מודלים חישוביים, חישוביות וסיבוכיות - 67521

22 באוגוסט 2024

1 עודד - שיעור 1 - 05.05.2024

1.1 מבוא

בקורסים מבני נתונים ואלגוריתמים למדנו מה אפשר לעשות באופן יעיל בעזרת מחשב, בקורס הזה נתרכז במה **אי אפשר לעשות בעזרת מחשב.** וגם אילו בעיות יקחו כל כך הרבה זמן ומשאבים שהן בלתי פתירות. הקורס מתחלק לשלושה חלקים :

1. מודלים חישוביים.
2. חישוביות.
3. סיבוכיות.

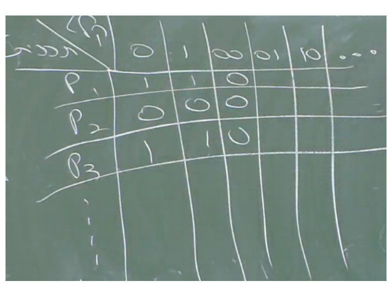
בחלק הראשון נדבר על איך אפשר להגדיר מה זה מחשב, מה זה חישוב, מה זה ריצה. בחלק השני נדבר על מה אפשר ואי אפשר לחשב, ובחלק השלישי נדבר על מגבלות משאבים - מה אפשר לחשב בהינתן מגבלת זיכרון או זמן.
הציון בקורס 85% מבחן, 15% מטלות בית.

דוגמא לטענה מהחלק השני של הקורס :

טענה. קיימת פונקציה בוליאנית (מקבלת מחרוזת בינארית ומחזירה $0/1$) שלא ניתנת לחישוב על ידי תכנית $C++$.

הוכחה. כל תכנית $C++$ מוגדרת על ידי מחרוזת של קבוצה סופית של סימנים באורך סופי. לכן יש $\aleph_0$ מחרוזות כאלה. מצד שני, פונקציה בוליאנית היא מחרוזת אינסופית מעל $\{0,1\}$ כלומר יש $\aleph$ כאלה, ולכן יש יותר פונקציות בוליאניות מתכניות $C++$ ובפרט יש פונקציה שאין עבורה תכנית שמחשבת אותה. $\square$

הוכחה. (ישירה) נגדיר טבלה שבה יש שורה לכל תכנית מחשב שמחשבת פונקציה בוליאנית, ועמודה לכל קלט אפשרי. נסדר את התכניות בסדר לקסיקוגרפי עולה, וכנ"ל את הקלטים. תוכן הטבלה בקואורדינטה ה-(i,j) הוא הפלט של התכנית בשורה ה-i עבור הקלט בעמודה ה-j.



איור 1 :

נגדיר פונקציה f לפי האלכסון ההפוך, כלומר לכל קלט x, מתקיים $f(x)=0$ אם במקום המתאים באלכסון רשום 1 ואחרת $f(x)=1$.
נראה כי f לא חשיבה על ידי תכנית $C++$. נניח בשלילה שיש תכנית $C++$ שמחשבת את f. אז לפי הגדרת הטבלה התכנית הזו מופיעה באחת השורות. נקרא לתכנית p_i (בשורה i). נסתכל על תוצאת הריצה של p_i על הקלט x שמתאים לעמודה i. אם התוצאה של הריצה היא 0, אז לפי הגדרת f, מתקיים $f(x)=1$, ואם $p_i(x)=1$ אז לפי הגדרת f מתקיים $f(x)=0$. כלומר p_i לא מחשבת את f. סתירה. $\square$

בהמשך הקורס נרצה להראות שיש דוגמאות נוספות, אולי יותר רלוונטיות, שגם הן לא ניתנות לחישוב. למשל:

טענה. נתבונן בבעיה של לקבל בתור קלט זוג תכניות $C++$, ולהחזיר בתור פלט האם הן שקולות או לא. בעיה זו אינה חשיבה. (כלומר לא ניתן לכתוב תכנית מחשב אשר תגיד האם שני אלגוריתמים הם שקולים או לא). ניגש לבעיה זו בהמשך הקורס.

כעת נתבונן בטעימה מהחלק השלישי של הקורס:

בעיה. בעיית מעגל **אוילר**.
קלט: גרף G לא מכוון.
פלט: האם קיים ב-G מעגל אוילר, כלומר מסלול מעגלי שמבקר בכל **קשת** פעם אחת בדיוק.

בעיה. בעיית מעגל **המילטוני**.
קלט: גרף G לא מכוון.
פלט: האם קיים ב-G מעגל המילטוני, כלומר מסלול מעגלי שמבקר בכל **צומת/קודקוד** פעם אחת בדיוק.

ניזכר בתנאי הכרחי ומספיק לקיומו של מעגל אוילר: G קשיר וגם כל הדרגות זוגיות. זה תנאי פשוט שניתן לבדוק על ידי אלגוריתם יעיל. על כן זו בעיה חשיבה.
לגבי בעיית המעגל ההמילטוני, קיים אלגוריתם לא יעיל הפותר את הבעיה. האלגוריתם עובר על כל הפרמוטציות האפשריות על הצמתים ולכל אחד בודק האם יש קשת מכל צומת לבא אחריו בפרמוטציה, וכן מהאחרון לראשון.
שאלת $\$1M$ של מדעי המחשב (ליטרלי): האם קיים אלגוריתם יעיל לבעיית מעגל המילטוני?
לצורך הקורס הזה זמן ריצה יעיל מוגדר להיות פולינומי באורך הקלט. למשל $O(n^3)$ או $O(n^{100})$.
בחלק השלישי של הקורס נדבר על הבעיה הזו ועל משפחה שלמה של בעיות השקולות לה. לא נצליח להוכיח האם היא ניתנת או לא ניתנת לפתרון יעיל, אבל נצליח להבין אותה יותר טוב.

1.2 חלק 1 של הקורס - מודלים חישוביים

1.2.1 הגדרות, א"ב, מילה, שפה

הגדרה. **א"ב** הוא קבוצה סופית של סימנים (אותיות). יסומן בדר"כ על ידי Σ.

דוגמאות: $\Sigma=\{0,1\}$, $\Sigma=\{a,b,c\}$, $\Sigma=$אותיות בעברית

הגדרה. **מילה** היא סדרה סופית של אותיות א"ב. את המילה הריקה נסמן ב-ε.

