תכנון דינאמי

בתרגול זה נדון בבעיית הכפלת סדרת מטריצות (CLR 16.1). ראשית נראה דוגמא:

דוגמא:
תהינה ארבע מטריצות: $A^1_{1\times10}, A^2_{10\times20}, A^3_{20\times15}, A^4_{15\times1}$. נסמן את גודל המטריצות בסדרה ע"י סדרת גדלים $P=\{1,10,20,15,1\}$, כלומר A_i היא בגודל $p_{i-1}\times p_i$, כאשר $1\le i\le 4$

הכפלה של שתי מטריצות $A_{p_1 x p_2}, A_{p_2 x p_3}$ בדרך הנאיבית היא בסיבוכיות של מכפלת הגדלים, כלומר $p_1 \cdot p_2 \cdot p_3$.
למשל: $A_1\times A_2$ היא בסיבוכיות של $1*10*20=200$, והתוצאה היא מטריצה בגודל 1×20.
במקרה הכללי: $A_i\times A_{i+1}$ היא בסיבוכיות $p_{i-1}\cdot p_i\cdot p_{i+1}$, והמטריצה המתקבלת היא בגודל $p_{i-1}\times p_{i+1}$.

הכפלת מטריצות היא אסוציאטיבית: $((A_1\times A_2)\times(A_3\times A_4))=(A_1\times((A_2\times A_3)\times A_4))$, כלומר סדר הכפלים אינו משנה את התוצאה. לעומת זאת סדר הכפלים משנה את מספר הכפלים לחישוב: אם נכפיל את המטריצות כך: $((A_1\times A_2)\times(A_3\times A_4))$ – ה"עלות" תהיה 200 לזוג המטריצות השמאליות, ו-$20*15*1=300$ למכפלה $A_3\times A_4$. הכפלת שתי התוצאות – $20*1*20=1$. סך הכול – 520 חישובים.
אם נכפיל את המטריצות כך: $(A_1\times((A_2\times A_3)\times A_4))$, רק עלות המכפלה $A_2\times A_3$ תהיה $10*20*15=3000$, ועלות זו גבוהה בהרבה מהכפלת הסדרה כולה לפי סדר ההכפלות הקודם.

נגדיר פורמאלית את הבעיה:

מופע: סדרת מטריצות $\{A_1, A_2..., A_n\}$ הניתנות להכפלה, כלומר, לכל $1\le i<n-1$ מימד העמודות של A_i שווה למימד השורות של A_{i+1}.
פתרון אפשרי: סדר הכפלה של המטריצות $\{A_1, A_2..., A_n\}$ שנסמנו ב U
עלות הפתרון: עבור פתרון U נגדיר את $\cos t(U)$ להיות מספר הפעולות הנדרש להכפלת
יש למצוא: סדר הכפלה של המטריצות עם עלות מינימום.

נשים לב שפתרון לבעיה הינו הצבת סוגריים על סדרת המטריצות, שכן הסוגריים מגדירים סדר הכפלה. הפתרון הנאיבי הוא לנסות את כל האפשרויות להצבת סוגריים על הסדרה.

בכמה אופנים ניתן להכניס סוגריים מאוזנים בצורה מלאה במחרוזת באורך n? התשובה היא מספר קָטָלָן (Catalan) של n שהוא אקספוננציאלי ב-n. ליתר דיוק, $\Omega(4^n/n^{1.5})$ הוא חסם תחתון אקספוננציאלי למספר הזה.

נראה כי קיים פתרון יעיל בהרבה.

פתרון על ידי תכנון דינמי

הגדרת תתי הבעיות ו OPT

נסמן ב- $OPT[i,j]$ את מספר פעולות הכפל המינימאלי הדרוש להכפלת סדרת המטריצות $\{A_i,...,A_j\}$.
אנו מחפשים את $OPT[1,n]$, כלומר מספר פעולות הכפל המינימאלי בהכפלת כל סדרת המטריצות.
למשל, אם $\{A_i,...,A_j\}=\{A^1_{20\times10}, A^2_{10\times1}\}$ אז $OPT[i,j]=200$.

