נושא 2 - סיבוכיות זיכרון

א) הקדמה

בעבר לא היה חשיבות גדולה לסיבוכיות זיכרון (כיוון שתמיד ניתן להוסיף) והמחקרים התמקדו בעיקר בסיבוכיות זמן ריצה. אולם כיום, כיוון שהמידע בעולם רק הולך וגדל, כמות האינפורמציה שמכניסים לתוכנות היא עצומה. במקרים כאלה יש חשיבות מכרעת לסיבוכיות הזכרון, כאשר עיקר ההתעניינות היא באלגוריתמים שסיבוכיות הזכרון שלהם היא סאב-לינארית. כלומר קטנה מסיבוכיות לינארית. ומקיימת $s(n) < n$, כמו $\log(n)$ ו-$\sqrt{n}$.

ב) המחלקות DSPACE ו- NSPACE

עבור פונקציה $S:\mathbb{N}\to\mathbb{N}$ ושפה $L\subseteq\{0,1\}^*$, נאמר כי $L\in DSPACE(S(n))$ אם קיימת מכונת טיורינג דטרמיניסטית M המכריעה את L, כך שלכל $x\in\{0,1\}^*$, M בחישוב על x משתמשת בכל היותר $S(|x|)$ תאי זיכרון בסרט האינסופי. באופן דומה נגדיר את $NSPACE(S(n))$ עבור מ"ט אי-דטרמיניסטית.
סיבוכיות הזכרון של מכונת טיורינג נקבעת לפי החישוב הדורש את מספר תאי הזכרון הגדול ביותר.

__פונקציות space-constructible:__ פונקציה $S:\mathbb{N}\to\mathbb{N}$ תקרא space-constructible אם קיימת מ"ט דטרמיניסטית M אשר בהינתן קלט 1^n כותבת על סרט הפלט את המספר $S(n)$ ורצה בסיבוכיות זכרון $O(S(n))$.
למעשה, אף שזו נראית הגדרה לא טריוויאלית, כמעט כל הפונקציות הן space-constructible

__משפט:__ לכל פונקציה $S:\mathbb{N}\to\mathbb{N}$ שהיא space-constructible מתקיים:

$$DTIME(S(n)) \underset{(1)}{\subseteq} DSPACE(S(n)) \underset{(2)}{\subseteq} NSPACE(S(n)) \underset{(3)}{\subseteq} DTIME\left(2^{O(S(n))}\right)$$

הוכחת (1): תהי $L\in DTIME(S(n))$. אזי קיימת מ"ט דטרמיניסטית M הרצה בזמן $S(n)$ ומכריעה את L.
כיוון שזמן הריצה $S(n)$, M לא יכולה לגשת ליותר מ-$c\cdot S(n)$ תאי זיכרון, ולכן $L\in DSPACE(S(n))$.

הוכחת (2): טריוויאלי. שכן מ"ט דטרמיניסטית היא מקרה פרטי של מ"ט אי-דטרמיניסטית.

הוכחת (3): תהי $L\in NSPACE(S(n))$, אזי קיימת מ"ט אי-דטרמיניסטית M_L המכריעה את L ומשתמשת בכל היותר $O(S(n))$ תאי זיכרון. נשים לב כי ב-M_L כל קונפיגורציה היא באורך לכל היותר $|Q|\cdot|\Gamma|\cdot S(n)=O(S(n))$ שכן על הסרט ניתן לכתוב לכל היותר $S(n)$ תא'ם. לכן בהרצת M_L על קלט $x\in\{0,1\}^*$ יש לכל היותר $2^{O(S(n))}$ קונפיגורציות אפשריות. נבנה גרף קונפיגורציות $G_{M_L,x}$ שבו כל צומת מייצגת קונפיגורציה, יהיה קשת בין שני צמתים c_i,c_j רק אם ב-δ_0 או δ_1 ב-M_L יש מעבר חוקי מ-c_i ל-c_j. $G_{M_L,x}$ הוא גרף מכוון.
נוסיף צומת c^* ונמתח קשת מכל קונפיגורציה מקבלת אל c^*. כעת נוכל להגדיר מכונת טיורינג דטרמיניסטית $\hat{M}_L$ שתכריע את L. עבור כל קלט $x\in\{0,1\}^*$ $\hat{M}_L$ בונה את $G_{M_L,x}$ ומחפשת בעזרת BFS (או DFS) מסלול מ-c_0 אל c^*. אם יש כזה תקבל, אחרת תדחה.

