19/11 בס"ד

14

נושא 3 - תכנון דינאמי

שיטת התכנון הדינאמי היא שיטה לבניית אלגוריתם לפתרון בעיות שאינן ניתנות לפתרון יעיל בשיטת הפרד ומשול. בשיטת הפרד ומשול פותרים בעיה ע"י חלוקתה לתת בעיות קטנות יותר, שאז אותן אנו מחלקים לתתי בעיות עד שמתקבלות תתי בעיות קטנות מספיק שיהיה ניתן לפתור אותן באופן ישיר. אמנם ישנן בעיות שלאחר שחולקו למספר תתי בעיות יש בהן חלקים משותפים. בשיטת הפרד ומשול נאיבית נפתור את החלקים המשותפים בכל תת בעיה בנפרד, מה שבבעיות מסוימות עלול לגרום לכך שהפתרון יתבצע בסיבוכיות גבוהה מאוד. האלטרנטיבה שמציעה שיטת התכנון הדינאמי היא לפתור את כל התת בעיות באופן סדרתי אחת אחרי השניה, לאחסן את כל הפתרונות לשימוש עתידי (בד"כ נכניס את הפתרונות למטריצה). באלגוריתמים אלו יותר יעיל לפתור את כל תתי הבעיות האפשריים, גם כאלו שלא בהכרח נדרכים לפתרון הבעיה הגדולה, מאשר לפתור את אותן תתי בעיות שוב ושוב מספר רב של פעמים.

באלגוריתם הפרד ומשול אנו פותרים מלמעלה למטה, ואילו בתכנון דינאמי פותרים מלמטה למעלה.

קווי פעולה למתי צריך להשתמש התכנון דינאמי:

(1) מציגים תיאור רקורסיבי לפתרון בעיה.

(2) מגלים כי זמן ריצת האלגוריתם הרקורסיבי הוא אקספוננציאלי.

(3) מגלים כי מספר תת הבעיות הוא פולינומי, והסיבה לסיבוכיות אקספוננציאלית היא קריאה חוזרת לאלגוריתם עבור אותה תת בעיה.

(4) פותרים את הבעיה ע"י פתרון כל תתי הבעיות-עד שמגיעים לבעיה המקורית.

1) סדרת פיבונאצ'י

א) תיאור הבעיה: אם נרצה למצוא את האיבר ה-n בסדרת פיבונאצ'י, נוכל להשתמש בהגדרה הרקורסיבית של הסדרה, כאשר תנאי העצירה הוא $F(1)=1$, $F(2)=1$. אמנם בדרך רקורסיבית נבצע את אותם חישובים פעמים רבות:

$$F(5)=F(4)+F(3)=\overbrace{F(3)+F(2)}^{F(4)}+\overbrace{F(2)+F(1)}^{F(3)}=F(2)+F(1)+F(2)+F(2)+F(1)$$

דרך כזאת לא יעילה שכן בכל שלב מכפילים כל איבר לשניים, ולכן זמן הריצה הוא $\Omega(2^n)$.

ב) פתרון בתכנון דינאמי: במקום ללכת מלמעלה למטה נחשב את כל הערכים מלמטה למעלה, כלומר מ-$F(1)$ ועד $F(n)$, ונאחסן אותם במערך. כך נמצא את $F(n)$ בזמן ריצה של $\Theta(n)$.

19/11 בס"ד

5

2) מקדם בינומי

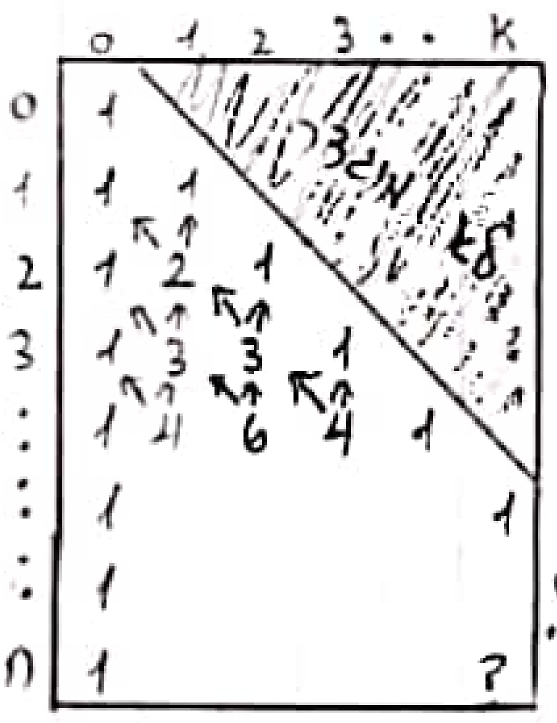
א) <u>תיאור הבעיה:</u> יש שני דרכים לחשב מקדם בינומי $\binom{n}{k}$. הדרך הראשונה היא באמצעות חישוב ישיר $\frac{n!}{k!(n-k)!}=\binom{n}{k}$, אמנם חישוב של עצרת בס"ת למחשב, משום שגם ערכים נמוכים של עצרת תהפוך למספר גדול ויכול להיגרם גיבוב מידע. הדרך השנייה היא באמצעות חישוב רקורסיבי $\binom{n}{k}=\binom{n-1}{k}+\binom{n-1}{k-1}$. אמנם דרך רקורסיה פשוטה נבצע את אותם החישובים פעמים רבות:

$$\binom{n}{k}=\binom{n-1}{k}+\binom{n-1}{k-1}=\binom{n-2}{k}+\underline{\binom{n-2}{k-1}}+\underline{\binom{n-2}{k-1}}+\binom{n-2}{k-2}$$

כאן הריצה היא: $T(n,k)\ge 2T(n-2,k-1)\ge 2^2T(n-4,k-2)\ge\dots\ge 2^{\frac{n}{2}}T(0,0)=\Omega\left(\sqrt{2^n}\right)$

תוספנו לפי המקרה הגרוע כאשר $k=\frac{n}{2}$. במקרה זה כמות הפעולות המינימלית היא $\Omega(\sqrt{2^n})$.

ב) <u>פתרון התכנון דינאמי:</u> נבנה טבלה עם k עמודות ו-n שורות. כל ערך (i,j) בטבלה ייצג את מקדם הבינומי $\binom{i}{j}$. בכל תא שבו $i<j$ יהיה ערך ריק כי בהו ערך לא מוגדר.



כאשר $i=j$ (אלכסון) או כאשר $j=0$ יהיה בתא 1. שאר התאים חם חיבור של התא מעליהם והתא משמאלו, בהתאם להגדרה הרקורסיבית של מקדם בינומי $\binom{n}{k}=\overset{\text{מעל}}{\binom{n-1}{k}}+\overset{\text{תא מעל משמאל}}{\binom{n-1}{k-1}}$. הבעיה בו עדיף לנו למלא את כל הטבלה שהסיבוכיות היא <u>$O(nk)$</u>, מאשר הדרך רקורסיבית כמו שניסינו לעיל.

ג) <u>פסאודו-קוד:</u>

```
DP-Binom(n, k) {
    C[n+1][k+1]        // two-dimension array
    for(i=0 to n) {
        for(j=0 to min(i,k)) {
            if(j=0 or j=i)  C[i][j]=1
            else  C[i][j]=C[i-1][j-1]+C[i-1][j]
        }
    }
    return C[n][j]
}
```

3) בעיית הטוב מ-7

א) <u>תיאור הבעיה:</u> שתי קבוצות A ו-B מתחרות ביניהם מי הראשונה שתנצח n משחקים. אנו מניחים כי לשתי הקבוצות סיכוי שווה של 50% לנצח במשחק בודד. נגדיר $p(i,j)$ ההסתברות <u>שקבוצה A תנצח</u> כאשר A צריכה לנצח i משחקים ו-B צריכה לנצח j משחקים. נשים לב כי $p(i,0)=0$ ו-$p(0,j)=1$.

כדי לחשב $P(i,j)$ כללי, נוכל להשתמש בנוסחה הרקורסיבית: $P(i,j)=\frac{1}{2}P(i-1,j)+\frac{1}{2}P(i,j-1)$.
כלומר, חצי כפול ההסתברות ש-A תנצח ועוד חצי כפול ההסתברות ש-B תנצח.
נחשב את זמן הריצה של $P(i,j)$ אותו נסמן $T(i,j)$: $T(0,j)=T(i,0)=O(1)$

$$T(i,j)=T(i-1,j)+T(i,j-1)=T(i-2,j)+\underline{T(i-1,j-1)}+\underline{T(i-1,j-1)}+T(i,j-2)$$

$$> 2T(i-1,j-1) > 2^2(i-2,j-2) > \dots > 2^k T(i-k,j-k)$$

עבור כאשר $k=\min(i,j)$, לכן זמן הריצה הוא: $\Omega\left(2^{\min(i,j)}\right)$. כלומר
תכנית זו אינה יעילה.

