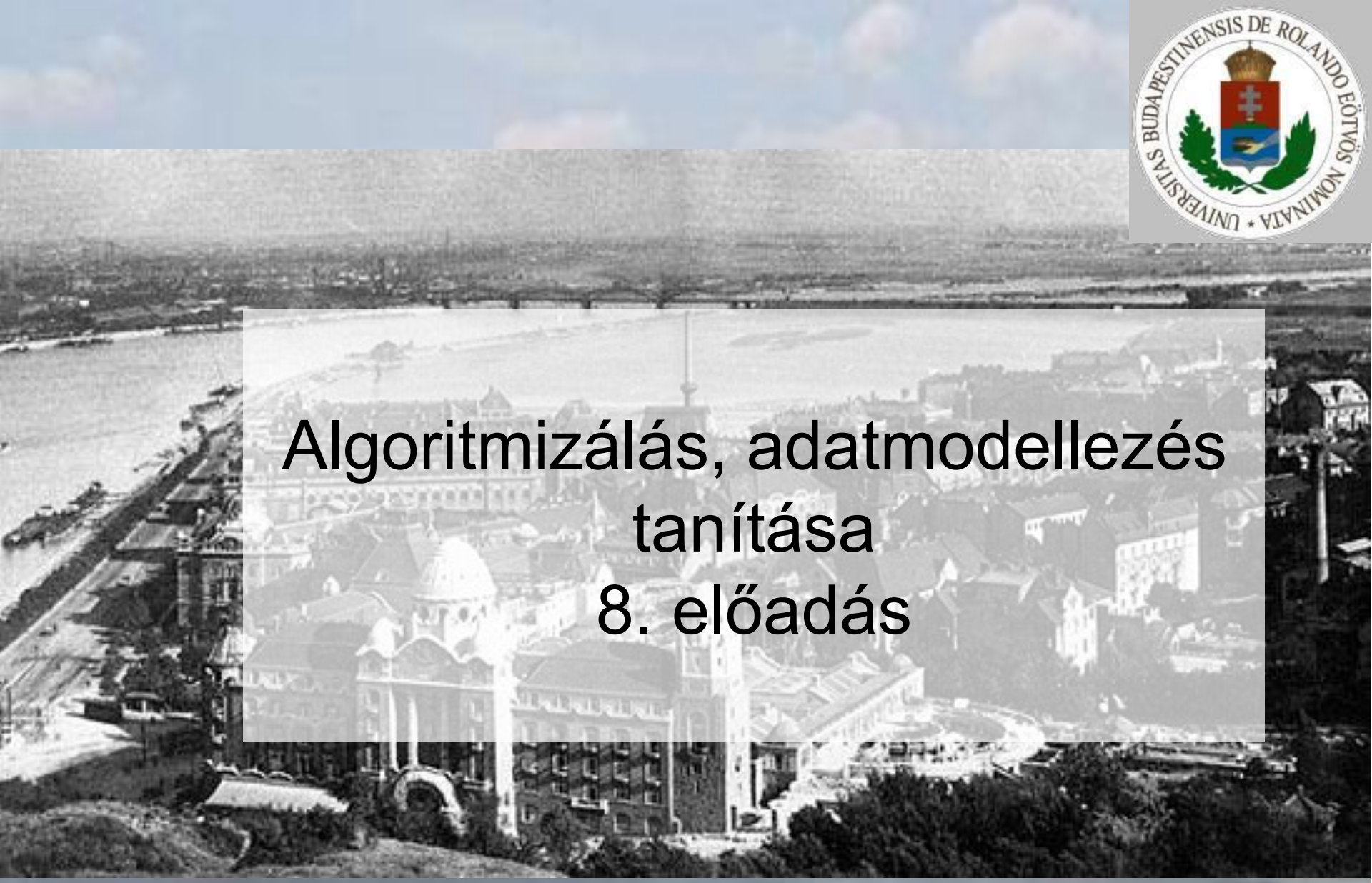
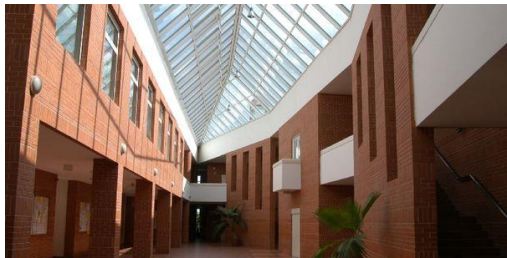


Algoritmizálás, adatmodellezés tanítása
8. előadás

Feladatmegoldási stratégiák
Elágazás és korlátozás

A backtrack alkalmas-e optimális megoldás keresésére? Van költség, és a legkisebb költségű megoldást szeretnénk előállítani.