הגדרה. **שפה** היא קבוצה (סופית/אינסופית) של מילים. את השפה הריקה נסמן ב-$\emptyset$. את שפת כל המילים בא"ב Σ נסמן ב-Σ^*.

דוגמאות:

$$L_1 = \{aba, ab, aaaa\}$$
$$L_2 = \{w : w \in \{0,1\}^* \; s.t. \; \# \; of \; 0's \; in \; w \; is \; even\}$$
$$L_3 = \{w : w \in \{0,1\}^* \; s.t. \; w \; contains \; the \; sequence \; 001\}$$

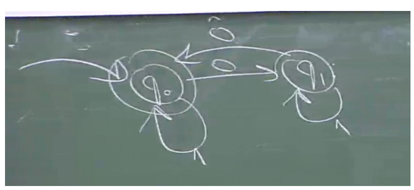
אז:
$0011 \in L_2$,$01010 \notin L_2$
$010 \notin L_3$,$011001 \in L_3$

הגדרה. **אורך של מילה** $w \in \Sigma^*$ יסומן על ידי $|w|$ ויוגדר להיות מספר הסימנים ב-w. $|\varepsilon| = 0$.

1.2.2 אוטומטים סופיים - דטרמיניסטים

הגדרה. **אוטומט סופי דטרמיניסטי (DFA - Deteministic Finite Automata)** הוא חמישייה $A = \langle Q, \Sigma, q_0, \delta, F\rangle$ כאשר:

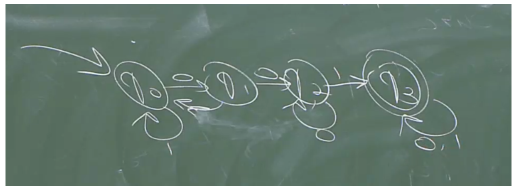
- Q קבוצה סופית של מצבים.
- Σ א"ב של הקלטים.
- $q_0 \in Q$ מצב התחלתי.
- $F \subseteq Q$ מצבים מקבלים.
- $\delta : Q \times \Sigma \to Q$ פונקציית מעבר.


איור 2:

דוגמה. בציור הנ"ל של DFA יש צומת לכל מצב של האוטומט. קשת נכנסת מיוחדת לסימון המצב ההתחלתי, ועיגול נוסף סביב כל מצב מקבל (מצב ב-F). קשתות מכוונות מסומנות באותיות Σ לתיאור פונקציית המעבר δ.
אוטומט מקבל מילה בתור קלט (מצד שמאל). הוא קורא אותו אות אחרי אות. על פי האות, הוא עובר בקשת המכוונת המתאימה. לדוגמא עבור 01101 נעבור במצבים בסדר הבא: $q_0 \to q_1 \to q_1 \to q_1 \to q_0 \to q_0$.
האוטומט סיים ב-q_0. זה מצב מקבל (כי $q_0 \in F$ ומסומן בעיגול כפול) ועל כן נגיד שהאוטומט קיבל את המילה. (אחרת היינו אומרים שהריצה היא "לא מקבלת"). נשים לב כי האוטומט מקבל רק מילים שמספר האפסים בהן זוגי, כלומר בדיוק השפה $L_2 = \{w : w \in \{0,1\}^* \; s.t. \; \# \; of \; 0's \; in \; w \; is \; even\}$.

הגדרה. עבור DFA, A, **השפה של A** המסומנת ב-$L(A)$ היא קבוצת כל המילים w כך שהריצה של A על w מסתיימת במצב מקבל, כלומר במצב $q \in F$.
אם ריצה של A על w מסתיימת במצב מקבל, נגיד ש-A **מקבל את** w ואחרת נגיד ש-A **לא מקבל את** w.

תרגיל. ציירו DFA כך ש-$L(A) = L_3$ כלומר קבוצת כל המילים מעל $\Sigma = \{0,1\}$ שמכילות את הרצף 001.



איור 3 :

הגדרה. **(ריצה של אוטומט)** על קלט $w \in \Sigma^*$ $A = \langle Q, \Sigma, q_0, \delta, F\rangle$ היא סדרה של מצבים $r = r_0, r_1, ..., r_n$ כאשר $n = |w|$. נסמן $w = w_1 w_2 ... w_n$ כאשר לכל i, $w_i \in \Sigma$. r היא **ריצה** של A על w אמ"ם לכל $i \geq 0$ מתקיים $r_{i+1} = \delta(r_i, w_{i+1})$ וגם $r_0 = q_0$.
אם $r_n \in F$ נגיד שהריצה **מקבלת** אם $r_n \notin F$ נגיד שהריצה **לא מקבלת**.

הגדרה. עבור שפה $L \subseteq \Sigma^*$ נגיד ש-L **רגולרית** אם קיים DFA, A כך ש-$L(A) = L$. (כלומר אפשר לקבל אותה בדיוק על ידי אוטומט).

הגדרה. נסמן ב-$\underline{REG}$ את מחלקת כל השפות הרגולריות.
כדי להראות ששפה $L \in REG$, אפשר להראות אוטומט ספציפי שמקבל. נראה בהמשך כלים נוספים וגם איך להראות ששפה $L \notin REG$.

1.2.3 פעולות על שפות

הגדרה. **פעולות על שפות:**

1. משלים: $\overline{L} = \{w : w \in \Sigma^* \wedge w \notin L\}$
2. חיתוך: $L_1 \cap L_2 = \{w : w \in L_1 \wedge w \in L_2\}$
3. איחוד: $L_1 \cup L_2 = \{w : w \in L_1 \vee w \in L_2\}$
4. שרשור: $L_1 \circ L_2 = \{w : w = w_1 \circ w_2 \wedge w_1 \in L_1 \wedge w_2 \in L_2\}$ ($\circ$ מסמל שרשור)
5. כוכבית: $L^* = \{w : w = w_1 \circ ... \circ w_t \wedge \forall i, w_i \in L \wedge t \in \mathbb{N} \cup \{0\}\}$ (נשים לב כי $\varepsilon \in L^*$)

1.2.4 סגירות המחלקה REG לפעולות

טענה. המחלקה REG סגורה לפעולת משלים. כלומר אם $L \in REG$ אז $\overline{L} \in REG$. כלומר אם ל-L יש DFA A, כך ש-$L(A) = L$, אז ל-$\overline{L}$ גם יש DFA A' כך ש-$L(A') = \overline{L}$.

