

סיבוכיות מקום

ראינו פעם שעברה את משפט Savitch:

$$\text{NSPACE}(\log n) \subseteq \text{DSPACE}(\log^2 n)$$

משפט Savitch מוכלל

עבור $s(n) \geq \log n$ מתקיים

$$\text{NSPACE}(s(n)) \subseteq \text{DSPACE}\left(O\left[(s(n))^2\right]\right)$$

הוכחה

עבור $A \in \text{NSPACE}(s(n))$ צ"ל $A \in \text{DSPACE}(s^2(n))$.
נגדיר A' באופן הבא:

$$A' = \left\{ \frac{x \cdot 1 \cdot 0^{2^{s(|x|)}}}{n} \mid x \in A \right\}$$

נשים לב:

$$A' \in \text{NSPACE}(\log n)$$

הסבר: כדי להכריע את A' צריך לוודא שני דברים:

1. צריך לוודא שהמחרוזת בפורמט נכון.
2. צריך לוודא ש $x \in A$.

$$n = |x| + 1 + 2^{s(|x|)} \quad \log n \geq s(|x|)$$

כדי לממש את 1 צריך לשמור בזיכרון העבודה מונה שיכול לספור עד $1 + |x| + 2^{s(|x|)}$. מונה כזה ניתן לממש ב $\log n$ ביטים ולכן גודל הזיכרון הנדרש $\log n \geq s(|x|)$.

$$s(|x|) \leq \log n = \log(1 + |x| + 2^{s(|x|)}) \leq \log(2 \cdot 2^{s(|x|)}) = s(|x|) + 1$$

$$s(|x|) \geq \log |x| \quad 2^{s(|x|)} \geq |x|$$

כדי לוודא ש $x \in A$, זאת ניתן בזיכרון $s(|x|)$, שכן $A \in \text{NSPACE}(s(n))$, ולכן 2 ניתן לחישוב ב $\log n \geq s(|x|)$ ביטים.

ראינו שכדי לוודא את 1 ו 2 נדרשים $\log n$ ביטים של זיכרון, ולכן $A' \in \text{NSPACE}(\log n)$.

בעזרת משפט Savitch:

$$A' \in \text{DSPACE} \left((\log n)^2 \right)$$

כדי להראות ש $A \in \text{DSPACE} (s^2(n))$ נראה את האלגוריתם הבא שמחשב את A בסיבוכיות זיכרון דטרמיניסטית $O(s^2(n))$:
בהינתן קלט x :

1. צור קלט $y = x \cdot 1 \cdot 0^{2^{s(|x|)}}$

2. הכרע האם $y \in A'$ והחזר את התשובה המתקבלת

ברור מהגדרת A' , שהאלגוריתם שתיארנו מכריע נכונה את A . נסביר כעת כי עבור קלט x , האלגוריתם עובד בסיבוכיות מקום דטרמיניסטית $O(s^2(|x|))$:
אמנם יש ליצור קלט באורך $|x| + 1 + 2^{s(|x|)}$, אך אין צורך להחזיר יצוג מפורש של קלט זה, ולמעשה כאשר ידרש הביט i של y נפעל באופן הבא:

• אם $1 \leq i \leq |x|$ נחזיר את x_i מתוך הקלט

• אם $i = |x| + 1$ נחזיר 1

• אם $|x| + 2 \leq i \leq 1 + |x| + 2$ נחזיר 0

לצורך זה נדרש רק מונה בגודל $\log(1 + |x| + 2^{s(|x|)})$ שניתן לממש ב $O(s(|x|))$ תאי זיכרון.

את שלב 2 ניתן לממש בסיבוכיות זיכרון דטרמיניסטית שהינה

$$[\log(|x|)]^2 \leq \left[\log \left(2 \cdot 2^{s(|x|)} \right) \right]^2 \cdot \log^2(|y|)$$