ב) <u>פתרון בתכנון דינאמי:</u> נכתוב פונקציה $P(n,m)$ (נשתמש ב-n ו-m ולא ב-i ו-j כדי למנוע בלבול בטבלאות) הבונה טבלה עם n שורות ו-m עמודות, כך שכל תא בטבלה (i,j) יכיל את ההסתברות $P(i,j)$. השורה הראשונה בטבלה כאשר $i=0$ יהיה שורת 1, והעמודה השמאלית כאשר $j=0$ תהיה עמודת 0.

	0	1	2	3	....		m
0	0	1	1	1	1	1	1
1	0	1/2	3/4	7/8			
2	0	1/4	1/2				
3	0	1/8	5/16				
⋮	0						
⋮	0						
n	0						?

הערך $P(0,0)$ לא מוגדר לכן נכניס לתוכו 0, כל שאר התא'ים הם סכום התא שמעליהם והתא שמשמאלם כפול חצי.
סיבוכיות פונקציה זו היא $\underline{O(n\cdot m)}$, יותר טוב משמעותית מרקורסיה.

ג) <u>פסאודו-קוד:</u>

```
P(n,m) {
    C[n+1][m+1]          // two-dimension array
    for (i=0  to n)   C[i][0] = 0
    for (j=1  to m)   C[0][j] = 1
    for (i=1  to n) {
        for (j=1 to n) {
            C[i][j] = 1/2 · C[i-1][j] + 1/2 C[i][j-1]
        }
    }
    return C[n][m]
}
```

4) <u>הכפלת מטריצות</u>

א) <u>תיאור הבעיה:</u> בהינתן n מטריצות $M_1, M_2, \dots, M_n$, כך שכל מטריצה M_i המימדים הם: $d_{i-1}\times d_i$, כלומר מספר השורות הוא מספר העמודות של מטריצה M_{i-1}. נרצה לחשב את תוצאת הכפל בין כל המטריצות $M_1\cdot M_2\cdot\dots\cdot M_n$ עם מספר מינימלי של הכפלות סקלריות:
נשים לב שלפי הגדרת המימדים ניתן להכפיל כל שתי מטריצות. בהכפלת מספר מטריצות

26/11

(א)

לא ניתן לשנות את המיקום של מטריצה (אין קומוטטיביות), אך כן ניתן לשנות את סדר ההכפלה, כלומר ניתן לשים סוגריים היכן שנרצה (אסוציאטיביות). למיקום הסוגריים חשיבות מכרעת בהכפלת מספר מטריצות.*

$\underset{2\times5}{M_1}\cdot \underset{5\times3}{M_2}\cdot \underset{3\times4}{M_3}$

$\underset{2\times5}{M_1}(\overset{60}{\underset{5\times4}{M_2\cdot M_3}}) = 60+40=100$

$\overset{30}{\underset{2\times3}{(M_1 M_2)}}\underset{3\times4}{M_3} = 30+24=54$

בהכפלת שתי מטריצות: A עם מימדים $p\times q$ ו-B עם מימדים $q\times r$, נקבל מטריצה C עם מימדים $p\times r$, ומספר ההכפלות היא $p\cdot q\cdot r$. $\underset{p\times q}{A}\cdot \underset{q\times r}{B} = \underset{p\times r}{C} = \underset{\text{הכפלות}}{p\cdot q\cdot r}$.

נסמן ב-$m(i,j)$ את מספר ההכפלות המינימלי של המטריצות $M_i, M_{i+1}, \dots, M_{j-1}, M_j$ כאשר $1\le i\le j\le n$. מטרתנו היא בסוף לחשב את $m(1,n)$.

בהינתן מטריצות $M_i\cdot M_{i+1}\cdots M_{j-1}\cdot M_j$ נוכל לחשב את $m(i,j)$ בדרך רקורסיבית, כאשר אנו מחלקים את ההכפלה לשתיים: מ-i עד k ומ-k+1 עד j, ומניחים שאנו יודעים לחשב את $m(i,k)$ ו-$m(k+1,j)$, מכיוון שתוצאת המכפלה השמאלית היא מטריצה עם מימדים $d_{i-1}\times d_k$ ותוצאת המכפלה הימנית היא מטריצה עם מימדים $d_k\times d_j$, נצטרך להוסיף את מספר ההכפלות בין שתי מטריצות אלו, שהוא $d_{i-1}\cdot d_k\cdot d_j$. מכיוון שאנו לא יודעים איזה k הוא אופטימלי נחשב עבור כל $i\le k<j$, וניקח את ה-k שמחזיר מספר הכפלות סקלרים מינימלי.

תנאי העצירה לרקורסיה היא כאשר $i=k$ שאז מספר המכפלות הוא 0.