- Van egy induló költségkorlát (felső becslés).
- Ennél a költségkorlátnál nem költségesebb megoldást keresünk.



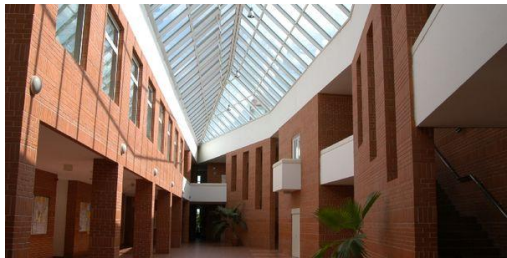
Feladatmegoldási stratégiák
Elágazás és korlátozás

Az aktuális pontot tetszőlegesen választhatjuk az aktív pontok közül.

A lényeg, hogy a választott aktuális pontból elérhető összes pontot generáljuk, és ha lehetséges megoldás, akkor betesszük az aktív pontok halmazába.

Tehát az algoritmus egy, az aktív pontokat tartalmazó adagolót használ az aktív pontok tárolására.

A visszalépéses stratégia esetén elég volt egy pontot, az aktuális pontot tárolni, mert a következő aktív pont mindig ennek fia, testvére, vagy apja.



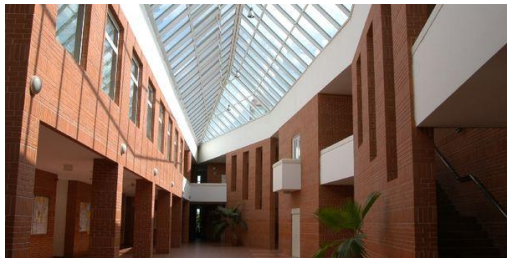
Feladatmegoldási stratégiák
Elágazás és korlátozás

Adott a C(X) valós értékű célfüggvény, és olyan X megoldást keresünk, amelyre a célfüggvény C(X) értéke minimális.

A megoldáskezdeményekre meg tudunk adni olyan AK(X) alsó korlát függvényt, amelyekre teljesül az alábbi egyenlőtlenség.

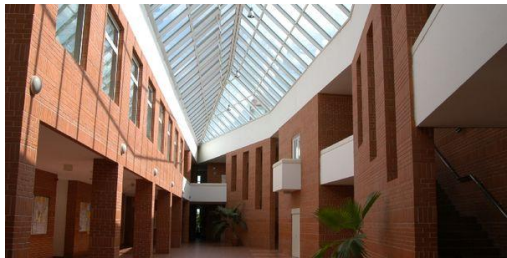
- Az Y megoldás bármely X részmegoldására: $AK(X) \leq C(Y)$

Ekkor az adagoló lehet az AK szerinti minimumos prioritási sor, tehát az aktív pontok közül mindig a legkisebb alsó korlátú pontot választjuk aktuálisnak.



