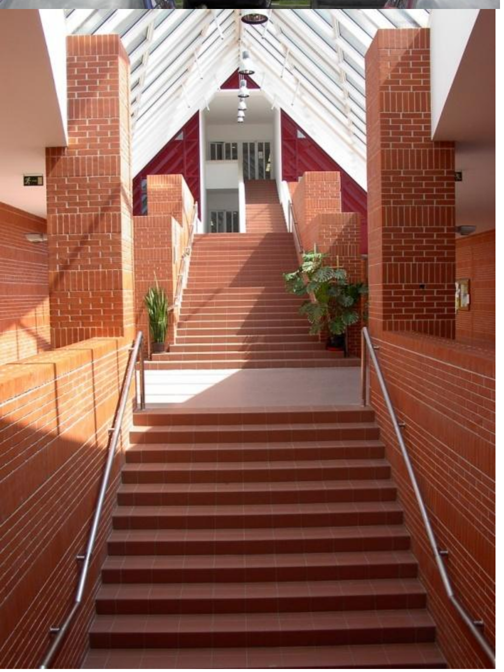
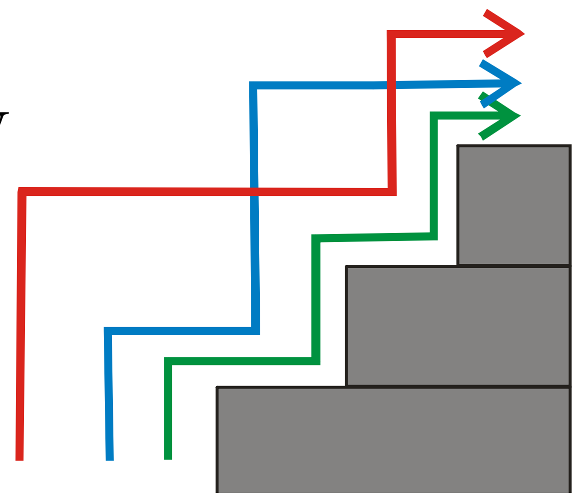
Programozási alapismeretek
13. előadás

Érdekességek - kombinatorika



- Az iskola bejáratánál N lépcsőfok van. Egyszerre maximum K fokot tudunk lépni, ugrani fölfele. Minden nap egyszer megyünk be az iskolába.
- Készíts programot, amely megadja, hogy hány napig tudunk más és más módon feljutni a lépcsőkön!

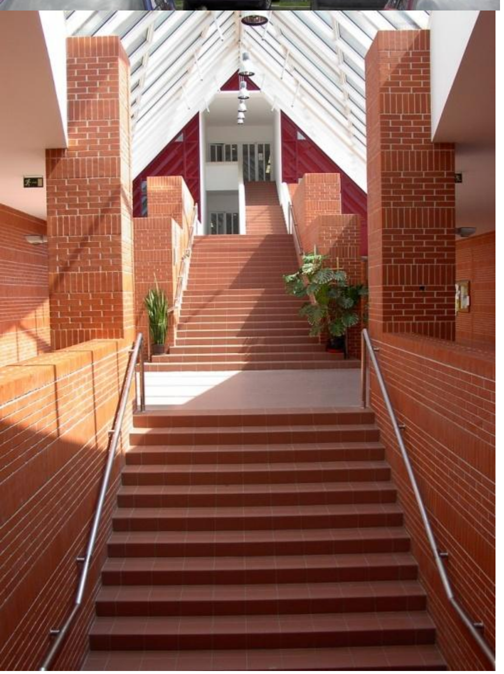


Érdekességek - kombinatorika

- Bemenet: N,K∈Egész
- Kimenet: Db∈Egész
- Előfeltétel: —
- Utófeltétel: ???

A probléma az, hogy nem látszik köz-vetlen összefüggés a bemenet és a kime-net között.

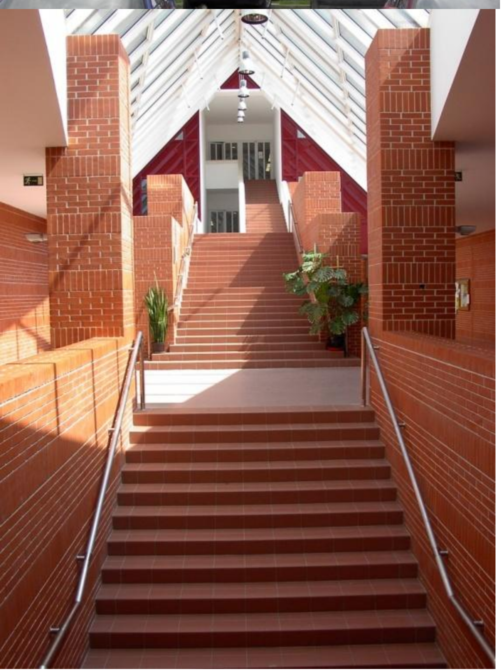
Érdekességek - kombinatorika



Próbáljuk megfogalmazni minden egyes lépcsőfokra, hogy hányféleképpen érhetünk el oda!

- Bemenet: N,K∈Egész
- Kimenet: Db∈Tömb[0..N: Egész]
- Előfeltétel: —
- Utófeltétel: ???

Továbbra sem látszik közvetlen összefüggés a bemenet és a kimenet között.



Érdekességek - kombinatorika

Próbáljunk meg összefüggést felírni a kimenetre önmagában!

Észrevétel: Az N-edik lépcsőfokra vagy az N-1-edikről lépünk, vagy az N-2-edikről, … vagy pedig az N-K-adikról!