טענה: (מבנה של פתרון אופטימאלי)

$$OPT[i,j]=\begin{cases}0, & ,i=j\\ \min\limits_{i\le k<j}\{OPT[i,k]+OPT[k+1,j]+p_{i-1}\cdot p_k\cdot p_j\} & ,i<j\end{cases}$$

הוכחה:

א. ניתוח מקרים – נחלק את הפתרונות האפשריים לקבוצות. נגדיר $j-i$ תת-קבוצות של פתרונות אפשריים, $S_i,S_{i+1},\ldots,S_{j-1}$, כאשר S_k היא אוסף של כל הפתרונות בהם הכפל האחרון הוא בין תוצאת מכפלת הסדרה $\{A_i,...,A_k\}$ לתוצאת מכפלת הסדרה $\{A_{k+1},...,A_j\}$, ז"א כל הפתרונות מהצורה:
$$\left((A_i\times A_{i+1}\times.....\times A_k)\times(A_{k+1}\times A_{k+2}\times.....\times A_j)\right).$$

ב. תת-הקבוצות מסעיף (א) מכסות את קבוצת כל הפתרונות האפשריים - ניקח פתרון כלשהו לבעיה U, שהוא סדר להכפלת סדרת מטריצות $\{A_i,A_2...,A_j\}$. נתבונן בכפל האחרון בסדר. נניח כי הכפל האחרון הוא בין המכפלות של הסדרות $\{A_i,...,A_k\}$ ו- $\{A_{k+1},...,A_j\}$, אזי $U\in S_k$.

ג. מסקנה לצורת נוסחת המבנה
תזכורת: לכל פתרון $U\in S_k$ הגדרנו את $\cos t(U)$ להיות מספר הפעולות הנדרש להכפלת המטריצות $\{A_i,A_2...,A_j\}$ לפי הסידור של הפתרון U עבור S_k, $i\le k\le j-1$.
נגדיר את $O^*(S_k)$ להיות המחיר המינימאלי של הפתרונות בקבוצה S_k, כלומר,
$O^*(S_k)=\min\limits_{U\in S_k}\{\cos t(U)\}$.

מסקנה: מ-(א) ו- (ב)

$$OPT[i,j]=\min\left\{O^*(S_i),O^*(S_{i+1}),....,O^*(S_{j-1})\right\}$$

ד. <u>ניתוח האופטימום של כל קבוצה</u>.

נוכיח כי לכל $i \le k < j$ מתקיים $O^*(S_k) = OPT[i,k] + OPT[k+1,j] + p_{i-1} \cdot p_k \cdot p_j$.

<u>כיוון 1</u>: $\le$
נסתכל כל צד ימין, יהיו P_1 ו- P_2 הסידורים של המטריצות $\{A_i,...,A_k\}$ ו- $\{A_{k+1},...,A_j\}$ בהתאמה כך ש: $\cos t(P_1) = OPT[i,k]$ ו- $\cos t(P_2) = OPT[k+1,j]$. אזי הסידור P^* של סדרת המטריצות $\{A_i, A_2..., A_j\}$ כך שהמטריצות $\{A_i,...,A_k\}$ מוכפלות לפי הסידור P_1 והמטריצות $\{A_{k+1},...,A_j\}$ מוכפלות לפי הסידור P_2, והכפל האחרון $\left((A_i \times A_{i+1} \times ..... \times A_k) \times (A_{k+1} \times A_{k+2} \times ..... \times A_j)\right)$ הוא פתרון ב- S_k ועלותו היא: $\cos t(P^*) = OPT[i,k] + OPT[k+1,j] + p_{i-1} \cdot p_k \cdot p_j$. ולפי הגדרת $O^*(S_k)$ בהכרח מתקיים $O^*(S_k) \le \cos t(P^*)$.