הוכחה. בהינתן $A = \langle Q, \Sigma, q_0, \delta, F\rangle$ נבנה את האוטומט $A' = \langle Q, \Sigma, q_0, \delta, F'\rangle$ כאשר $F' = Q \setminus F$.
הוכחת נכונות: נראה ש-$L(A') = \overline{L}$. תהי $w \in \Sigma^*$. נסתכל על ריצה של A על w, כלומר סדרת מצבים $r = r_0, ..., r_n$ וקלט-$w' = w_1, ..., w_n$ כך שלכל i, $w_i \in \Sigma$. וגם לכל $i \geq 0$ מתקיים $\delta(r_i, w_{i+1}) = r_{i+1}$ וגם $r_0 = q_0$. אותה הסדרה r מתארת גם ריצה של A' על w כי δ משותפת לשני האוטומטים וכך גם q_0.

- אם $w \in L$, אז לפי נכונות A, $w \in L(A)$ כלומר r היא ריצה שמסתיימת במצב מקבל, כלומר $r_n \in F$. לכן $r_n \notin F'$ (לפי הגדרת F') ולכן ריצת A' על w היא ריצה לא מקבלת, כלומר $w \notin L(A')$.
- אם $w \notin L$, אז לפי נכונות A, $w \notin L(A)$ כלומר r היא ריצה שמסתיימת במצב לא מקבל, כלומר $r_n \notin F$. לכן $r_n \in F'$ (לפי הגדרת F') ולכן ריצת A' על w היא ריצה מקבלת, כלומר $w \in L(A')$.

על כן נסיק כי $L(A') = \overline{L}$. $\square$

> טענה. המחלקה REG סגורה לפעולת חיתוך. כלומר אם $L_1, L_2 \in REG$ אז $L_1 \cap L_2 \in REG$.

הוכחה. צריך להראות שאם יש זוג אוטומטים A_1, A_2 כך ש-$L(A_1) = L_1$ וגם $L(A_2) = L_2$, אז קיים אוטומט A כך ש- $L(A) = L_1 \cap L_2$. אי אפשר להריץ את A_1 ו-A_2 אחד אחרי השני כי אוטומט סופי דטרמיניסטי קורא כל אות בקלט פעם אחת ויחידה. לכן נרצה "להריץ במקביל" את שני האוטומטים. נגדיר אוטומט מכפלה שבו כל מצב מייצג זוג מצבים, אחד מכל אוטומט, והריצה עליו מדמה ריצה בו זמנית על שני האוטומטים. זה הרעיון, **עכשיו באופן פורמלי:**
נסמן:

$$A_1 = \langle Q_1, \Sigma_1, q_0^{(1)}, \delta_1, F_1 \rangle$$
$$A_2 = \langle Q_2, \Sigma_2, q_0^{(2)}, \delta_2, F_2 \rangle$$
$$A = \langle Q, \Sigma, q_0, \delta, F \rangle$$

כאשר:

$$Q = Q_1 \times Q_2$$
$$q_0 = (q_0^{(1)}, q_0^{(2)})$$
$$F = F_1 \times F_2 = \{(q_1, q_2) : q_1 \in F_1 \wedge q_2 \in F_2\}$$

ו- $\delta : Q \times \Sigma \to Q$ מוגדרת באופן הבא: $\delta\left((q_1, q_2), \sigma\right) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$ לכל $q_1 \in Q_1$,$q_2 \in Q_2$ ו- $\sigma \in \Sigma$.
הערה: הנחנו כאן ש-Σ זהה בשתי השפות L_1, L_2. אפשר להכליל למקרה שאינו זהה.

צריך להוכיח כי $L(A) = L_1 \cap L_2$. נסתכל על ריצה r של A על קלט w. $r = r_0, ..., r_n$ וקלט-$w' = w_1, ..., w_n$ כך שלכל i, $w_i \in \Sigma$ ו-$r_i \in Q$. אז מהגדרה $r_0 = q_0 = (q_0^{(1)}, q_0^{(2)})$. נסמן: $r = (r_0^{(1)}, r_0^{(2)}), ..., (r_n^{(1)}, r_n^{(2)})$, כאשר לכל i $r_i^{(1)} \in Q_1$ וגם $r_i^{(2)} \in Q_2$.
נפריד את r לשתי סדרות. $r^{(1)} = r_0^{(1)}, ..., r_n^{(1)}$ ו- $r^{(2)} = r_0^{(2)}, ..., r_n^{(2)}$. לפי נכונות הסדרה r, ידוע שלכל i מתקיים: $\delta(r_i, w_{i+1}) = r_{i+1}$, כלומר $\delta\left((r_i^{(1)}, r_i^{(2)}), w_{i+1}\right) = (r_{i+1}^{(1)}, r_{i+1}^{(2)})$.
על כן לפי הגדרת δ מתקיים: $\delta_1(r_i^{(1)}, w_{i+1}) = r_{i+1}^{(1)}$ וגם $\delta_2(r_i^{(2)}, w_{i+1}) = r_{i+1}^{(2)}$. וכן $r_0^{(1)} = q_0^{(1)}$ וכן $r_0^{(2)} = q_0^{(2)}$. כלומר:
r_1 היא ריצה חוקית של A_1 על w ו-r_2 היא ריצה חוקית של A_2 על w.
על כן: $r_n = (r_n^{(1)}, r_n^{(2)}) \in F$ אמ"ם (לפי הגדרת F), $r_n^{(1)} \in F_1$ וגם $r_n^{(2)} \in F_2$ אמ"ם (לפי הגדרת A_1, A_2) $w \in L(A_1)$ וגם $w \in L(A_2)$.
כלומר A מקבלת את w אמ"ם $w \in L_1 \cap L_2$ כנדרש. $\square$

2 עודד - שיעור 2 - 12.05.2024

2.1 המחלקה REG

ראינו ש-REG סגורה לחיתוך ע"י אוטומט המכפלה. ניתן לשנות בקלות את ההוכחה כך שתראה סגירות REG לאיחוד. אזי יש לנו שני דרכים להראות ששפה היא ב-REG - או להראות DFA ספציפי, או להראות ששפה היא חיתוך או איחוד של שפות ב-REG. כעת נרצה להראות שיטות להוכחה ששפה לא ב-REG.

