תכנון דינאמי

גיא פליישר 2006/7

תרגיל – תת מחרוזת משותפת ארוכה ביותר

תת-מחרוזת $Z = z_1 z_2, \dots z_k$ של מחרוזת נתונה מעל א"ב Σ, היא מחרוזת כך שקיימת סדרה עולה ממש $i_1, i_2, \dots, i_k$ של אינדקסים כך שלכל $j = 1, 2, \dots, k$ מתקיים כי $x_{i_j} = z_j$.

דוגמה:

כפגחקבבדטדגובסרכ

Z היא תת-מחרוזת משותפת של X ו- Y אם היא תת-מחרוזת של X וגם תת-מחרוזת של Y.

תת מחרוזת משותפת ארוכה ביותר

סימון: עבור מחרוזת נתונה X מאורך m, נסמן ב- X_i, $0 \le i \le m$, את הרישא הבאה של X: $X_i = x_1 x_2 \ldots x_i$ (עבור $i = 0$ נקבל רישא ריקה).

יהיו X ו- Y שתי מחרוזות מעל א״ב Σ מאורך m ו-n בהתאמה.

אנו רוצים למצוא את תת המחרוזת המשותפת הארוכה ביותר של X ו- Y.

טענה

תהא $Z = z_1 z_2 ,...z_k$ תת-המחרוזת המשותפת הארוכה ביותר של X ו- Y הנתונים. אזי:

1. אם $x_m = y_n$, אזי $z_k = x_m = y_n$ ו- Z_{k-1} היא תת-מחרוזת משותפת ארוכה ביותר של X_{m-1} ו-Y_{n-1}.

2. אם $x_m \neq y_n$:

א. אם $z_k \neq x_m$ אזי Z היא תת-מחרוזת משותפת ארוכה ביותר של X_{m-1} ו- Y .

ב. אם $z_k \neq y_n$ אזי Z היא תת-מחרוזת משותפת ארוכה ביותר של X ו- Y_{n-1} .

<u>טענה</u>

תהא $Z = z_1 z_2, \ldots z_k$ תת-המחרוזת המשותפת הארוכה ביותר של X ו- Y הנתונים. אזי:

1. $\boxed{\text{אם } x_m = y_n \text{, אזי } z_k = x_m = y_n}$ ו- Z_{k-1} היא תת-מחרוזת משותפת ארוכה ביותר של X_{m-1} ו-Y_{n-1}.

<u>הוכחה</u>:

נניח בשלילה כי $z_k \neq x_m$.
אזי ניתן יהיה לשרשר לסוף Z את האיבר x_m (כי $x_m = y_n$), ובכך נקבל מחרוזת משותפת של X ו- Y אשר יותר ארוכה מ- Z.
סתירה לכך ש- Z ארוכה ביותר.

טענה

תהא $Z = z_1 z_2,\ldots z_k$ תת-המחרוזת המשותפת הארוכה ביותר של X ו- Y הנתונים. אזי:

1. אם $x_m = y_n$, אזי $z_k = x_m = y_n$ ו- Z_{k-1} היא תת-מחרוזת משותפת ארוכה ביותר של X_{m-1} ו-Y_{n-1}.

הוכחה:

ברור כי $Z = z_1 z_2,\ldots z_{k-1}$ היא מחרוזת משותפת של X ו- Y, כל שנותר להראות הוא שהיא גם ארוכה ביותר.

נניח בשלילה שלא, אזי קיימת מחרוזת משותפת ארוכה ביותר של X_{m-1} ו- Y_{n-1} שנסמנה W, אשר אורכה גדול ממש מ-$k-1$.

על-ידי שרשור האיבר x_m (כי $x_m = y_n$) ל- W, נקבל מחרוזת משותפת ל- X ו- Y באורך גדול ממש מ- k – סתירה.

