תכנון דינאמי

סדר הכפלת מטריצות

נתחיל מתזכורת לדברים הרלוונטיים מאלגברה לינארית:

- ע"מ לבצע את מכפלת המטריצות $A\cdot B$, על מס' העמודות של המטריצה השמאלית להיות שווה למספר השורות של המטריצה הימנית. עבור מטריצות $A\in M_{n\times m}(F), B\in M_{m\times k}(F)$ נקבל כי $A\cdot B\in M_{n\times k}(F)$.
- ע"מ לחשב את האיבר c_{ij} בתוצאת המכפלה, יש לבצע $O(m)$ פעולות [m מכפלות וכ-m סכומים].
- ע"מ לחשב את המכפלה כולה, יש לבצע $O(n\cdot m\cdot k)$ פעולות.

ידוע ש-$A\cdot(B\cdot C)=(A\cdot B)\cdot C$ – איזו מבין המכפלות יש לבצע?
לדוגמה, נניח שאנו רוצים לבצע את המכפלה:

$$A^1_{1\times 10}\cdot A^2_{10\times 15}\cdot A^3_{15\times 20}\cdot A^4_{20\times 1}$$

[[הסימון A^i משמעותו המטריצה ה-i, ולא מטריצה בחזקת i.]]
אם נכפול קודם את הזוג השמאלי והזוג הימני ואז את התוצאה, נצטרך לבצע 150 פעולות עבור הזוג השמאלי, 300 פעולות עבור הזוג הימני, ואז כיוון שנשאר לבצע מכפלה מהצורה $B_{1\times 15}\cdot C_{15\times 1}$, יש לבצע עוד 15 פעולות.
סה"כ: 465 פעולות.

עם לעומת זאת נתחיל מלכפול את הזוג האמצעי, רק הוא כבר ידרוש לבצע $10\cdot 15\cdot 20=3000$ פעולות.

כלומר זה לא רק משנה – זה משנה **מאוד**.

הגדרת הבעיה

- <u>מופע</u>: n מטריצות להכפלה, $A^1\cdot A^2\cdots A^n$ כך שכל מטריצה A^i היא מסדר $p_{i-1}\times p_i$.
- <u>פתרון</u>: סידור סוגריים חוקי להכפלת מטריצות.
- <u>עלות פתרון</u>: $cost(U)=$ מספר הפעולות הנדרש לחישוב המכפלה.
- <u>פתרון אופטימלי</u>: פתרון בעל **עלות** מינימלית.

מספר האפשרויות לסידור סוגריים הוא יותר מ-2^n (החל מ-n מסוים).

תת-בעיה אופיינית

- זוג (i,j) כך ש-$1\le i\le j\le n$.
- (i,j) – תת-הבעיה של הכפלת המטריצות $A^i\cdot A^{i+1}\cdot A^{i+2}\cdots A^j$.

הגדרת OPT ונוסחת מבנה

נגדיר פונקציה OPT באופן הבא:

$$OPT\left(i,j\right)=A^{i}\cdots A^{j}$$ מס' פעולות מינימלי לחישוב

<u>נוסחת מבנה:</u>

נבנה את הנוסחה עבור פונקציה זו:

$$OPT\left(i,j\right)=\begin{cases}0, & i=j\\ ? & i<j\end{cases}$$

נחשוב רקורסיבית: בהנחה ואנו יודעים את הפתרון של כל הבעיות הקטנות יותר, מה יהיה הערך הרצוי?

נבחן את כל האפשרויות לפצל את המכפלה בנקודה $i\le l\le j$:

$$\underbrace{\left(A^{i}A^{i+1}\cdots A^{l}\right)}_{B_{p_{i-1}\times p_l}}\underbrace{\left(A^{l+1}\cdots A^{j}\right)}_{C_{p_l\times p_j}}$$

עלות הפתרון האופטימלי עבור $\left(i,j\right)$ יהיה המינימום מבין העלויות של כל האפשרויות לבחור את l.

תוצאת המכפלה $\left(A^{i}A^{i+1}\cdots A^{l}\right)$ היא מטריצה B מסדר $p_{i-1}\times p_l$ [[כי אנו מבצעים מכפלה של מטריצות מהסדרים $\left(p_{i-1}\times p_i\right)\cdot\left(p_i\times p_{i+1}\right)\cdots\left(p_{l-1}\times p_l\right)$]], ובדומה, תוצאת המכפלה $\left(A^{l+1}\cdots A^{j}\right)$ היא מטריצה C מסדר $p_l\times p_j$.

לכן מכפלת מטריצות התוצאה לוקחת $p_{i-1}\cdot p_l\cdot p_j$ פעולות.