- Utófeltétel: Db[0]=1 és

$$\forall j(1 \le j \le N) : Db[j] = \sum_{\substack{i=1 \\ i \le j}}^{K} Db[j-i]$$

Tehát eljutottunk a sorozatszámítás (összegzés) programozási tételhez.

Érdekességek - kombinatorika

Db[0]:=1

j=1..N

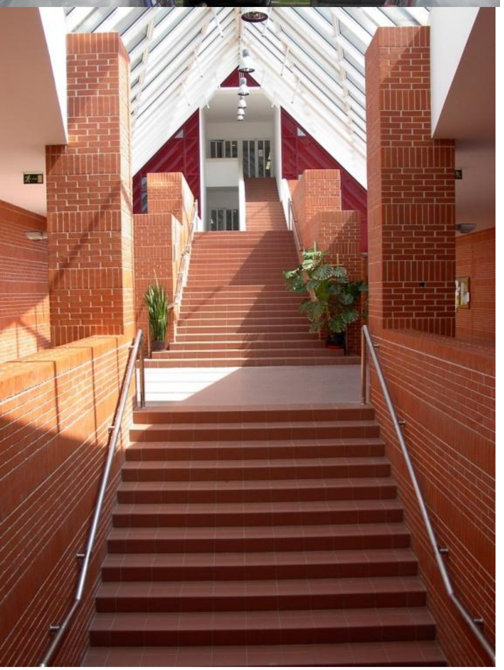
Db[j]:=0

i=1..K

j≥i

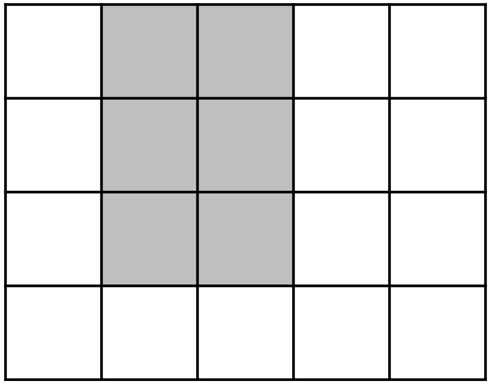
Db[j]:=Db[j]+Db[j-i]	—

```
Db[0]=1;
for(int j=1, j<=N; j++) {
  Db[j]=0;
  for(int i=1; i<=K; i++) {
     if  (j>=i) Db[j]=Db[j]+Db[j-i]  }   }
```



Érdekességek - segédösszegek

- Egy földműves egy téglalap alakú területet szeretne vásárolni egy ***NxM***-es téglalap alakú földterületen. Tudja minden megvásárolható földdarabról, hogy azt megművelve mennyi lenne a haszna vagy vesztesége.
- Add meg azt a tég-lalapot, amelyen a legnagyobb haszon érhető el!

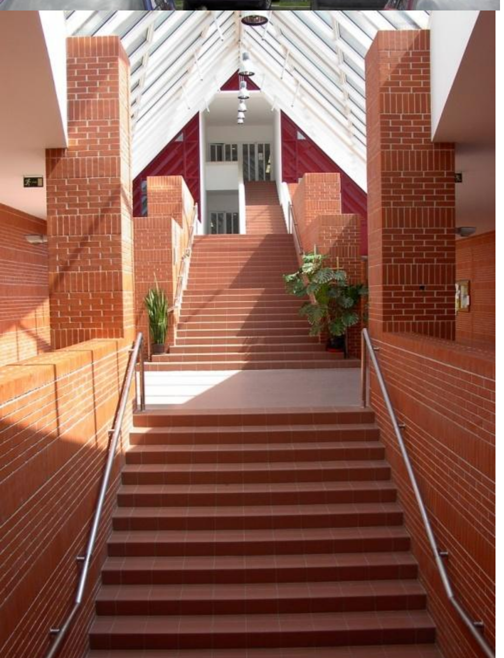


Érdekességek - segédösszegek

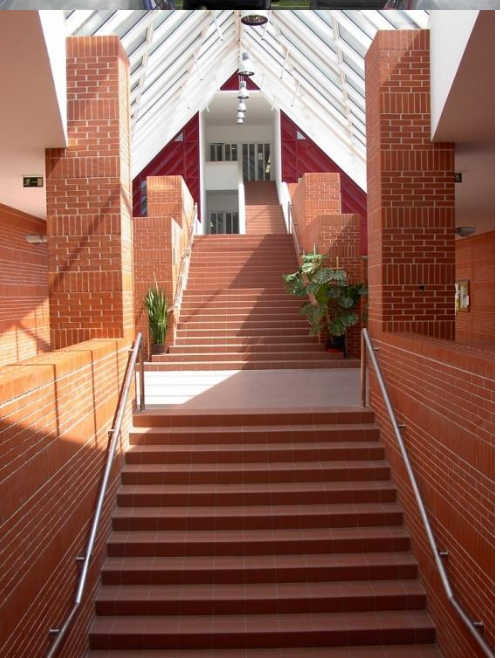
- Bemenet: $N,M \in$ Egész
  $T \in$ Tömb[1..N,1..M: Egész]
- Kimenet: $P,Q,R,S \in$ Egész
- Előfeltétel: —
- Utófeltétel: $1 \leq P \leq R \leq N$ és $1 \leq Q \leq S \leq M$ és $\forall$ i,j,k,l ($1 \leq i \leq k \leq N, 1 \leq j \leq l \leq M$): érték(P,Q,R,S)$\geq$érték(i,j,k,l)

$$\text{érték}(a, b, c, d) = \sum_{x=a}^{c} \sum_{y=b}^{d} T[x, y]$$

Érdekességek - segédösszegek



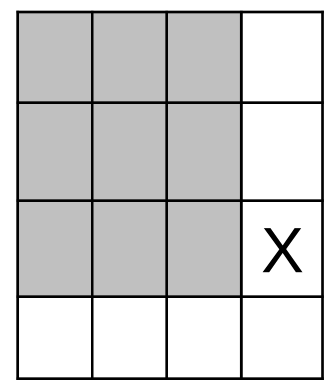
- Az eddigiek alapján az utófeltétel segített a ciklusok meghatározásában.
- Most ciklust kellene írni i-re, j-re, k-ra, l-re, x-re és y-ra, azaz 6 ciklus lenne egymás belsejében. **Ez sok!**