Feladatmegoldási stratégiák
Elágazás és korlátozás

```
Korlátozás(F):
  Minért:=+∞; Betesz(A,F)
  Ciklus amíg nem üres?(A)
    Kivesz(A,F)
    Ciklus p=F-ből kapható megoldáslépések
      Ha Megoldás(p) akkor
         Ha C(p)<Minért akkor Minért:=C(p)
                              Min:=p
      különben ha AK(p)<Minért akkor Betesz(A,p)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```



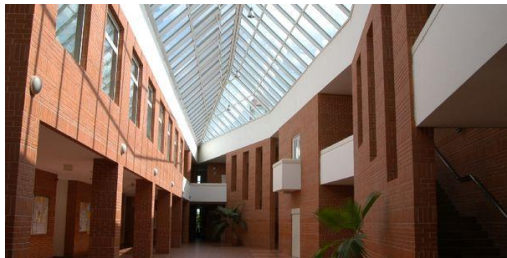
Feladatmegoldási stratégiák
Elágazás és korlátozás

Ha a megoldáskezdeményekre meg tudunk adni felső korlátot is, akkor az adagoló lehet a felső korlát szerinti minimumos prioritási sor.

- Felső korlát olyan FK(X) függvény, amelyre teljesül, hogy minden Y megoldás minden X részmegoldására:

$$C(Y) \leq FK(X).$$

- Azaz egy részmegoldásnál járva tudjuk, hogy az ebből kiinduló megoldásoknak mi a felső korlátja.
- Mindig a legkisebb felső korlátú ágat válasszuk!

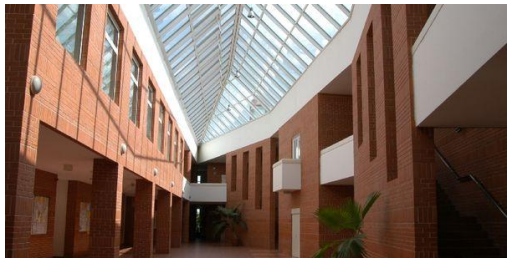


Feladatmegoldási stratégiák
Mohó stratégia

Az optimalizálási probléma megoldására szolgáló algoritmus sokszor olyan lépések sorozatából áll, ahol minden lépésben adott halmazból választhatunk.

A mohó algoritmus mindig az adott lépésben optimálisnak látszót választja: a lokális optimumot választja, feltételezve, hogy ez globális optimumhoz fog majd vezetni.

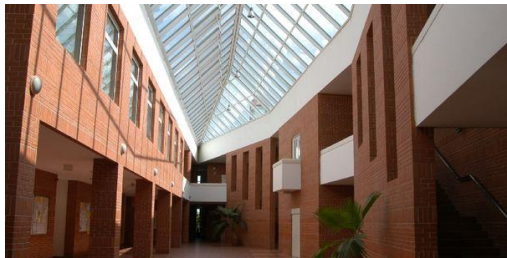
Ez nem mindig ad optimális megoldást, azonban sok probléma megoldható mohó algoritmussal.



Feladatmegoldási stratégiák
Mohó stratégia

Adott az L halmaz és egy T tulajdonság az L részhalmazaira (igaz vagy hamis). Keresünk egy optimális M részhalmazt kiválogatás programozási tétellel.

```
Mohó(L,Van,M):
  M:=∅
  Ciklus amíg M nem megoldás és L≠∅
    Válasszuk ki L-ből a legjobb x elemet
    L:=L\{x}
    Ha T(M ∪ {x}) akkor M := M ∪ {x}
  Ciklus vége
  Van:=M megoldás
Eljárás vége.
```



Feladatmegoldási stratégiák
Mohó stratégia

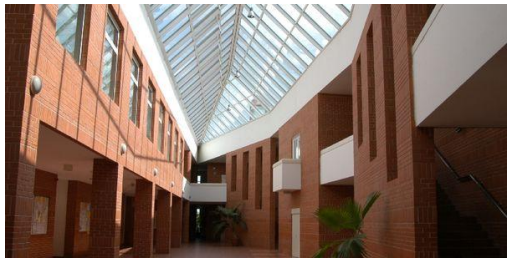
Egy esemény-kiválasztási feladat

Bemenet:

- E={1..n} esemény egy erőforrásért.
- Minden esemény ideje: $[k_i,v_i)$.
- i,j kompatibilisek $\Leftrightarrow [k_i,v_i) \cap [k_j,v_j) = \varnothing$

Kimenet:

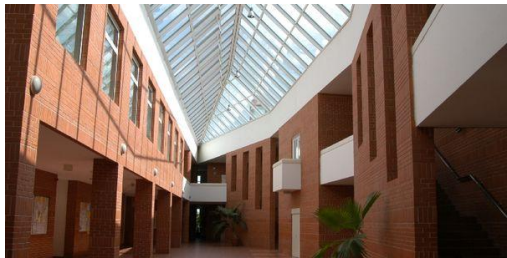
- $M=\{e_1,e_2 \ldots e_{Db}\}$ $e_i \in E$
- M maximális elemszámú
- e_i-k páronként kompatibilisek



Feladatmegoldási stratégiák
Mohó stratégia

Egy esemény-kiválasztási feladat

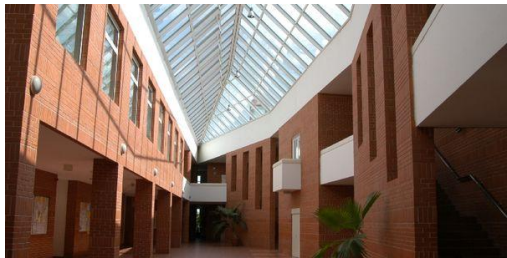
- Eseményenként döntsünk arról, hogy bevesszük vagy nem!
- Bevehető, ha „kompatibilis” az eddigiekkel.
- Válasszunk az események közül úgy, hogy a legtöbb lehetőség maradjon a továbbiak számára!
- Az az optimális, amelyik leghamarabb ér véget.
- Ehhez szükséges a végidőpontok szerinti rendezés.



Feladatmegoldási stratégiák
Mohó stratégia

Egy esemény-kiválasztási feladat

