# Dinamikus programozás II.

# Dinamikus programozás stratégiája

## A dinamikus programozás stratégiája

1. Az [optimális] megoldás szerkezetének tanulmányozása.
2. Részproblémákra és összetevőkre bontás úgy, hogy:
   - az összetevőktől való függés körmentes legyen;
   - minden részprobléma [optimális] megoldása kifejezhető legyen (rekurzívan) az összetevők [optimális] megoldásaival.
3. Részproblémák [optimális] megoldásának kifejezése (rekurzívan) az összetevők [optimális] megoldásaiból.

# Dinamikus programozás stratégiája

4. Részproblémák [optimális] megoldásának kiszámítása alulról-felfelé haladva:

- A részproblémák kiszámítási sorrendjének meghatározása. Olyan sorba kell rakni a részproblémákat, hogy minden p részprobléma minden összetevője (ha van) előbb szerepeljen a felsorolásban, mint p.
- A részproblémák kiszámítása alulról-felfelé haladva, azaz táblázatkitöltéssel.

5. Egy [optimális] megoldás előállítása a 4. lépésben kiszámított (és tárolt) információkból.

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyekkel kifizethető-e F forint?

**Megoldás:**

Tegyük fel, hogy $F = P_{i_1} + P_{i_2} + \ldots + P_{i_m} \left(i_1 < \ldots < i_m\right)$ !

Ekkor $F - P_{i_m} = P_{i_1} + P_{i_2} + \ldots + P_{i_{m-1}} \left(i_1 < \ldots < i_m\right)$, azaz az adott részproblémának megoldása a megoldás megfelelő részlete.

Ezek alapján számoljuk ki V(X,i) értékeket, ahol V(X,i) igaz, ha az X összeg kifizethető az első i pénzjeggyel!

*Az előadás Horváth Gyula tananyagai felhasználásával készült.*

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyekkel kifizethető-e F forint?

**Megoldás:**

Három esetet vizsgálunk: az összeg megegyezik az i-edik pénzjeggyel; az i-edik pénzjegyet nem használjuk fel; az i-edik pénzjegyet felhasználjuk (és a maradék kifizethető):

$$V(X,i) = igaz \quad \Leftrightarrow \quad \begin{cases} P_i = x \\ i > 1 \quad és \quad V(X, i-1) \\ i > 1 \quad és \quad X \geq P_i \quad és \quad V(X - P_i, i-1) \end{cases}$$

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyekkel kifizethető-e F forint?

**Megoldás:**

```
V(X,i):
   V:=X=P(i) vagy
       i>1 és V(X,i-1) vagy
       i>1 és X>P(i) és V(X-P(i),i-1)
Függvény vége.
```

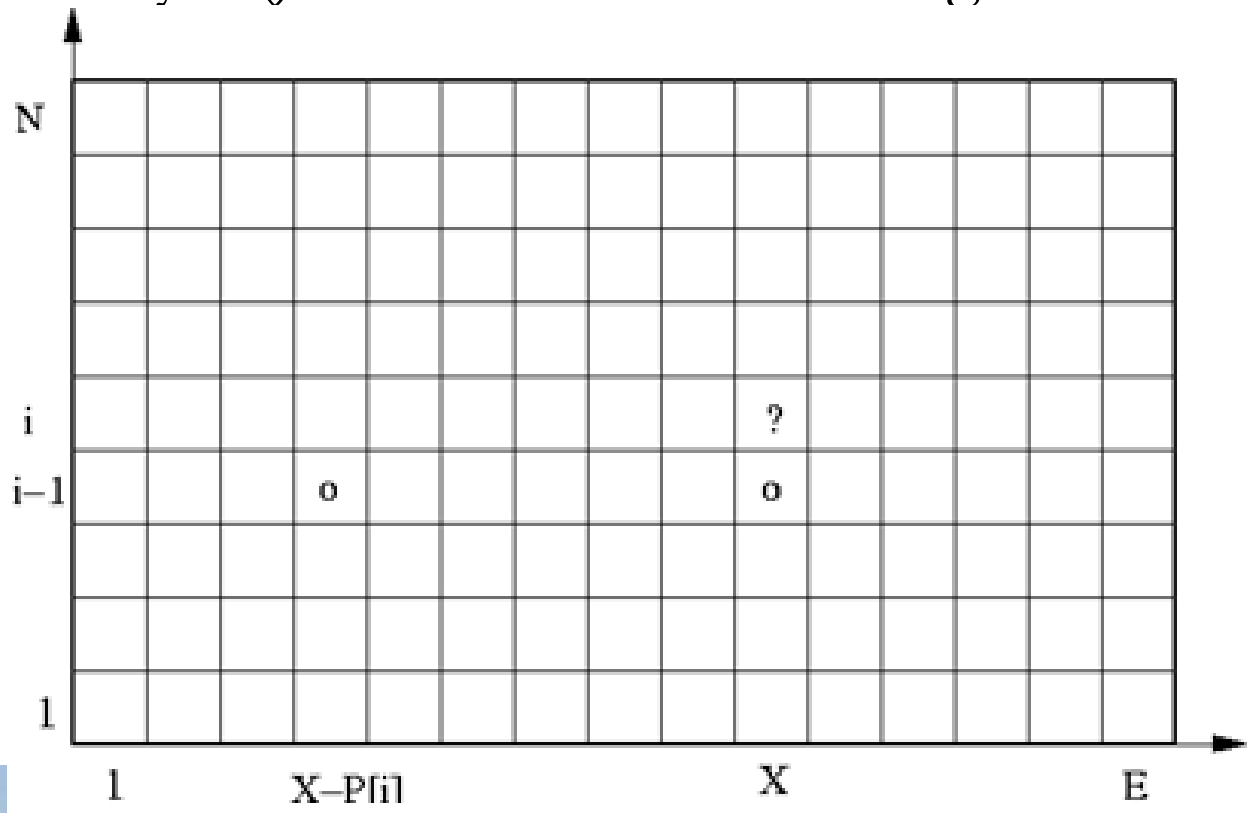
A megoldás: V(F,N) kiszámolása.

