Matrix-chain multiplication

Mislav Žanić

Siječanj 2021. godine

Contents

1 Uvod **3**

2 Algoritam **4**
2.1 Broj redoslijeda 4
2.2 Dinamičko programiranje 4
2.3 Implementacija algoritma 5
2.4 Dokaz optimalnosti rješenja 5

3 Analiza složenosti **5**

4 Primjeri i mjerenja **6**

1 Uvod

Za dani niz ulančanih matrica $A_1, A_2, \ldots, A_n$ potrebno je izračunati produkt $A_1 A_2 \ldots A_n$. Produkt možemo izračunati standardnim algoritmom za množenje matrica nakon što postavimo sve zagrade na njihova mjesta (tj. odredimo redoslijed množenja). Npr. u produkt matrica $A_1 A_2 A_3 A_4$ sve zagrade možemo postaviti na 5 načina:

$$\begin{aligned}
&(A_1(A_2(A_3A_4))),\\
&(A_1((A_2A_3)A_4)),\\
&((A_1A_2)(A_3A_4)),\\
&((A_1(A_2A_3))A_4),\\
&(((A_1A_2)A_3)A_4).
\end{aligned}$$

Odabir načina postavljanja zagrada može uvelike utjecati na složenost algoritma. To slijedi iz činjenice da u standardnom algoritmu za množenje matrica, vrijeme računjanja produkta najviše ovisi o broju operacija množenja skalara.

Primjer 1.1 *Za matrice A, B, C dimenzija $4 \times 3, 3 \times 2, 2 \times 1$ odredite ABC koristeći sto manje operacija množenja skalara.*
Rješenje. Koristimo standardni algoritam za množenje matrica:

```
function MULTIPLY(A, B)
    if A.columns ≠ B.rows then
        error "incompatible dimensions"
    else
        C ← A.rows × B.columns matrix
        for i = 1 to A.rows do
            for j = 1 to B.columns do
                c_ij ← 0
                for k = 1 to A.columns do
                    c_ij ← c_ij + a_ik · b_kj
        return C
```

Produkt ABC možemo izračunati redoslijedom $((AB)C)$ ili $(A(BC))$. Prvi redoslijed zahtijeva $4 \cdot 3 \cdot 2 + 4 \cdot 2 \cdot 1 = 32$ operacija množenja skalara, dok drugi zahtijeva $3 \cdot 2 \cdot 1 + 4 \cdot 3 \cdot 1 = 18$ operacija množenja skalara. Zaključujemo da produkt trebamo računati drugim redoslijedom.

Matrix-chain multiplication je optimizacijski problem pronalaženja redoslijeda množenja u nizu ulančanih matrica.
Formalno, za dani niz od n ulančanih matrica $A_1, \ldots, A_n$, u kojem matrica A_i ima dimenzije $p_{i-1} \times p_i$, $i = 1, \ldots, n$, odredite redoslijed množenja matrica u produktu $A_1 A_2 \ldots A_n$ koji minimizira broj operacija množenja skalara.

2 Algoritam

2.1 Broj redoslijeda

Označimo produkt niza ulančanih matrica $A_1, \ldots, A_n$ sa $A_{1\ldots n}$ i broj rasporeda množenja ulančanih matrica u produktu $A_{1\ldots n}$ sa R_n.
Za $n = 1$ vrijedi $R_1 = 1$ jer imamo samo jedan raspored. Za $n \geq 2$ možemo produkt $A_{1\ldots n}$ podijeliti na dva potprodukta $A_{1\ldots k}$ i $A_{k+1\ldots n}$ $((A_1 \ldots A_k)(A_{k+1} \ldots A_n))$, $k = 1, \ldots, n$, i računati broj rasporeda za svaki od njih. Sumiranjem po k dobivamo rekurzivnu formulu

$$R_n = \begin{cases} 1 & n = 1 \\ \sum_{k=1}^{n-1} R_k R_{n-k} & n \geq 2. \end{cases}$$

Iz ovoga slijedi da je R_n jednak $n-1$-om *Catalanovom broju* $C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$.
Iz činjenice da Catalanov n-ti broj asimptotski raste eksponencijalno, možemo zakljuciti da *brute-force* rješenje nije najbolja ideja.

