16.3.09 סיבוכיות - שיעור 3 (2 היה שיעור בונוס)

1

תזכורת: להרשם לפורום של הקורס (באתר הקורס)

בוחן: כנראה ביום שישי 15 במאי

סיבוכיות זמן ריצה

- P
- NP
  - הגדרה חלופית
  - רדוקציות זמן פולינומיאלי
  - בעיות NP שלמות
  - משפט Cook-Levin

<u>הגדרה</u> (NP) - שפה $L \subseteq \{0,1\}^*$ היא ב-NP אם קיים פולינום p ואלגוריתם M של בזמן פולי׳ כך ש: $x \in L \Leftrightarrow \exists w \in \{0,1\}^{p(|x|)}$ $st. M(x,w)=1, x \in \{0,1\}^*$.

<u>דוגמאות:</u>

1. קבוצה בת״ל (Independent set): בהינתן גרף לא מכוון G ומספר k, האם יש ב-G קבוצה בגודל k קודקודים שאין בהם אף קשת?

/* האם הבעיה ב-NP? כן. ניתן להסביר את הקבוצה שהיא IS (ובעצם NP שלמה) */

העד יכול להיות רשימה של k קודקודים והאלגוריתם המוודא M, בהינתן $<G,x>$ ועד w בודק שב-w יש k קודקודים ושאין קשת בין אף זוג.

2. HamPath (מסלול המילטון) - בהינתן גרף לא מכוון G, האם יש מסלול שמבקר בכל <u>קודקוד</u> בדיוק פעם אחת?

גם ב-NP (שלמה).

3. Ham Cycle - האם יש מעגל שמבקר בכל קודקוד בדיוק פעם אחת? גם ב-NP (שלמה).

4. Max IndSet - בהינתן גרף G ומספר k, האם גודל הקבוצה הבת״ל המקסימלית הוא k?

/* זו לא ממש בעיה ב-NP כיוון שקשה להשתכנע שלא קיים משהו גדול מ-k */ לא מאמינים שב-NP. קוראים לקבוצה $\sum_2^P$.

5. Liner Programming: האם יש פתרון למערכת אי שוויונים, למשל $2x+3y-5z \geq 2$, $x-4y-8z \leq 6$ ....

/* סבירה, "ברור" שהבעיה ב-NP כי אפשר להשתמש בהצבה מספקת כדוגמא. אם יכול להיות שהפתרונות יהיו הצבה גדולה מאוד. יש הוכחה לכך שזה ב-NP */

הבעיה ב-NP כי ניתן לתת בתור עד את ההצבה המספקת והמוודא יבדוק שכל האי-שוויונים מסתפקים. חשוב להוכיח שקיימת הצבה במספרים לא גדולים מדי, כדי שהעד יהיה בגודל פולינומיאלי.

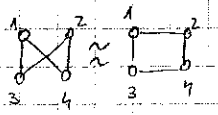
בעצם הבעיה ב-P בגלל אלגוריתם האליפסויד והמציאות יש הרבה אלגוריתמים מהירים.

6. Integer Programming: האם יש פתרון <u>בשלמים</u> למערכת אי שוויונים לינארים נתון?

גם כאן זה ב-NP (צריך להוכיח שגודל ההצבה יכולה להיות פולינומית) נשים לב שזו בעיה NP שלמה.

16.3.09 | סיבוכיות - שיעור 3

②



7. Graph Isomorphism - בהינתן שני גרפים (לא מכוונים) G_1 ו- G_2, האם הגרפים איזומורפים? (כלומר, האם זה אותו גרף תחת שינוי שמות הקודקודים?).

המוודא מצפה לקבל בתור עד את החלפת השמות (פרמוטציה) שמעבירה מ- G_1 ל- G_2 ובודק שעבור כל i ו- j בקבוצת הקודקודים, $(i,j)\in E_1 \Leftrightarrow (i',j')\in E_2$. וכמובן לוודא שהעד באמת מייצג פרמוטציה.

לכן, הבעיה היא ב- NP.

*) בינתיים לא מצליחים להוכיח שהבעיה NP שלמה ולא מצליחים להוכיח שהבעיה היא ב-P. לפני 10 שנים הוכח שכנראה היא לא NP שלמה *)

בנוסף, לא מאמינים שהיא NP שלמה כי זה היה קורה משהו שלא מאמינים בו (למשל NP=coNP).

8. Compositeness: בהינתן מספר N, האם הוא פריק (כלומר לא ראשוני)?

*) נשים לב שגודל הקלט הוא $\log N$ ולא N *)

גודל הקלט הוא $\log N$. לכן, אנחנו מודדים יעילות יחסית ל- $\log N$.