סיבוכיות: בגרף $G_{M_1,x}$ יש $2^{O(s(n))}$ צמתים ולכל היותר $2^{O(s(n))^2}$ קשתות. סיבוכיות BFS היא לינארית לכן סה"כ נקבל כי סיבוכיות $\hat{M}_1$ היא $O(2^{O(s(n))})$. כיוון ש-$\hat{M}_1$ גם מכריע את L, מתקיים כי $L\in DTIME(2^{O(s(n))})$. מ.ש.ל.

<u>גרף קונפיגורציות</u>: יהי $G_{M,x}$ גרף קונפיגורציות עבור מ"ט M עם סיבוכיות זכרון $s(n)$ והקלט $x\in\Sigma^*$ אזי:

(1) ב-$G_{M,x}$ יש $2^{c\cdot s(n)}$ צמתים עבור $c>0$ קבוע, התלוי ב-M.

(2) קיימת נוסחת CNF $\varphi_{M,x}$, כך שלכל שני צמתים (קונפיגורציות) c_1,c_2, מתקיים כי $(c_1,c_2)\in E(G_{M,x})$ אם ורק אם $\varphi_{M,x}(c_1,c_2)=1$ וגם $|\varphi_{M,x}|=O(s(n))$.

2) <u>המחלקות PSPACE ו-NPSPACE</u>

הגדרה: $PSPACE=\bigcup_{i\in\mathbb{N}} DSPACE(n^i)$. כלומר PSPACE היא קבוצת כל השפות שיש להן מ"ט דטרמיניסטית המכריעה אותן בשימוש לכל היותר $O(n^i)$ תאי זכרון, פולינומיאלי בגודל הקלט.

באותו אופן נגדיר $NPSPACE=\bigcup_{i\in\mathbb{N}} NSPACE(n^i)$.

<u>משפטים:</u>

(1) $NP\subseteq PSPACE$.

הוכחה: נוכיח ש-$SAT\in PSPACE$. הוכחה זו מספיקה שכן עבור כל $L\in NP$ קיימת רדוקציה פולינומית ל-SAT, ולכן אם יש מ"ט דטרמיניסטית M המכריעה את SAT, נוכל להעביר כל קלט $x\in L$ באמצעות פונקציית הרדוקציה לקלט ל-SAT ולהעביר ל-M, שתכריע בסיבוכיות זיכרון פולינומיאלית.

נגדיר מ"ט דטרמיניסטית M שעבור כל נוסחת CNF φ עם n משתנים, תבדוק את כל 2^n ההשמות האפשריות. ועבור כל השמה z תכניס את $\langle\varphi,z\rangle$ לאלגוריתם וידוא. אם נמצאה השמה מספקת M תקבל אחרת תדחה. נשים לב ש-M משתמשת ב-$O(n)$ תאים מכיוון שכל השמה משתמשת באותם תאים בזכרון שוב ושוב. לכן $SAT\in PSPACE$.

(2) $PSPACE\subseteq EXP$. ללא הוכחה.

(3) לא ידוע אם $PSPACE\stackrel{?}{=}NP$ או $PSPACE\stackrel{?}{=}P$.

3) <u>המחלקות DL ו-NL</u>

הגדרה: $DL=DSPACE(\log(n))$. כלומר DL* קבוצת כל השפות שיש להן מ"ט דטרמיניסטית המכריעה אותן בשימוש לכל היותר $O(\log(n))$ תאי זיכרון. באותו אופן נגדיר $NL=NSPACE(\log(n))$*.

*ניתן נקראת גם בקיצור L.
*D. מהו קיצור של Deterministic Log-space

<u>משפטים:</u>

*הגדרה נוספת בעמוד 1

(4) לא ידוע אם $DL\stackrel{?}{=}NP$ אך כן ידוע ש-$DL\subseteq P\subseteq EXP$.