<u>טענה</u>

תהא $Z = z_1 z_2 ,...z_k$ תת-המחרוזת המשותפת הארוכה ביותר של X ו- Y הנתונים. אזי:

2. אם $x_m \neq y_n$:

א. אם $z_k \neq x_m$ אזי Z היא תת-מחרוזת משותפת ארוכה ביותר של X_{m-1} ו- Y.

<u>הוכחה</u>:

Z תת-מחרוזת משותפת של X_{m-1} ו- Y, כי $z_k \neq x_m$.
נותר להראות כי Z תת-מחרוזת ארוכה ביותר.

נניח בשלילה כי קיימת תת-מחרוזת משותפת של X_{m-1} ו- Y, W, אשר אורכה גדול ממש מ- k.

אזי W גם מחרוזת משותפת של X ו- Y (ארוכה יותר מ- Z) – סתירה.

פתרון

נסמן ב- $c(i,j)$ את אורך המחרוזת המשותפת הארוכה ביותר של X_i ו- Y_j, $0 \le i \le m$, $0 \le j \le n$.

אם $i=0$ או $j=0$, אזי ברור כי $c(i,j)=0$.

לפי עובדה זו והטענה שהוכחנו קודם, ניתן לקבל את נוסחת הנסיגה הבאה:

$$c(i,j)=\begin{cases} 0 & i=0 \text{ or } j=0 \\ c(i-1,j-1)+1 & i,j>0, x_i=y_j \\ \max\{c(i,j-1),c(i-1,j)\} & i,j>0, x_i \ne y_j \end{cases}$$

פתרון

$$c\left(i,j\right)=\begin{cases}0 \quad i=0 \text{ or} & j=0\\ c\left(i-1,j-1\right)+1 & i,j>0, x_i=y_j\\ \max\left\{c\left(i,j-1\right),c\left(i-1,j\right)\right\} & i,j>0, x_i\neq y_j\end{cases}$$

אם נחשב פונקציה זו באופן רקורסיבי, סיבוכיות הזמן תהיה אקספוננציאלית, זאת משום שיהיו ערכים של הפונקציה אשר יחושבו מספר פעמים.

ניתן לחשב את הפונקציה c ביעילות בעזרת מטריצה $\left(m+1\right)\times\left(n+1\right)$ (מספור האינדקסים מתחיל מאפס), כאשר הערך שמעניין אותנו הוא $c\left(i,j\right)$.

ניתן למלא את ערכי מטריצה זו שורה אחר שורה, ובאופן זה נקבל אלגוריתם יעיל לפתרון הבעיה כי יש $O\left(mn\right)$ תאים במטריצה כאשר זמן החישוב של כל תא הוא $O\left(1\right)$.

(1) For $i=1$ to m do: $c(i,0) \leftarrow 0$

(2) For $j=1$ to n do: $c(0,j) \leftarrow 0$

(3) For $i=1$ to m do:

(4) For $j=1$ to n do:

(5) If $x_i = y_j$ then $c(i,j) \leftarrow c(i-1,j-1)+1$

(6) else $c(i,j) \leftarrow \max\{c(i-1,j), c(i,j-1)\}$

<u>פתרון</u>

נעיר כי ניתן בזמן של $O\left(m+n\right)$ למצוא את תת-המחרוזת המשותפת הארוכה ביותר של X ו- Y בעזרת המטריצה c.

<u>הסבר</u>: מתחילים מהתא $c\left(m,n\right)$.

בשלב שבו נמצאים בתא $c\left(i,j\right)$ עוברים לתא ה:

- $c\left(i-1,j-1\right)$ במידה ו- $c\left(i,j\right)=c\left(i-1,j-1\right)+1$. במידה ומתבצע מעבר כזה, מתקיים $x_i=y_j$ ותו זה נמצא בתת המחרוזת המשותפת הארוכה ביותר.
- $c\left(i,j-1\right)$ במידה ו- $c\left(i,j\right)=c\left(i,j-1\right)$. התו y_j לא נמצא בתת המחרוזת המשותפת הארוכה ביותר.
- $c\left(i-1,j\right)$ במידה ו- $c\left(i,j\right)=c\left(i-1,j\right)$. התו x_i לא נמצא בתת המחרוזת המשותפת הארוכה ביותר.