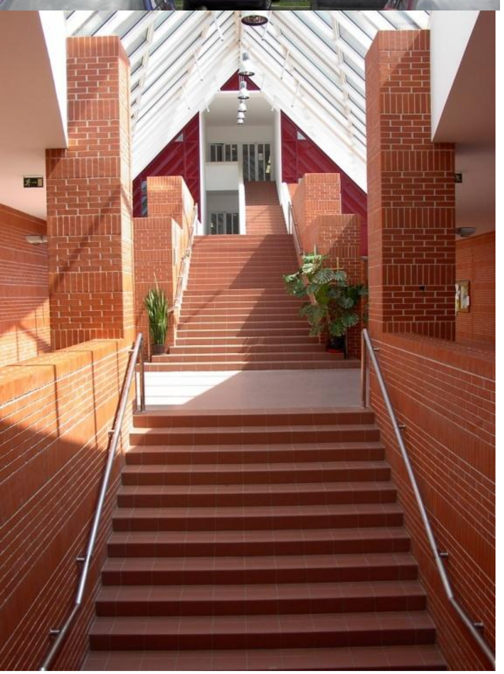
Érdekességek - segédösszegek

- Próbáljunk az előző feladathoz hasonlóan valami részcélt kitűzni: számoljuk ki az (1,1) bal felső (u,v) jobb alsó sarkú téglalapok értékét!

X= a szürke téglalap értéke + az X fölötti számok összege



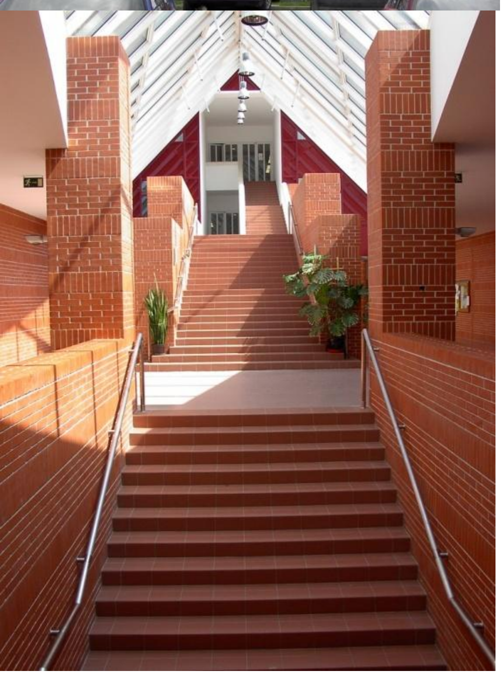
Érdekességek - segédösszegek



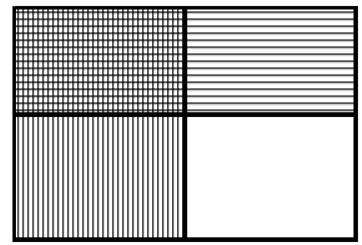
v=1..M		
	x=0	
	u=1..N	
		x=x+T[u,v]
		E[u,v]:=E[u,v-1]+x
…		

```
for(int v=1, v<=M; v++) {
   x=0;
   for(int u=1; u<=M; u++) {
      x=x+T[u][v]; E[u][v]=E[u][v-1]+x
   }
}
```

Érdekességek - segédösszegek



- Definiáljuk E[u,v] segítségével az érték(i,j,u,v)-t!

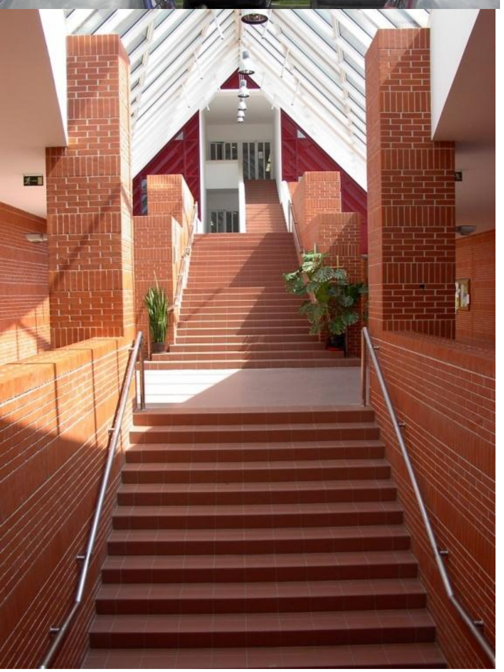


- érték(i,j,u,v)=

  E[u,v]-E[i-1,v]-E[u,j-1]+E[i-1,j-1]

- A módszer neve: kumulatív összegzés.