דוגמה. נגדיר את השפה $L=\{0^n1^n : n\in\mathbb{N}\cup\{0\}\}$. שפה זו אינה רגולרית (לעומת למשל $L'=\{0^n1^m : n,m\in\mathbb{N}\cup\{0\}\}$).
האינטואיציה היא שצריך "זיכרון"=מספר מצבים אינסופי.
נניח בשלילה ש-$L\in REG$, כלומר קיים DFA A, שמקבל את L. במילים אחרות $L(A)=L$. נסמן ב-$|Q|$ את מספר המצבים באוטומט A. נסתכל על כל הרצפים של אפסים באורך 0 עד $|Q|$. כלומר $\varepsilon,0,00,000,...$. נסתכל לאיזה מצב המחרוזות הללו מגיעות. על פי עקרון שובך היונים, יש לפחות 2 מחרוזות מהנ"ל שמגיעות לאותו המצב ב-A. בה"כ, נקרא להן $0^i,0^j$ כאשר $i<j$ וגם $i,j\in\mathbb{N}\cup\{0\}$. לפי תכונת האוטומט $L(A)=L$, ידוע ש-0^i1^i שייך לשפה ולכן ריצת A על המילה 0^i1^i מסתיימת במצב $q\in F$. מכאן שגם המילה 0^j1^i מסתיימת במצב $q\in F$, כי בשתי המילים אחרי רצף האפסים מגיעים לאותו המצב, ואחריו מגיע רצף של i אחדות ולכן בשני המקרים מגיעים לאותו המצב. קיבלנו ש-$0^j1^i\in L(A)$ אבל $i\neq j$, כלומר $0^j1^i\notin L$. סתירה.

נרצה להכליל את הרעיון שמאחורי שיטת ההוכחה הנ"ל. יש שתי הכללות:

1. למת הניפוח לשפות רגולריות.
2. משפט מייהיל נרוד. (הסמסטר מקוצר אז לא נראה את שיטה זו).

טענה. **(למת הניפוח לשפות רגולריות)** אם L רגולרית, אז קיים קבוע $p\in\mathbb{N}\cup\{0\}$ כך שלכל $w\in L$, המקיים $|w|\geq p$ מתקיים:
קיימת חלוקה של w: $w=xyz$ כך ש-

1. $x,y,z\in\Sigma^*$
2. $|y|\geq 1$
3. $|xy|\leq p$
4. לכל $i\in\mathbb{N}\cup\{0\}$, מתקיים $xy^iz\in L$

(נשים לב כי למשל $xz\in L$, $xyyyz\in L$)
(ה- p המינימלי הוא תכונה של השפה)

דוגמה. יישום הטענה על שפה $L'\in REG$. נגדיר $L'=\{0^n1^m : n,m\in\mathbb{N}\cup\{0\}\}$. הקבוע $p=1$ מקיים את התנאים. אם $w=0...01...1$ נבחר את $x=\varepsilon$ ו-y האות הראשונה. ואז גם אם $y=0$ וגם אם $y=1$ המחרוזת המנופחת נשארת בשפה.

השימוש הנפוץ בלמת הניפוח יהיה כדי להראות ששפה כלשהי אינה ב-REG.

דוגמה. נוכיח כי $L=\{0^n1^n : n\in\mathbb{N}\cup\{0\}\}\notin REG$. נניח בשלילה כי $L\in REG$. אז קיים p קבוע שלם עבורו מתקיימת למת הניפוח לשפות רגולריות עבור L. נסתכל על המילה $w=0^p1^p$. מתקיים $|w|>p$ ולכן מתקיימת למת הניפוח. כלומר קיימת חלוקה $w=xyz$, $x,y,z\in\{0,1\}^*$, $|y|\geq 1$, $|xy|\leq p$ ולכל $i\in\mathbb{N}\cup\{0\}$, $xy^iz\in L$. לפי בחירת w ולפי העובדה כי $|xy|\leq p$, ידוע ש-$x,y\in\{0\}^*$. נסמן $|y|=k$. על פי למת הניפוח $k\geq 1$. על פי למת הניפוח $xz\in L$ אבל $xz=0^{p-k}1^p$. כלומר $xz=xy^0z\notin L$, בסתירה (להנחה ש-L רגולרית).

הערה. עבור מילים w שקצרות מ-p, למת הניפוח לא דורשת שנוכל לנפח אותן.

הערה. שפות סופיות תמיד רגולריות. למת הניפוח תתקיים עבורן באופן "לא מספק" - p יהיה גדול יותר מאורך המילה הארוכה ביותר בשפה. (כי אחרת אפשר היה לייצר אינסוף מילים בזכות הלמה).

הוכחה. **(הוכחת למת הניפוח לשפות רגולריות).**
אם L שפה רגולרית, אז יש DFA A, שמקבל אותה. כלומר $L(A)=L$. נסמן ב-$|Q|$ את מספר מצבי האוטומט. נסמן $p=|Q|$. נסתכל על מילה $w\in L$ כך ש-$p\leq n=|w|$. נסתכל על ריצה של A על w, $r=r_0,r_1,...,r_n$ ומתקיים $r_0=q_0$ המצב ההתחלתי, ולכל $i\geq 1$ $\delta(r_{i-1},w_i)=r_i$ כאשר $w=w_1...w_n$ וכן $r_n\in F$.

על פי עקרון שובך היונים, קיימים $i < j$ כך ש-$r_i = r_j$. (יש יותר מצבים בריצה מאשר $|Q|$). נתבונן בריצה:

$$\underbrace{r_0 r_1 ...}_{x}\underbrace{r_i ... r_j}_{y}\underbrace{...r_n}_{z}$$

כלומר נסמן $x = w_1, ..., w_i$,$y = w_{i+1}, ..., w_j$,$z = w_{j+1}, ..., w_n$.

טענת עזר: $|y| \geq 1$. הוכחה: $i < j$.

טענת עזר: $|xy| \leq p$. הוכחה: יש מצב שחוזר פעמיים בין $|Q| + 1$ המצבים הראשונים בריצה, כלומר עד r_j עבור $j = |Q|$ או עבור j קטן יותר. מסקנה: $|xy| \leq p$.