הנוסחה הרקורסיבית היא:

$$m(i,j)=\begin{cases}0 & i=j\\ \min\limits_{i\le k<j}\left(m(i,k)+m(k+1,j)+d_{i-1}\cdot d_k\cdot d_j\right) & i<j\end{cases}\qquad (\underbrace{M_i\cdots M_k}_{d_{i-1}\times d_k})(\underbrace{M_{k+1}\cdots M_j}_{d_k\times d_j})$$

זמן הריצה בשיטה זו עבור $m(1,n)$ אותו נסמן ב-$T(n)$ הוא:

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

$T(k)$ ו-$T(n-k)$ הם שקולים ב-Σ רק בכיוון ההפוך.

$$= n-1+2\sum_{k=1}^{n-1}T(k).$$

לפי מה שקיבלנו בשורה למעלה נחשב את $T(n-1)$.

$$T(n-1)=n-2+2\sum_{k=1}^{n-2}T(k).$$

נחסר ונעביר אגף

$$T(n)-T(n-1)=\left(n-1+2\sum_{k=1}^{n-1}T(k)\right)-\left(n-2+2\sum_{k=1}^{n-2}T(k)\right)=1+2T(n-1).$$

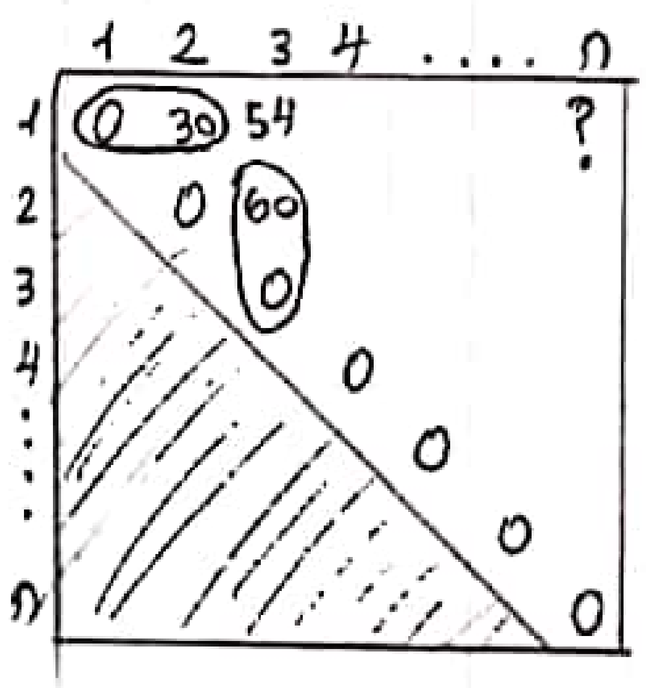
$$T(n)=1+3T(n-1)>3T(n-1)>3^2T(n-2)>\dots>3^kT(n-k)$$

נעצור כאשר $n=0$, ומכאן $k=n$ ולכן זמן הריצה הוא: $\Omega(3^n)$.

(ב) <u>פתרון בתכנון דינאמי</u>: נבנה טבלה עם n שורות ו-n עמודות, כך שבכל תא בטבלה יהיה הערך $m(i,j)$ שהוא מספר ההכפלות המינימלי. נרצה להחזיר $m(1,n)$. נשים לב שכל תא שבו $j<i$ לא מוגדר, ולכן כל תא מתחת לאלכסון ראשי הוא ריק. בנוסף בכל תא באלכסון ראשי שבו $i=j$ הוא 0. נמלא את הטבלה החל מאלכסון ראשי

26/11 בס"ד

(18)



ונעלה באלכסון מעלה, כך אלכסון אחר אלכסון, עד ל-$m(1,n)$.
כל תא $m(i,j)$ נחשב לפי הנוסחא הרקורסיבית, כאשר אנו בודקים
בכל k אפשרי $i \le k < j$ מהו הסכום המינימאלי לפי הנוסחא.
לאחר שמצאנו מהו הסכום המינימאלי נחשב את הערך $m(i,j)$ לפיו.

ג) פסאודו קוד:

```
mul(n) {
    m[n,n]          // two-dimension array
    For (i=1; i≤n; i++)  m(i,i)=0
    For (diff=1; diff≤n-1; diff++) {     // move across the diagonal line
        For (i=1; i≤n-diff; i++) {
            j = i+diff
            x = ∞
            For (k=i; k≤j-1; k++) {
                y = m[i,k] + m[k+1,j] + d_{i-1}·d_k·d_j
                IF (y<x)   x = y        // take the sum for min k
            }
            m[i,j] = x
        }
    }
    return m[1,n]
}
```

ניתן לראות שעוברים בשלוש לולאות והחישובים שבתוך הלולאות הם קבועים לכן
זמן הריצה הוא $O(n^3)$.

