**הרצאה 10** – יום ראשון 24.8

**מחלקות *L*, *NL***

כל *TM* אמור לקרוא את כל הקלט, אז המקום הוא לפחות $n$. מה אם, חוץ מקריאת הקלט, הוא משתמש רק במספר קטן של תאים, אפילו פחות מ-$n$? נשתמש במודל אחר: יש סרט אחד שהוא רק קלט (קריאה בלבד, *read-only*) וסרט עבודה. נאמר שיש ל-*M* סיבוכיות מקום $f(n)$ או ש-*M* רץ במקום $f(n)$, אם מספר **תאי העבודה** ש-*M* קורא הוא לכל היותר $f(n)$.

עבור *N NTM*, נגדיר את סיבוכיות המקום ע"י המספר המקסימלי של תאי עבודה שנסרקים מעל כל הענפים של עץ החישוב, עבור קלט באורך $n$.

$$L \coloneqq SPACE(\log n), \qquad NL \coloneqq NSPACE(\log n)$$

**לדוגמה**

$$L_1 \coloneqq \{0^{n/2}1^{n/2} : n \in \mathbb{N}\} \in L$$

נוכיח ע"י תיאור אלגוריתם שמכריע את השפה, במקום $\log n$. נניח שהקלט מתקבל על סרט 1, ויש את סרט 2 – סרט עבודה.

1. נעבור על סרט 1 עד ה-1 הראשון. בכל 0 שרואים, נוסיף 1 לספירה בסרט 2.
2. נמשיך על סרט 1 עד הסוף. לכל 1 שרואים, נחסיר 1 מהספירה בסרט 2.
3. אם קוראים מקום ריק בסרט 1 ובסרט 2 כתוב 0, נקבל. אחרת, נדחה.

המקום הדרוש הוא $O(\log n)$ עבור שמירת מספר עד $n$.

**דוגמה 2**

$$PATH \coloneqq \{\langle G, s, t\rangle : directed\ graph\ G\ has\ a\ path\ from\ s\ to\ t\} \in NL$$

נוכיח ע"י תיאור אלגוריתם פולינומי אי-דטרמיניסטי שמכריע את השפה. עבור קלט $\langle G, s, t\rangle$ כאשר יש ב-$G$ $m$ קודקודים:

1. נשמור מצביע לקודקוד $s$.
2. נרוץ עד שנגיע ל-$t$, או $m$ פעמים:
   a. המצביע השמור מצביע על $a$. נבחר (באופן אי-דטרמיניסטי) צלע $(a, b)$, ונצביע על $b$.
   b. אם הגענו ל-$m$ פעמים, נדחה.
3. נקבל.

בכל ענף, ה-*NTM* שומר רק מצביע לקודקוד (מספר עד $m$) ומונה (שיכול להיות עד $m$). בסה"כ, מקום $O(\log m)$.

**קונפיגורציה של *TM* עם שני סרטים**

ב-*TM* עם סרט קריאה בלבד, התוכן שלו לא משתנה – ולכן הוא לא חלק מהקונפיגורציה (רק המיקום של הראש). אם *M* הוא מ"ט שרץ במקום תת-לינארי $f(n)$, אז מספר הקונפיגורציות שלו הוא $n \cdot f(n) \cdot 2^{O(f(n))}$.

$n \cdot f(n)$ זה מספר המיקומים האפשריים של שני הראשים. $2^{O(f(n))}$ זה מספר האפשרויות לתוכן של סרט 2.

אם $f(n) = \log n$ אז ה-*TM* רץ בזמן לכל היותר $O(n^2 \log n)$. כלומר, *L* ו-*NL* הם בזמן פולינומי: $L, NL \in P$.

**משפט סביץ' בהרחבה למקום תת-לינארי**

ניזכר במשפט סביץ': לכל פונקציה $f: \mathbb{N} \to \mathbb{R}^+$ כך ש- $f(n) \geq n$, מתקיים: $NSPACE(f(n)) \subseteq SPACE(f^2(n))$.

משפט סביץ' המורחב: לכל פונקציה $f: \mathbb{N} \to \mathbb{R}^+$ כך ש- $f(n) \geq \log n$, מתקיים: $NSPACE(f(n)) \subseteq SPACE(f^2(n))$.

ההוכחה זהה להוכחה של משפט סביץ' הרגיל, כאשר משתמשים ב-*TM* עם שני סרטים (אחד לקריאה בלבד).

משפט סביץ' מוכיח ש- $PSPACE = NPSPACE$, אבל לא ש- $L = NL$. למה?