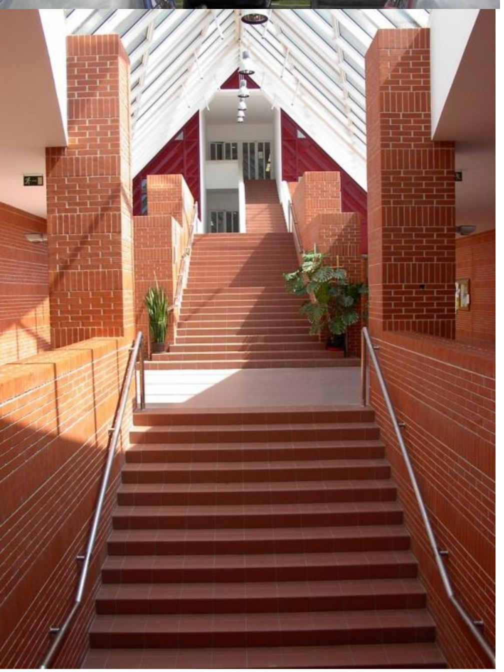
Érdekességek - segédösszegek



P:=1
Q:=1
R:=1
S:=1
Maxért:=T[1,1]
…

```
int P=1; int Q=1;
int R=1; int S=1;
int Maxert=T[1][1];
…
```

Érdekességek - segédösszegek



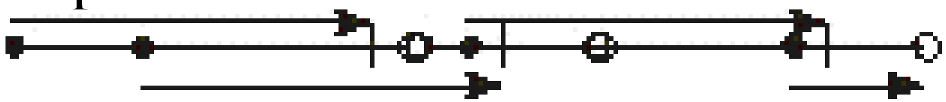
```
…
i=1..N
  j=1..M
    k=i..N
      l=j..M
        értékd(i,j,k,l)>Maxért
          P:=i
          Q:=j
          R:=k
          S:=l
          Maxért:=érték(i,j,k,l)
        —
```

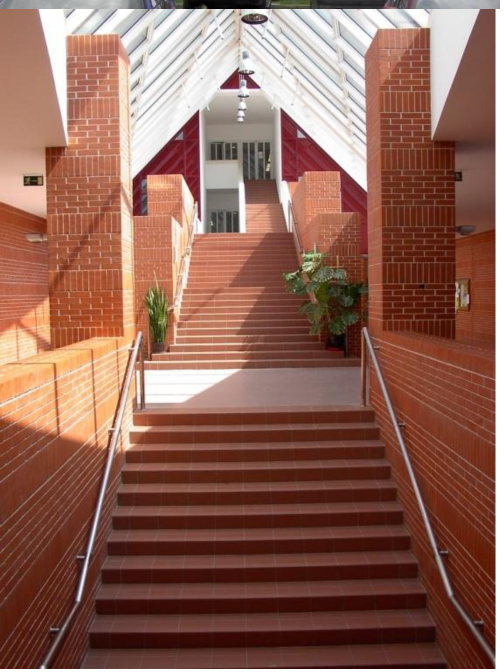
Érdekességek - segédösszegek

```
…
for(int i=1; i<=N; i++) {
 for(int j=1; j<=N; j++) {
  for(int k=1; k<=N; k++) {
   for(int l=1; l<=N; l++) {
    if(E[u][v]-E[i-1][v]-E[u][j-1]+E[i-1][j-1]>Maxert)
    {
       P=i; Q=j; R=k; S=l;
       Maxert=E[u][v]-E[i-1][v]-E[u][j-1]+E[i-1][j-1]
    }
   }
  }
 }
}
```

Érdekességek – Mohó stratégia

- A Budapest-Párizs útvonalon N benzinkút van, az i-edik B_i távolságra Budapesttől (az első Budapesten, az utolsó Párizsban). Egy tankolás az autónak K kilométerre elég.
- Készíts programot, amely megadja a lehető legkevesebb benzinkutat, ahol tankolni kell, úgy, hogy eljuthassunk Budapestről Párizsba!