<u>כיוון 2</u>: $\ge$
נסתכל על צד שמאל, יהי $P^* \in S_k$ כך ש- $O^*(S_k) = \cos t(P^*)$, כלומר קודם מכפילים מטריצות $\{A_i,...,A_k\}$, אח"כ מכפילים מטריצות $\{A_{k+1},...,A_j\}$ ואח"כ מכפילים את התוצאות.
יהיו P_1 ו- P_2 הסידורים של המטריצות $\{A_i,...,A_k\}$ ו- $\{A_{k+1},...,A_j\}$ (בהתאמה) ב- P^*.
מחיר P^* הוא $\cos t(P_1) + \cos t(P_2) + p_{i-1} \cdot p_k \cdot p_j$ ולכן
$O^*(S_k) = \cos t(P^*) = \cos t(P_1) + \cos t(P_2) + p_{i-1} \cdot p_k \cdot p_j$.
לפי הגדרת $OPT[i,j]$ מתקיים כי: $OPT[i,k] \le \cos t(P_1)$ וגם $OPT[k+1,j] \le \cos t(P_2)$ ולכן
$O^*(S_k) \ge OPT[i,k] + OPT[k+1,j] + p_{i-1} \cdot p_k \cdot p_j$.

<u>הערה</u>: <u>הוכחנו רק את הנוסחא ולא את "האלגוריתם". שימו לב שעד כה לא תיארנו אלגוריתם.</u>

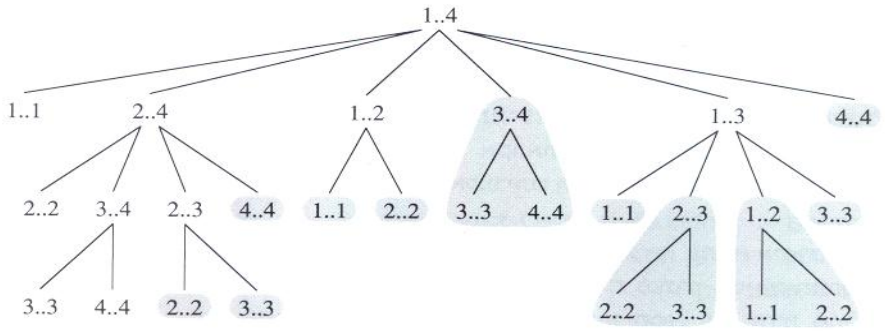
אלגוריתמים מבוססי תכנון דינאמי לבעיית המטריצות

נתחיל מאלגוריתם רקורסיבי לחישוב $OPT[i,j]$.

```
Recursive-Matrix-Chain(i,j)
if i = j then
        return 0
else
        m ← ∞
        for k ← i to j − 1 do
                q ← Recursive-Matrix-Chain(i,k) +
                        Recursive-Matrix-Chain(k+1, j)+ p_{i−1} p_k p_j
                if q < m then
                        m ← q
        return m
```

המחיר המינימאלי של הכפלת המטריצות $A_1,...,A_n$ יתקבל ע"י ההפעלה
Recursive-Matrix-Chain(1,n).

הבעיה באלגוריתם הנ"ל היא שנחשב מספר גדול של חישובים חופפים – נתבונן לדוגמא בעץ הקריאות הרקורסיבי שראינו קודם:



זמן הריצה הנדרש לחישוב $OPT[i, j]$ הוא <u>לפחות אקספוננציאלי</u> ב-n {*פתרון נוסחת המבנה: מקרה בסיס* $T(1)\geq 1$ *ועבור* $n>1$*:* $T(n) \geq 1+\sum_{k=1}^{n-1}(T(k)+T(n-k)+1)$}

פתרון רקורסיבי כזה הוא לא הכי טוב שניתן להשיג.

זה הזמן לשים לב לכך שכמה זוגות ערכים מופיעים בעץ פעמים רבות. כלומר,מספר תת הבעיות הקטנות שהאלגוריתם שלנו פוגש במהלך הפתרון הוא קטן יחסית:

מספר האפשרויות לבחור זוג $<i, j>$ המקיים $1\leq i < j \leq n$ הוא $\binom{n}{2}+n=\Theta(n^2)$.