כי המעבר הריבועי גדול יותר מהלוגריתם. כלומר, המשפט נותן לנו: $NL \subseteq SPACE(\log^2 n)$, שזה לא זמן לוגריתמי.

אז השאלה $L =_? NL$ היא שאלה פתוחה.

### *NL-reducibility*

מכונת *log space transducer (LST)* היא *TM* עם:

- סרט קלט לקריאה בלבד.
- סרט פלט לכתיבה בלבד – שלא יכול לזוז שמאלה.
- סרט עבודה לקריאה וכתיבה – שיכול להכיל עד $O(\log n)$ תווים.

פונקציה **חשיבה במקום לוגריתמי** (*log space computable function, LSC*) היא פונקציה: $f: \Sigma^* \to \Sigma^*$ כך שקיים *LST* כלשהו *M* שמחשב את $f$ – כלומר עבור קלט $x$ שמתקבל על סרט הקלט, יהיה כתוב $f(x)$ על סרט הפלט בסוף הריצה אחרי ש- $M(x)$ עוצר.

שפה *A* תיקרא *log space reducible* לשפה *B* (נסמן $A \leq_L B$) אם קיימת פונקציה *LSC* כך ש: $x \in A \Leftrightarrow f(x) \in B$.

שפה *B* תיקרא ***NL-hard*** אם כל שפה *A* ב-*NL* היא *log space reducible* ל-*A*. כלומר אם מתקיים: $\forall A \in NL : A \leq_L B$.

אם בנוסף, $B \in NL$, אז *B* היא ***NL-complete (NLC)***.

נשים לב שרדוקציה שהיא $\leq_L$ (לוגריתמית במקום) היא גם $\leq_p$ (פולינומית בזמן), כי לוגריתמי במקום גורר פולינומי במקום, ופולינומי במקום גורר פולינומי בזמן.

### *NL-completeness*

<u>טענה</u>: אם $A \leq_L B$, אז $B \in L \Longrightarrow A \in L$.

<u>הוכחה</u>: ראשית, נשים לב לבעיה: $f(x)$ המתקבל מה-*LST* הוא לא בהכרח בגודל $\log n$. אז איך אפשר לומר שהרדוקציה לוגריתמית? כדי לשמור את הפלט, צריך מקום לא-לוגריתמי.

נעשה טריק: במקום לחשב את כל $f(x)$ ואז להעביר את הפלט בתור קלט ל-*TM* של *B*, נריץ את ה-*TM* של *B* במקביל ל*LST*, וכשה-*TM* של *B* צריך עוד תו, נחשב אותו. פורמלית:

יהי $M_B$ ה-*TM* המכריע את *B* (משתמש במקום לוגריתמי). ויהי $M_A$ ה-*LST* המחשב את $f(x)$. בהינתן $x \in_? A$, נריץ את $M_A$ עד שנקבל תו. נריץ את $M_B$ עד שהראש רוצה לקרוא את התו הבא. ואז נריץ את $M_A$ עד שמקבלים את התו הבא, ונריץ את $M_B$ עד שהוא רוצה לקרוא עוד תו... וכו'.

אנחנו צריכים רק לשמור את המיקום של הראש הקורא של $M_B$, שזה דורש מקום $O(\log|f(x)|)$.
ובגלל ש- $f(x) \in O(n^2 \log n) < O(n^3)$, נקבל ש- $O(\log|f(x)|) \in O(\log n)$.

כלומר, המערכת $(M_A, M_B)$ היא *DTM* לוגריתמי במקום שמכריע את *A*.

מסקנה: אם יש שפה אחת שהיא *NL-complete* וגם ב-*L*, אז $L = NL$.

### *PATH* היא *NLC*

כבר הראינו ש- $PATH \in NL$. נראה עכשיו שלכל שפה $A \in NL$, מתקיים $A \leq_L PATH$.

יהי *N*, *NTM* שמכריע את *A* במקום $O(\log n)$. נבנה את הרדוקציה:

אנחנו צריכים לבנות גרף שיתאר את תהליך הריצה של *N*. נשנה את *N* כך שתהיה לו קונפיגורציה מקבלת יחידה (פשוט מכל קונפיגורציה מקבלת, נמחק את הסרט ונזיז את הראש למיקום הכי שמאלי). עכשיו, בהינתן קלט $x$ עבור *N*, נבנה את $\langle G, s, t \rangle$:

- כל קונפיגורציה של *N* תהיה קודקוד של *G*. זה נותן לנו $O(n^2)$ קודקודים.
- נגדיר $s := c_{start},\ t := c_{accept}$.
- נוסיף צלע $(c_i, c_j)$ אם $c_i$ מניבה את $c_j$ בצעד אחד.