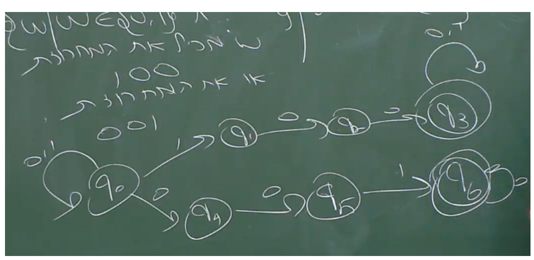
כיוון ש-$r_i = r_j$, גם הריצה $r_0 ... r_i r_{j+1} ... r_n$ היא ריצה חוקית ומקבלת ($r_n \in F$) וזו ריצה על הקלט xz כלומר $xz \in L$. באופן דומה, הריצה $r_0 ... r_i (r_{i+1} ... r_j)^k r_{j+1} ... r_n$, כלומר ריצה שבה חוזרים k פעמים על $(r_{i+1} ... r_j)$, היא ריצה חוקית ומקבלת. כלומר $xy^k z \in L$ לכל $k \geq 1$ שלם. □

2.2 אוטומטים סופיים לא דטרמיניסטים

הגדרה. אוטומט סופי לא דטרמיניסטי\Non-Determinstic Finite Automata\NFA הוא חמישייה $A = \langle Q, \Sigma, \delta, Q_0, F \rangle$ כאשר:

1. Q קבוצה סופית של מצבים
2. Σ א"ב סופי של הקלט.
3. Q_0 קבוצת מצבים התחלתיים.
4. F קבוצת מצבים מקבלים.
5. $\delta : Q \times \Sigma \cup \{\varepsilon\} \to 2^Q$

דוגמה. $L = \{w : w \in \{0,1\}^* \wedge w \ contains \ 100 \ or \ 001\}$.

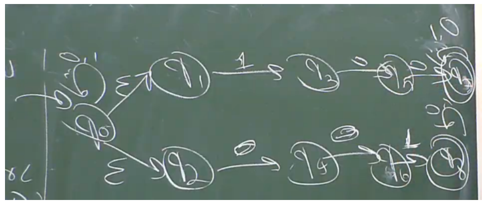


איור 4:

נשים לב שעל אותו קלט יש ריצות שונות. $\delta(q_0, 0) = \{q_0, q_4\}$.

הערה. כאשר חסר חץ יוצא עם אות כלשהי, המוסכמה היא שיש חץ כזה אל מצב שהוא **בור דוחה** (מצב עם לולאה ועליה כל Σ והמצב לא ב-F).

דוגמה. ε בקלט של δ זה צעד שלא "מבזבז" אות בקלט.



איור 5:

הערה. פורמלית δ יכולה, עבור זוג מסוים של מצב ואות להביא לקבוצה הריקה. אפשר לומר שבמקרה כזה אם עושים את הצעד הנ"ל הריצה לא מקבלת, או לאסור מצב כזה, כלומר לדרוש ש-δ מחזירה לפחות מצב אחד.

הגדרה. **ריצה של <u>NFA</u>**, A על קלט $w = w_1 ... w_n$, היא סדרה $r = r_0 r_1 ... r_t$ כאשר קיימת מחרוזת $w' = w'_1 ... w'_t$ כאשר w תת סדרה של w' וכל שאר הסימנים ב-w' הם ε. בנוסף $r_0 \in Q_0$, וגם לכל $i \geq 1$, $r_i \in \delta(r_{i-1}, w'_i)$.
אם $r_t \in F$ נגיד שהריצה **מקבלת.**

הערה. לכל קלט w ייתכן שיש יותר מריצה אחת.

הגדרה. אם A הוא NFA, נגיד ש-$w \in L(A)$ אם קיימת ריצה מקבלת של A על w.

הגדרה. REG^{NFA} תהיה מחלקת כל השפות L שיש עבורן NFA, A כך ש-$L(A) = L$.

הערה. ברור ש-$REG \subseteq REG^{NFA}$ כי כל DFA הוא גם NFA, כלומר NFA שבמקרה אין לו מעברי ε, $|Q_0| = 1$ ולכל $q \in Q$ ו-$\sigma \in \Sigma$ מתקיים $|\delta(q, \sigma)| = 1$.

הגדרה. נוסיף הגדרה שמכלילה את הגדרת δ לאוטומטים DFA וNFA כך שניתן יהיה להפעיל אותה גם על מחרוזות מאורך שונה מ-1.
עבור NFA נגדיר לכל $w \in \Sigma^*$:

$$\delta^*(Q_0, w) = \begin{cases} Q_0 & w = \varepsilon \\ \bigcup_{s \in \delta^*(Q_0, u)} \delta(s, \sigma) & w = \sigma u, s.t.\ u \in \Sigma^*, \sigma \in \Sigma \end{cases}$$

עבור DFA נגדיר לכל $w \in \Sigma^*$:

$$\delta^*(q_0, w) = \begin{cases} q_0 & w = \varepsilon \\ \delta(\delta^*(q_0, u), \sigma) & w = \sigma u, s.t.\ u \in \Sigma^*, \sigma \in \Sigma \end{cases}$$

טענה. $REG = REG^{NFA}$.

הוכחה. נראה ש-$REG = REG^{NFA}$ כלומר לכל NFA, A יש $DFA\ A'$ כך ש- $L(A) = L(A')$. (ראינו בהערה הקודמת הכלה בכיוון השני).

נניח עבור ההוכחה כי אין ב-A מעברי ε (יש דרך לקבל NFA שקול בלי מעברי ε, נעשה את זה בתרגול אולי).

יהי $A=\langle Q,\Sigma,Q_0,\delta,F\rangle$, NFA. נגדיר DFA $A'=\langle Q',\Sigma,q_0',\delta',F'\rangle$ באופן הבא:

$$\begin{aligned}Q'&=2^Q\\ q_0'&=Q_0\\ F'&=\{q'\in Q' : q'\cap F\neq\emptyset\}\\ \forall q'\in Q',\forall\sigma\in\Sigma\ \delta'(q',\sigma)&=\bigcup_{s\in q'}\delta(s,\sigma)\end{aligned}$$

סיימנו עם הבנייה (קוראים לה Subset Construction). נותר להוכיח נכונות. כלומר ש-$L(A)=L(A')$. במילים אחרות $w\in L(A)\iff w\in L(A')$. במילים אחרות, יש ריצה מקבלת של A על w אמ"ם הריצה של A' על w היא מקבלת.