```
Mohó(N,K,V,Db,X):
  Rendezés V szerint; Db:=1; X(1):=1
  Ciklus i=2-től N-ig
    Ha K(i)≥V(X(Db)) akkor Db:=Db+1; X(Db):=i
  Ciklus vége
Eljárás vége.
```

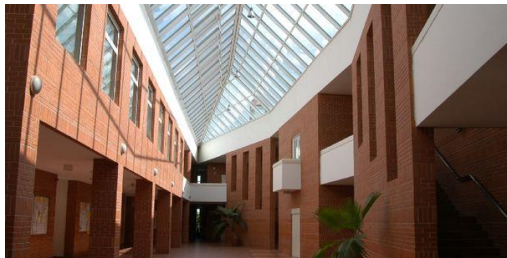


Feladatmegoldási stratégiák
Mohó stratégia

Egy esemény-kiválasztási feladat

$Kezd_j$: a j-ben végződő, legkésőbb kezdődő intervallum, S_j a sorszáma. $1 \le K_i, V_i \le M$.

```
Kiválogatás(N,K,V,Db,X):
  DB:=0; idő:=0; Kezdetek(N,K,V,Kezd,S)
  Ciklus i=1-től M-ig
    Ha S(i)≠0 és idő≤Kezd(i)
       akkor Db:=Db+1; X(Db):=S(i); idő:=i
  Ciklus vége
Eljárás vége.
```

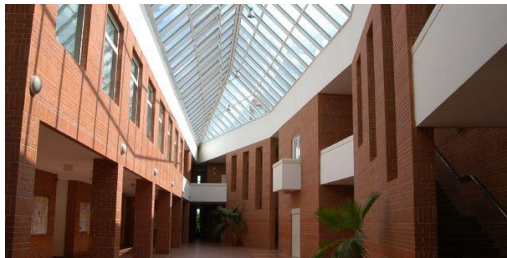


Feladatmegoldási stratégiák
Mohó stratégia

Egy esemény-kiválasztási feladat

Kezd$_j$ előállítása:

```
Kezdetek(N,K,V,Kezd,S):
  Kezd:=(0,…,0)
  Ciklus i=1-től N-ig
    Ha K(i)>Kezd(V(i))
       akkor Kezd(V(i)):=K(i); S(V(i)):=i
  Ciklus vége
Eljárás vége.
```



Feladatmegoldási stratégiák
Mohó stratégia

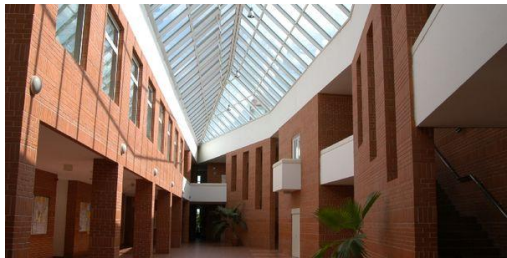
Egy intervallum lefedési feladat

Bemenet:

- E={1..n} intervallum, [A,B] lefedendő intervallum.
- Minden intervallum kezdete és hossza: $[k_i,h_i)$.
- i,j kompatibilisek $\Leftrightarrow k_i+h_i \geq k_j$, de $k_i+h_i < k_{j+1}$

Kimenet:

- $M=\{e_1,e_2 \ldots e_{Db}\}$ $e_i \in E$
- A szomszédos e_i-k páronként kompatibilisek
- M minimális elemszámú
- $k_1=A$, $k_{Db}+h_{Db} \geq B$

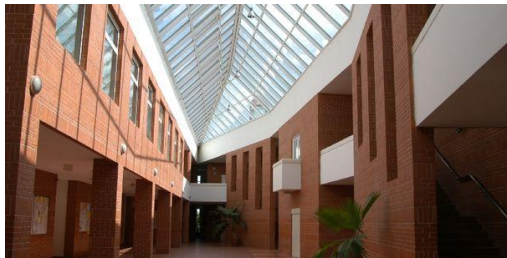


Feladatmegoldási stratégiák
Mohó stratégia

Egy intervallum lefedési feladat

- Intervallumonként döntsünk arról, hogy bevesszük vagy nem! Bevehető, ha „kompatibilis” az eddigiekkel – kompatibilis, ha átfedi az előzőt.
- Optimális, ha a kompatibilisek közül a legnagyobb a végpontja.
- Adott elemig akkor ismerjük az optimálist, ha az azt követő már nem kompatibilis.
- Ehhez szükséges a kezdőidőpontok szerinti rendezés.

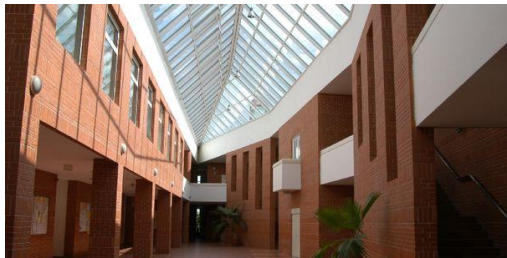
Tegyük fel, hogy van megoldás!



Feladatmegoldási stratégiák
Mohó stratégia

Egy intervallum lefedési feladat