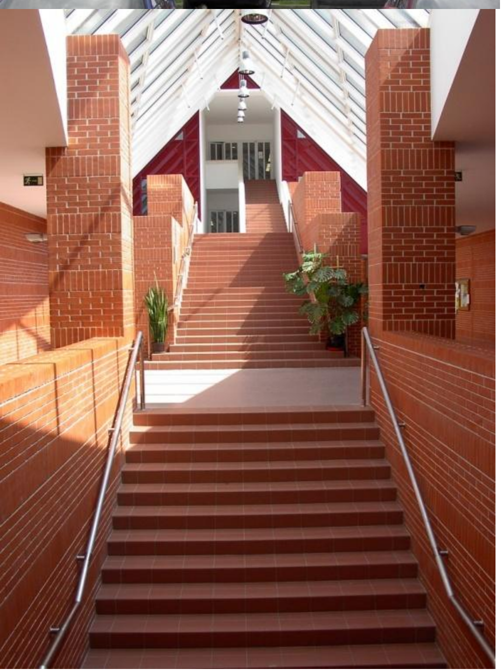




Érdekességek – Mohó stratégia

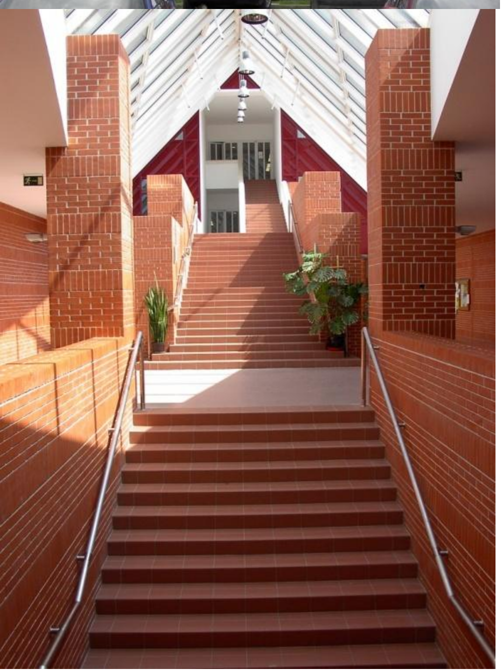
- Bemenet: $N,K \in$ Egész
  $B \in$ Tömb[1..N: Egész]
- Kimenet: $Db \in$ Egész
  $T \in$ Tömb[1..N-1: Egész]
- Előfeltétel: $\forall i(1 \le i < N)$: $B[i+1]-B[i] \le K$
- Utófeltétel: $Db=???$ és $T[1]=1$ és
  $\forall i(1 \le i < Db)$: $B[T[i+1]]-B[T[i]] \le K$
  és $B[N]-B[T[Db]] \le K$ és $T \subseteq (1,\ldots,N-1)$

Érdekességek – Mohó stratégia



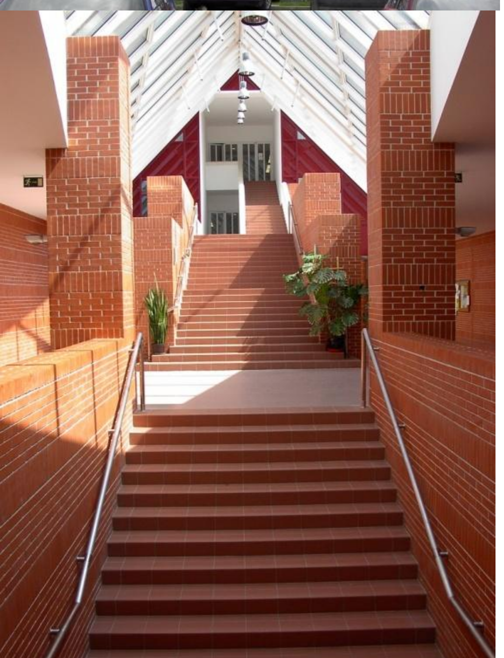
- A megfogalmazásból látható, hogy a tankolási helyek halmaza az összes benzinkút halmazának egy részhalmaza lesz.
- Állítsuk elő az összes részhalmazt, majd válogassuk ki közülük a jókat (amivel el lehet jutni Párizsba), s végül adjuk meg ezek közül a legkisebb elemszámút!
- Probléma: 2^N részhalmaz van!

Érdekességek – Mohó stratégia



A megoldás (tegyük fel, hogy van megoldás):

- Budapesten mindenképpen kell tankolni!
- Menjünk, ameddig csak lehet, s a lehető legutolsó benzinkútnál tankoljunk!
- …
- Belátható, hogy ezzel egy optimális megoldást kapunk.
- **Kiválogatás!**



Érdekességek – Mohó stratégia

Db:=1

T[1]:=1

i=2..N-1

B[i+1]-B[T[Db]]>K

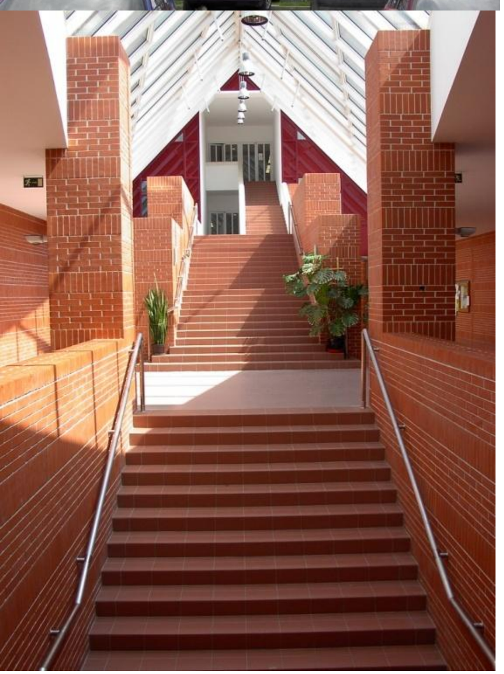
Db:=Db+1 | —

T[Db]:=i

```
Db=1; T[1]=1;
for(int i=2; i<=N; i++){
   if(B[i+1]-B[T[Db]]>K)
      { Db++; T[Db]=i }
}
```

Érdekességek - keverés

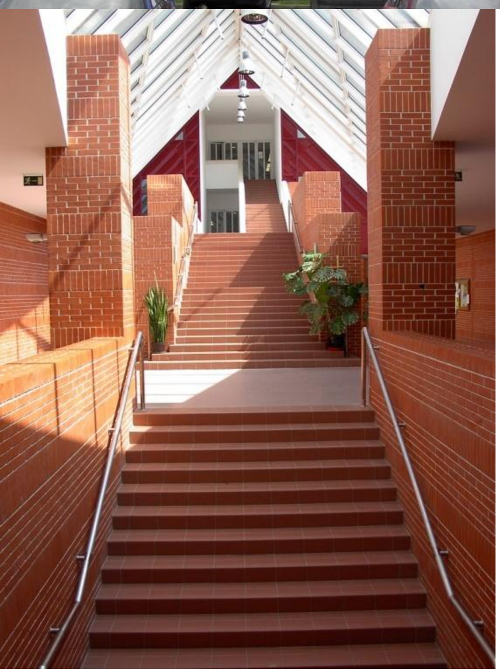
- Van N elemünk (1,2,…,N), keverjük össze őket véletlenszerűen!
- Mit jelent a keverés? Az N elem összes lehetséges sorrendje egyenlő eséllyel álljon elő a keverésnél!
- $(1,2,\ldots,N) \rightarrow (X_1, X_2,\ldots, X_N)$
- (A hamiskártyások egyik trükkje, hogy nem így keverik a kártyákat!)