טענת עזר: לכל $w\in\Sigma^*$, $\delta'^*(q_0',w)=\delta^*(Q_0,w)$. למה הטענה הזו מספיקה? כי $w\in L(A)$ אמ"ם $\delta^*(Q_0,w)\cap F\neq\emptyset$, אמ"ם (לפי הטענה) $\delta'^*(q_0',w)\cap F\neq\emptyset$ אמ"ם (לפי הגדרת F') $\delta'^*(q_0,q)\in F'$ אמ"ם $w\in L(A')$.

הוכחת טענת העזר: נוכיח את הטענה באינדוקציה על אורך הקלט $|w|$.

עבור $|w|=0$ כלומר $w=\varepsilon$, לפי ההגדרה:

$$\delta'^*(q_0',w)=\delta'^*(q_0',\varepsilon)=q_0'=Q_0=\delta^*(Q_0,\varepsilon)=\delta^*(Q_0,w)$$

כעת נניח שהטענה נכונה עבור כל w כך ש-$|w|\leq n$ ונוכיח עבור $|w|=n+1$.

$$\begin{aligned}\delta^*(Q_0,w)&=\delta^*(Q_0,\sigma u)\\ &=\bigcup_{s\in\delta^*(Q_0,u)}\delta(s,\sigma)\\ &\overset{\spadesuit}{=}\delta'^*(\delta^*(Q_0,u),\sigma)\\ &\overset{\heartsuit}{=}\delta'^*(\delta'^*(q_0',u),\sigma)\\ &=\delta'^*(q_0',\sigma u)\\ &=\delta'^*(q_0',w)\end{aligned}$$

כאשר $\heartsuit$ נובע מהנחת האינדוקציה. $\spadesuit$ מהגדרת δ'. $\square$

הערה. בהמשך הקורס נראה שבמודלים אחרים השאלה האם אי דטרמיניזם מוסיף כוח חישובי היא בדר"כ אינה פתורה.

הערה. מעתה כדי להראות ששפה $L\in REG$ אפשר גם להראות $L\in REG^{NFA}$.

הערה. הניפוח האקספוננציאלי במספר המצבים בבניית Subset Construction הוא הכרחי. כלומר יש דוגמאות שבהן ה-DFA הקטן ביותר הוא גדול אקספוננציאלית מה-NFA הקטן ביותר.

3 עודד - שיעור 3 - 19.05.2024

3.1 חלק 2 של הקורס - חישוביות

3.1.1 הגדרות מכונת טיורינג

נעבוד עם מודל מכונת טיורינג שפותח על ידי אלן טיורינג בשנת 1936.

הגדרה. **מכונת טיורינג** היא שביעייה $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} \rangle$ כאשר :

1. Q קבוצה סופית של מצבים פנימיים.
2. Σ א"ב של השפה (סופי).
3. Γ א"ב עבודה ומקיים $\sqcup \in \Gamma$, $\Sigma \subseteq \Gamma$.
4. $q_0 \in Q$ מצב התחלתי.
5. $q_{accept} \in Q$ מצב מסיים מקבל.
6. $q_{reject} \in Q$ מצב מסיים דוחה.
7. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$.

תיאור ציורי ומילולי :

סרט אינסופי לכיוון ימין. בכל תא יש אות מתוך Γ. הקלט (שהוא מילה) בתחילת הריצה מופיע בתחילת הסרט (כל אות תופסת תא) ואחריו הסימן $\sqcup$ עד אינסוף. (זה במובן מסוים הזיכרון של המכונה).

למכונה ראש קורא, שנמצא בתחילת הריצה בתא הראשון (השמאלי ביותר). בכל צעד, הראש קורא את האות בתא הנוכחי, ולפי האות והמצב הפנימי, המכונה פועלת.

כלומר הראש כותב בתא אות מתוך Γ, המכונה עוברת למצב מתוך Q, והראש זז ימינה או שמאלה, תא אחד.

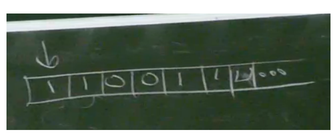
המצב בתחילת הריצה הוא q_0. המכונה ממשיכה לרוץ עד שמגיעה למצב q_{accept} או q_{reject} ואז היא עוצרת. אם המכונה הגיעה למצב q_{accept}, נגיד שהיא **קיבלה** את הקלט, ואם המכונה הגיעה למצב q_{reject}, נגיד שהיא **דחתה** את הקלט.

הערה. נשים לב שעבור מכונת טיורינג M, וקלט $w \in \Sigma^*$, יתכן ש-M תעצור ותחזיר q_{accept} או q_{reject}, **אבל** ייתכן גם ש-M לא תעצור ותמשיך לרוץ עד אינסוף.

דוגמה. נגדיר את השפה Pal :

$$Pal = \{uu^{rev} : u \in \Sigma^* \wedge u^{rev} \ is \ mirror \ of \ u \wedge \Sigma = \{0,1\}\}$$

למשל $0110 \in Pal$ אך $110 \notin Pal$. במילים אחרות, Pal היא שפת כל הפלינדרומים מאורך זוגי מעל $\{0,1\}$. נרצה לתאר מ"ט M כך שלכל קלט $w \in \{0,1\}^*$ מחזירה q_{accept} אם $w \in Pal$, ו-q_{reject} אם $w \notin Pal$.

