

Dinamikus programozás alkalmazása mátrixszorzás sorrendjének optimalizálására

Dinamikus programozás

- Oszd meg és uralkodj: független részproblémák
- Dinamikus programozás esetén nem függetlenek. (Az Oszd meg és uralkodj feleslegesen többet dolgozna)

Jellemzése:

A dinamikus programozás éppúgy, mint az oszd meg és uralkodj módszer, a feladatot részfeladatokra való osztással oldja meg.

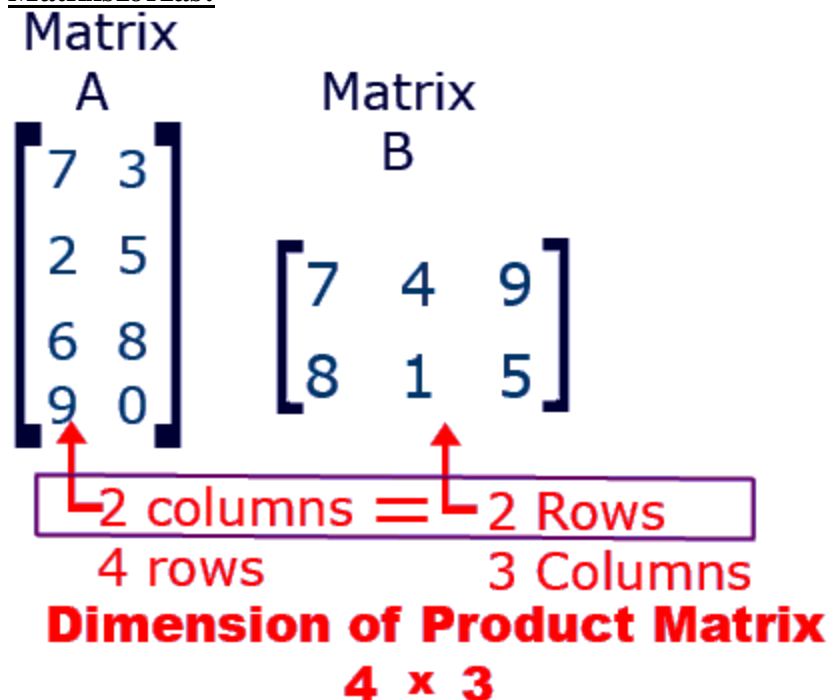
A feladatot felosztja részfeladatokra, és a részfeladat eredményét egy táblázatban tárolja el, és ezáltal elkerüli az ismételt számítást, ha a részfeladat megint felmerül.

A dinamikus programozást optimalizálási feladatok megoldására használjuk.

A dinamikus programozás lépései:

1. részproblémákra bontás, optimális megoldás szerkezete
2. a részproblémák közötti összefüggések meghatározása (rekurzív képlettel)
3. a részproblémák optimális értékének kiszámítása alulról felfelé haladva (táblázat kitöltése)
4. az optimális megoldás megalkotása

Mátrixszorzás:



Dinamikus programozási példa: Mátrixsorozat összeszorozása

- Adott n mátrix (A_1, A_2, \dots, A_n) és ezek $A_1 * \dots * A_2 * A_n$ szorzatát akarjuk kiszámítani.
- Mátrixok szokásos szorzása + zárójelezés
- Mátrixok egy szorzata teljesen zárójelezett, ha vagy egyetlen mátrix, vagy két zárójelbe rakott teljesen zárójelezett mátrixszorzat szorzata.
- Asszociatív, bármilyen zárójelezés ugyanazt az eredményt adja.
($A_1(A_2(A_3A_4))$),
($A_1((A_2A_3)A_4)$),

$((A_1 A_2)(A_3 A_4))$,
 $((A_1 (A_2 A_3))A_4)$,
 $((A_1 A_2)A_3)A_4$

- Zárójelzés alapvetően befolyásolja a kifejezés kiértékelésének költségét.

– a szorzótényezők sor-oszlop kompatibilisek kell, hogy legyenek

Számítási idő: Egy $p \times q$ és egy $q \times r$ dimenziós mátrix összeszorzásának a költsége $p \cdot q \cdot r$.

Mátrixszorzás

- Melyik az oszlop és melyik a sor?