2.2 Dinamičko programiranje

Optimalan broj operacija množenja skalara niza ulančanih matrica dobit ćemo metodom dinamičkog programiranja.
Označimo sa m_{ij} minimalan broj operacija množenja skalara potreban za računanje matrice $A_{i\ldots j}$, $1 \leq i \leq j \leq n$. Definiramo rekurzivnu formulu

$$m_{ij} = \begin{cases} 0 & i = j \\ \min\limits_{i \leq k < j} \{m_{ik} + m_{k+1j} + p_{i-1}p_k p_j\} & i < j. \end{cases}$$

Ocito, za računanje $A_{i\ldots i} = A_i$ potrebno je 0 operacija množenja skalara, pa je $m_{ii} = 0$, dok za $i < j$ trebamo prvo podijeliti matricu $A_{i\ldots j}$ na matrice $A_{i\ldots k}$ i $A_{k+1\ldots j}$, $k = i, \ldots, j-1$, kojima je potrebno m_{ik} i m_{k+1j}, operacija množenja skalara. Nakon toga trebamo pomnožiti matrice $A_{i\ldots k}$ i $A_{k+1\ldots j}$ za što je potrebno $p_{i-1}p_k p_j$ operacija množenja skalara. Uzimanjem minimuma po k dobivamo najmanji broj operacija množenja skalara potrebnih da bi smo dobili matricu $A_{i\ldots j}$.

2.3 Implementacija algoritma

Algoritam ćemo implementirati koristeći *bottom-up* metodu i tablicu vrijednosti $m[1 \dots n][1 \dots n]$, gdje je n duljina niza ulančanih matrica u funkciji MATRIX-CHAIN-ORDER koja prima niz $p = \{p_0, \dots, p_n\}$, $p_{i-1} \times p_i$ dimenzija matrice A_i.

```
function MATRIX-CHAIN-ORDER(p)
    n ← p.length − 1
    m[1...n][1...n] ← new table
    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 = 1 to j − 1 do
                q ← m[i][k] + m[k + 1][j] + p_{i−1} p_k p_j
                if q < m[i][j] then
                    m[i][j] ← q
    return m
```

Prva for petlja iterira po svim mogućim duljinama niza ulančanih matrica, druga iterira po svim mogućim početnim indeksima i, a treća po svim mogućim srednjim indeksima k. Tablicu $m[1 \dots n][1 \dots n]$ popunjavamo t.d. na mjesto $m[i][j]$ upisujemo vrijednost m_{ij} koju računamo u funkciji.

2.4 Dokaz optimalnosti rješenja

Pretpostavimo da rješenje koje nam je dao algoritam nije optimalno. Slijedi da postoji matrica $A_{i \dots j}$ za koju smo mogli odabrati bolji indeks $k \in \{i, \dots, j-1\}$ za koji će $m[i][j]$ biti minimalan. Ali, algoritam već bira k za koji će $m[i][j]$ biti minimalan, pa možemo zaključiti da je pretpostavka pogrešna, tj. da je algoritam dao optimalno rješenje.

3 Analiza složenosti

U pseudokodu algoritma vidimo da imamo sveukupno $O(n^2)$ potproblema (podnizovi duljine $l = 2, \dots, n$, od kojih svaki počinje na $n - l + 1$ različitih mjesta). U svakom od potproblema promatramo $j - 1$ različitih mogućnosti za optimalno rješenje, a pošto $j \in \{2, \dots, n\}$, slijedi da je složenost algoritma $O(n^3)$.