Probléma: a futási idő $O(2^N)$

# Dinamikus programozás – Pénzváltás

V(X,i) kiszámításához mely V() értékekre lehet szükség?

Az ábra alapján V értékei alulról felfelé, azon belül balról jobbra számolhatók.




# Dinamikus programozás – Pénzváltás

```
Pénzváltás(Lehet):
  V(1..F,1):=hamis
  Ha P(1)≤F akkor V(P(1),1):=igaz
  Ciklus i=2-től N-ig
    Ciklus X=1-től F-ig
      V(X,i):=X=P(i) vagy
              V(X,i-1) vagy
              X>P(i) és V(X-P(i),i-1)
    Ciklus vége
  Ciklus vége
  Lehet:=V(F,N)
Függvény vége.
```

A futási idő O(N*F)

Memóriaigény: N*F

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül melyekkel fizethető ki F forint?

**Megoldás:**

Legyen V(X,i) a legutolsó olyan k index, amelyre igaz, hogy $P_k$ előfordul X felváltásában!

Legyen V(X,i)=N+1, ha X nem váltható fel az első i pénz-jeggyel!

# Dinamikus programozás – Pénzváltás

## Feladat:

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül melyekkel fizethető ki F forint?

## Megoldás:

$$V(X,i) = \begin{cases} N+1 & ha \quad i = 0 \quad és \quad X > 0 \\ 0 & ha \quad X = 0 \\ V(X,i-1) & ha \quad V(X,i-1) \le N \\ i & ha \quad X \ge P_i \quad és \quad V(X-P_i,i-1) \le N \\ N+1 & egyébként \end{cases}$$

# Dinamikus programozás – Pénzváltás

```
Pénzváltás:
  V(1..F,0):=N+1; V(0,0):=0
  Ciklus i=1-től N-ig
    V(0,i):=0
    Ciklus X=1-től F-ig
      Ha V(X,i-1)≤N akkor V(X,i):=V(X,i-1)
      különben ha X≥P(i) és V(X-P(i),i-1)≤N
                    akkor V(X,i):=i
      különben V(X,i):=N+1
    Ciklus vége
  Ciklus vége
  …
```

A V mátrix kiszámolása.

# Dinamikus programozás – Pénzváltás

Csak akkor van megoldás, ha $V(F,N) \leq N$ – azaz az előző algoritmust ezzel helyettesíthetjük.

Ekkor $k=V(F,N)$ olyan pénz indexe, hogy $F-P(k)$ felváltható az első $k-1$ pénzzel.

Tehát a $V(F-P(k),k-1)$ probléma megoldásával kell folytatnunk, mindaddig, amíg a maradék pénz 0 nem lesz.

# Dinamikus programozás – Pénzváltás

```
  …
  Db:=0; X:=F; i:=N
  Ha V(F,N)≤N
     akkor Ciklus
              i:=V(x,i)    {i. pénz felhasználva}
              Db:=Db+1; A(Db):=i
              X:=X-P(i) i:=i-1
           amíg X≠0
           Ciklus vége
Eljárás vége.
```

A megoldás előállítása.