בעיית תרמיל הגב (Knapsack):

נתון תרמיל שלו קיבולת משקל אפשרית W ונתונים n עצמים $a_1, a_2, \dots, a_n$ באופן שלכל עצם a_i נתונים משקל w_i ורווח p_i.

בעיית תרמיל הגב היא בעיית בחירת תת-קבוצה של עצמים לאריזה בתרמיל באופן שהרווח הכולל עבורם יהיה מקסימאלי תוך שאיננו מפרים את אילוץ המשקל.

$W = 25$

$w_1 = 19$

$p_1 = 10$

$w_4 = 5$

$p_4 = 5$

$w_2 = 12$

$p_2 = 3$

$w_3 = 3$

$p_3 = 6$

$w_5 = 9$

$p_5 = 3$

פתרון:

נסמן ב- $F\left(i,w\right)$ את הרווח המקסימאלי כאשר בוחרים עצמים מתוך הקבוצה $\left\{a_1,a_2,...,a_i\right\}$ בלבד כך שסכום משקליהם לא יעלה על w.

אזי:

$$F\left(i,w\right)=\begin{cases}F\left(i-1,w\right) & \text{if } w_i>w\\ \max\left\{F\left(i-1,w\right),\, p_i+F\left(i-1,w-w_i\right)\right\} & \text{else}\end{cases}$$

סיבוכיות:

נממש בעזרת מטריצה בגודל $\left(n+1\right)\times\left(W+1\right)$ (נתעניין כמובן בתא ה- $\left(n,W\right)$) בסיבוכיות $O\left(nW\right)$.

פתרון – מילוי המערך:

$$F(i,w)=\begin{cases}F(i-1,w) & \text{if } w_i > w\\ \max\left\{F(i-1,w),\ p_i+F(i-1,w-w_i)\right\} & \text{else}\end{cases}$$

מספר הפריט \ גודל התרמיל	0	1		• • •			$W-1$	W
0	0	0	0	0	0	0	0	0
1	0							
⋮	0							
	0							
$n-1$	0							
n	0							

דוגמה:

$$w_1 = 5 \quad w_2 = 4 \quad w_3 = 6 \quad w_4 = 3$$
$$p_1 = 10 \quad p_2 = 40 \quad p_3 = 30 \quad p_4 = 50$$

i / w	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10	10	10	10
2	0	0	0	0	40	40	40	40	40	50	50
3	0	0	0	0	40	40	40	40	40	50	70
4	0	0	0	50	50	50	50	90	90	90	90

שאלה: לאחר חישוב המטריצה, כיצד נמצא את הפתרון האופטימלי (פריטים 2 ו-4 בדוגמה)?

תרגיל

נתונה פיסת בד בגודל $m \times n$, כאשר m ו- n מספרים טבעיים.

נתונות k מידות (n_i, m_i), $1 \le i \le k$, כאשר n_i ו-m_i מספרים טבעיים לכל $1 \le i \le k$.

לכל מידה נתון רווח מכירה p_i, $1 \le i \le k$. (לכל מידה שאינה נתונה, הרווח הוא 0) .

המטרה היא לגזור את הבד כך שסך הרווח שיתקבל מגזרי הבד יהיה מרבי.

בכל גזירה מותר לגזור פיסת בד לשניים, לאורכה או לרוחבה, כך שמתקבלים שני מלבנים שמידות כל אחד מהם הן מספרים טבעיים.

לא ניתן לסובב את פיסת הבד או המלבנים שהתקבלו. מספר הגזירות אינו מוגבל.

תרגיל

הציעו אלגוריתם אשר מוצא את הרווח המרבי שניתן לקבל מפיסת הבד הנתונה. על האלגוריתם לעבוד בזמן של $O\left(nm\left(n+m\right)\right)$.