על מנת להימנע מחישוב חוזר של תת בעיות, נשמור במבנה נתונים כל פתרון שנחשב לתת בעיה. במקרה שלנו, נבנה טבלה בגודל $n\times n$ כך שאם חישבנו כבר את $OPT[i, j]$ אזי ערך הפתרון מאוכסן בטבלה במקום ה-$[i, j]$.

שיפור זמן ריצה על ידי שיטת הפתקאות (memoization)

בזמן ריצת האלגוריתם הרקורסיבי, לפני כל קריאה רקורסיבית נבדוק האם תת הבעיה הזו כבר חושבה (על פי הערך בטבלה – בעלת ערך שונה מ-∞), אם הערך חושב נשתמש בו ונחסוך קריאה רקורסיבית מיותרת (במחיר של חיפוש במבנה הנתונים), אם הערך לא חושב, נחשב אותו ונאכסן את התוצאה במבנה הנתונים לשימוש עתידי. שיטה זו נקראת שיטת התזכור - memoization.

Memoized-Matrix-Chain(n)
for $i \leftarrow 1$ *to* n *do*
    for $j \leftarrow i$ *to* n *do*
        $m[i, j] \leftarrow \infty$
return Lookup-Chain(1,n)

Lookup-Chain(i,j)

```
if m[i, j] < ∞ then
        return m[i, j]
else
        if i = j then
                m[i, j] ← 0
        else
                for k ← i to j −1 do
                        q ← Lookup-Chain(i,k) +
                                Lookup-Chain(k+1, j)+ p_{i−1} p_k p_j
                        if q < m[i, j] then
                                m[i, j] ← q
        return m[i, j]
```

ניתן לראות כי זמן החישוב של הפרוצדורה הוא $O(n^3)$.

אלגוריתם איטראטיבי

במקום לחשב את הפתרון בצורה רקורסיבית, נחשב את הפתרון "מלמטה למעלה" בצורה איטראטיבית. החישוב לתת בעיה באורך 2 משתמש רק בתוצאות החישובים לתתי הבעיות באורך 1 וכן הלאה. ז"א הנתונים הנדרשים לחישוב תת בעיה באורך ℓ הם פתרונות עבור תתי בעיות באורך קטן מ-ℓ אשר כבר חושבו ונמצאים בטבלה. זו בעצם בניית הטבלה מהאלכסון הראשי ומעלה.

השגרה משתמשת בטבלת עזר m[1..n,1..n] לאחסון העלויות ובטבלת עזר s[1..n,1..n] שבה נרשמים האינדקסים k שעבורם התקבלה עלות אופטימאלית בעת חישוב m[i,j], כלומר זהו הערך שמפצלים בו את המכפלה $A_i \cdot ... \cdot A_j$ כדי לקבל הצבת סוגריים אופטימאלית.

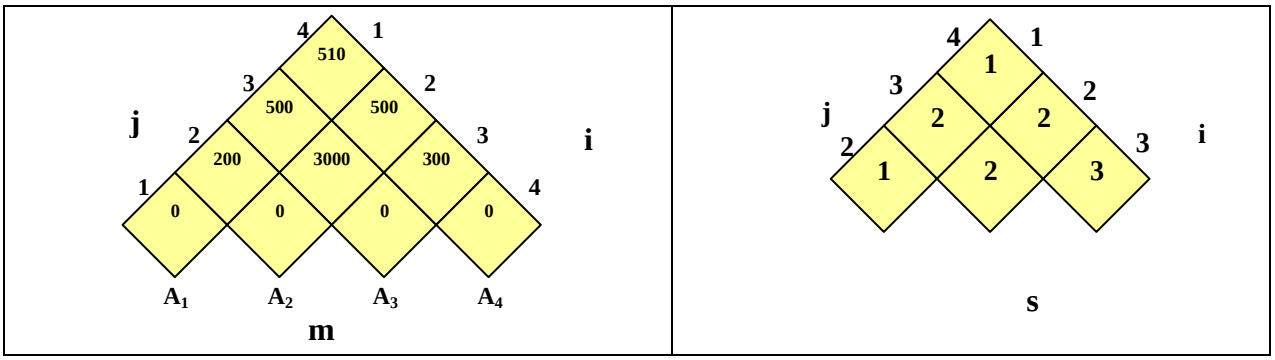
Matrix-Chain-Order(n)

```
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
                        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
                                s[i, j] ← k
return m and s
```