Érdekességek - keverés

A megoldás elve:

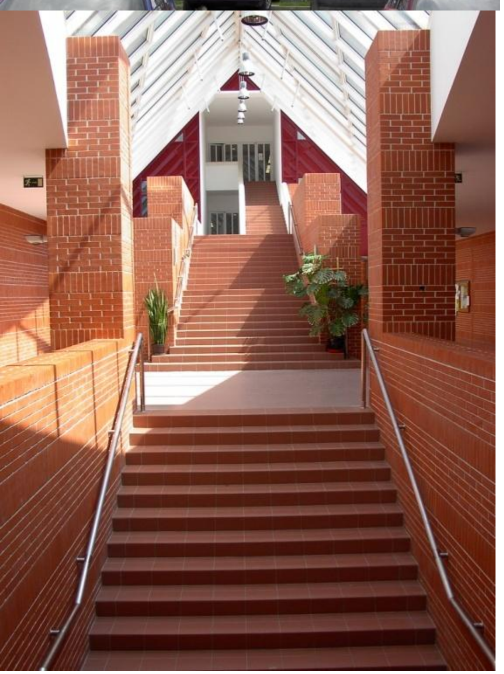
- Válasszunk az N elem közül egyet véletlenszerűen, és cseréljük meg az elsővel! Így az első helyre egyenlő (1/N) eséllyel kerül bármely elem.
- A maradék N-1-ből újra válasszunk véletlenszerűen egyet, s cseréljük meg a másodikkal!



Érdekességek - keverés

Be kellene látnunk, hogy így jó megoldást kapunk! (nem bizonyítás, csak gondolatok)

- Nézzük meg, hogy mi annak az esélye, hogy az I kerül a második helyre!
- Ez úgy történhet, hogy az első helyre nem az I került (esélye (N-1)/N), a másodikra pedig igen (esélye 1/(N-1)).
- Tehát (N-1)/N*1/(N-1)=1/N!

Érdekességek - keverés

i=1..N-1
j:=Véletlen(i..N)
Csere(X[i],X[j])

```
for(int i=1; i<N; i++){
    int j=i+rand() % (N-i+1);
    int y=X[i]; X[i]=X[j]; X[j]=y;
}
```

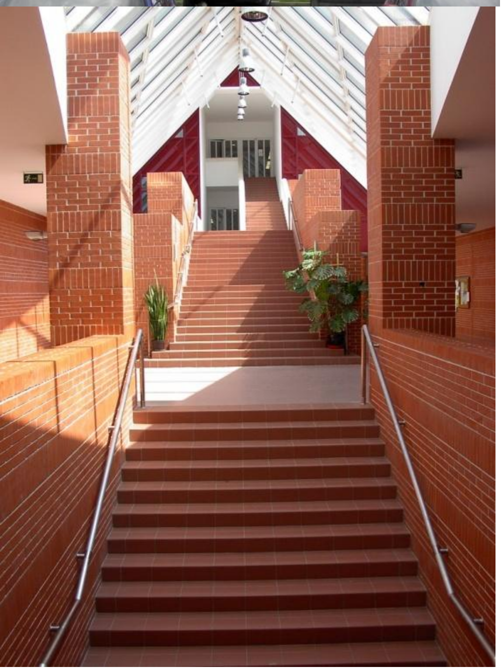
Ez olyan, mint a rendezés, csak nagyság szerinti hely helyett véletlenszerű helyre cserélünk.

Érdekességek – közelítő számítások

- Feladat: Számítsuk ki gyök(2) értékét!
- Probléma: irracionális számot biztosan nem tudunk ábrázolni a számítógépen!
- Új feladat: Számítsuk ki azt a P,Q egész számpárt, amire P/Q elég közel van gyök(2)-höz!
- Probléma: Mi az, hogy „elég közel”?
- Ötlet: $|P^2/Q^2-2|<E$, ahol E egy kicsi pozitív valós szám.

Érdekességek – közelítő számítások

- Bemenet: $E \in$ Valós
- Kimenet: $P,Q \in$ Egész
- Előfeltétel: $E>0$
- Utófeltétel: $|P^2/Q^2-2|<E$
- Probléma: nem látszik egyszerű összefüggés P, Q és E között.
- Ötlet: Állítsunk elő (P_i,Q_i) számpárokat úgy, hogy $|P_{i+1}^2/Q_{i+1}^2-2| < |P_i^2/Q_i^2-2|$ legyen!
- Ha felülről közelítünk, az abszolút érték jel elhagyható!

