המשך סיבוכיות

אמרנו:

- P – כל השפות L כך שקיים אלגוריתם פולינומי לבדיקת האם קלט x נמצא ב-L.
- NP – כל השפות L כך שקיים אלגוריתם אימות A ל-L.
  - A מקבל $x=$ קלט ל-L ו-$w=$ עד ("פתרון לבעיית חיפוש").
    - אם $x \in L$ אזי קיים עד w בגודל פולינומי ב-$|x|$ כך ש-$A(x,w)$ מחזיר "כן".
    - אם $x \notin L$ אז לכל עד w, $A(x,w)$ מחזיר "לא".
  - A רץ בזמן פולינומי ב-$|x|+|w|$.

<u>הערה</u>: אם A מקבל קלט x ועד w באורך $|w| \le poly_1(|x|)$ ורץ בזמן $poly_2(|\text{input for } A|)$ – אז בסה"כ A רץ בזמן $poly_2(|x|+poly_1(|x|))$. זה עדיין פולינומי בגודל הקלט $|x|$.

משפט [P מוכל ב-NP]

$$P \subseteq NP$$

<u>הוכחה</u>:

[פשוט נתעלם מהעד, נפתור את הבעיה בעצמנו (הבעיה ב-P אז אפשר), ואז נחזיר תשובה בהתאם לתוצאה.]

תהי $L \in P$.

קיים אלגוריתם A הבודק בזמן פולינומי האם $x \in L$.

נגדיר אלגוריתם A' שמתעלם מהעד w, מריץ את A על x ומחזיר כן/לא בהתאם.

- אם $x \in L$, אזי A' יחזיר "כן" עבור (למשל) $w=1$.
- אם $x \notin L$, A' יחזיר "לא" בהתאם לתשובה של A (לכל w אפשרי).

[[<u>הסבר</u>: אנחנו מגדירים אלגוריתם שפותר את הבעיה עצמה (L) ו"עונה בהתאם".

אם יש פתרון לבעיה, האלגוריתם שלנו **תמיד** מחזיר "כן", לא משנה מה נותנים לו ב-w.

לכן קיים w (נתנו פה דוגמה של $w=1$, אבל כל דבר יעבוד) שעבורו האלגוריתם מחזיר "כן".

אם אין פתרון לבעיה, האלגוריתם שלנו **תמיד** מחזיר "לא", לא משנה מה נותנים לו ב-w.

זה מתאים לדרישות שהגדרנו עבור אלגוריתם אימות (אם כי לא ממש מתאים לרעיון של "האלגוריתם בודק אם w הוא פתרון לבעיה x", אבל זו לא אחת מהדרישות הפורמליות שהגדרנו, אז זה פחות רלוונטי).]]

רדוקציות

<u>תכנית</u>:

[ציור]

<u>תזכורת</u>:

רדוקציה מ-A ל-B:

[ציור]

[הרדוקציה צריכה להיות **יעילה** (לעבוד בזמן פולינומי) **ונכונה**.]

הגדרה [רדוקציה פולינומית]

רדוקציה פולינומית משפה A לשפה B היא פונקציה f שמקיימת:

- <u>יעילות</u>: f ניתנת לחישוב בזמן פולינומי
- <u>נכונות</u>: $f(x) \in B \Leftrightarrow x \in A$ (לכל x)

(צ"ל: $\begin{matrix} f(x) \in B \Leftarrow x \in A \\ x \in A \Leftarrow f(x) \in B \end{matrix}$)

סימון: אם קיימת רדוקציה כנ"ל, נסמן $A \leq_p B$.

תזכורות

<u>תזכורת לאינטואיציה מתחילת הקורס</u>:

אם יש רדוקציה מ-A ל-B, אז:

- B קלה $\Leftarrow A$ קלה
- A קשה $\Leftarrow B$ קשה

<u>למה (תזכורת)</u>: אם $A \leq_p B$ ו-$B \in P$ אזי $A \in P$.

<u>הוכחה</u>:

אפשר לבנות **אלגוריתם** תלוי-רדוקציה באופן הבא:

- נקבל קלט x ל-A
- נפעיל את f על x ונקבל קלט y ל-B
- נפעיל אלגוריתם פולינומי שפותר את B ונחזיר את התשובה

זמן הריצה של האלגוריתם:

- הפעלת f – פולינומי ב-$|x|$
- מכאן ש-f מייצר y בגודל ב-$|x|$ (אין לו זמן לייצר y יותר ארוך)
- האלגוריתם שפותר את B רץ בזמן פולינומי של $|y|$

מכיוון ש- $y \le poly_1(|x|)$ וזמן הריצה של $poly_2(|y|) \ge B$ – קיבלנו ש- B רץ בזמן פולינומי ב- $|x|$ ($\ge poly_2(poly_1(|x|))$)

[את כל הנ"ל כבר ראינו בעבר, כשדיברנו על רדוקציה בתחילת הקורס וכשהראינו את העניין של פולינום של פולינום.]

