אלגוריתמים
הרצאה 9
29.12.08
9.12.08

אלגוריתמים - הרצאה 9

המשך תכנון דינמי...

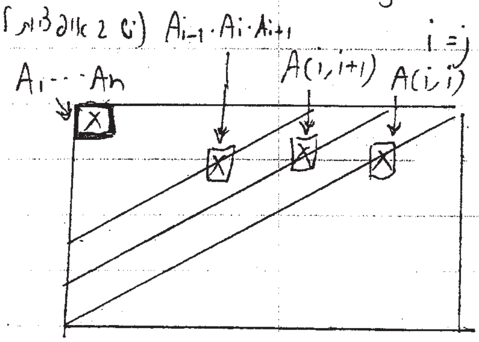
הדרישות בתכנון דינמי:

1) רקורסיה - מתקבלת מתוך תכונות של הפתרון האופטימלי. הקריאה הרקורסיבית תהיה על כל האפשרויות ל-k. הפתרון של $A_1\cdots A_n$ הוא: $Cost(A_1\cdots A_n)=\min_k\{Cost(A_1\cdots A_k)+Cost(A_{k+1}\cdots A_n)+$ הכפל הסופי$\}$

2) שיתוף בין תת בעיות - מאפשרת להוריד את זמן הריצה מאקספוננציאלי לפולינומי.
דוגמה טובה לכך זה מספרי פיבונאצ'י (אין צורך לחשב כמה פעמים את אותו דבר)

נחשב את: $A_1\cdots A_n$ בצורה אופטימלית, הצורה הכללית של כל תת הבעיות שאנחנו רוצים לפתור זה רצף של הכפלות מהצורה הזו: $A_i\cdots A_j$

נייצג זאת ע"י טבלה, שהאלכסונים יהיה המקרה בו $j=i$



נמלא את הטבלה אלכסון אחרי אלכסון A(i,i) עד שנגיע למשבצת המודגשת שם נקבל את התשובה.

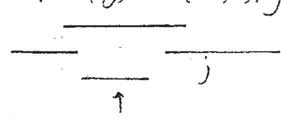
איך נדע באילו נקודות לעשות שבירות אופטימליות?
צריך לזכור בכל שלב איפה התקבל ה-min (מה היתה נקודת השבירה המינימלית).
באמצעות אינפורמציה זו נדע איפה למקם את הסוגריים.

דוגמה:

יש אוסף של משימות $I_1, I_2, \ldots, I_n$. לרשותנו עומדת מכונה יחידה, את המשימות חייבים לבצע ברצף. לכל משימה יש נקודת התחלה ונקודת סיום. s_i - התחלה, f_i - סיום.
אם מבצעים משימה i אז היא חייבת להתחיל ב s_i, להתבצע ברצף ולסיים ב f_i.
רק משימה אחת יכולה להתבצע בו"ז במכונה.

אם משימה I_i מתבצעת בהצלחה הרווח הוא P_i,
המטרה שלנו היא למקסם את הרווח, כלומר למצוא שיבוץ חוקי S (בלי חיתוכים בין האינטרוולים) כך שנקבל: $\max_S \sum_{i\in S} P_i$
(מקרה פרטי $P_i=1$ לכל i ואז המטרה היא למקסם את מספר המשימות)
(מקרה נוסף למקסם את הזמן שהמכונה רצה כי $P_i=$ אורך האינטרוול).

נניח שהמשימות ממוספרות עפ"י זמן הסיום, כלומר I_1 מסתיימת ראשונה לכן: $f_1<f_2<\ldots<f_n$
לכל אינטרוול j נגדיר את $p(j)$: היות האינטרוול בעל האינדקס המקסימלי כך ש $f_{p(j)} \le s_j$
לדוגמה:



זה יהיה אינטרוול $p(j)$ כי מסתיים הכי קרוב לפני j.