בקטע קוד הזה אנחנו מחפשים ומכניסים ל - s[i,j] את הערך שמפצלים בו את המכפלה $A_i \cdot ... \cdot A_j$ כדי לקבל הצבת סוגריים אופטימאלית ע"י בדיקת הערכים k=i,..,j-1 ומציאת המינימום.

האיור הבא מדגים תהליך זה על המטריצות $A^1_{1\times10}$, $A^2_{10\times20}$, $A^3_{20\times15}$, $A^4_{15\times1}$. מכיוון שהגדרנו את m[i,j] עבור $i \le j$ בלבד, אנו משתמשים רק במשולש הנמצא מעל האלכסון הראשי של הטבלה. באיור הבא, הטבלאות מסובבות כך שהאלכסון הראשי הוא בכיוון האופקי. העלות המינימאלית m[i,j] של חישוב תת המכפלה

$A_i \cdot \ldots \cdot A_j$ תימצא בתא שבנקודת החיתוך בין הקו היוצא מ- A_i לכיוון צפון-מזרח, והקו היוצא מ- A_j לכיוון צפון-מערב. כל שורה אופקית בטבלה מכילה תאים עבור סדרות מטריצות באותו האורך. השגרה ***Matrix-Chain-Order(n)*** מחשבת את השורות מלמטה למעלה ובכל שורה – משמאל לימין. ערך התא m[i,j] מחושב תוך שימוש במכפלות $p_{i-1}p_k p_j$ עבור $k = i, i+1, \ldots, j-1$ ובכל ערכי התאים הנמצאים מדרום-מערב ומדרום-מזרח ל-m[i,j].



הוכחת נכונות של האלגוריתם האיטראטיבי

יש לוודא כי בעת חישוב תא m[i,j] כל התאים בהם תלוי החישוב של תא זה חושבו כבר ואלו מכילים ערכי פתרונות אופטימאליים עבור תתי-הבעיות אותן הם מייצגים.
יש להוכיח את שתי הטענות הבאות:

טענה(ניתן להראות באינדוקציה):
בעת חישוב התא ה- m[i,j] כל התאים בשורה ה- i מהעמודה ה-i ועד העמודה ה-(j-1) וכל התאים בעמודה ה-j מהשורה ה-(i+1) ועד השורה ה-j כבר חושבו על מבנה הנתונים (במקרה זה מטריצה).

טענה: (הטענה שמנסחת את הנוסחה הרקורסיבית):
התא m[i,j] אכן שווה ל- OPT(i,j) לאחר החישוב.

כלומר יש להראות (באינדוקציה) שערך התא שווה לערך שמחושב על ידי הנוסחא שהוגדרה קודם

$$OPT[i,j] = \begin{cases} 0, & ,i = j \\ \min\limits_{i \le k < j} \{OPT[i,k] + OPT[k+1,j] + p_{i-1} \cdot p_k \cdot p_j\} & ,i < j \end{cases}$$

שחזור: כיצד למצוא סדרת הכפלות אופטימאלית ולא רק עלות

בזמן מילוי הטבלה, בכל תא נחזיק גם את ה-k שממנו הגענו לתוכן התא. כלומר, את הערך k שמגדיר כיצד לחלק את בעיית ההכפלה $\{A_i,...,A_j\}$ בצורה אופטימאלית. בעזרת ערכים אלו נוכל ע"י הליכה "אחורה" מהתא $[1,n]$ ו"איסוף" ה-k-ים לשחזר את הצבת הסוגריים.