בסה"כ נקבל ש-$OPT\left(i,j\right)$ במקרה בו $i<j$ הוא:

$$\boxed{*}\ \boxed{\min\left\{OPT\left(i,l\right)+OPT\left(l+1,j\right)+p_{i-1}\cdot p_l\cdot p_j \,\middle|\, i\le l\le j\right\}}$$

ולכן:

$$OPT\left(i,j\right)=\begin{cases}0, & i=j\\ \min\left\{OPT\left(i,l\right)+OPT\left(l+1,j\right)+p_{i-1}\cdot p_l\cdot p_j \,\middle|\, i\le l\le j\right\} & i<j\end{cases}$$

[**<u>שימו לב</u>**: הפונקציה OPT היא מספרית, ומוגדרת במילים להיות הערך המתאים לפתרון הטוב ביותר. הנוסחה הנ"ל היא **לא הגדרת הפונקציה**; אנו רק טוענים שניתן לחשב את הערך $OPT\left(i,j\right)$ בעזרת נוסחה זו.]

<u>הוכחת נוסחת המבנה:</u>

- <u>מקרה א'</u>: $i=j$

אז מספר הפעולות הנדרש הוא 0 – ואכן $OPT\left(i,j\right)$ על פי נוסחת המבנה.

- <u>מקרה ב'</u>: $i<j$
צ"ל כי $\boxed{*}=OPT(i,j)$.

1. נראה כי $OPT(i,j)\le\boxed{*}$ על ידי כך שנראה ש-$\boxed{*}=cost(U)$ כאשר u פתרון כלשהו.
ניח שהמינימום מתקבל בנקודה $l=k$.
אזי:

$$\boxed{*}=OPT(i,k)+OPT(k+1,j)+p_{i-1}\cdot p_k\cdot p_j$$

נבנה את U כך:
   - נשים סוגריים על $(A^i\cdots A^k)$ ועל $(A^{k+1}\cdots A^j)$.
   - נבחר סידור חוקי U_1 ל- $A^i\cdots A^k$ כך ש-$cost(U_1)=OPT(i,k)$.
   - נבחר סידור חוקי U_2 ל- $A^{k+1}\cdots A^j$ כך ש-$cost(U_2)=OPT(k+1,j)$.

[[<u>כלומר</u>: חלוקה בנקודה זו מניב את המכפלות:

$$(A^i\cdots A^k)(A^{k+1}\cdots A^j)$$

(כאשר יתכנו עוד סוגריים פנימיים).
נסמן:
   - $U_1=$ פתרון אופטימלי עבור (i,k)
   - $U_2=$ פתרון אופטימלי עבור $(k+1,j)$

]]

אז:

$$cost(U)=cost(U_1)+cost(U_2)+p_{i-1}\cdot p_k\cdot p_j=$$
$$=OPT(i,k)+OPT(k+1,j)+p_{i-1}\cdot p_k\cdot p_j=\boxed{*}$$

[בכך הראינו כי $OPT(i,j)\le\boxed{*}$, ולא כי $OPT(i,j)=\boxed{*}$.
למה? כי הראינו שקיים פתרון שהעלות שלו היא $\boxed{*}$.
לא הוכחנו שפתרון זה הוא אופטימלי, לכן עלות הפתרון האופטימלי היא **לכל היותר** הערך של הפתרון שמצאנו (כי העלות של הפתרון האופטימלי ≥ לעלות של כל הפתרונות החוקיים).]

2. נראה כי $OPT(i,j)\ge\boxed{*}$ על ידי כך שנראה שלכל פתרון U מתקיים $cost(U)\ge\boxed{*}$.
[מדוע זה נכון?]

כי כל U חוקי מכיל פירוק ראשי לשתי מכפלות, $\left(A^i \cdots A^k\right)\left(A^{k+1} \cdots A^j\right)$, והעלות של מכפלה זו היא אחד המרכיבים ב- $\min$ שהגדרנו.]

ניקח פתרון U ונניח שהסוגריים החיצוניים הם על $\left(A^i \cdots A^k\right)$ וְ-$\left(A^{k+1} \cdots A^j\right)$.

נסמן:

- U_1 – סידור הסוגריים ב- $A^i \cdots A^k$.
- U_2 – סידור הסוגריים ב- $A^{k+1} \cdots A^j$.

אז:

$$\begin{aligned} cost(U) &= cost(U_1) + cost(U_2) + p_{i-1} \cdot p_k \cdot p_j \geq \\ &\geq OPT(i,k) + OPT(k+1,j) + p_{i-1} \cdot p_k \cdot p_j \geq \boxed{*} \end{aligned}$$

מ-1 ו-2 אנו מקבלים שוויון.