השערה

$$NP \neq CO - NP$$

ראינו כי אם

$$NP = CO - NP$$

אז

$$PH = \Sigma_1$$

משפט Immerman

$$NL = CO - NL$$

$$NL = \bigcup_c \text{NSPACE}(l_c)$$

$$l_c = c \log n$$

$$CO - NL = \{ \{0, 1\}^* \setminus A \mid A \in NL \}$$

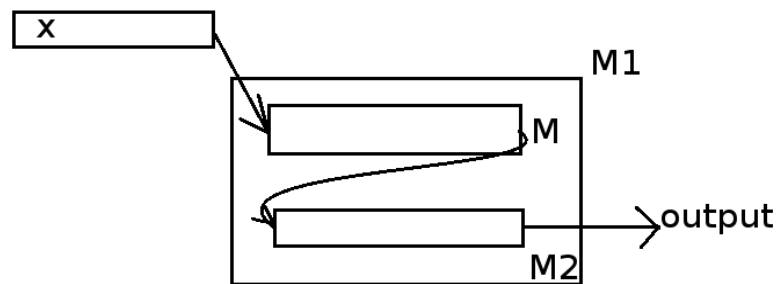
טענה

אם קיימת רדוקציית log-space מ L_1 ל L_2 וכן $L_2 \in NL$ אזי $L_1 \in NL$.

הוכחה

כזכור רדוקציית log-space מ L_1 ל L_2 היא פונקציה f חשיבה בסיבוכיות מקום לוגריתמית, כלומר $f(x)$ ניתן לחישוב במקום $\log(|x|)$ ולכן ע"פ משפט משיעור שעבר בסיבוכיות זמן פולינומית ב $|x|$, כך שמתקיים

$$f(x) \in L_2 \iff x \in L_1$$



- M_1 - מחשבת את L_1
- M - מחשבת רדוקציה $f(x)$
- M_2 - מחשבת את L_2

$$|f(x)| \leq |x|^c$$

נשים לב שמכונה M_1 שמכריעה את L_1 צריכה לסמלץ בתורה את הפעלת מכונות M ו M_2 . הפלט $f(x)$ של M גודלו חסום פולינומית ב $|x|$, ולכן M_1 לא יכולה לשמור $f(x)$ בשטח

העבודה שלה. הדרך להתגבר על כך, היא להפעיל את M מחדש עבור כל ביט i מתוך $f(x)$ שמכונה M מבקשת. כלומר, עבור כל ביט(מתוך $f(x)$) ש M_2 דורשת נפעיל את M מחדש. על כן נוכל להסיק ש M_1 עובדת בסיבוכיות זיכרון לוגריתמית, ולכן $L_1 \in NL$ כנדרש.

■

טענה

אם A היא NL -שלמה, וכן $A \in CO - NL$, אזי $NL = CO - NL$.

הוכחה

נראה ראשית שתחת ההנחות לעיל $NL \subseteq CO - NL$. תהי $L \in NL$, צ"ל $L \in CO - NL$. מכיוון ש A היא NL -שלמה, קיימת רדוקציית f ל M log-space, כלומר f חשיבה בזיכרון לוגריתמי, כך ש:

$$\begin{aligned} f(x) \in A &\iff x \in L \\ f(x) \notin A &\iff x \notin L \\ f(x) \in \bar{A} &\iff x \in \bar{A} \end{aligned}$$

כלומר f הינה גם רדוקציית log-space מ \bar{L} ל \bar{A} . ע"פ טענה \odot , $\bar{L} \in NL$ ולכן $L \in CO - NL$, ולכן קיבלנו $NL \subseteq CO - NL$. נניח כי קיימת $\tilde{L} \in CO - NL \setminus NL$.

$$\bar{\tilde{L}} \in NL \subseteq CO - NL$$

$$\tilde{L} \in NL$$

אבל זוהי סתירה שכן הנחנו ש

$$\tilde{L} \in CO - NL \setminus NL$$

ולכן

$$NL = CO - NL$$

מסקנה

לפיכך, כדי להוכיח את משפט Immerman, די למצוא שפה שהיא NL -שלמה וגם משלימתה של אותה שפה שייכת ל NL .