נגדיר רקורסיה לבעיה:

אפשרות 1: I_n שייך ל OPT (לפיתרון האופטימלי)

אפשרות 2: I_n לא שייך ל OPT (לפיתרון האופטימלי)

באפשרות 2: הפיתרון האופטימלי על $I_1 \cdots I_n$ הוא האופטימלי גם עבור $I_1 \cdots I_{n-1}$
כי I_n לא נמצא בפיתרון האופטימלי ולכן משרה פיתרון אופטימלי על I_{n-1}.
באפשרות 1: אין ב OPT אף אינטרוול עם אינדקס j כך ש: $d(n) < j < n$
ולכן OPT משרה פיתרון אופטימלי עבור: $I_1 \cdots I_{d(n)}$

נוסחאת הרקורסיה שלנו היא:

$$OPT(I_1, \ldots, I_n) = \max\{OPT(I_1, \ldots, I_{d(n)}) + P_n, OPT(I_1, \ldots, I_{n-1})\}$$

הדוגמה הרקורסיבית שלנו היא:

```
compute-Opt(n)
    if n=0 then
        Return 0
    Else
        V=Compute-Opt(d(n))
        V'=compute-Opt(n-1)
        if Wn+V ≥ V'
            Return Wn+V
        Else
            Return V'
        Endif
    Endif
```

הבעיה בפונקציה הזו שהיא אקספוננציאלית.
נהפוך את הפתרון לפיתרון עם מימוש פולינומי.
הטבלה שלנו היא מערך: $M(0, \ldots, n)$, $M(i)$ מהווה את הפתרון האופטימלי עבור האינטרוולים: $I_1, \ldots, I_i$
נממש פונקציה כזו.

```
M-Compute-Opt(n)
    if M(n)="empty" then
        V=Compute-Opt(n)
        M(n)=V
        Return V
    Else Return M(n)
```

אלגוריתמים
הרצאה 9
29.12.08
דף 2 מ 2

הקריאה הרקורסיבית הראשונה נעשית ל M-compute-opt עם הפרמטר n
(M מאותחל ל "empty")

כל פעם שיש קריאה רקורסיבית בודקים האם הערך של M(n) כבר נמצא בטבלה,
אם לא אז עדיין לא חישבנו את הערך הזה ולכן עושים קריאה רקורסיבית ל Compute-opt

<u>הסיבוכיות של האלגוריתם הוא:</u> O(n) בלי לקחת בחשבון את מיון האינטרוולים
ואת האתחול.

מידת ההתקדמות תהיה מספר הערכים שאינם "empty" במערך M.
בהתחלה מידת ההתקדמות היא 0. בכל קריאה ל compute-opt מספר
הערכים שאינם "empty" גדל ב-1. לכן מספר הקריאות ל compute-opt
הוא לכל היותר: n+1. בכל קריאה ל compute-opt נוצרות 2 קריאות ל compute-opt
לכן מספרם חסום ע"י: 2(n+1). כל שאר הפעולות אפשר לחסום ע"י קבוע
לכן הסיבוכיות הוא: O(n).

<u>בעיית תת הסדרה הארוכה ביותר</u>

נתונה סדרה מעל א"ב סופי: C A E A D

דוגמה לתת סדרה: C A A (כלומר האיברים בתת סדרה מופיעים באותו הסדר,
שהופיעו בסדרה המקורית אך לא חייבים להיות רצופים)

<u>הבעיה שרוצים לפתור:</u> נתונות 2 סדרות: $X=x_1,x_2,\ldots,x_n$; $Y=y_1,y_2,\ldots,y_m$
וצריך למצוא את תת הסדרה באורך מקסימלי
שמשותפת לשתי הסדרות. (נראה איך אפשר לעשות זאת באמצעות תכנון דינמי).

נסמן: X_i את האיברים $x_1,x_2,\ldots,x_i$ (כלומר הרישא של X עד המקום ה-i) אופטימלית

<u>משפט:</u> נניח ש $Z=z_1,z_2,\ldots,z_k$ היא תת סדרה של X ו Y. (תת סדרה שהיא LCS)
אזי מתקיימים לפחות אחד משני התנאים הבאים:

1) אם $x_n=y_m$ אז: $z_k=x_n=y_m$ ו z_k הוא LCS של X_{n-1} ו Y_{m-1}

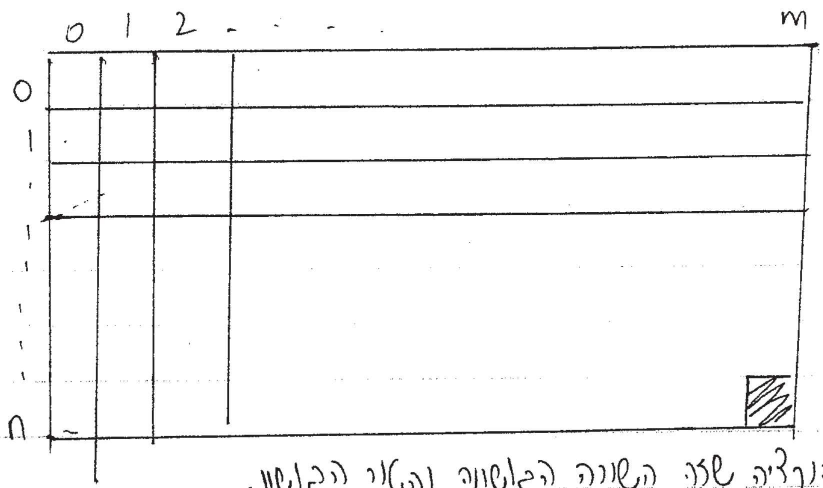
2) אם $x_n\neq y_m$ אז: אם $z_k\neq x_n$ אז Z הוא LCS של X_{n-1} ו Y.
אם $z_k\neq y_m$ אז Z הוא LCS של X ו Y_{m-1}

נגדיר טבלה כאשר $C(i,j)$ יהיה אורך ה LCS של X_i ו Y_j.

$C(i,0)=0$, $C(0,j)=0$

$$C(i,j)=\begin{cases} 0 & i=0 \lor j=0 \\ C(i-1,j-1)+1 & \text{if } i,j>0,\ x_i=y_j \\ \max(C(i,j-1),C(i-1,j)) & \text{if } i,j>0,\ x_i\neq y_j \end{cases}$$

נבנה טבלה:



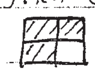
נתחיל מבסיס האינדוקציה שזה השורה הראשונה והטור הראשון.
נמשיך לשורה השנייה והטור השני והשאיפה היא להגיע בסוף לריבוע המושחר.
נבצע דוגמת ריצה:

	0	B 1	D 2	C 3	A 4	B 5	A 6
0	0	0	0	0	0	0	0
A 1	0	0	0	0	1	1	1
B 2	0	1	1	1	1	2	2
C 3	0	1	1	2	2	2	2
B 4	0	1	1	2	2	3	3
D 5	0	1	2	2	2	3	3
A 6	0	1	2	2	3	3	4
B 7	0	1	2	2	3	4	4

← בסיס האינדוקציה

בסיס האינדוקציה

איך הגענו לסימנים המתאימים?
החל מהמקום שמצאנו התאמה
בטור ובשורה סימנו 1
כשמצאנו התאמה נוספת בשורה
הבאה הוספנו 1 למס האלכסוני לו.



סיבוכיות האלגוריתם:

A^2 ערכים בטבלה כל אחד $O(1)$ פעולות כי בודקים
3 ערכים נוספים ולכן סיבוכיות: $O(m\cdot n)$

דוגמה - Subset Sum:

קלט: $a_1,...,a_n$ ועוד K
שאלה: האם יש תת קבוצה של $a_1...a_n$ שמסתכמת בדיוק ל-K?
הפיתרון: נחזיק טבלה P כך ש: $P(i,K)=True$ אם יש תת קבוצה מתוך
$a_1,...,a_i$ שמסתכמת ל-K. $0\le K\le \sum_{j=1}^{i} a_j$
איך נבנה את הטבלה?
$P(0,K)=False$ בהנחה שכל הערכים מהצורה $P(j,K)$ $0\le j\le i$
כבר ידועים, כיצד נחשב את $P(i,K)$?
אם $P(j,K)=True$ כך ש $j<i$ אזי גם: $P(i,K)=True$, בנוסף: אם: $P(j,K)=True$
אזי גם: $P(i,K+a_i)=True$