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül a lehető legkevesebbet használva mennyivel fizethető ki F forint?

**Megoldás:**

Tegyük fel, hogy $F = P_{i_1} + P_{i_2} + \ldots + P_{i_m} \left(i_1 < \ldots < i_m\right)$ optimális felbontás!

Ekkor $F - P_{i_m} = P_{i_1} + P_{i_2} + \ldots + P_{i_{m-1}} \left(i_1 < \ldots < i_m\right)$ is optimális, azaz az adott részproblémának megoldása a megoldás megfelelő részlete.

# Dinamikus programozás – Pénzváltás

## Feladat:

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül a lehető legkevesebbet használva mennyivel fizethető ki F forint?

## Megoldás:

$$O(X,i)=\begin{cases} N+1 & ha \quad i=0 \quad és \quad X>0 \\ 0 & ha \quad X=0 \\ O(X,i-1) & ha \quad i>0 \quad és \quad X<P_i \\ \min(O(X,i-1),1+O(X-P_i,i-1)) & ha \quad i>0 \quad és \quad X\geq P_i \end{cases}$$

Ha az i. pénzjegy felhasználható, akkor meg kell nézni, hogy mi van vele, illetve nélküle!

# Dinamikus programozás – Pénzváltás

```
Pénzváltás(Minimum):
  O(1..F,0):=N+1; O(0,0):=0
  Ciklus i=1-től N-ig
    O(0,i):=0
    Ciklus X=1-től F-ig
      Ha X<P(i) akkor O(X,i):=O(X,i-1)
      különben O(X,i):=min(O(X,i-1),
                           1+O(X-P(i),i-1))
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

# Dinamikus programozás – Pénzváltás

**Feladat:**

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül a lehető legkevesebbet használva melyekkel fizethető ki F forint?

**Megoldás:**

Tegyük fel, hogy $F = P_{i_1} + P_{i_2} + \ldots + P_{i_m} \left(i_1 < \ldots < i_m\right)$ optimális felbontás!

Ekkor $F - P_{i_m} = P_{i_1} + P_{i_2} + \ldots + P_{i_{m-1}} \left(i_1 < \ldots < i_m\right)$ is optimális, azaz az adott részproblémának megoldása a megoldás megfelelő részlete.

# Dinamikus programozás – Pénzváltás

## Feladat:

Adott $P_1, P_2, \ldots P_n$ pénzjegyek közül a lehető legkevesebbet használva melyekkel fizethető ki F forint?

## Megoldás:

$$V(X,i)=\begin{cases} N+1 & ha \quad i=0 \quad és \quad X>0 \\ 0 & ha \quad X=0 \\ i & ha \quad X \geq P_i \quad és \quad O(X,i)>1+O(X-P_i,i-1) \\ V(X,i-1) & egyébként \end{cases}$$

# Dinamikus programozás – Pénzváltás

```
Pénzváltás(O,V):
  O(1..F):=N+1; O(0):=0; V(1..F,0):=N+1
  Ciklus i=1-től N-ig
    O(0):=0; V(0,i):=0
    Ciklus X=F-től 1-ig -1-esével
      Ha X≥P(i) akkor S:=O(X-P(i))+1
                különben S:=N+1
      Ha S<O(X) akkor O(X):=S; V(X,i):=i
                különben V(X,i):=V(X,i-1)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

# Dinamikus programozás

## Feladat:

Két testvér adott értékű ajándékokon osztozkodik. Minden egyes ajándékot pontosan ez egyik testvérnek kell adni. Készíts testvéries osztozkodást, azaz olyat, amelyben a két testvér által kapott ajándékok értéke különbségének abszolút értéke minimális!

## Megoldás:

Ha az ajándékok összértéke E, akkor a feladat megoldása a legnagyobb olyan $A \leq E/2$ szám, amely érték az ajándékok értékeiből kétféleképpen is előállítható.

Tehát a megoldás visszavezethető a pénzváltás probléma megoldására.

# Dinamikus programozás – Hátizsák feladat

## Feladat:

Adott N tárgy (értékük $F_i$, súlyuk $S_i$). Egy hátizsákba maximum H súlyú tárgyat pakolhatunk. Mennyi a maximális elszállítható érték?

## Megoldás:

Tegyük fel, hogy $H \geq S_{i_1} + S_{i_2} + \ldots + S_{i_m} \; (i_1 < \ldots < i_m)$ optimális megoldás, azaz $F_{i_1} + F_{i_2} + \ldots + F_{i_m}$ maximális!