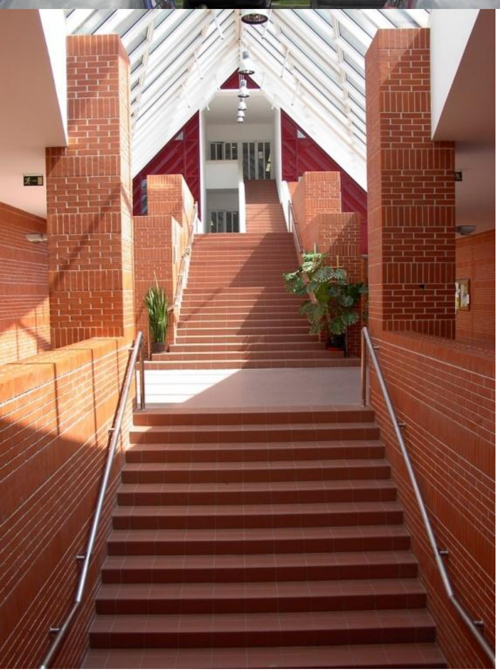



Érdekességek – közelítő számítások

- Állítás: a P^2-$M*Q^2=4$ egyenletnek végtelen sok megoldása van, ha M nem négyzetszám. (Most nem bizonyítjuk.)
- Állítás: az alábbi sorozat értéke gyök(2)-höz tart, ha n tart végtelenhez (most ezt sem bizonyítjuk):

$$x_{n+1} := \frac{1}{2} * \left(x_n + \frac{2}{x_n} \right)$$

- Legyen $x_n=P_n/Q_n$!



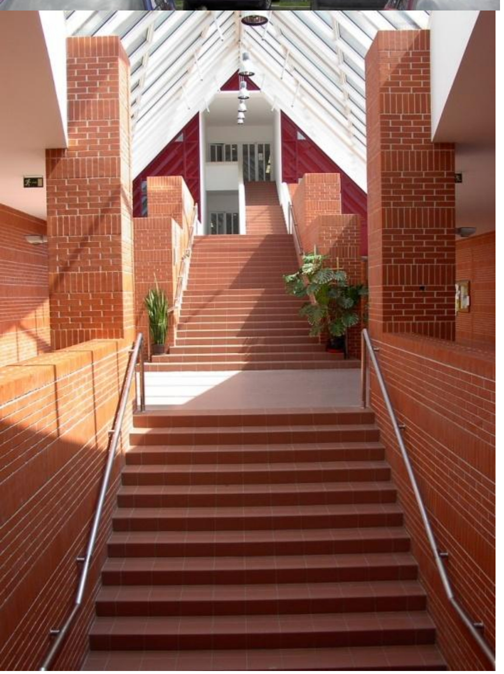
Érdekességek – közelítő számítások

- Legyen (P_n,Q_n) a fenti egyenlet megoldása. Ekkor:

$$x_{n+1} := \frac{1}{2} * \left(x_n + \frac{2}{x_n} \right) = \frac{1}{2} * \left(\frac{P_n}{Q_n} + \frac{2 * Q_n}{P_n} \right) =$$

$$= \frac{1}{2} * \left(\frac{P_n^2 + 2 * Q_n^2}{P_n * Q_n} \right) = \frac{P_n^2 - 2}{P_n * Q_n} = \frac{P_{n+1}}{Q_{n+1}}$$

- Belátható, hogy (P_{n+1},Q_{n+1}) is megoldása az egyenletnek.
- Legyen $P_0=6$, $Q_0=4$, ami megoldása az egyenletnek!



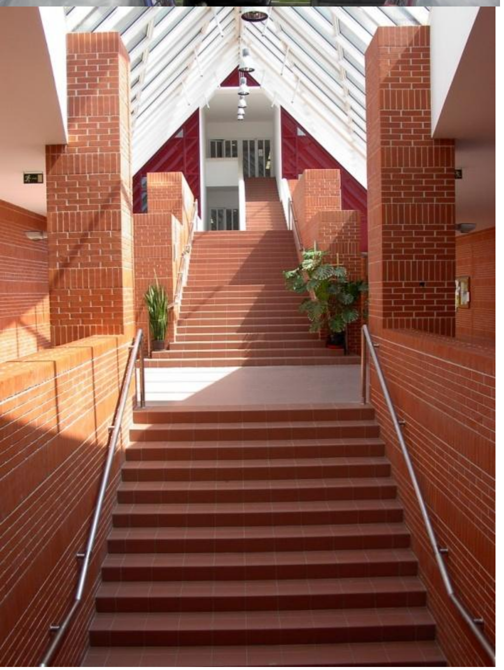
Érdekességek – közelítő számítások

P:=6
Q:=4
P*P-2*Q*Q≥E*Q*Q
Q:=P*Q
P:=P*P-2

```
P=6; Q=4;
while(P*P-2*Q*Q≥E*Q*Q){
    Q=P*Q; P=P*P-2
}
```

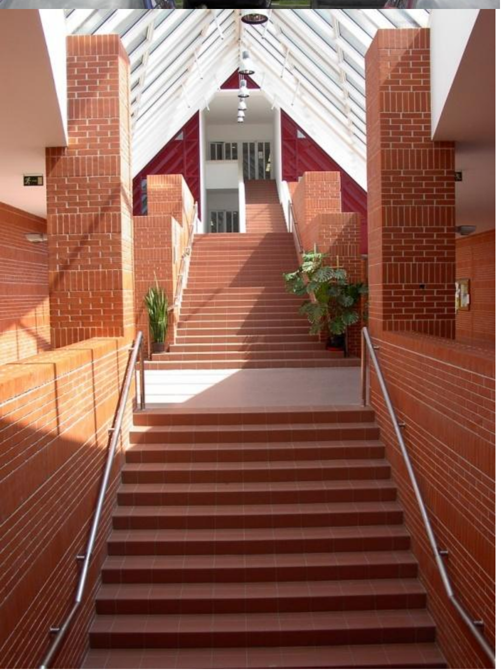
- Most nem foglalkozunk a megoldás lépésszámának vizsgálatával.