אז אם $x \in A$, זה אומר שיש סדרת מעברים בין קונפיגורציות שמסתיימת במצב המקבל. שזה בדיוק מסלול $s \rightsquigarrow t$. והפוך, מסלול $s \rightsquigarrow t$ מתאר סדרת קונפיגורציות שמחילה ב- $c_{start}$ ומגיעה ל- $c_{accept}$.

נוכיח שהפונקציה היא לוגריתמית במקום. ניזכר שאנחנו לא צריכים לשמור את כל הפלט ביחד – רק להעביר אותו למכונה של *PATH*. נתייחס לקונפיגורציה ע"י מספר סדרתי – עבור $O(n^2)$ קונפיגורציות, זה דורש רק $O(\log n^2) = O(2\log n) = O(\log n)$ מקום. לכל אחת, גם נבדוק לאיזה קונפיגורציות אפשר להגיע בצעד יחיד (בשביל הצלעות). אפשר לבדוק כל קונפיגורציה בנפרד ולא צריך לשמור מידע בין לבין, אז כל זה קורה ב- $O(\log n)$ מקום. והבדיקה היא פולינומית בזמן.

**מסקנה: $NL \subseteq P$**

הוכחה: כל $TM$ שמשתמש במקום $f(n)$ רץ בזמן $n \cdot 2^{O(f(n))}$, אז גם הרדוקציה שתיארנו יכולה לרוץ בזמן פולינומי. כלומר לכל שפה $A \in NL$, מתקיים $A \leq_p PATH$. ואפשר לפתור את $PATH$ בזמן פולינומי (ע"י $BFS$, $DFS$). אז כל שפה ב-$NL$ היא גם ב-$P$.

**מחלקת $coNL$**

$$coNL := \{A : \overline{A} \in NL\}$$

טענה – משפט אימרמן (*Immerman – Szelepcsényi*): $NL = coNL$. כלומר כל שפה שהמשלים שלה מוכרע ע"י $DTM$ במקום לוגריתמי, בעצמה גם ניתנת להכרעה ע"י $DTM$ במקום לוגריתמי.

הוכחה: נוכיח ש- $\overline{PATH} \in NL$. אנחנו כבר יודעים שלכל $A \in NL$, מתקיים $x \in A \Leftrightarrow f(x) \in PATH$. כלומר:

$$x \in \overline{A} \Leftrightarrow x \notin A \Leftrightarrow f(x) \notin PATH \Leftrightarrow f(x) \in \overline{PATH}$$

אז תהי שפה $A \in NL$ כלשהי, אנחנו יודעים שמתקיים $\overline{A} \leq_L \overline{PATH}$, כלומר $\overline{A} \in NL$. אז $A \in coNL$, כנדרש.

נותר להוכיח ש- $\overline{PATH} \in NL$. תהי $A \in NL$, ויהי $NTM\ N$ שמכריע את $A$ במקום $O(\log n)$ עבור קלט בגודל $n$.

נשתמש באותה בנייה של הגרף מהרדוקציה של $A \leq_L PATH$, אבל הפעם נרצה לבדוק אם **אין** מסלול $s \rightsquigarrow t$. לכאורה, נוכל פשוט לבדוק את כל המסלולים באורך $m$ שיוצאים מ-$s$, ואם הגענו ל-$t$, נדחה. הבעיה היא שנצטרך לשמור רשימה של המסלולים והקודקודים שעברנו בהם, וזה דורש יותר מקום מ- $O(\log n)$.

נציע רעיון – ספירה בשכבות. עבור $m = |V(G)|$, נשאל: לכמה קודקודים אפשר להגיע מ-$s$ בצעד אחד, לכמה קודקודים אפשר להגיע בשני צעדים... לכמה אפשר להגיע ב-$m$ צעדים. פורמלית, נגדיר:

$$R_i := \{v : v \text{ is reacable from } s \text{ in} \leq i \text{ steps}\}, \qquad c[i] := |R_i|$$

ברור שמתקיים: $R_0 = \{s\},\ c[0] = 1$. ו- $R_m$ זה כל הקודקודים שאפשר להגיע אליהם מ-$s$. אנחנו רוצים לחשב את $c[m]$ בזמן לוגריתמי.