פתרון:

נגדיר:

$c'\left(i,j\right)$ - הרווח המקסימאלי מפיסת בד בגודל $i\times j$ שנגזרה ל-2 או יותר פיסות.

$p\left(i,j\right)$ - הרווח ממכירת פיסת בד (לא גזורה) בגודל $i\times j$.

$c\left(i,j\right)$ - הרווח המקסימלי מפיסת בד בגודל $i\times j$ (גזורה או לא).

פתרון (המשך)

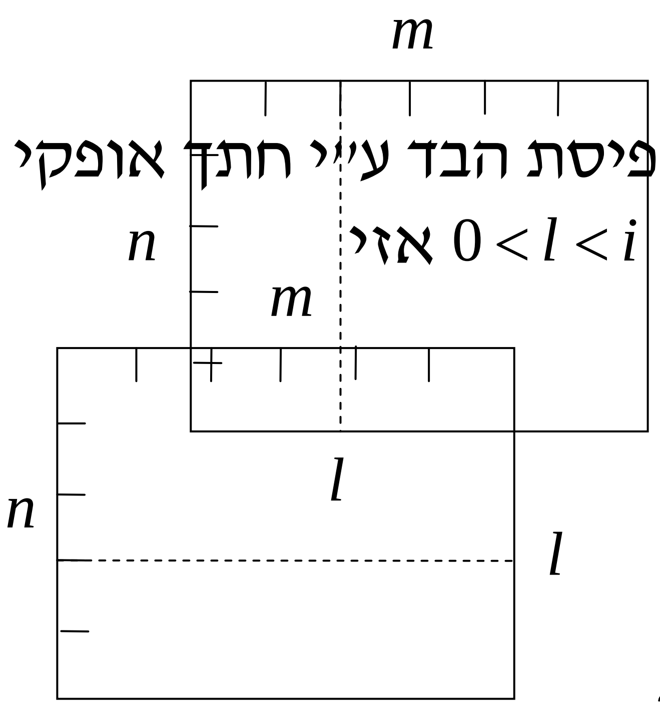
אבחנה 1:

לכל i ו- j מתקיים: $c(i,j)=\max\{c'(i,j),p(i,j)\}$.

אבחנה 2:

א- אם $c'(i,j)$ מתקבל כאשר חוצים את פיסת הבד ע"י חתך אנכי ל-2 פיסות בד $i\times l$ ו- $i\times(j-l)$ כך ש- $0<l<j$ אזי $c'(i,j)=c(i,l)+c(i,j-l)$.

ב- אם $c'(i,j)$ מתקבל כאשר חוצים את פיסת הבד ע"י חתך אופקי ל-2 פיסות בד $l\times j$ ו- $(i-l)\times j$ כך ש- $0<l<i$ אזי $c'(i,j)=c(l,j)+c(i-l,j)$.



פתרון (המשך)

טענה 1:

לכל i ו- j מתקיים:

$$c(i,j)=\max\left\{\begin{array}{l} p(i,j), \\ \max\limits_{0<l<i}\left\{c(l,j)+c(i-l,j)\right\} \\ \max\limits_{0<l<j}\left\{c(i,l)+c(i,j-l)\right\} \end{array}\right\}$$

הוכחה:

מיידית מההגדרות ומאבחנות 1 ו-2.

פתרון (המשך)

על מנת לחשב את $c(i,j)$ ביעילות נשתמש במטריצה $C_{n\times m}$ בה התא ה-(i,j) מייצג רווח אפשרי מפיסת בד בגודל $i \times j$.

$$C(i,j) = \max \left\{ \begin{array}{l} p(i,j), \\ \max\limits_{0<l<i} \left\{ C(l,j) + C(i-l,j) \right\} \\ \max\limits_{0<l<j} \left\{ C(i,l) + C(i,j-l) \right\} \end{array} \right\}$$

פתרון (המשך)

```
for i=1 to m do:
  for j=1 to n do:
    C(i,j)←0
for i=1 to k do:
  C(n_i,m_i)←p_i
for i=1 to m do:
  for j=1 to n do:
    for l=1 to i-1
      C(i,j)←max{C(i,j), C(l,j)+C(i-l,j)}
    for l=1 to j-1
      C(i,j)←max{C(i,j), C(i,l)+C(i,j-l)}
return C(n,m)
```

פתרון (המשך)

סיבוכיות:

ניתן לראות כי סיבוכיות האלגוריתם היא $O\left(nm(n+m)\right)$.