הוכחה: לפי משפט אסטל ב' מתקיים: $DSPACE(\log(n)) \subseteq DTIME(2^{O(\log(n))}) = DTIME(n^{O(1)}) = P$

(2) מתקיים $PATH = \{\langle G,s,t\rangle \mid$ G גרף מכוון וקיים בו מסלול מ-s ל-t $\} \in NL$

הוכחה: נגדיר מכונת טיורינג אי-דטרמיניסטית M שתכריע את PATH. M על הקלט $\langle G,s,t\rangle$:

- הגדר $r_0 = s$. אם $t = r_0$ תקבל וסיימנו.
- בכל שלב מ-$i=1$ עד $i=|V(G)|$ לכל היותר:
- בחר שכן* n_i של r_{i-1}; וקבע $r_i := n_i$. אם $t = r_i$ קבל.
- אם לא נמצא $r_i = t$ לכל $i \in |V(G)|$ אזי עברנו על כל הצמתים ולא נמצא מסלול מ-s ל-t. לכן נדחה.

*...מנחשת ...ן כיון שהיא ...דטרמיניסטית

סיבוכיות הזיכרון: של M היא רק כמות שמירת צומת אחת בעץ, שאורכה לכל היותר $\log(n)$.
לכן M משתמש בלכל היותר $O(\log(n))$ תאי זיכרון בסרט.

נכונות: $\langle G,s,t\rangle \in PATH \Leftarrow$ יש מסלול ב-G מ-s ל-t $\Leftarrow$ יש סדרת בחירות של שכנים הקטן מ-$|V(G)|$ שמגדירה מסלול מ-s ל-t $\Leftarrow$ M ינחש נכון מסלול זה $\Leftarrow \langle G,s,t\rangle \in L(M)$.

$\langle G,s,t\rangle \notin PATH \Leftarrow$ לא קיים מסלול מ-s ל-t ב-G $\Leftarrow$ M יגיע ל-$i=|V(G)|$ וידחה.

כיוון ש-M מכריעה את PATH בלכל היותר $O(\log(n))$ תאי זיכרון, מתקיים $PATH \in NL$. מ.ש.ל.

ה) משפט ההיררכיה לסיבוכיות זיכרון

תהיינה f,g שתי פונקציות space-constructible המקיימות $f=o(g)$ (f קטן מ-g) אזי
מתקיים: $DSPACE(f(n)) \subsetneq DSPACE(g(n))$. הוכחת משפט זה היא בשיטת הלכסון כמו שראינו עבור סיבוכיות זמן.

ו) משפט Savitch

משפט: לכל פונקציה שהיא space-constructible $S:\mathbb{N} \to \mathbb{N}$ המקיימת $S(n) \geq \log(n)$ מתקיים:

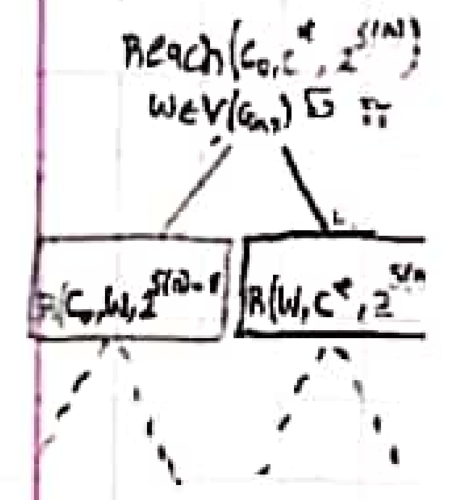
$$NSPACE(S(n)) \subseteq DSPACE(S(n)^2)$$

מסקנות: (1) $NPSPACE = PSPACE$.

(2) $NL \subseteq DSPACE(\log(n)^2)$.

הוכחת המשפט: תהי $L \in NSPACE(S(n))$, אזי קיימת מ"ט אי-דטרמיניסטית M המכריעה את L בסיבוכיות זיכרון $S(n)$. נרצה לבנות מ"ט דטרמיניסטית $\hat{M}$ שעבור הקלט $x \in \Sigma^*$ תבנה את גרף הקונפיגורציות $\underbrace{G_{M,x}}_{\text{עם } 2^{S(n)} \text{ צמתים}}$ ותחפש מסלול בין c_0 ל-c^*. אמנם נשים לב שאין לנו מספיק זיכרון לבנות את הגרף. לכן נשתמש בשיטה זה שמריק כל מסלול חישוב של M בנפרד. נגדיר פונקציה $Reach(u,v,i)$ המחזירה 1 אם יש מסלול ב-$G_{M,x}$ מ-u ל-v באורך i, אחרת מחזירה 0. נרצה ש-$\hat{M}$ תענה כמו הפונקציה $Reach(c_0,c^*,2^{S(n)})$. נשים לב כי $Reach(u,v,2^i)=1$ אם ורק אם קיים $w \in V(G_{M,x})$ עבורו $Reach(u,w,2^{i-1})=1 \cap Reach(w,v,2^{i-1})$. לכן נגדיר