```
Mohó(N,K,H,Db,X):
  Rendezés K szerint; Db:=1; X(1):=1; max:=1
  Ciklus i=2-től N-ig
    Ha K(i)+H(i)>K(max)+H(max) akkor max:=i
    Ha K(i+1)>K(X(Db))+H(X(Db))
       akkor Db:=Db+1; X(Db):=max
  Ciklus vége
Eljárás vége.
```

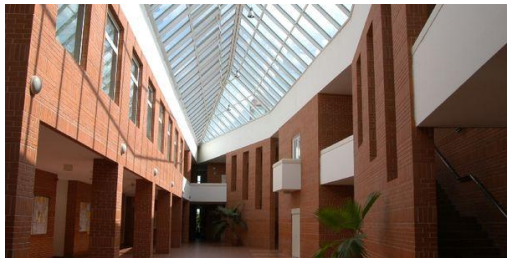


Feladatmegoldási stratégiák
Mohó stratégia

Egy erőforrás kiosztási feladat

N folyamat igényel egy erőforrást, különböző időintervallumokban. Az erőforrásból minimális mennyire van szükség, hogy minden igényt kielégíthessünk?

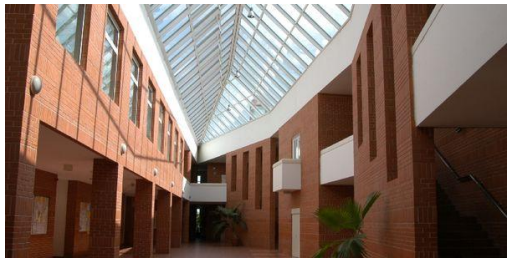
Rendezzük sorba az igényeket kezdési idő szerint! Vegyük sorra őket és rendeljük hozzájuk az első szabad erőforrást! Ha mindegyik erőforrás foglalt, akkor új erőforrást kell üzembe állítanunk!



Feladatmegoldási stratégiák
Mohó stratégia

Egy erőforrás kiosztási feladat

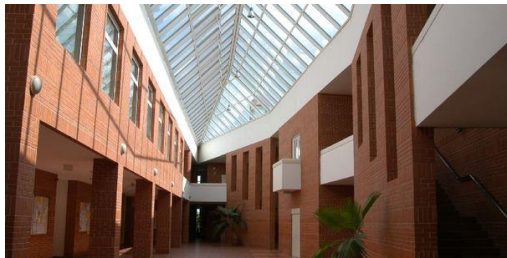
```
Mohó(N,K,V,Db,X):
  Rendezés K szerint
  Db:=1; X(1):=1; U(1):=V(1)
  Ciklus i=2-től N-ig
    j:=1
    Ciklus amíg j≤Db és K(i)<U(j)
      j:=j+1
    Ciklus vége
    Ha j≤Db akkor X(i):=j; U(j):=V(i)
    különben Db:=Db+1; X(i):=Db; U(Db):=V(i)
  Ciklus vége
Eljárás vége.
```



Feladatmegoldási stratégiák
Mohó stratégia

A mohó megoldó stratégia elemei

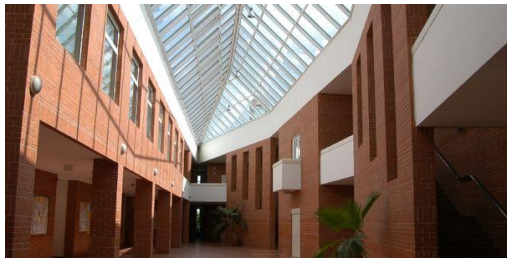
- Fogalmazzuk meg az optimalizációs feladatot úgy, hogy minden egyes választás hatására egy megoldandó részprobléma keletkezzen!
- Bizonyítsuk be, hogy mindig van olyan optimális megoldása az eredeti problémának, amely tartalmazza a mohó választást, tehát a mohó választás mindig biztonságos!
- Mutassuk meg, hogy a mohó választással olyan részprobléma keletkezik, amelynek egy optimális megoldásához hozzávéve a mohó választást, az eredeti probléma egy optimális megoldását kapjuk!



Feladatmegoldási stratégiák
Mohó stratégia

Néhány további klasszikus feladat:

- Huffman-kódolás
- Minimális költségű feszítőfa
- Darabolási feladat
- Pénzváltási feladat
- Töredékes hátizsák feladat
- Ütemezési feladat

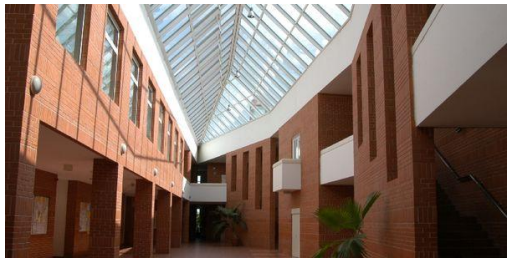


Feladatmegoldási stratégiák
Dinamikus programozás