- נתחיל עם $c[0] \leftarrow 1$. $R_0 = \{s\}$
- **שלב $L$:** לכל $0 \leq i < m$: לכל שכבה, נחשב את $c[i+1]$ מתוך $c[i]$.
  - נאתחל $c[i+1] \leftarrow 1$. כי תמיד $s \in R_{i+1}$.
  - לכל קודקוד $v \neq s$: אנחנו רוצים לקבוע אם $v \in R_{i+1}$
    - נאתחל $d \leftarrow 0$ סופר כמה קודקודים יהיו ב- $R_{i+1}$
    - לכל קודקוד $u$: אנחנו רוצים לבדוק את הקודקודים שפוטנציאלים שיגיעו לפני $v$ בדרך.
      - באופן לא דטרמיניסטי, נבחר אם לבדוק את $u$ או לא. אם נבחר לבדוק, אז:
        - נלך $i$ צעדים מ-$s$ (נבחר את הצעדים באופן לא דטרמיניסטי), ואם לא הגענו ל-$u$, נדחה את הענף הזה.
        - אם מצאנו את $u$, נוסיף 1 ל-$d$: $d \leftarrow d + 1$.
        - אם בנוסף, $(u, v) \in E(G)$, אז $c[i+1] \leftarrow c[i+1] + 1$, ונחזור לשלב $L$ (נעבור לקודקוד $v$ הבא).
  - אחרי שבדקנו את כל ה-$u$, נבדוק אם $d = c[i]$. אם לא, נדחה את הענף. זה מוודא שהענף הזה מצא בדיוק את כל הקודקודים של $R_i$.

באינטואיציה, למה זה עובד:

- האי-דטרמיניזם נותן לנו לבחור בדיוק את ה- $c[i]$ קודקודים של $R_i$ שאנחנו רוצים לוודא.
- הבדיקה $d = c[i]$ מבטלת כל ענף עם יותר מדי או פחות מדי.
- לכל $v$, מספיק שיהיה $u \in R_i$ יחיד עם $(u, v) \in E(G)$ כדי שנספור את $v$ לתוך $c[i+1]$.
- אנחנו לא שומרים קבוצות – רק את האינדקס והספירה. זה דורש רק מקום $O(\log m)$.

עכשיו, אנחנו צריכים לבדוק ש-$t$ **לא נמצא** ב- $R_m$:

- נאתחל מונה: $d \leftarrow 0$. זה יספור לכמה קודקודים ב- $R_m$ אפשר להגיע.
- לכל קודקוד $u$:
  - באופן לא דטרמיניסטי, נבחר אם לבדוק את $u$ או לא. אם נבחר לבדוק, אז:

- נעשה $m$ צעדים (נבחר את הצעדים באופן לא דטרמיניסטי), ואם לא הגענו ל-$u$, נדחה את הענף הזה.
- אם $u = t$, כלומר יש מסלול $s \rightsquigarrow t$, **נדחה.**
- אחרת, $d \leftarrow d + 1$.

- בדיקה אם $d = c[m]$.
  - אם לא, אז **נדחה** (כי אחרת, היינו אולי שומרים ענף לא תקין. הדרישה הזו גורמת לנו לשמור רק ענפים שמגיעים ל-$t$, אם יש מסלול כזה).
  - אחרת, **נקבל.**

**סיבוכיות מקום:** שומרים את:

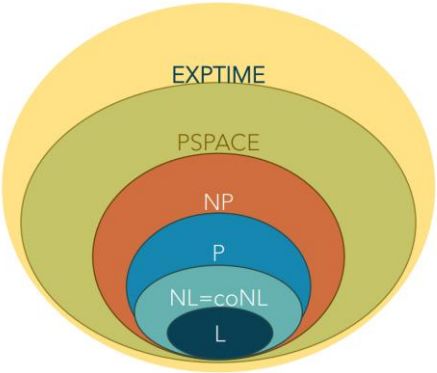
- $m$ (מספר הקודקודים),
- $u, v$ (מספרים סידוריים של הקודקודים הנוכחיים שנבדקים),
- $i$ (אינדקס),
- $c[i],\ c[i+1]$ (מונים מהשלב הראשון),
- $d$ (מונה כללי לווידוא).

זה מספר קבוע של מונים וקידודים, אז $O(\log m)$. שזה קטן יותר מ- $O(\log n)$ ($n$ זה גודל הקלט המקורי). בסה"כ:

$$\overline{PATH} \in NL\text{-}complete \Longrightarrow PATH \in coNL\text{-}complete \Longrightarrow NL = coNL$$

כנדרש.

**סיכום ביניים**




**משפט היררכיית המקום – *The Space Hierarchy Theorem***

המשך יבוא...

.