5) בעיית תרמיל הגב השלמים

א) תיאור הבעיה: בשיעור בפרק אלגוריתמים חמדניים (שיעור 2) הראינו בעיה שנקראת "בעיית תרמיל הגב". נחזור על הבעיה שוב. נתונים n עצמים, כך שלכל עצם i נגדיר w_i משקל העצם ו-v_i ערכו. ישנו תרמיל המסוגל להכיל עד C קילוגרם. מטרתנו היא למלא את השק בעצמים כך שלא נחרוג מהקיבולת, וערך העצמים מקסימאלי. הוכחנו שם שהבחירה החמדנית לקחת את העצם שלו ה-$\frac{v_i}{w_i}$ הוא הכי גבוה, מחזיר פתרון אופטימאלי אם ניתן לקחת שבר מהעצם, אך לא אם חייב לקחת את העצם כולו, ולכן הבאנו דוגמא נגדית*. כעת ננסה למצוא לבעיית תרמיל הגב השלמים פתרון אופטימאלי באמצעות תכנון דינאמי.

i	v_i	w_i
1	60	1
2	100	2
3	120	3

$C = 5$

(19)

ב) פתרון בתכנון דינאמי: נגדיר $A[i,j]$ להיות הערך המקסימלי כאשר ניתן לבחור מ-i הפריטים הראשונים $\{1,2,...,i\}$ אל תרמיל שקיבולו j, כאשר $0\le i\le n$ ו- $0\le j\le c$. מטרתנו הסופית היא להחזיר את $A[n,c]$.

נבנה טבלה עם n שורות ו-c עמודות, כך שבכל תא (i,j) יהיה הערך המקסימלי $A[i,j]$.
כאשר $i=0$ או $j=0$ נקבל $A[0,j]=A[i,0]=0$, מפני שכאשר לא ניתן להכניס עצמים לתיק או שאין עצמים לבחור מהם, הערך הוא 0. לכן השורה העליונה והעמודה השמאלית מלאות ב-0. בכל תא אחר נחלק את החישוב לשני מקרים:

(1) כאשר $j<w_i$ - כלומר את פריט i לא ניתן להכניס לתרמיל, מפני שהוא כבד מדי, נחשב באותו תא את $A[i-1,j]$, שהוא התא מעליו. כלומר ננסה להכניס את הערך המקסימלי מ-$i-1$ הפריטים הראשונים.

(2) כאשר $w_i\le j$ - כלומר ניתן להכניס את פריט i לתרמיל, אז יש שני אפשרויות: או שבחרנו את פריט i, ואז צריך למצוא מהו הערך המקסימלי שנבחר מאשר הפריטים $A[i-1,j-w_i]+v_i$ או שלא בחרנו את i ואז החישוב הוא $A[i-1,j]$. ניקח את הערך המקסימלי מבין שני ערכים אלו. כלומר כל תא הוא או הערך מעליו או הערך שמשמאל לערך מעליו בתוספת v_i.

	0	1	2	...
0	0	0	0	0 0 0
1	0			
2	0			
⋮	0			
	0			
n	0			

לסיכום:

$$A[i,j]=\begin{cases}0 & i=0 \text{ או } j=0\\ A[i-1,j] & w_i>j\\ \max(A[i-1,j],\ A[i-1,j-w_i]+v_i) & w_i\le j\end{cases}$$

חישוב כל תא הוא $O(1)$ מפני שמסתמך על חישובים קודמים. לכן הסיבוכיות היא $O(c\cdot n)$.

ג) פסאודו-קוד:

```
Integer_Knapsack(n, c) {
    For(i=0 to n)   A[i,0] = 0
    For(j=0 to c)   A[0,j] = 0
    For(i=1 to n) {
        For(j=1 to c) {
            IF(Wi > j)   A[i,j] = A[i-1,j]
            else         A[i,j] = max(A[i-1,j], A[i-1,j-Wi]+Vi)
        }
    }
    return A[n,c]
}
```

תרגיל דוגמא נגדית

i	v_i	w_i
1	60	1
2	100	2
3	120	3

$c=5$

		0	1	2	3	4	5
	0	0	0	0	0	0	0
$v_i=60$	1	0	60	60	60	60	6
$v_i=100$	2	0	60	100	160	160	16
$v_i=120$	3	0	60	100	160	180	22

return 220 ✓