Az optimalizálás alapelve: az optimális megoldás optimális részmegoldásokból épül fel.

A dinamikus programozás feltétele:

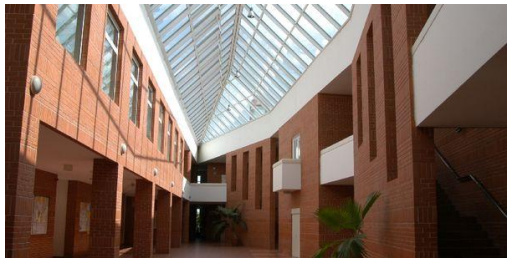
- optimalizálási elv
- ne legyen mohó választás (akkor is lehetne, de nem érdemes)
- a részfeladatok között legyen sok azonos.



Feladatmegoldási stratégiák
Dinamikus programozás

A dinamikus programozás lépései:

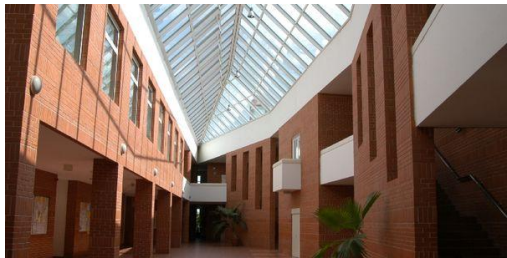
- a megoldás szerkezetének tanulmányozása – részproblémákra bontás megsejtése
- részproblémákra és összetevőkre bontás – részproblémák és paramétereik körvonalazása: a rekurzió előkészítése
- részproblémák megoldásának kifejezése rekurzívan az összetevők megoldásaiból – formalizálás: függvény-definíció
- részproblémák megoldásának kiszámítása – a táblaszámítás algoritmizálása



Feladatmegoldási stratégiák
Dinamikus programozás

A dinamikus programozás lépései:

- kiszámítási sorrend meghatározása: minden részprobléma minden összetevője előbb szerepeljen a felsorolásban
- az „alulról-felfelé” haladó számítás
- a megoldás előállítása a 4. lépésben előállított táblázat segítségével – a megoldás algoritmizálása



Feladatmegoldási stratégiák
Dinamikus programozás

Toronyépítés: a legmagasabb torony

Bemenet:

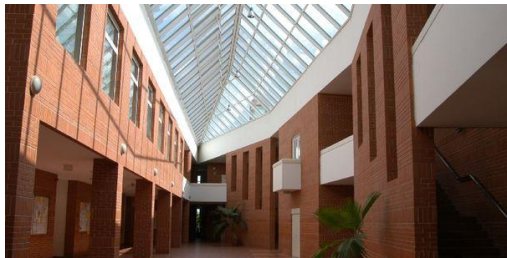
- n kocka, M_i méretű, S_i súlyú

Feltétel:

- nagyobb kocka kisebbre nem tehető
- nehezebb kocka könnyebbre nem tehető

Kimenet:

- H a torony magassága
- Db a felhasznált kockák száma
- $K=\{k_1,...k_{Db}\}$ a felhasznált kockák



Feladatmegoldási stratégiák
Dinamikus programozás

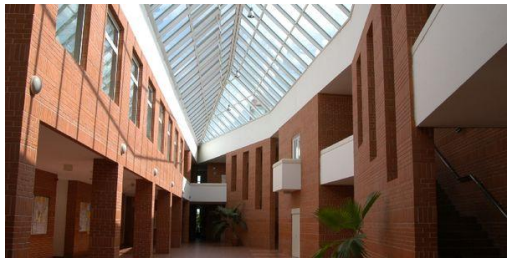
Toronyépítés: a legmagasabb torony

Ötlet:

- rendezzük a kockákat súly szerint csökkenő sorrendbe
- ekkor a kockasorozat egy részsorozata a megoldás

Részproblémák:

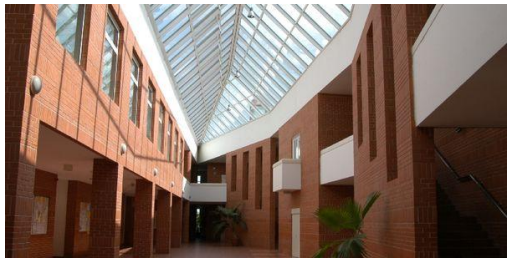
- Mekkora torony építhető az első K kockából úgy, hogy a K-adik van legfelül?
- A K-adikat olyan toronyra tehetjük, ahol vagy az 1., vagy a 2., ... vagy a K-1. kocka volt legfelül.



Feladatmegoldási stratégiák
Dinamikus programozás

Toronyépítés : a legmagasabb torony