גם כאן סיבוכיות האלגוריתם אינה פולינומיאלית
אלא פסאדו-פולינומיאלית.

n ו-m בקלט מיוצגים על ידי
$\log_{10} n$ ו-$\log_{10} m$ תווים

פתרון (המשך)

נכונות:

נובעת מנכונות הטענה הבאה:

טענה 2: עם ההגעה לשורה 12 מתקיים $C(i,j)=c(i,j)$.

הוכחה: באינדוקציה על $i\times j$.

בסיס: $i\times j=1$

מאחר ולא ניתן לחתוך פיסת בד בגודל 1×1 אז $c(1,1)=p(n_i,m_i)$ אם קיים i כך ש-$(n_i,m_i)=(1,1)$, אחרת $c(1,1)=0$.

עפ"י האתחול $C(1,1)$ מאותחל בהתאם ולא משתנה בהמשך.

פתרון (המשך)

<u>צעד</u>: נניח כי $C\left(i',j'\right)=c\left(i',j'\right)$ לכל $\left(i',j'\right)$ כך ש- $i'\times j'<i\times j$, ונסתכל על התא ה-$\left(i,j\right)$ברגע חישוב ערכו.

עפ"י טענה 1 מתקיים כי:

$$c\left(i,j\right)=\max\left\{\begin{matrix}p\left(i,j\right),\max\limits_{0<l<i}\left\{c\left(l,j\right)+c\left(i-l,j\right)\right\},\\ \max\limits_{0<l<j}\left\{c\left(i,l\right)+c\left(i,j-l\right)\right\}\end{matrix}\right\}$$

לכן עפ"י הנחת האינדוקציה מתקיים כי:

$$c\left(i,j\right)=\max\left\{\begin{matrix}p\left(i,j\right),\max\limits_{0<l<i}\left\{C\left(l,j\right)+C\left(i-l,j\right)\right\},\\ \max\limits_{0<l<j}\left\{C\left(i,l\right)+C\left(i,j-l\right)\right\}\end{matrix}\right\}$$

פתרון (המשך)

נשים לב כי לאחר האתחול $C(i,j)=p(i,j)$.

לאחר שורות 8-9 מתקיים כי

$$C(i,j)=\max\left\{p(i,j),\max_{0<l<i}\left\{C(l,j)+C(i-l,j)\right\}\right\}$$

לאחר שורות 10-11 מתקיים כי

$$C(i,j)=\max\left\{\begin{array}{l} p(i,j),\max\limits_{0<l<i}\left\{C(l,j)+C(i-l,j)\right\}, \\ \max\limits_{0<l<j}\left\{C(i,l)+C(i,j-l)\right\} \end{array}\right\}$$

תרגיל 3 – קבוצה בלתי-תלויה מקסימום בעצים

בהינתן גרף לא מכוון $G=(V,E)$ קבוצת צמתים $V' \subseteq V$ נקראת קבוצה בלתי-תלויה (Independent Set) אם לא קיימים $u,v \in V'$ כך ש- $(u,v) \in E$.

הציעו אלגוריתם שבהינתן עץ לא-מכוון $T=(V,E)$ מוצא קבוצה בלתי-תלויה מקסימום ב- T.

למרבה ההפתעה,

גם בעיה זו היא NP קשה

עבור גרפים כלליים.