אלגוריתם 1 [רקורסיה נאיבית]

$Algo_1(i,j)$:

1. If $i = j$ return 0
2. $Min \leftarrow \infty$
3. For $l = i$ to $j-1$:
   3.1. $Temp \leftarrow Algo_1(i,l) + Algo_1(l+1,j) + p_{i-1} \cdot p_l \cdot p_j$
   3.2. If $Temp < Min$:
      3.2.1. $Min \leftarrow Temp$
4. Return Min

[כרגע אנחנו מעוניינים רק לדעת מה הערך של הפתרון האופטימלי.
אח"כ נבנה את הפתרון עצמו.]

סיבוכיות:

$$\begin{cases} T(1) = c \\ T(n) = \sum_{k=1}^{n-1} \left(T(k) + T(n-k)\right) \end{cases}$$

מספר הקריאות הרקורסיביות כאן עצום – $T(n) > 2^n$.

ננסה גישה אחרת.

אלגוריתם 2: Memoization

נחזיק מערך $M[1\cdots n, 1\cdots n]$ [[כלומר מערך דו-ממדי בגודל $n \times n$]].

בכל פעם שנחשב את $OPT(i,j)$ נכתוב את הערך ל-$M[i,j]$.

בחישוב ערך חדש $OPT(i,j)$ נבדוק במערך – אם יש ערך נחזיר אותו, אחרת נבצע קריאה רקורסיבית.

[שרטוט עץ קריאות לדוגמה]

[את האלגוריתם המלא ניתן למצוא בדפי התרגול באתר הקורס.]

אלגוריתם 3

[נחשב בהתחלה את כל הערכים במקומות בהם $i=j$ (כלומר על האלכסון הראשי).

אז נחשב את הערכים על האלכסון בו $j-i=1$, נתקדם ל-$j-i=2$ וכן הלאה.

נשים לב שיש למלא רק חצי של המערך הדו-ממדי (את המשולש העליון, שבו $i \le j$).

ייצא בסוף שהסיבוכיות הכוללת היא $O(n^3)$.]

$Algo_3(A^1,\dots,A^n)$:

1. For $i=1$ to n:
   1.1. $M[i,i] \leftarrow 0$
2. For $diff=1$ to $n-1$:
   2.1. For $i=1$ to $n-diff$:
      2.1.1. $min \leftarrow \infty$
      2.1.2. For $l=i$ to $n-diff-1$:
         2.1.2.1. $temp \leftarrow M[i,l]+M[l+1,i+diff]+p_{i-1}\cdot p_l \cdot p_{n-diff}$
         2.1.2.2. If $temp < min$:
            2.1.2.2.1. $min \leftarrow temp$
      2.1.3. $M[i,i+diff] \leftarrow min$
3. Return $M[1,n]$

הוכחת נכונות

1. כאשר מחשבים את $M[i,j]$, כבר חישבנו את ערך כל תת-בעיה קטנה יותר.
2. (באינדוקציה) ל-$M[i,j]$ כותבים את $\boxed{*}$.

שחזור הפתרון

[כעת נרצה לבנות את חלוקת הסוגריים בעזרת המערך הדו-ממדי שבנינו באלגוריתם.]

$Algo_4(M,i,j)$:

1. אם $i=j$: Return [[כלומר אל תעשה כלום וצא]]
2. מצא l כך ש: $M[i,j] = M[i,l]+M[l+1,j]+p_{i-1}\cdot p_l\cdot p_j$.
3. שים סוגריים על $A^1\cdots A^l$ ועל $A^{l+1}\cdots A^j$.
4. $Algo_4(M,i,l)$
5. $Algo_4(M,l+1,j)$

<u>זמן ריצה</u>: $O(n^2)$ [צריך להוכיח זאת].

<u>עוד דרך לשחזור הפתרון</u>:
בזמן ריצת האלגוריתם המקורי נוסיף בניה עבור מערך דו-ממדי נוסף שישמור את נקודות החלוקה הטובות ביותר.
להלן השינוי באלגוריתם:

4. For $i=1$ to n:
    4.1. $M[i,i]\leftarrow 0$
5. For $diff=1$ to $n-1$:
    5.1. For $i=1$ to $n-diff$:
        5.1.1. $min\leftarrow\infty$
        5.1.2. For $l=i$ to $n-diff-1$:
            5.1.2.1. $temp\leftarrow M[i,l]+M[l+1,i+diff]+p_{i-1}\cdot p_l\cdot p_{n-diff}$
            5.1.2.2. If $temp<min$:
                5.1.2.2.1. $min\leftarrow temp$
                5.1.2.2.2. $S[i,i+diff]\leftarrow l$
        5.1.3. $M[i,i+diff]\leftarrow min$
6. Return $M[1,n]$

לאחר שינוי זה ניתן לבצע את $Algo_4$ מבלי להצטרך לבצע את החיפוש, מה שהופך את הסיבוכיות שלו ל-$O(n)$ [המקרה הכי גרוע הוא, למשל, כאשר l תמיד הכי גדול שאפשר].