Ekkor $H - S_{i_m} \geq S_{i_1} + S_{i_2} + \ldots + S_{i_{m-1}} \; (i_1 < \ldots < i_m)$ is optimális, azaz az adott részproblémának megoldása a megoldás megfelelő részlete.

Emlékeztető: a mohó stratégia nem működött rá!

# Dinamikus programozás – Hátizsák feladat

**Feladat:**

Adott N tárgy (értékük $F_i$, súlyuk $S_i$). Egy hátizsákba maximum H súlyú tárgyat pakolhatunk. Mennyi a maximális elszállítható érték?

**Megoldás:** Jelölje E(X,i) a legnagyobb elszállítható értéket, amely az első i tárgyból egy X kapacitású hátizsákba bepakolható.

$$E(X,i)=\begin{cases} F_1 & \text{ha } i=1 \text{ és } S_1 \le X \\ 0 & \text{ha } i=1 \text{ és } S_1 > X \\ E(X,i-1) & \text{ha } i>1 \text{ és } S_i > X \\ \max(E(X,i-1), F_i + E(X-S_i,i-1)) & \text{egyébként} \end{cases}$$

# Dinamikus programozás

```
Hátizsák():
  E(0..S(1)-1,1):=0; E(S(1)..H,1):=F(1)
  Ciklus i=2-től N-ig
    Ciklus X=0-tól H-ig
      E(X,i):=E(X,i-1)
      Ha S(i)≤X és E(X,i)<E(X-S(i),i-1)+F(i)
         akkor E(X,i):=E(X-S(i),i-1)+F(i)
    Ciklus vége
  Ciklus vége
  Kiírás
Eljárás vége.
```

# Dinamikus programozás

```
Kiírás:
  i:=N; X:=H
  Ciklus amíg E(X,i)>0
    Ciklus amíg i>1 és E(X,i)=E(X,i-1)
      i:=i-1
    Ciklus vége
    Ki: i, X:=X-S(i); i:=i-1
  Ciklus vége
Eljárás vége.
```

Ha E(X,i)=E(X,i-1), akkor az i. tárgy nem kell a hátizsákba!

# Dinamikus programozás – Hátizsák típusú feladatok

**Feladat:**

Adott N tárgy (értékük $F_i$, súlyuk $S_i$). Egy hátizsákba maximum H súlyú tárgyat pakolhatunk. Mennyi a maximális elszállítható érték?

A. Mindegyikből legfeljebb $F_i$ darabot.

B. Mindegyikből legalább $A_i$ darabot.

C. A hátizsáknak pontosan tele kell lenni (pénzfelváltás).

D. Minimális elszállítható érték.

# Dinamikus programozás – Tükörszó

**Feladat:**

Egy szóba minimálisan hány karaktert kell beszúrni, hogy tükörszó legyen belőle (azaz ugyanaz legyen balról-jobbra és jobbról balra olvasva is)!

Tetszőleges S szóra az $SS^T$ szó biztos tükörszó, tehát a feladatnak kell legyen megoldása!

**Definíció:**

S tükörszó, ha üres vagy egybetűs. S tükörszó, ha a két szélső betűje megegyezik és a közötte levő rész tükörszó!

# Dinamikus programozás – Tükörszó

**Megoldás:**

Az egybetűs szavak biztosan tükörszavak.

Az azonos kezdő- és végbetűjű szavak tükörszóvá alakításához elég a középső rész átalakítása.

Ha az első és az utolsó betű különbözik, akkor két lehetőségünk van a tükörszóvá alakításra:

- az első betűt szúrjuk be a szó végére;
- az utolsó betűt szúrjuk be a szó elejére.

# Dinamikus programozás – Tükörszó

**Megoldás:**

Jelölje M(i,j), hogy minimum hány karaktert kell beszúrni, hogy a szöveg i. és j. karaktere közötti részét tükörszóvá alakítsuk!

Három eset van:

- egybetűs részek
- a két szélső karakter egyforma
- az első és az utolsó karakter különbözik

$$M(i,j)=\begin{cases} 0 & ha \quad i \geq j \\ M(i+1,j-1) & ha \quad i<j \quad és \quad S_i = S_j \\ 1+\min(M(i+1,j),M(i,j-1)) & ha \quad i<j \quad és \quad S_i \neq S_j \end{cases}$$

Probléma: a rekurzió nem hatékony!