<u>פתרון</u>

נכוון את קשתות T כך שיתקבל עץ מכוון (למשל ע״י הרצת DFS החל מצומת $r \in V$)

ונגדיר את הערכים הבאים לכל צומת $v \in V$:

- $S^+(v)$ - גודל קבוצה בלתי-תלויה מקסימום <u>המכילה</u> את v בתת-העץ שמושרש ב-v.
- $S^-(v)$ - גודל קבוצה בלתי-תלויה מקסימום <u>שלא מכילה</u> את v בתת-העץ שמושרש ב-v.

פתרון (המשך)

נחשב את הערכים הנ״ל עבור צומת v, בהנחה שהם חושבו כבר לכל בניו של v, $children(v)$, כך:

$$S^+(v) = 1 + \sum_{u \in children(v)} S^-(u)$$

$$S^-(v) = \sum_{u \in children(v)} \max\left\{S^+(u), S^-(u)\right\}$$

גודל הקבוצה הבלתי תלויה מקסימום הוא כמובן $\max\left\{S^+(r), S^-(r)\right\}$.

פתרון (המשך)

r שייך לקבוצה אם $S^{+}\left(r\right) < S^{-}\left(r\right)$.

לכל צומת אחר u שהוא בן של צומת v נחליט האם u חלק מהקבוצה הבלתי תלויה מקסימום באופן הבא:

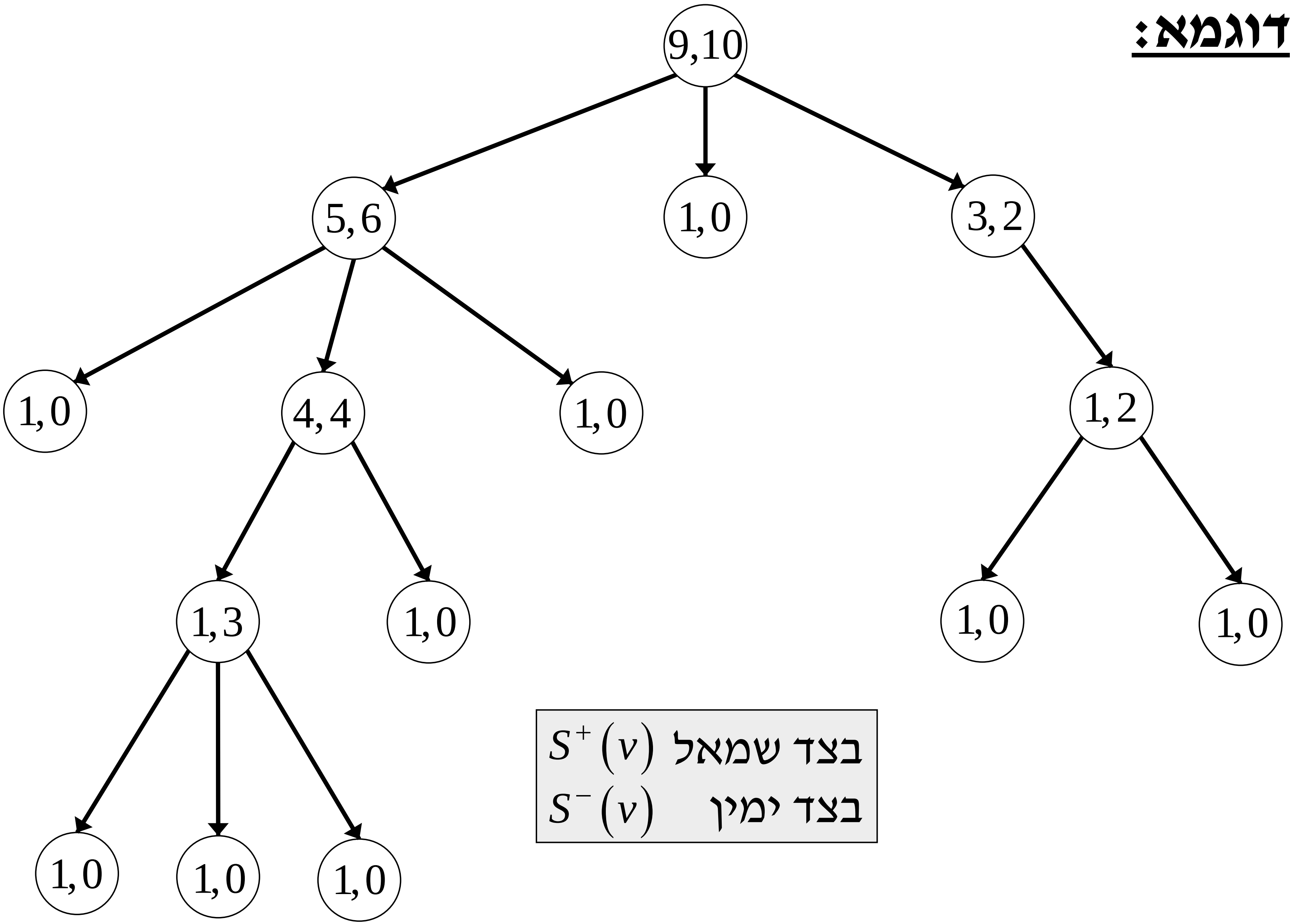
אם v בקבוצה אז u לא בקבוצה.
אחרת, נחליט בהתאם לשימוש ב-$S^{-}\left(u\right)$ או ב-$S^{+}\left(u\right)$ בחישוב $S^{-}\left(v\right)$.