נכונות:
צ"ל שהאלגוריתם מחזיר "כן" $x \in A \Leftrightarrow$.

דרישת נכונות הרדוקציה

$x \in A \Leftrightarrow y = f(x) \in B \Leftrightarrow$ האלגוריתם ל- B מחזיר "כן" עבור $y \Leftrightarrow$ האלגוריתם תלוי-הרדוקציה מחזיר "כן".

מסקנה חשובה

אם $A \notin P$ וְ- $A \le_p B$ אזי $B \notin P$.

הגדרה [NP-קשה, NP-שלמה]

B בעיה NP-קשה אם לכל שפה $A \in NP$ מתקיים $A \le_p B$.

B בעיה NP-שלמה אם:

- $B \in NP$
- B היא NP -קשה.

[ציור]

[ישנן בעיות שהן NP-קשות אך לא נמצאות ב-NP – למשל בעיית העצירה לא ניתנת לפתרון בזמן סופי **כלל**, אבל יש רדוקציה מכל שפה ב-NP אליה, לכן היא NP-קשה.]

משפט [מרכזי לשאלה הגדולה]

אם השפה B היא NP -שלמה אז:

$$B \in P \Leftrightarrow P = NP$$

[כלומר אם מוכיחים **שאחת** מאלפי הבעיות שידועות בתור NP-שלמות נמצא ב- P (כלומר ניתן לפתור אותה בזמן פולינומי) – אז זה פותר את השאלה הפתוחה המרכזית של מדעי המחשב ומוכיח כי $P = NP$.]

<u>הוכחה</u>:

- $\underline{\Leftarrow}$:

נניח $P = NP$, B שלמה ובפרט $B \in NP = P$.

- <u>$\Rightarrow$</u>:
  נניח ש- $B \in P$ (ראינו ש- $P \subseteq NP$, צ"ל $NP \subseteq P$ – כלומר שלכל $A \in NP$ מתקיים $A \in P$).
  ניקח שפה $A \in NP$.
  B היא NP-קשה, לכן מתקיים $A \le_p B$.
  אז לפי הלמה, גם $A \in P$.

□

<u>הערה:</u>
בשביל להראות ש- $P = NP$ מספיק להוכיח $B \in P$ עבור בעיה B שהיא NP-**<u>קשה</u>**,
ובשביל להראות ש- $P \ne NP$ מספיק להוכיח $B \notin P$ עבור בעיה $B \in NP$.
[זאת כי בהוכחה השתמשנו רק בעובדה ש- B היא NP-קשה.]

טענה (הרכבת רדוקציות)

אם $A \le_p B$ ו- $B \le_p C$ אזי $A \le_p C$.

<u>הוכחה:</u>
ניקח את הרדוקציה f מ- A ל- B ו- g מ- B ל- C.
אזי $g \circ f$ נותנת רדוקציה מ- A ל- C:

- <u>זמן ריצה:</u>
  f רצה בזמן פולינומי בגודל הקלט $|x|$, ומייצרת פלט $f(x)$ פולינומי בגודל הקלט $|x|$, ו-
  $g(f(x))$ ניתן לחשב בזמן פולינומי ב- $|f(x)|$ – שהוא פולינומי גם ב- $|x|$.

- <u>נכונות:</u>

$$g(f(x)) \in C \overset{B \le_p C}{\Leftrightarrow} f(x) \in B \overset{A \le_p B}{\Leftrightarrow} x \in A$$

□

מסקנה [רדוקציה פועלת גם על NP-קשה]

אם B היא NP-קשה ו- $B \le_p C$ אזי גם C היא NP-קשה.

<u>הסבר:</u>
לכל שפה $A \in NP$ מתקיים $A \le_p B$ (מהגדרת NP-קשה).
נתון ש- $B \le_p C$ ולכן לפי הטענה $A \le_p C$.

IS וְ-VC

<u>תזכורת</u>:
השפה IS:

$\left(G=\left(V,E\right),k\right)\in IS$ אם"ם קיימת קבוצה $U\subseteq V$ כך ש- $\left|U\right|\geq k$ ולכל קשת $\left(u,v\right)\in E$ **<u>לכל</u>** **<u>היותר</u>** אחד מבין u,v שייך ל- U.

<u>נראה בהמשך</u>: בעיית IS (קבוצה בלתי תלויה) היא NP-שלמה.

<u>שפה נוספת</u>:
$\left(G=\left(V,E\right),k\right)\in VC$ אם"ם קיימת קבוצה $C\subseteq V$ בגודל $\left|t\right|\geq$ ולכל קשת $\left(u,v\right)$ **<u>לפחות</u>** אחד מבין u,v שייך ל- C.

[<u>שימו לב</u>: עבור כל קבוצה U שמתאימה ל- IS, המשלים שלה, $C=\bar{U}$, מתאים ל- VC.
לכן יש רדוקציה פשוטה מהאחת לשנייה.]