```
Torony(N,M,H,Db,K):
  Mag(1):=M(1); R(1):=0
  Ciklus i=2-től N-ig
    Mag(i):=M(i); R(i):=0
    Ciklus j=1-től i-1-ig
      Ha M(i)≤M(j) és Mag(j)+M(i)>Mag(i)
         akkor Mag(i):=Mag(j)+M(i); R(i):=j
    Ciklus vége
  Ciklus vége
  Megoldáskeresés(N,H,R,Db,K)
Eljárás vége.
```



Feladatmegoldási stratégiák
Dinamikus programozás

Mátrix bejárás: a legnagyobb haszon

Bemenet:

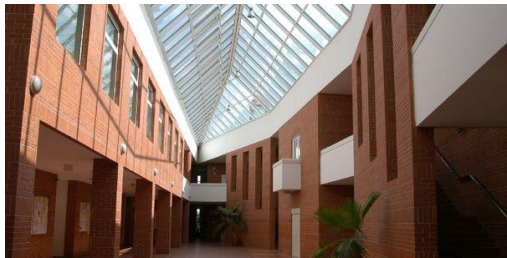
- mátrix jutalomértékekkel $(T(i,j) \geq 0)$

Feltétel:

- a bal felső sarokból indulunk, a jobb alsó sarokba kell eljutnunk
- csak jobbra és lefelé léphetünk

Kimenet:

- Max a maximálisan elérhető haszon
- $K = \{k_1, \ldots k_{N+M-2}\}$ a lépések

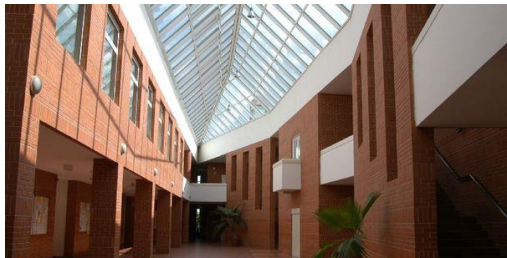


Feladatmegoldási stratégiák
Dinamikus programozás

Mátrix bejárás: a legnagyobb haszon

Részproblémák:

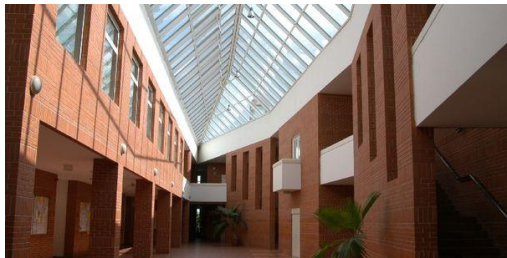