סיבוכיות

לאחר כיוון העץ ($O\left(V\right)$) יש לטייל עליו בסדר pre-order ולחשב לכל צומת את הערכים הנ"ל.
הזמן שיש להשקיע בכל צומת הוא $O\left(d\left(v\right)\right)$.
לאחר חישוב הערכים ניתן לטייל על העץ בסדר post-order על מנת למצוא את אברי הקבוצה.

שוב הזמן שיש להשקיע בכל צומת הוא $O\left(d\left(v\right)\right)$.

דוגמא:



בצד שמאל $S^{+}(v)$
בצד ימין $S^{-}(v)$

תרגיל – בעיית הסוכן הנוסע (Traveling Salesman Problem -TSP)

אחת הבעיות המפורסמות ביותר במדעי המחשב היא בעיית הסוכן הנוסע: נתונה רשת של n ערים $\{1,\ldots,n\}$ וביניהן כבישים כך ש- $d_{i,j}$ מייצג את המרחק בין עיר i לעיר j.

סוכן-נוסע רוצה לצאת מעיר "1", לבקר בכל שאר הערים – בכל עיר פעם אחת בדיוק, ולחזור לעיר ממנה יצא. באיזה סדר עליו לבקר בערים כך שהמרחק שייסע יהיה קטן ככל האפשר?

איזה בעיה מוכרת זה מזכיר לנו?

<u>פתרון</u>

פתרון נאיבי יהיה לבדוק את כל $(n-1)!$ האפשרויות.

נראה פתרון יעיל יותר (אך לא פולינומיאלי – לא ידוע פתרון פולינומיאלי לבעיה) המבוסס על תכנון דינאמי.

נסמן ב-$c(A,k)$ את מחיר המסלול הקל ביותר מ-"1" ל-k אשר עובר דרך כל הצמתים בקבוצה $A\subseteq\{2,\ldots,n\}$.

הפתרון האופטימאלי יהיה: $\min\limits_{2\le k\le n}\left\{c\left(\{2,3,\ldots,n\}\setminus\{k\},k\right)+d_{1k}\right\}$

חישוב $c(A,k)$ יעשה עפ"י הנוסחה הבאה:

$$c(A,k)=\min_{2\le k\le n}\left\{c\left(A\setminus\{i\},i\right)+d_{ik}\right\}$$

<u>סיבוכיות</u>: יש לחשב $O\left(n\cdot 2^n\right)$ ערכים, חישוב כל ערך מתבצע ב-$O(n)$ לכן סה"כ – טוב יותר מהפתרון הנאיבי.