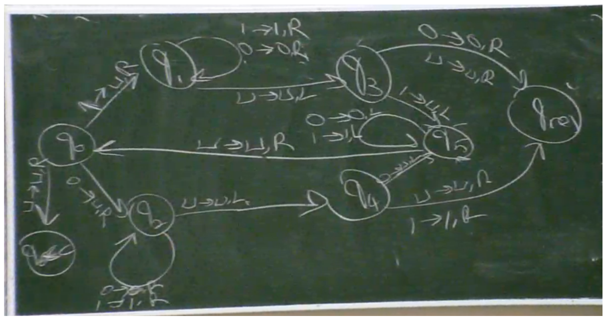


איור 6 :

נתאר במילים חופשיות את אופן פעולת M (לא פורמלי) : M תקרא את האות הראשונה. תזכור אותה על ידי מצב פנימי (אחד שמתאים לאחד ושני שמתאים לאפס). תמחק את האות הראשונה, ותרוץ עם הראש ימינה כל הזמן מבלי לשנות את מה שכתוב על הסרט. כשהראש יגיע לקצה

הקלט (התא הראשון עם $\sqcup$) - יחזור אחד אחורה ויבדוק האם האות הנוכחית מתאימה למצב הפנימי. אם לא, המכונה תעצור ותחזיר q_{reject}. אם כן, המכונה תמחק את האות הזו, ותעבור למצב שמחזיר את הראש שמאלה עד הסוף, מבלי לשנות מה שכתוב על הסרט. כשהראש יגיע ל-$\sqcup$ הראשון שיתקל בו - יחזור תא אחד ימינה, ונחזור על התהליך מהתחלה. אם כל המילה נמחקה, M תעצור ותחזיר q_{accept}. אם לא, היא תחזור על אותה פעולה על האות השנייה וכו.

תיאור פורמלי של δ: כדי להגדיר את M צריך בפרט להגדיר את δ. דרך אחת היא ע"י טבלה, אבל זה פחות ישים וקריא. דרך אחרת היא על ידי ייצוג גרפי, בדומה לתיאור DFA. יהיה קודקוד\צומת לכל מצב פנימי, וקשת לכל מעבר של δ. על הקשת נסמן מידע נוסף:


איור 7:

הערה: אם נרצה ש-M תחזיר q_{accept} גם על פלינדרומים מאורך אי זוגי, אז נעשה את השינוי הבא: מ-q_3 ומ-q_4, אם רואים סימן ריק אז עוברים ל-q_{accept}.

הגדרה. נגיד שמכונת טיורינג M **מכריעה** שפה L אם לכל קלט $w \in \Sigma^*$, ריצת M על w תסתיים ב-q_{accept} או q_{reject}, וכן: $w \in L$ אמ"ם ריצת M על w מסתיימת ב-q_{accept}.

הגדרה. נגיד שמכונה טיורינג M **מקבלת** שפה L אם לכל קלט $w \in L$, ריצת M על w מסתיימת ב-q_{accept}, ולכל $w \notin L$, ריצת M על w מסתיימת ב-q_{reject} או לא מסתיימת.

נסמן ב-$L(M)$ את השפה ש-M מקבלת.

נסמן $M(w) = q_{accept}$ אם ריצת M על w מסתיימת ב-q_{accept}.

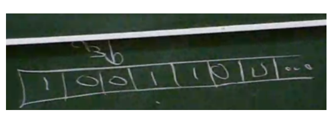
נסמן $M(w) = q_{reject}$ אם ריצת M על w מסתיימת ב-q_{reject}.

נסמן $M(w) = \perp$ אם ריצת M על w לא מסתיימת.

נרצה להגדיר באופן פורמלי ריצה של מכונה על קלט.

הגדרה. **קונפיגורציה של מכונה** מתארת את המצב הפנימי, מיקום הראש, ותוכן הסרט. באופן פורמלי, קונפיגורציה היא מחרוזת מהצורה uqv, כאשר $q \in Q$, $u, v \in \Gamma^*$, והיא מתארת מכונה שבה המצב הוא q, תוכן הסרט הוא uv והראש נמצא על האות הראשונה ב-v.

דוגמה. עבור המכונה הבאה, $u = 10, q = q_3, v = 0110$. במילים אחרות, הקונפיגורציה היא $10q_30110$


איור 8:

הגדרה. נתאר כמה קונפיגורציות מיוחדות:

- **קונפיגורציה ההתחלתית** - כאשר הראש בתחילת הסרט, המצב q_0, תוכן הסרט הוא מילת הקלט $w \in \Sigma^*$. כלומר הקונפיגורציה בתור מחרוזת היא $q_0 w$.
- **קונפיגורציה מסיימת** - היא כל קונפיגורציה שיש בה q_{accept} או q_{reject}.
- **קונפיגורציה מקבלת** - כל קונפיגורציה שיש בה q_{accept}.
- **קונפיגורציה דוחה** - כל קונפיגורציה שיש בה q_{reject}.

הגדרה. זוג קונפיגורציות c_1, c_2 יקראו **קונפיגורציות עוקבות** אם הקונפ' c_2 מתקבלת על ידי צעד אחד של δ שמופעל על c_1.

הערה. c_1, c_2 זהות כמעט בכל מקום, למעט בסביבת המקום של הראש, ושם השינוי לפי הפונקציה δ.

הגדרה. **ריצה** של מכונה M על קלט w, היא סדרה (אולי אינסופית) של קונפיגורציות: $r = c_0 c_1 c_2 \ldots$ כאשר c_0 היא הקונפיגורציה ההתחלתית $q_0 w$, ולכל i מתקיים ש-c_i, c_{i+1} הן קונפיגורציות עוקבות.
אם הסדרה היא סופית, אז הקונפיגורציה האחרונה c_t היא קונפיגורציה מסיימת.
הריצה מקבלת אמ"ם הקונפיגורציה האחרונה היא קונפיגורציה מקבלת. (תזכורת: מכילה את q_{accept}).
הריצה דוחה אמ"ם הקונפיגורציה האחרונה היא קונפיגורציה דוחה. (תזכורת: מכילה את q_{reject}).

הגדרה. $\underline{R}$ היא מחלקת כל השפות שניתנות להכרעה. כלומר כל השפות L כך שקיימת מכונת טיורינג M שמכריעה את L.
$\underline{RE}$ היא מחלקת כל השפות שניתנות לקיבול. כלומר כל השפות L כך שקיימת מכונת טיורינג M שמקבלת את L.
$\underline{coRE}$ היא מחלקת כל השפות L כך ש$\overline{L} \in RE$.

3.1.2 טענות על מחלקות טיורינג

טענה. R סגורה למשלים.
כלומר אם $L \in R$ אז גם $\overline{L} \in R$.
כלומר אם קיימת מ"ט M שמכריעה את L, אז קיימת מ"ט M' שמכריעה את $\overline{L}$.