את $\hat{M}$ כך שיחשב את $Reach(c_0,c^*,2^{s(n)})$ ברקורסיה על ידי הפעלת אותו חישוב על כל $w\in V(G_{M,x})$ ניחשב את $Reach(c_0,w,2^{s(n)-1})$ וגם את $Reach(w,c^*,2^{s(n)-1})$ ורק אם שניהם יחזירו 1 יקבל אם אין w כזה יחזיר 0. גם על כל אחד משני חישובים אלו יופעל הרקורסיה אלגוריתם זה, וכן הלאה, כאשר לא חוזרים על אותה צומת באותו מסלול חישוב.



כך $\hat{M}$ עובר על כל המסלולים האפשריים אך לכל היותר מסלול באורך $s(n)$ בכל זמן נתון, שכן אם קיים מסלול מקבל הוא באורך $s(n)$ כי אחרת סיבוכיות M הייתה גדולה יותר. בנוסף, צריך לשמור כל הקונפיגורציות באותו מסלול חישוב, וכל קונפיגורציה באורך לכל היותר $s(n)$. לכן סיבוכיות $O(s(n)^2)$. מ.ש.ל

ג) PSPACE - Complete

<u>הגדרה</u>: שפה $L\in PSPACE\text{-}Complete$ אם מקיימת שני תנאים:

(1) $L\in PSPACE$.

(2) קיימת רדוקציה פולינומית (בזמן) מכל שפה $L'\in PSPACE$ אל L, $L'\le_p L$.

דוגמה: השפה $SPACETM=\{\langle M,w,1^n\rangle : M$ מ"ט דטרמיניסטית המקבלת את w בסיבוכיות $n\}$ היא שלמה ב-PSPACE.

הוכחה: (1) $SPACETM\in PSPACE$, שכן כל מ"ט דטרמיניסטית המסמלצת את M ועונה כמוה מכריעה את SPACETM.

(2) תהי $L\in PSPACE$, אזי קיימת לה מ"ט M המכריעה אותה בסיבוכיות זיכרון $c_1 n^{c_2}$ עבור c_1,c_2 קבועים. בהינתן קלט $x\in\{0,1\}^*$ נגדיר את פונקציית הרדוקציה $f(x)=\langle M,x,1^{c_1|x|^{c_2}}\rangle$. סיבוכיות הזמן פולינומית. אם $x\in L$ אזי M מקבלת בסיבוכיות $c_1|x|^{c_2}$ ואזי $\langle M,x,1^{c_1|x|^{c_2}}\rangle\in SPACETM$.

זוהי שפה מאוד מלאכותית. נחפש שפה שלמה ב-PSPACE טבעית יותר שתעזור לנו למצוא שפות נוספות.

ד) השפה TQBF

<u>Quantified Boolean Formula (QBF)</u>: זוהי נוסחה בוליאנית עם כמתים $(\forall,\exists)$ מהצורה $Q_1x_1Q_2x_2\dots Q_nx_n\varphi(x_1,x_2,\dots,x_n)$, כאשר $Q\in\{\forall,\exists\}$, ו-φ נוסחה בוליאנית (לאו דווקא CNF) מעל n משתנים $x_1,x_2,\dots,x_n$. כל נוסחת QBF מקבלת ערך 1 (True) או 0 (False).

דוגמאות: (1) $\forall x\exists y(x\cap y)\cup(\bar{x}\cap\bar{y})$ - ערך הנוסחה T כי עבור כל x נבחר $y=x$.

(2) $\forall x\forall y(x\cap y)\cup(\bar{x}\cap\bar{y})$ - ערך הנוסחה F כי לא מתקיים עבור $x\neq y$, $x=0$ ו-$y=1$ לדוגמה.