לדוגמא, במקרה של המטריצות שלנו $A^1_{1\times10}, A^2_{10\times20}, A^3_{20\times15}, A^4_{15\times1}$:

<u>שלב 1:</u> $l=2$

j / i	1	2	3	4
1	0	0+0+p0*p1*p2 = 1* 10*20 = 200 K = 1		
2	X	0	0 + 0 + p1*p2*p3 = 10 * 20 * 15 = 3000 K = 2	
3	X	X	0	0+ 0 +p2*p3*p4 = 20*15*1 = 300 K=3
4	X	X	X	0

<u>שלב 2:</u> $l=3$

j / i	1	2	3	4
1	0	200 k=1	200+0+p0*p2*p3 = 200+1*20*15 = 500 k=2	
2	X	0	3000 K=2	0+300+p1*p2*p4 = 300+10*20*1 = 500 k=2
3	X	X	0	300 K=3
4	X	X	X	0

<u>שלב 3:</u> $l=4$

	1	2	3	4
1	0	200 k=1	500 k=2	0+500+p0*p1*p4= 500+1*10*1 = 510 k=1
2	X	0	3000 K=2	500 k=2
3	X	X	0	300 K=3
4	X	X	X	0

<u>ערך הפתרון האופטימאלי נמצא ב $[1,4]$:</u>

	1	2	3	4
1	0	200 k=1	500 k=2	510 k=1
2	X	0	3000 K=2	500 k=2
3	X	X	0	300 K=3
4	X	X	X	0

שיחזור: נסתכל על התא $[1,4]$. כיוון ש $k=1$ ההכפלה האחרונה בסדרה היא בין תת הסדרה $\{A_1\}$ ותת הסדרה $\{A_2,A_3,A_4\}$, ז"א $A_1\times(A_2\times A_3\times A_4)$. כעת נביט בתא $[2,4]$ על מנת למצוא את ההצבה האופטימאלית של הסוגריים בתוך $\{A_2,A_3,A_4\}$. בתא זה $k=2$, משמע סדר ההכפלות הוא $A_1\times(A_2\times(A_3\times A_4))$, ובזאת סיימנו את השחזור.

ניתוח זמן :

מילוי הטבלה: ישנם $O(n^2)$ תאים בטבלה. n תאי האלכסון מאותחלים בזמן קבוע לכל תא.

כל תא $[i,j]$ מצריך בדיקה של $j-i$ זוגות של תאים שכבר חושבו, ובמקרה הכללי $O(n)$ פעולות לכל תא.

סה"כ זמן מילוי הטבלה: $O(n^3)$.

שיחזור ההצבה – הליכה לאחור $n-1$ שלבים כשכל צעד בזמן קבוע - $O(n)$.

וכמה מקום צריך?

לסיכום: בהינתן בעיה שאתם שוקלים לפתור אותה (חושדים או יודעים שפתרונה הוא) ע"י תכנון דינאמי, אלה השלבים שמרכיבים את הפתרון:

- הגדרת הבעיות באופן מילולי והגדרת OPT.
- ניסוח נוסחת המבנה, כולל מקרי בסיס ומיקום הפתרון לבעיה המקורית.
- הוכחת נכונות של נוסחת המבנה:
  - הגדרת תתי-קבוצות של פתרונות אפשריים לפי חלוקה למקרים.
  - הוכחה שתתי-הקבוצות הנ"ל מכסות את כל הפתרונות האפשריים.
  - הסקת הצורה הסכמאטית של נוסחת המבנה.
  - ניתוח כל תת-קבוצת פתרונות בנפרד, והוכחת המרכיבים המתאימים בנוסחת המבנה.
- מימוש האלגוריתם – איטראטיבי/רקורסיבי למציאת ערך הפתרון האופטימאלי ו/או הפתרון האופטימאלי עצמו (לפי הדרישה).
- הוכחת נכונות של האלגוריתם:
  - האלגוריתם פועל בהתאם לנוסחת המבנה.
  - בכל שלב בריצת האלגוריתם, כל ערך לו האלגוריתם זקוק, חושב באחד מהשלבים הקודמים.
- ניתוח זמן ריצה.