4 Primjeri i mjerenja

Primjer 4.1 *Optimalan broj množenja i broj množenja dobiven množenjem po redu za dane nizove.*

(a) $p = 10, 5, 30, 100, 77$
(b) $p = 26, 15, 67, 3, 9, 63$
(c) $p = 75, 14, 9, 4, 239, 8, 46$
(d) $p = 141, 9, 4, 99, 38, 29, 12, 74$
(e) $p = 94, 23, 43, 90, 9, 34, 6, 33, 10$
(f) $p = 14, 3, 43, 50, 97, 74, 9, 53, 59, 73$

Rješenje. Označimo sa b_{ij}, $i < j$ broj množenja dobiven množenjem po redu.

(a) $m_{1,5} = 57350$, $b_{1,5} = 108500$, redoslijed: $(A_1((A_2A_3)A_4))$
(b) $m_{1,6} = 10800$, $b_{1,6} = 46800$, redoslijed: $((A_1(A_2A_3))(A_4A_5))$
(c) $m_{1,7} = 27624$, $b_{1,7} = 254850$, redoslijed: $((A_1(A_2A_3))((A_4A_5)A_6))$
(d) $m_{1,8} = 107728$, $b_{1,8} = 438228$, redoslijed: $((A_1A_2)((((A_3A_4)A_5)A_6)A_7))$
(e) $m_{1,9} = 56442$, $b_{1,9} = 630458$, redoslijed: $((A_1(A_2(A_3(A_4(A_5A_6)))))(A_7A_8))$
(f) $m_{1,10} = 71331$, $b_{1,10} = 320376$, redoslijed: $(A_1(((((((A_2A_3)A_4)A_5)A_6)A_7)A_8)A_9))$

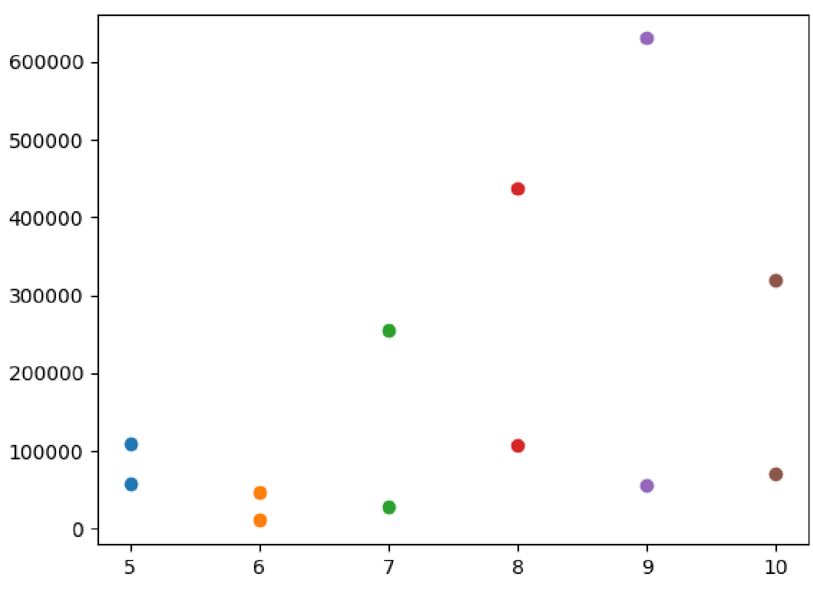


Figure 1: Grafički prikaz rješenja.

Primjer 4.2 *Optimalan broj množenja i broj množenja dobiven množenjem po redu za dane nizove.*

(a) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10$
(b) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10, 70$
(c) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10, 70, 120$
(d) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10, 70, 120, 60$
(e) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10, 70, 120, 60, 10$
(f) $p = 300, 35, 50, 500, 10, 200, 20, 50, 100, 10, 70, 120, 60, 10, 4$

Rješenje. Označimo sa b_{ij}, $i < j$ broj množenja dobiven množenjem po redu.