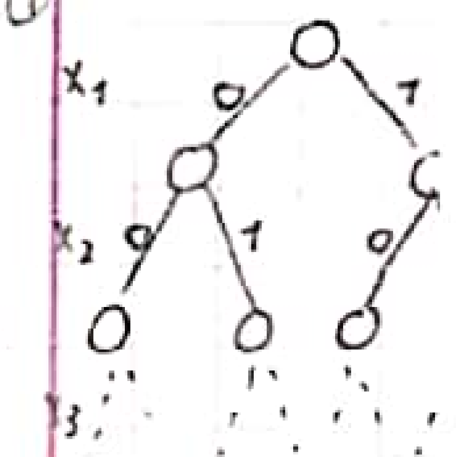
(3) ניסוח חדש ל-SAT באמצעות QBF: $SAT=\{\psi=\exists x_1\exists x_2\dots\exists x_n\varphi(x_1,x_2,\dots,x_n)\mid \varphi$ נוסחת CNF ו-$\psi=T\}$

(4) ניסוח חדש ל-Tautology באמצעות QBF: $Tautology=\{\psi=\forall x_1\forall x_2\dots\forall x_n\varphi(x_1,\dots,x_n)\mid \varphi$ נוסחת CNF ו-$\psi=T\}$

<u>השפה TQBF</u>: $TQBF=\{\psi\mid \psi$ היא נוסחת QBF וגם $\psi=T\}$.

<u>משפט</u>: $TQBF\in PSPACE\text{-}Complete$.

לא פורשים
רק דרך פולינומית
רק שכן זה לא
שעשתה רדוקציה.
שלפונקציה זו
לכל שפה ב-P
P-complete, לפי
Q, תא' 3 נדרוש
תחשים הרדוקציה
צריך החישוב
למחלקה, אחרת
אינו כלום.



① נוכיח $TQBF\in PSPACE$.

נגדיר מ"ט דטרמיניסטית M שעבור כל קלט $\psi\in\{0,1\}^*$ קודם תבדוק אם ψ הוא אכן נוסחת QBF. במידה וכן, M תעבור באופן רקורסיבי על כל ההשמות של ψ. בכל פעם שמפעילה את הקריאה הרקורסיבית על משתנה x_i, היא זוכרת את כל ההשמות הנוכחיות עבור $x_1,x_2,\ldots,x_{i-1}$, קודם בודקים את הרקורסיה עבור $x_i=0$, ולאחר מכן על $x_i=1$ (במידה וצריך). אם x_i הוא עלה, כלומר $i=n$, אזי לוקחים את ההשמה הנוכחית ובודקים אותה על ψ. ומחזירים מה שיצא.
אם הכמת על x_i הוא $\exists$ (קיים) אזי אם עבור $x_i=0$ חזר True אין צורך לבדוק $x_i=1$ ומחזירים מהרקורסיה True. אם לא חזר False עבור $x_i=0$, מפעילים את הרקורסיה על $x_i=1$ ועונים כמוה.
אם הכמת על x_i הוא $\forall$ (לכל) צריך על $x_i=0$ אם חזר True צריך על $x_i=1$ ועונה כמוהו, אם לא חזר False עבור $x_i=0$, אין צורך לבדוק $x_i=1$ ונחזיר False.
M מכריעה את TQBF שכן בודקת כל השמה אפשרית המטפלת לתת ל-ψ ערך True. סיבוכיות הזיכרון של M היא פולינומיאלית, שכן בכל רגע נתון, צריך רק לשמור את ההשמה הנוכחית ואת הערך שחזר מהרקורסיה. לכן סיבוכיות הזיכרון היא $O(n)$. מסקנה $TQBF\in PSPACE$.

② נוכיח לכל $L\in PSPACE$ מתקיים $L\le_p TQBF$.

תהי $L\in PSPACE$, אזי יש מ"ט דטרמיניסטית M המכריעה את L בסיבוכיות זיכרון פולינומיאלית $s(n)$. נרדה למצוא פונקציה F שעבור כל קלט $x\in\{0,1\}^*$, $f(x)$ היא נוסחת QBF המקבלת ערך True אם ורק אם בגרף הקונפיגורציות של G על x - $G_{M,x}$, יש מסלול מקבל מ-c_0 (קונפיגורציה התחלתית) ל-c^* (קונפיגורציה מקבלת). נזכור כי בגרף הקונפיגורציות של מ"ט עם סיבוכיות זיכרון פולינומית יש לכל היותר $2^{O(s(n))}$ קונפיגורציות, לכן אין באפשרותנו לבנות גרף כזה. נוכיח בדרך דומה להוכחת משפט סביץ'.