הבעיה ב- NP כי המוודא יכול לצפות לקבל מחלק k של N ולבדוק שבאמת k מחלק את N וגם $2\le k$, גודל "העד" k הוא $\log N \ge \log k$. ⓟ

9. Primality: בהינתן מספר N, האם הוא ראשוני?

הבעיה ב- coNP כי עד יכול לשכנע את המוודא שהתשובה היא לא
מסתבר שהבעיה היא ב- NP (Pratt 1975).

בסוף שנות ה- 70 גילו שהבעיה קיימת ב- BPP (כלומר, יש אלגוריתם רנדומי יעיל)
וזו השיטה שמשתמשים בה היום
בשנת 2002 גילו שהבעיה בעצם ב- ⓟ.

10. Factoring - בהינתן מספרים K, N האם יש ל-N מחלק ראשוני בתחום $\{k,k+1,\dots,N-1\}$?

(זה מודבק ע"י כך שפתרון לבעיית ההחלטה הזאת גורר פתרון לבעיית פירוק לגורמים בעזרת חיפוש בינארי).

הבעיה ב- NP כי יכולים לתת לי בתור עד, מספר בתחום הזה ונבדוק שהוא מחלק את N ושהוא ראשוני. (אין בעזרת האלגוריתם מ- 2002 או בעזרת המודא של PRATT ותוספת לעד).

מסתבר שהבעיה גם ב- coNP כי עד יכול לשכנע אותי שהתשובה היא לא.
הוא יתן לי רשימה של מספרים $a_1,\dots,a_j$ ואני אבדוק ש:

(1) $a_1\cdot a_2\cdots a_j = N$
(2) כולם ראשוניים.
(3) כולם קטנים מ- k וגדולים מ- 1.

צריך להוכיח שהעד בגודל פולינומיאלי (ב- $\log N$) וגם זמן ריצת המוודא פולינומיאלי (ב- $\log N$).

בעיות NP
- לבדוק מה גודל הקלט של האלגוריתם (לשים לב ל- 8).
- לבדוק מה גודל העד.
- לבדוק שבאמת ניתן לוודא.
- לחשוב תמיד על מוודא חוקי.

כיוון שהבעיה היא ב- $NP \cap coNP$, לא מאמינים שהיא NP שלמה (כי אחרת יוצא משהו שלא מאמינים בו, $NP = coNP$).

טענה: $P \subseteq NP \subseteq EXP := \bigcup_{c \geq 1} DTIME(2^{n^c})$.

הוכחה: $P \subseteq NP$: ניקח שפה L כלשהי ב- P ונראה שהיא שייכת ל- NP. כיוון ש- $L \in P$, קיים אלגוריתם M שמכריע את L בזמן פולינומי (נגיד n^c עבור איזשהו $c > 0$). נבנה מוודא שמתעלם מהעד (שיכול להיות באורך 0) ופשוט מפעיל את M על הקלט x.

$NP \subseteq EXP$: תהי L שפה כלשהי ב- NP. נראה כי L שייכת ל- EXP. כיוון ש- $L \in NP$ קיים פולינום p ואלגוריתם מוודא M שרץ בזמן פולי כך שלכל x, $x \in L \Leftrightarrow \exists w \in \{0,1\}^{p(|x|)} . M(x,w) = 1$.

נבנה אלגוריתם M' שבהינתן x, מפעיל את $M(x,w)$ על כל $2^{p(|x|)}$ העדים w האפשריים. אם אחת הריצות קיבלה (לפחות) אז נקבל אחרת נדחה.

זמן הריצה הכולל $2^{p(|x|)} \cdot poly(|x|) = 2^{poly(|x|)}$ כנדרש ולכן $L \in EXP$.

הגדרה חלופית ל- NP:

ההגדרה משתמשת באי-דטרמיניסטיות (nondeterministic) וזה גם מקור השם NP.

דוגמה: התוכנית בודקת האם המערך הנתון אינו כולו 0:

```
is_non_zero(a[1...n])
{
    if (n==1)
        if a[1]==0
            terminate(reject)
        else
            terminate(accepte)

    non_det_choice {
        is_non_zero(a[1...n/2]);
    }or{
        is_non_zero(a[n/2+1...n]);
    }
}
```

*) אין הגבלה על מספר המעבדים והתוכנית יכולה להתפצל לכל חלק יחד במקביל.