- Közös dimenziójú oszlop- ill. sorpárok elemeit összeszorozzuk és összeadjuk. Az eredmény: sorok száma az elsőből, oszlopok száma a másodikból

Mátrixszorzat(A, B)

if oszlop(A) <> sor(B) then error

else for i=1 to sor(A)

do for j=1 to oszlop(B)

do C[i, j]=0

for k=1 to oszlop(A)

do C[i, j]=C[i, j]+A[i, k]*B[k, j]

return C

Szorások számának becslése az összes lehetőség végigvizsgálásával?

- Exponenciális algoritmus!!
- Egyetlen újabb mátrixtényező hozzávétele a lehetőségek számát legalább megkétszerezi (bár nem lenne muszáj mindent újra kiszámítani)
- $M_1 \cdot M_2 \dots M_n \rightarrow P_n$, akkor $P_{n+1} > P_n * 2$
- $P(1)=1$; $P(n) = \sum_{k=0}^{n-1} P(k) * P(n-k)$, ha $n \geq 2$
- Észrevétel: $(M_1 M_2 \dots M_k) * (M_{k+1} \dots M_n)$ Ha a teljes szorzat optimális, akkor a részszorzatainak is optimálisnak kell lenni.

Rekurzív (naív) megoldás

- Legyen $m[i,j]$ az $M_{i..j}$ szorzatszakasz optimális kiszámításának szorzatszám. ($1 < i, j < n$)
- $m[i,i] = 0$ $m[i,j] = \min_{i \leq k < j} (m[i,k] + m[k+1,j] + \text{sor}(i) * \text{oszlop}(k) * \text{oszlop}(j))$
- k szerinti ciklusban, az m szerinti rekurzív hívással (**FentrőlLefelé-TeljestőlRészekig-TopDown - FeketeLuk** megközelítés) \rightarrow exponenciális idő, mert a részfeladatokat esetleg többször is megoldja \rightarrow átfedő részfeladatok

Szorások számának becslése

- Hány részfeladat van összesen? Ahány $m[i,j]$ részoptimum, vagyis ahány $1 \leq i \leq j \leq n$ (i,j) pár, összesen $n*(n+1)/2 = \Theta(n^2)$
- Megoldás: **AlulrólFelfelé, a RészektőlEgészíg, BottomUp**, a részeredmények tárolásával
- Algoritmuselemzés: Futásidő: 1 részfeladat kiszámítása egy n hosszú vektor (átló) optimumkeresése. $\rightarrow \Theta(n^3)$
- Helyigény: $\Theta(n^2)$
- Filozófia: TopDown vagy BottomUp? – Lebontunk, vagy építkezünk? – Csőlátás vagy halszemoptika? – Holisztika vagy redukcionizmus? Ügyes szakbarbárok, vagy haszontalan próféták?

```
MSzorásSorrend(n)
  for i=1 to n
    do m[i,i]=0
  for l=2 to n
    do for i=1 to n-l+1
      do j=i+l-1
        m(i,j)=∞
        for k=i to j-1
          do Szorz=m[i,k]+m[k+1,j]+
            sor(i)*oszlop(k)*oszlop(j)
          if Szorz<m[i,j] then
            m[i,j]=Szorz
            s[i,j]=k
  return m, s
```

Mátrixok száma

Főátló kinullázása

Átlósan haladunk, l a kezdő j index

k töréspont ($i \leq k < j$)

Az optimális megoldás

- `MátrixLáncSzorzat(M, i, j)`
 if `j > i` then
 `X = MátrixLáncSzorzat(M, i, s[i, j])`
 `Y = MátrixLáncSzorzat(M, s[i, j] + 1, j)`
 return (`Mátrixszorzat(X, Y)`)
 else return `M[i]`
- A helyes zárójelezés: $(M1 * (M2 * M3)) * ((M4 * M5) * M6)$

Hol alkalmazható egyáltalán?

- Optimalizálási probléma
- Optimális részstruktúrák
- Egymást átfedő részfeladatok (az Oszd Meg és Uralkodj nem vizsgálja a visszatérő részfeladatokat, hanem újra megoldja őket)

Egy alternatív megoldás!! Rekurzív, felülről lefelé (naív) megoldás a részeredmények mátrixba történő feljegyzésével.