נגדיר את הפונקציה: $\psi_i(c,c')=\exists c''\,\forall D_1,D_2\,\big((D_1=c)\cap(D_2=c'')\big)\cup\big((D_1=c'')\cap(D_2=c')\big)\to\psi_{i-1}(D_1,D_2)$ *

נוסף: מתקיים: $\psi_1(c,c')=True \iff (c,c')\in E(G_{M,x})$

המיוצגת נוסחת QBF, שכן את הביטוי $A=B$ ניתן לייצג בתור נוסחת CNF הבאה $(A\cap B)\cup(\bar A\cap\bar B)$. וכן את הגרירה $A\to B$ ניתן לייצג בתור נוסחת CNF $(\bar A\cup B)$.

נוסחת QBF זו תחזיר True אם ורק אם ק"ים מסלול ב-$G_{M,x}$ מ-c ל-c' באורך לכל היותר 2^i. לפשט רק דרושה להיות קונפיגורציה c'' שיש מסלול באורך 2^{i-1} מ-c אליה, וכן יש מסלול באורך 2^{i-1} ממנה ל-c'.

פונקציית הרדוקציה שאנו מחפשים תהיה $F(x)=\psi_{s(n)}(c_0,c^*)$. נכונות הרדוקציה נובעת מההסבר במשפט סביץ'.
סיבוכיות: נשים לב כי $|\psi_1|=O(s(n))$ ו-$|\psi_i|=|\psi_{i-1}|+O(s(n))$. לכן כל הפעלה רקורסיבית מוסיפה $O(s(n))$. מכאן $|\psi_{s(n)}|=O(s(n)^2)$. נמצא כי סיבוכיות הרדוקציה אכן פולינומית. מ.ש.ל.

6) NL-Completenes

הגדרות:

(1) חישוב בזיכרון לוגריתמי (מובלע): נאמר כי פונקציה $f:\{0,1\}^* \to \{0,1\}^*$ ניתנת לחישוב בזיכרון לוגריתמי באופן מובלע (implicit) אם f חסומה פולינומית, כלומר קיים קבוע c כך שעבור כל $x\in\{0,1\}^*$ מתקיים $|f(x)|\le O(|x|^c)$, וגם שתי השפות הבאות שייכות ל-DL.

$$L_f=\{\langle x,i\rangle \mid f(x)_i=1\} \quad , \quad L'_f=\{\langle x,i\rangle \mid |f(x)|\ge i\}$$

$F(x)_i$ זוהי הסיבית ה-i ב-$F(x)$.

(2) רדוקציה בזיכרון לוגריתמי (מובלע): שפה L' ניתנת לרדוקציה בזיכרון לוגריתמי מובלע לשפה L אם קיימת פונקציית רדוקציה F הניתנת לחישוב בזיכרון לוגריתמי (מובלע)*, וגם לכל $x\in\{0,1\}^*$ $x\in L' \iff F(x)\in L$. במידה וקיימת פונקציה F כזו נאמר כי L' ניתנת לרדוקציה בזיכרון לוגריתמי מובלע ל-L, בסימון $L'\le_{\ell} L$.

* הגדרה שקולה: קיימת מ"ט המחשבת את F בזיכרון לוגריתמי בסרט העבודה ויש לה סרט פלט ... איננו נספר בסרט הפלט. ניתן לבצע בסוף כל ... יציאה סלפק.

NL-completenes

נאמר כי שפה $L\in NL\text{-complete}$ אם מתקיימים שני תנאים:

(1) $L\in NL$.

(2) לכל $L'\in NL$ מתקיים $L'\le_{\ell} L$.

* רדוקציות פולינומיות רגילות הן חזקות מדי עבור המחלקה NL לכן היינו צריכים להגדיר מושג חדש.

דוגמה לשפה שהיא NL-COMPLETE

משפט: $PATH \in NL\text{-COMPLETE}$

הוכחה: בסעיף ג' בפרק זה הוכחנו $PATH\in NL$. נותר להוכיח לכל $L\in NL$ מתקיים: $L\le_{\ell} PATH$.

תהי $L\in NL$, אזי קיימת לה מ"ט אי-דטרמיניסטית M_L המכריעה את L בזיכרון $O(\log(n))$.