תזכורת

בשבוע שעבר ראינו שהשפה הבאה היא NL -שלמה:

$$St - Conn = \left\{ (G, s, t) \mid \begin{array}{l} G \text{ is directed graph with} \\ \text{a path from } s \text{ to } t \end{array} \right\}$$

כדי להוכיח את המשפט, נראה שהשפה הבאה היא NL :

$$\overline{St - Conn} = \left\{ (G, s, t) \mid \begin{array}{l} G \text{ is directed graph without} \\ \text{a path from } s \text{ to } t \end{array} \right\}$$

הגדרה

$TR = \text{Total Reachability } (G, s)$

$$TR(G, s) = \# \text{ of nodes in Graph } G \text{ that there is} \\ \text{a directed path to them from node } s$$

טענה

אם $TR(G, s) \in NL$ אזי $\overline{St - Conn}(G, s, t) \in NL$

הגדרה

עבור $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$, נאמר כי f "מ"ט ל"ד מחשבת את f אם עבור קלט x :

- המכונה תחזיר $f(x)$ או סימן \perp ("לא יודעת")
- קיים מסלול שיגרום למכונה להחזיר $f(x)$

הוכחת הטענה

להלן מכונה NL שמכריעה את $\overline{St - Conn}$:
בהנתן מכונה NL שמכריעה את $TR(G, s)$
 $\overline{St - Conn}(G, s, t)$

1. $C_1 \leftarrow TR(G, s)$ אם נכשלנו נחזיר 0
2. $C_2 \leftarrow TR(G', s)$ אם נכשלנו נחזיר 0, כאשר G' הוא הגרף G בו הורדנו את קודקוד t
3. אם $C_1 = C_2$ נחזיר 1 אחרת 0

כלומר, כדי להוכיח את משפט Immerman נותר להראות $TR(G, s) \in NL$.

אסטרטגיה: ננחש את $TR(G, s)$, נקרא לניחוש g

1. נבדוק ש $TR(G, s) \geq g$ - קל

2. נבדוק ש $TR(G, s) \leq g$ - קשה

אם נוכל לבצע את 1 ו 2 בסיבוכיות זיכרון לוגריתמית, נוכל להראות $TR \in NL$.

עבור גרף $G = (V, E)$ וקודקוד $v \in V$:

Set of nodes that there is

$R_i =$ a path to them from v
with length $\leq i$

$$R_0 = \{v\}$$

$$R_i = R_{i-1} \cup \{u | w \in R_{i-1}, (w, u) \in E\}$$

ולכן

$$TR(G, s) = |R_n|$$

הרעיון: ישנו אלגוריתם הרץ בסיבוכיות זיכרון לוגריתמית המכריע את $|R_1|$ בהנתן $|R_{i-1}|$

נשים לב כי אם אפשר להכריע את $|R_i|$ בהנתן $|R_{i-1}|$ בסיבוכיות זיכרון לוגריתמית אזי $TR(G, s) \in NL$, וההסבר הינו הבא:

כדי לחשב את TR אנו נדרשים למצוא את $|R_n|$. $|R_0|$ ידוע לנו, וחישוב $|R_i|$ בהנתן $|R_{i-1}|$ יכול להעשות בסיבוכיות לוגריתמית וכן כל פעם ישמר רק ה $|R_i|$ האחרון שחושב, ולחץ סה"כ הזיכרון המנוצל הינו לוגריתמי.

לכן נותר להראות אלגוריתם שרץ בסיבוכיות זיכרון לוגריתמית לחישוב $|R_i|$ בהנתן ערך נכון עבור $|R_{i-1}|$.

0. ננחש $|R_i| = g$

1. נבדוק $|R_i| \geq g$

2. נבדוק $|R_i| \leq g \iff |V - R_i| \geq n \cdot g$

אם שלב 1 ו 2 הצליחו נחזיר g , אחרת נחזיר \perp .

- שלב 1 מבוצע בזיכרון לוגריתמי, כי אפשר להגריל g קודקודים, ולבדוק לגבי כל אחד מהם ב \log -space שקיים מסלול באורך $i \geq sm$ אליו.

- שלב 2 כדי לבדוק ש $|V - R_i| \geq n - g$ נגרול $n - g$ קודקודים. לכל קודקוד שהוגרל, נגריל $|R_{i-1}|$ קודקודים, ונבדוק לגבי כל אחד מהם ב \log -space שהוא אכן ב R_{i-1} . עבור כל קדקד w שוידינו שהוא ב R_{i-1} נוודא ש $w \neq u$, וכן שאין קשת (w, u) . אם כל הבדיקות עברו בהצלחה נשתכנע כי $u \notin R_i$, ולכן סה"כ שלב 2 מבוצע בסיבוכיות זיכרון לוגריתמית.