választjuk az úthossz-korlátot (túl alacsonyan vágjuk el a gráfot) akkor nem találunk megoldást.
  - Ha a start csúcsban áll elő a visszalépési feltétel, akkor:
  - 1. nincs megoldás
  - 2. túl rövidre választottuk az úthossz-korlátot



Feladatmegoldási stratégiák
Visszalépéses keresés

Visszalépéses keresés feladatok:

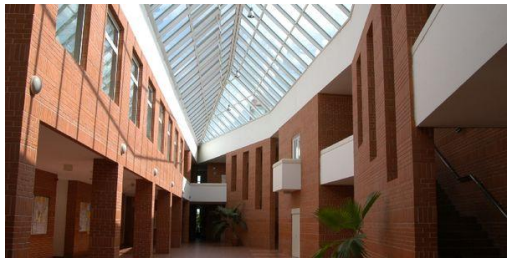
- Kör kiküszöbölése
  - lesz egy újabb visszalépési feltétel: az aktuális csúcs szerepelt-e már az aktuális úton
  - ha igen: rögtön visszalépés (így nem zárjuk be a kört).

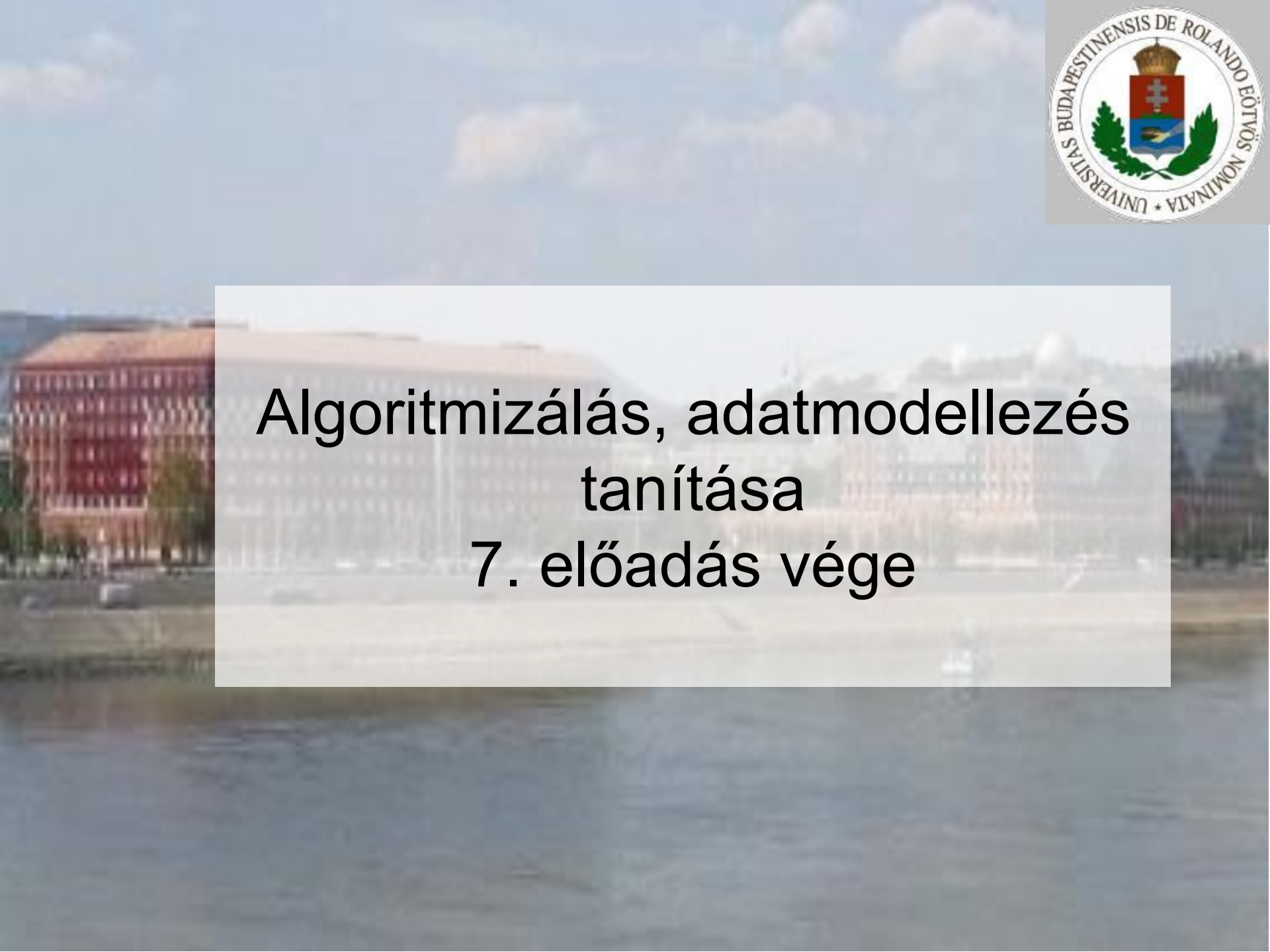


Feladatmegoldási stratégiák
Visszalépéses keresés

A visszalépéses stratégia

- véges fákban mindig terminál (véges fákban teljes);
- végtelen gráfban úthossz-korláttal terminál (kör kizárása: az aktuális út csúcsait nem engedjük ismételni;
- egy zsákutcát többször is bejár, ha több út vezet hozzá.





Algoritmizálás, adatmodellezés tanítása
7. előadás vége