פקודת non-det_choice מפצלת את התוכנית ל-2 ריצות מקבילות. נגיד שהתוכנית מקבלת אם קיים איזשהו פיצול שמוביל לקבלה. כל הפיצולים דרושים לרוץ כל עוד יש מזמן הריצה הנתון. (זמן הריצה הוא המקסימום מבין כל המסלולים).

הגדרה: נגיד ששפה $L \in \{0,1\}^*$ שייכת ל- $NTIME(T(n))$ אם קיים $c > 0$ וקיים אלגוריתם אי דטרמיניסטי M כך שלכל x, קיימת בחירה אי דטרמיניסטית $\Leftarrow x \in L$ שגורמת ל- M לקבל ובנוסף M עוצר אחרי $c \cdot T(n)$ צעדים עבור כל בחירה אי דטרמיניסטית.

משפט: $NP = \bigcup_{c \geq 1} NTIME(n^c)$

הוכחה: $\supseteq$: תהי $L \in NTIME(n^c)$ עבור c כלשהו. נראה ש- $L \in NP$. לפי הנחה קיים אלגוריתם אי דטרמיניסטי שרץ בזמן n^c ופותר את L. נבנה אלגוריתם מוודא (דטרמיניסטי רגיל) שמקבל בנוסף לקלט x עד w באורך n^c ומריץ את האלגוריתם האי-דטרמיניסטי כאשר כל פיצול אי דטרמיניסטי מוחלף בפיצול רגיל (של if) בהתאם לבחירת העד w.

16.3.09 (4)

כלומר,

```
non-det-choice {              if (w[i++] == 0) {
    ...                            ...
} or {               =>       } else {
    ...                            ...
}                             }
```

האלגוריתם המיוצג רץ בזמן פולינומיאלי כי כל הפעולים מוגבלים לזמן $\le n^c$,

$\subseteq$: תהא $L \in NP$. לכן קיים אלגוריתם מוודא M ופולינום p כך ש-
$x \in L \Leftrightarrow \exists w \in \{0,1\}^{p(|x|)} . M(x,w)=1$
נבנה אלגוריתם אי דטרמיניסטי ל- L:

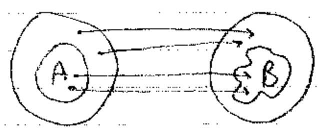
```
bool w[n];
for i=1 to p(|x|) do {
    non_det_choice {
        w[i] = 0
    } or {
        w[i] = 1
    }
}
Run M(x,w) and return its answer.
```

האלגוריתם שקיבלנו מקבל את x אמ"מ קיימת בחירה אי דטרמיניסטית שתגרום לקבלה וזה שקול לכך שקיים w שעבורו $M(x,w)=1$, כאשר זה שקול ל- $x \in L$ כנדרש.

<u>רדוקציות ושלמות ב- NP:</u>

<u>הגדרה</u> רדוקציה (KARP) פולינומיאלית משפה $A \subseteq \{0,1\}^*$ לשפה $B \subseteq \{0,1\}^*$ היא פונקציה f שניתנת לחישוב בזמן פולינ' כך שלכל x $x \in A \Leftrightarrow f(x) \in B$.

במקרה שקיימת כזו רדוקציה, נסמן $A \le_p B$



<u>טענה</u> 1. אם $A \le_p B$ וגם $B \le_p C$ אז $A \le_p C$. (טרנזיטיביות)

2. אם $A \le_p B$ וגם $B \in P$ אז גם $A \in P$ (P סגור תחת רדוקציות Karp פולי')

3. אם $A \le_p B$ וגם $B \in NP$ אז $A \in NP$ (NP סגור תחת רדוקציות Karp פולי').

<u>הוכחה:</u> 1,3 תרגיל.

2. לפי הנחה, קיימת רדוקציה פולי' f מ- A ל- B וגם קיים אלג' פולי' M שמחשב את B.
נבנה אלגוריתם שבהינתן קלט x מחשב את $f(x)$ ומחזיר את $M(f(x))$. קל לראות שזמן החישוב הוא פולי'. (כי גודל $f(x)$ הוא פולי' ב- x והרכבה של 2 פולינומים היא פולינום)
והאלגוריתם מחשב את A.

<u>הערה:</u> כדי להראות רדוקציה פולינומיאלית מ- A ל- B דרוש: (1) למצוא פונקצית רדוקציה f,
(2) צריך להראות שחישובה בזמן פולינומיאלי: (3) $x \in A \rightarrow f(x) \in B$, $f(x) \in B \rightarrow x \in A$
completeness soundness

<u>הגדרה:</u> נגיד ששפה B היא NP-קשה אם לכל $A \in NP$ מתקיים ש $A \le_p B$. תקרא NPC אם בנוסף היא גם ב- NP.