(a) $m_{1,10} = 478000$, $b_{1,10} = 13425000$
(b) $m_{1,11} = 688000$, $b_{1,11} = 13635000$
(c) $m_{1,12} = 922000$, $b_{1,12} = 16155000$
(d) $m_{1,13} = 814000$, $b_{1,13} = 18315000$
(e) $m_{1,14} = 641000$, $b_{1,14} = 18495000$
(f) $m_{1,15} = 288600$, $b_{1,15} = 18507000$

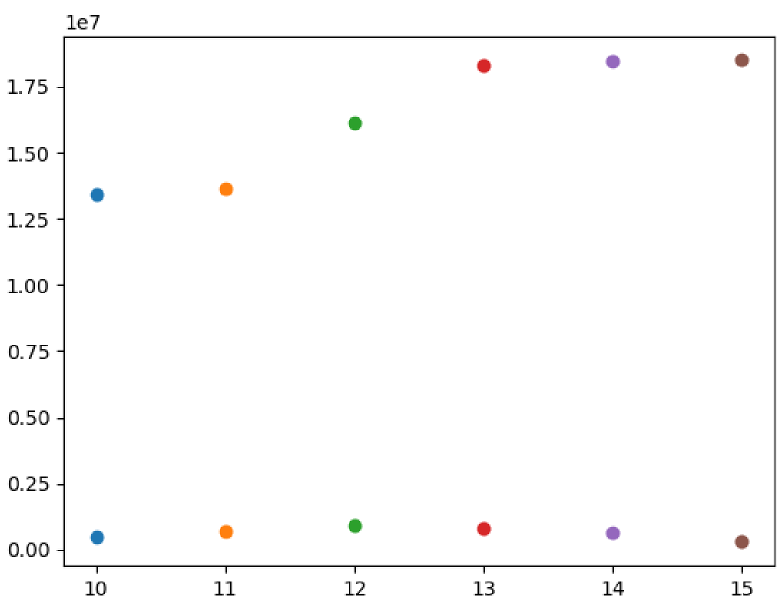


Figure 2: Grafički prikaz rješenja.

Primjer 4.3 *Brojevi množenj za dane nizove.*
(a) $p = 64, 369, 385, 875, 1411, 1767, 1799, 1946, 2410, 2447, 2767, 3206, 3299, 3565, 3769$
(b) $p = 3760, 3565, 3299, 3206, 2767, 2447, 2410, 1946, 1799, 1767, 1411, 875, 385, 369, 64$

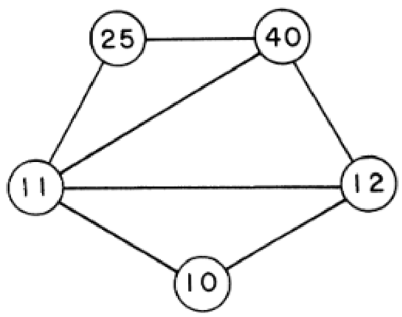
Rješenje. Označimo sa b_{ij}, $i < j$ broj množenja dobiven množenjem po redu.

(a) $m_{1,15} = 4662873536$, $b_{1,15} = 4662873536$
(b) $m_{1,15} = 4662873536$, $b_{1,15} = 223632072400$

Primjer 4.4 *Odredite broj množenja za sve cikličke rotacije danog niza.*

$$p = 12, 10, 11, 25, 40$$

Rješenje. Svi su jednaki i iznose 16620. Ovo svojstvo vrijedi za bilo koji niz ulančanih matrica. Redoslijed množenja: $(((A_1A_2)A_3)A_4)$.



Dokaz za ovo svojstvo je možete pronaći u [2], [3] zajedno sa algoritmom složenosti $O(nlogn)$ za ovaj problem.

Bibliografija

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms, 3rd Edition*, The MIT Press, 2009.

[2] T. C. Hu, M. T. Shing, *Computation of matrix chain products. Part 1*, Society for Industrial and Applied Mathematics, 1982.

[3] T. C. Hu, M. T. Shing, *Computation of matrix chain products. Part 2*, Society for Industrial and Applied Mathematics, 1984.