- Mekkora haszon érhető el az (i,j) pontig?
- Az (i,j) pontba az (i-1,j) vagy az (i,j-1) pontból léphetünk.



Feladatmegoldási stratégiák
Dinamikus programozás

Mátrix bejárás: a legnagyobb haszon

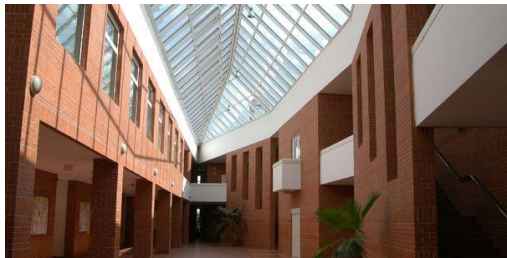
```
Haszon(N,M,T,Max,K):
  H(1,1):=T(1)
  Ciklus i=2-től N-ig
    H(i,1):=H(i-1,1)+T(i,1)
  Ciklus vége
  Ciklus j=2-től M-ig
    H(1,j):=H(1,j-1)+T(1,j)
  Ciklus vége
  ...
```

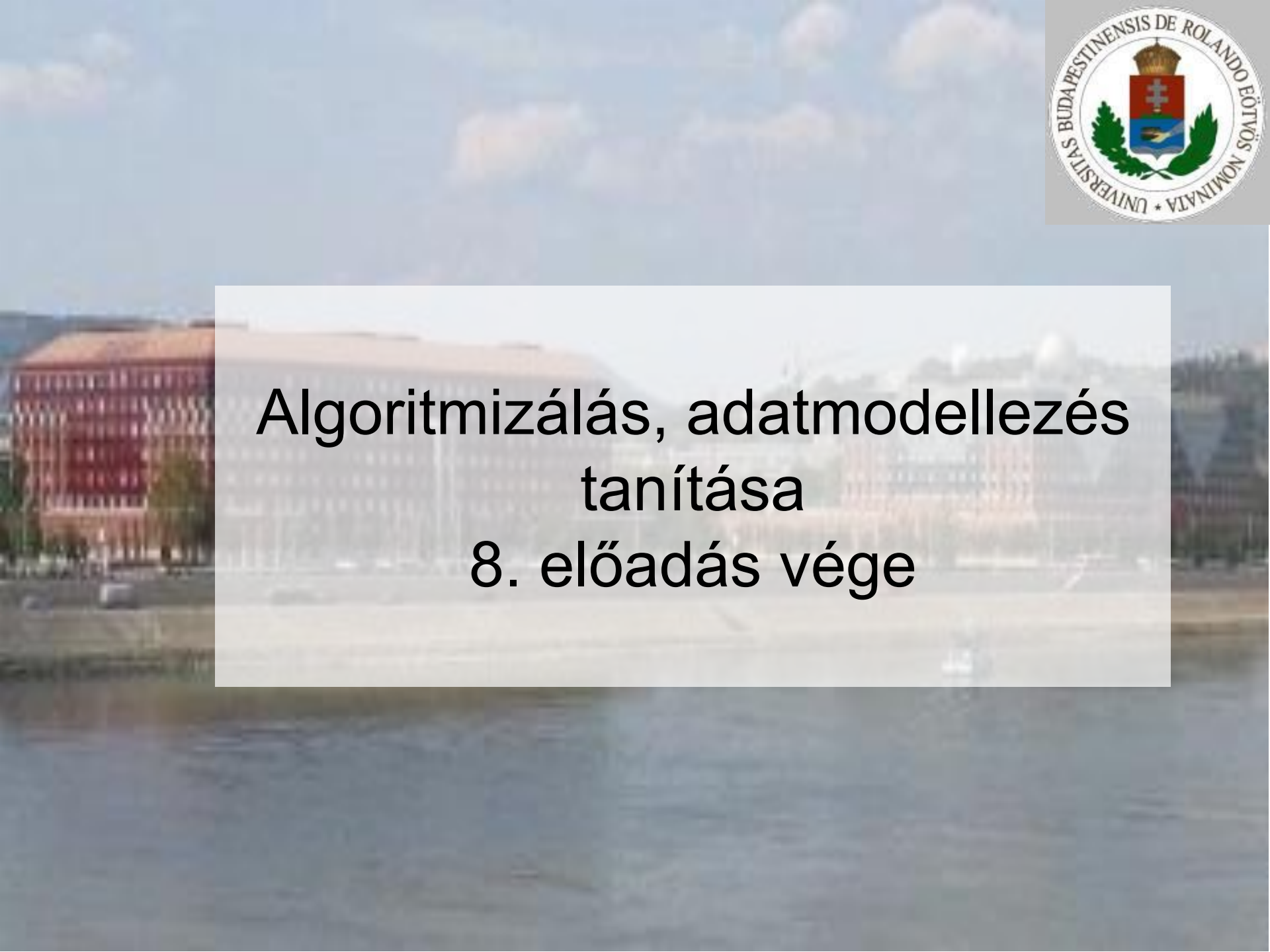


Feladatmegoldási stratégiák
Dinamikus programozás

Mátrix bejárás: a legnagyobb haszon

```
  ...
  Ciklus i=2-től N-ig
    Ciklus j=2-től M-ig
      Ha H(i-1,j)>H(i,j-1)
         akkor H(i,j):=H(i-1,j)+T(i,j)
      különben H(i,j):=H(i,j-1)+T(i,j)
    Ciklus vége
  Ciklus vége
  Max:=H(N,M); Megoldás(K)
Eljárás vége.
```





Algoritmizálás, adatmodellezés tanítása
8. előadás vége