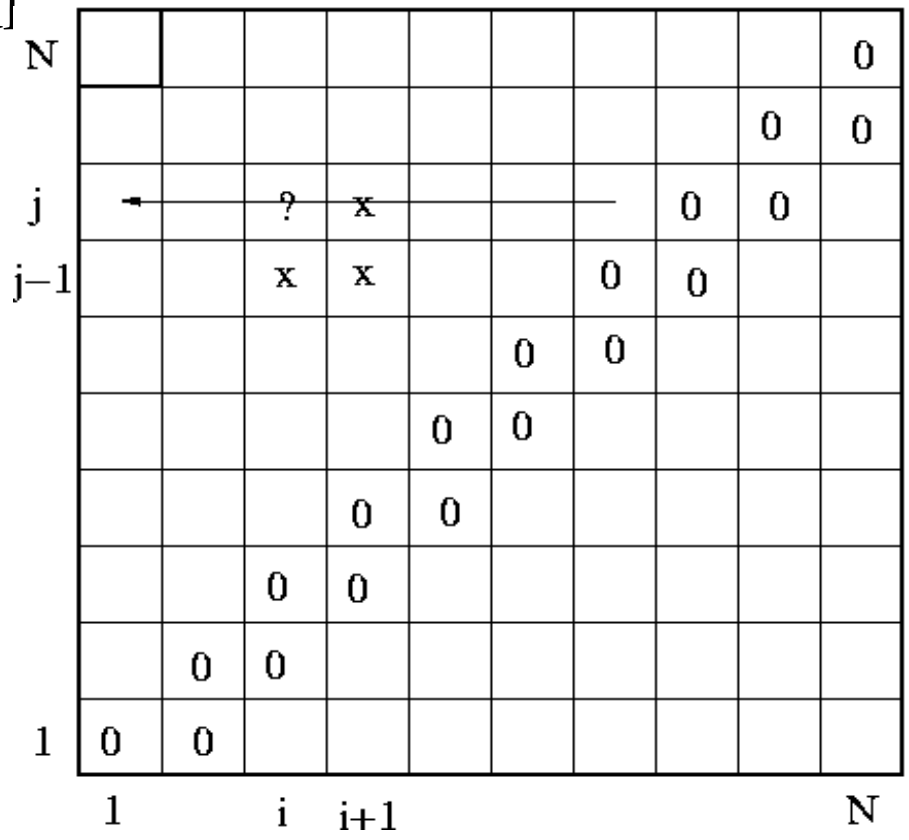
# Dinamikus programozás – Tükörszó

Jó sorrendű táblázatkitöltés kell:

Alulról felfelé, azon belül jobbról balra töltjük ki.

Itt is elég egyetlen sort tárolni a táblázatból (T(j)=M(i,j))?

Egy sor és egy elem kell a táblázatból.




# Dinamikus programozás – Tükörszó

```
Tükörszó(S,M):
  M() kezdő feltöltése
  Ciklus j=2-től N-ig
    M(j,j):=0
    Ciklus i=j-1-től 1-ig -1-esével
      Ha S(i)=S(j) akkor M(i,j):=M(i+1,j-1)
        különben M(i,j):=1+Min(M(i+1,j),M(i,j-1))
    Ciklus vége
  Ciklus vége
  Tükörszó:=M(1,N)
Függvény vége.
```

# Dinamikus programozás – Tükörszó

```
Tükörszó kiírása:
  Ha Tükörszó(S,E)
     akkor i:=1; j:=N
           Ciklus amíg i<j
             Ha S(i)=S(j) akkor i:=i+1; j:=j-1
             különben Ha M(i,j)=M(i+1,j)+1
             akkor {S(i) a j. betű mögé}
             különben {S(j) az i. betű elé}
           Ciklus vége
Eljárás vége.
```

# Dinamikus programozás stratégiája

## A dinamikus programozás stratégiája

1. Az [optimális] megoldás szerkezetének tanulmányozása.
2. Részproblémákra és összetevőkre bontás úgy, hogy:
   - az összetevőktől való függés körmentes legyen;
   - minden részprobléma [optimális] megoldása kifejezhető legyen (rekurzívan) az összetevők [optimális] megoldásaival.
3. Részproblémák [optimális] megoldásának kifejezése (rekurzívan) az összetevők [optimális] megoldásaiból.

# Dinamikus programozás stratégiája

4. Részproblémák [optimális] megoldásának kiszámítása alulról-felfelé haladva:

- A részproblémák kiszámítási sorrendjének meghatározása. Olyan sorba kell rakni a részproblémákat, hogy minden p részprobléma minden összetevője (ha van) előbb szerepeljen a felsorolásban, mint p.
- A részproblémák kiszámítása alulról-felfelé haladva, azaz táblázatkitöltéssel.

5. Egy [optimális] megoldás előállítása a 4. lépésben kiszámított (és tárolt) információkból.