Érdekességek – összes (i-edik) permutáció



- Feladat: Állítsuk elő egy N elemű sorozat (1,…,N) összes permutációját!
- Másik feladat: Állítsuk elő egy N elemű sorozat i-edik permutációját ($0 \le i < n!$)!
- Azaz, ha az i-edik permutációt elő tudjuk állítani, akkor abból az összes permutáció egy egyszerű ciklussal kapható meg.

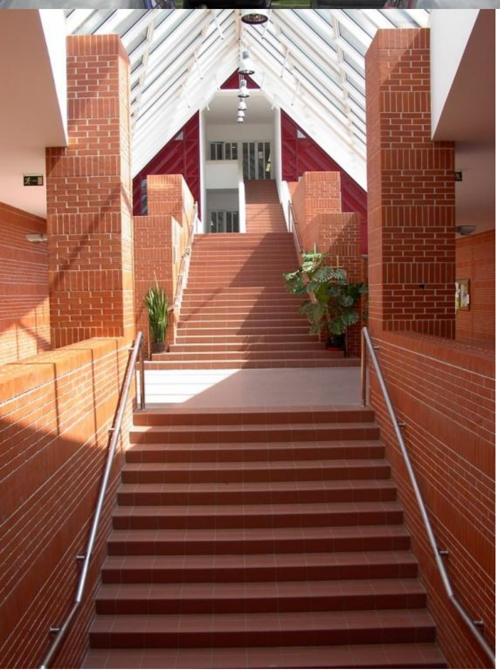
Érdekességek – összes (i-edik) permutáció



- Vegyünk egy rendező módszert!

i=1..N-1		
	Min:=i	
	j=i+1..N	
	X[min]>X[j]	
	Min:=j	—
	Csere(X[i],X[Min])	
	Táv[i]:=Min-i	

- Tároljuk azt, hogy az egyes lépésekben milyen messzire kellett cserélni!



Érdekességek – összes (i-edik) permutáció

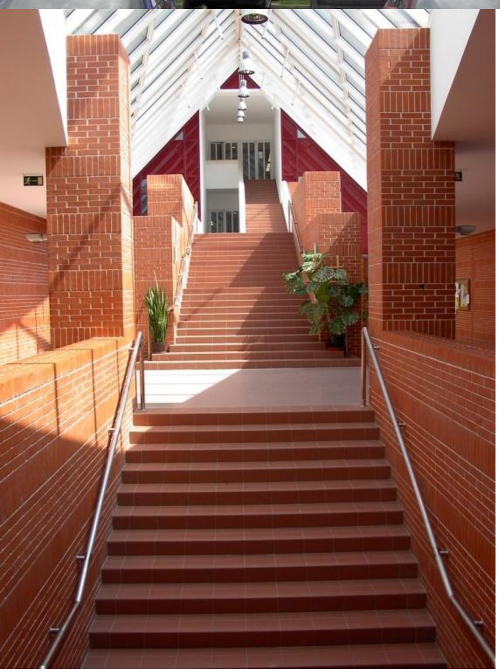
- A Táv vektor alapján a rendezés hatása visszaalakítható!

i=N-1..1, -1esével	
	Csere(X[i],X[i+Táv[i]])

```
for(int i=N-1; i>=1; i--){
  y=X[i]; X[i]=X[i+Tav[i]]; X[i+Tav[i]]=y;
}
```

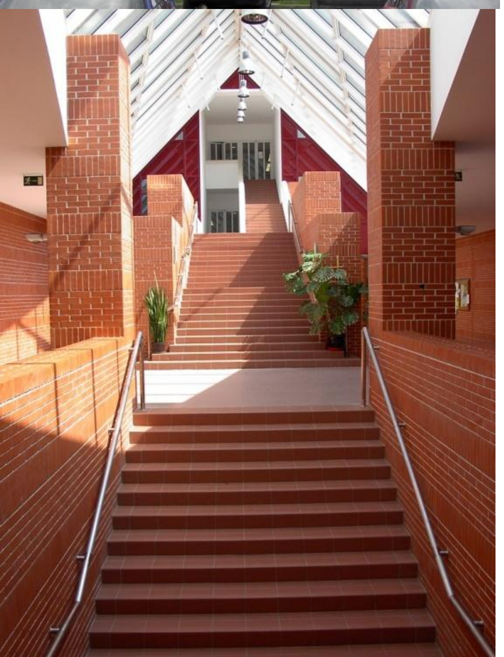
- Belátható, hogy minden permutáció-hoz más és más Táv vektor tartozik.
- Kérdés: hogyan lehet egy i értékhez Táv vektort rendelni?

Érdekességek – összes (i-edik) permutáció



- Táv[N-1] értéke 0 vagy 1.
- Táv[N-2] értéke 0, vagy 1, vagy 2.
- …
- Táv[1] értéke 0, vagy 1, …, vagy N-1.
- Azaz Táv egy N-1 jegyű egész szám egy olyan számrendszerben, aminek helyiértékenként más és más az alapszáma!
- Megoldás: Az i egész szám átírása ebbe a számrendszerbe.

Érdekességek – összes (i-edik) permutáció



j=1..N-1	
	Táv[N-j]:=i mod (j+1)
	i:=i div (j+1)

```
for(int j=1; j<=N-1; j++){
    Tav[N-j]=i % (j+1);
    i=i/(j+1);
}
```

- A fenti programrészt összeépítve a rendezés visszaalakításánál készítettel megadtuk az i-edik permutáció elő-állításának algoritmusát.

Programozási alapismeretek
13. előadás vége