בהינתן $x\in\{0,1\}^*$ נסמן ב-$G_{M_L,x}$ את גרף הקונפיגורציות של M_L בריצה על x. פונקציית הרדוקציה מ-L ל-$PATH$ היא $F(x)=(G_{M_L,x}, C_{start}, C_{acc})$. אכן מתקיים $x\in L \iff F(x)\in PATH$, שכן אם $x\in L$ יש מסלול מקבל בגרף הקונפיגורציות ולכן $F(x)\in PATH$, ואם $F(x)\in PATH$ יש מסלול מקבל בגרף הקונפיגורציות של M_L ולכן M יקבל את x כי הוא עוצר אז"ק ואז $x\in L$.

נותר להוכיח כי F ניתנת לחישוב בזיכרון לוגריתמי מובלע. נוכיח זאת בשני שלבים:

(1) F חסומה פולינומית - לפי משפט בסעיף ב' ב-$G_{M_L,x}$ יש $2^{O(\log(n))}$ צמתים על קלט באורך n, ולכן מתקיים $|G_{M_L,x}|=O(2^{O(\log(n))})\le O(n^c)$ עבור c קבוע חיובי כלשהו. מסקנה: F חסומה פולינומית.

(2) $L_F, L'_F\in DL$ - מתקיים $L'_F\in DL$ שכן ניתן לחשב את $F(x)$ בזיכרון לוגריתמי ואז בקלות לבדוק מה האורך שלו. מתקיים גם $L_F\in DL$ שכן את C_{start} ו-C^* גם כן ניתן לחשב בזיכרון לוגריתמי

ואז לחשב היכן יש אחדות. עבור הגרף $G_{M_L,x}$ יש לחשב כל תא $[i][j]$ במטריצת השכנויות ולבדוק היכן יש אחדות. לשם כך עבור כל שני קונפיגורציות c_i ו-c_j יש לחשב האם $(c_i,c_j)\in E(G_{M_L,x})$, כלומר האם קיים מעבר מ-$c_i$ ל-c_j לפי δ_0 או δ_1. הוכחנו בסעיף קודם שניתן לחשב זאת בזכרון לינארי ב-$|c_i|$ ו-$|c_j|$, ואנו כיוון ש $|c_i|,|c_j| = O(\log(n))$, כלומר באורך $O(\log(n))$ אזי נקבל שניתן לחשב זאת בזכרון של $O(\log(n))$, ולכן $L_F\in DL$.

י) הגדרה נוספת למחלקה NL

שפה L שייכת למחלקה NL אם קיימת לה מ"ט **דטרמיניסטית** עם סרט קלט נוסף לקריאה יחידה (כלומר ניתן לנוע רק ימינה), וקיים פולינום $p(\cdot)$ כך שלכל קלט $x\in\{0,1\}^*$ מתקיים:

$$x\in L \iff \exists\, y\in\{0,1\}^* \quad M(x,y)=1$$

כאשר y כתוב בסרט הקלט הנוסף של הקריאה היחידה בעת חישוב M על (x,y), וגם M משתמשת בזכרון לוגריתמי על כל סרטי העבודה.

יא) המחלקה $coNL$

נגדיר $coNL=\{L\subseteq\{0,1\}^* \mid \bar{L}\in NL\}$ כל השפות המשלימות של השפות ב-NL.

נשים לב כי $\overline{PATH}\in coNL$ וגם $\overline{PATH}\in coNL\text{-}COMPLETE$ שכן ניתן להשתמש באותה רדוקציה שראינו בסעיף ט' להוכחת $PATH\in NL\text{-}COMPLETE$ להוכחת $\overline{PATH}\in coNL\text{-}COMPLETE$, כלומר אם $L\le_\ell PATH$ לפי רדוקציה f אזי גם $\bar{L}\le_\ell \overline{PATH}$ לפי אותה f.

משפט Zimmerman-Szelepcsény: $\overline{PATH}\in NL$

הוכחה: נוכיח באמצעות הגדרה אלטרנטיבית ל-NL מסעיף קודם. נראה שקיימת מ"ט דטרמיניסטית M המקבלת כקלט $\langle G,s,t\rangle$ ועד y שבאמצעותו ניתן לוודא שאין ב-G מסלול מ-s ל-t ע"י קריאה יחידה של y ושימוש בזכרון לוגריתמי בסרטי העבודה. לפני הגדרת העד y נגדיר

$C_i=\{v\in V(G) \mid$ ניתן להגיע מ-s ל-v בלכל היותר i צעדים$\}$, ונסמן $S_i=|C_i|$ מספר הקודקודים ב-C_i.

העד y יהיה מורכב מ-$n+1$ חלקים* כך שמכל חלק i ניתן להסיק מהו S_i. מהחלק האחרון ניתן להסיק מהו S_n כלומר מהו מספר הקודקודים ב-G שאליהם ניתן להגיע מצומת s. בעד y יהיה חלק נוסף לאחר כל $n+1$ החלקים שבאמצעותו ניתן לבדוק האם $t\in C_n$ ובכך להכריע האם $\langle G,s,t\rangle$ שייך ל-$\overline{PATH}$ או לא. כעת נספר איך כל חלק עובד. כל חלק $0\le i\le n$ יכיל לכל צומת $v\in V(G)$ הוכחה האם $v\in C_i$ או $v\notin C_i$. למכונה M יהיה counter שיספור את כל הקודקודים $v\in C_i$ ובכך ידע מהו S_i. $n=|V(G)|$

אם $v\in C_i$ אזי ב-y יהיה מסלול באורך i שמתחיל ב-s ונגמר ב-v. המכונה M תוודא שכל שני צמתים

מוכלות במסלול זה אכן נמצאות ב-$E(G)$ שהצמתים הראשון והאחרון הם אכן s ו-v.

אם $v\notin C_i$ אזי בעץ y יהיה S_{i-1} מסלולים שמתחילים ב-s ונגמרים בכל צומת מ-$w\in C_{i-1}$ שהם באורך i-1. המכונה M תוודא ש-$w\neq v$ וגם ש-$(w,v)\notin E(G)$ לכל $w\in C_{i-1}$ ובכך תדע בוודאות ש-$v\notin C_i$. כך נחשב כל S_i עבור $0\le i\le n$.

כעת כדי להראות ש-$t\notin C_n$ בעץ y יהיו S_{n-1} מסלולים שמתחילים ב-s ונגמרים בכל הצמתים $w\in C_{n-1}$ שהם באורך n-1. המכונה M תוודא ש-$w\neq t$ וגם ש-$(w,t)\notin E(G)$ לכל $w\in C_{n-1}$.

אורך העץ y הוא $n^4\log(n)$, הזכרון ש-M משתמשת בו בסרט העבודה הוא לוגריתמי שכן צריכה לשמור רק מונה לכל חלק שעברנו, מונה שמוודא שעברנו על כל הצמתים בכל חלק.

מונה שמוודא שהמסלול לכל צומת הוא באורך i-1, ומונה אחרון שסופר את הצמתים $v\in C_i$.

הוכחת נכונות- אם $\langle G,s,t\rangle\in\overline{PATH}$ אזי קיים עד y שבאמצעותו M תחזיר נכון לכל הבדיקות שתיארנו ואזי $M(x,y)=1$. אם $\langle G,s,t\rangle\notin\overline{PATH}$ אזי לא קיים עד שעבורו M תחזיר נכון בכל הבדיקות, ובהכרח תהיה בדיקה שתיכשל, ולכן M על קלט (x,y) תחזיר 0.

מסקנה: $NL=coNL$.

הוכחה: כיוון 1- תהי $L\in coNL$ אזי קיימת רדוקציה $L\le_{L}\overline{PATH}$ כיוון ש-$\overline{PATH}\in coNL\text{-}COMPLETE$.
בנוסף, כיוון ש-$\overline{PATH}\in NL$ קיימת מ"ט א'-דטרמיניסטית בזכרון לוגריתמי שמכריעה את $\overline{PATH}$. לכן לכל $x\in\{0,1\}^*$ נוכל לבנות מכונת טיורינג א'-דטרמיניסטית בזכרון לוגריתמי שתחשב את $f(x)$, תכניס אותו ל-M ותענה כמוה.

כיוון 2- תהי $L\in NL$ אזי $\bar{L}\in coNL$. לפי ההוכחה בכיוון 1 גם מתקיים $\bar{L}\in NL$ ולכן $L\in coNL$. מ.ש.ל.