הוכחה. בהינתן M נחליף בין q_{accept} ו-q_{reject} ונקבל את M'. נשים לב ש-M מכונת הכרעה ולכן מובטח שהיא עוצרת על כל קלט. מכאן שגם M' עוצרת על כל קלט. התשובות שלהן על כל קלט הפוכות, ולכן M' מכריעה את $\overline{L}$. □

טענה. $R \subseteq RE$.
כלומר עבור שפה $L \in R$, ידוע שיש M שמכריעה את L. הטענה היא שיש גם M' שמקבלת את L.

הוכחה. נבחר את M' להיות M. נשים לב שלפי ההגדרה של מכונת הכרעה, M' היא בפרט גם מכונה שמקבלת את L כי על $w \in L$ היא מחזירה q_{accept} ועל $w \notin L$ היא מחזירה q_{reject}, ומותר לה להחזיר q_{reject} או לא לעצור. □

טענה. $R \subseteq coRE$.

הוכחה. אם $L \in R$, אז $\overline{L} \in R$. לכן $\overline{L} \in RE$. לכן $L \in coRE$. □

מסקנה. $R \subseteq RE \cap coRE$.

טענה. $R = RE \cap coRE$.

הוכחה. נראה ש- $RE \cap coRE \subseteq R$. (בכיוון השני כבר הראינו). תהי $L \in RE \cap coRE$. לפי $L \in RE$ קיימת מ"ט M_1 שמקבלת את L. לפי $L \in coRE$ קיימת מ"ט M_2 שמקבלת את $\overline{L}$.

נרצה לבנות מכונה M שמכריעה את L. זה יראה כי $L \in R$.

הפתרון השגוי: M תפעל כמו M_1 ואח"כ כמו M_2. אם M_1 מחזירה q_{accept}, ואם M_2 מחזירה q_{accept}, אז M מחזירה q_{reject}. הבעיה היא ש-M_1 עלולה שלא לעצור, ואז M לא תעצור (לא תגיע להרצת M_2).

נרצה פתרון ש"מריץ במקביל" את M_1 ואת M_2 ואז M תחזיר q_{accept} אם M_1 מחזירה q_{accept}, ו-M תחזיר q_{reject} אם M_2 מחזירה q_{accept}.

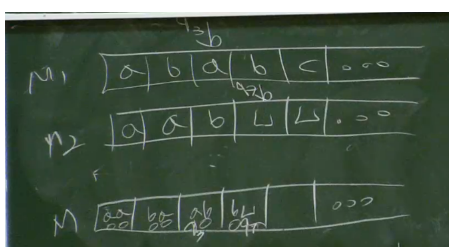
נשים לב שמובטח שלפחות אחת משתי המכונות M_1 ו-M_2 עוצרת ומחזירה q_{accept}. אם $w \in L$ אז ההבטחה היא על M_1, ואם $w \notin L$ אז ההבטחה היא על M_2. נותר רק להסביר איך אפשר להריץ במקביל 2 מכונות על אותו הקלט.

נראה בקורס שתי דרכים להרצה במקביל. 1. איחוד שתי מכונות (עם סרט לכל אחת) למכונת סרט יחיד שמדמה את שתיהן. 2. ריצה מדורגת וחסומה בזמן. נעשה את דרך 1.

נבנה M שמכריעה את L. נסמן:

$$M_1 = \langle Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0^{(1)}, q_{acc}^{(1)}, q_{rej}^{(1)} \rangle$$
$$M_2 = \langle Q_2, \Sigma_2, \Gamma_2, \delta_2, q_0^{(2)}, q_{acc}^{(2)}, q_{rej}^{(2)} \rangle$$
$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej} \rangle$$

נרצה לקודד על הסרט של M את התוכן הסרט של M_1, תוכן הסרט של M_2, מיקום הראשים של M_1, M_2, ואת המצב הפנימי של המכונות M_1, M_2. לכן, כל אות ב-Γ תייצג את המידע הנדרש. כלומר $\Gamma = \Gamma_1 \times \Gamma_2 \times (Q_1 \cup \{0\}) \times (Q_2 \cup \{0\})$. כלומר אות בתא במקום ה-$k$ בסרט של M מייצגת אות מ-Γ_1 בתא ה-k של M_1, אות מ-Γ_2 בתא ה-k של M_2, האם הראש של M_1 במקום k ואם כן מה המצב הפנימי של M_1, האם הראש של M_2 במקום k ואם כן מה המצב הפנימי של M_2. כמתואר בציור הבא:



איור 9:

בתחילת הריצה, הסרט יכיל את הקידוד של הקונפיגורציה ההתחלתית של M_1 ושל M_2, כלומר שני הראשים בתא הראשון, המצב הפנימי של שתי המכונות הוא המצב ההתחלתי, ותוכן הסרט הוא w בשתי המכונות. כלומר אם $w = w_1 ... w_n$ אז הסרט יכיל את:

$$w_1 w_1 q_0^{(1)} q_0^{(2)}, w_2 w_2 00, ..., w_n w_n 00, \sqcup, ...$$

M תדמה את צעדי הריצה של M_1 ו-M_2, צעד אחד בכל פעם, לסירוגין. כדי לדמות צעד ריצה, הראש (האמיתי) של M יסרוק את הסרט ויחפש תא שבו מסומן קיום של הראש של M_1. ואז M תבצע את השינויים (המקומיים) המאימים לצעד של M_1, כלומר לפי δ_1, כלומר בהתאם למצב

של M_1 (רשום באות) ובהתאם לאות הנוכחית של M_1 מתוך Γ_1 (גם רשום באות), צריך לבצע כתיבה של M_1, הזזת ראש, ועדכון מצב. בפועל זה אומר לעדכן את האות הנוכחית וכן את האות הסמוכה מימין או השמאל (לפי האם δ_1 אומרת שצריך להזיז את הראש ימינה או שמאלה).
בסוף העדכון הנ"ל, M תחזיר את הראש (האמיתי) לתחילת הסרט, ותחזור על הנ"ל עבור M_2. וחוזר חלילה, ל-M_1 ו-M_2 לסירוגין.
אם בשלב כלשהו M_1 או M_2 עוברות למצב מסיים, אז M עוצרת ומחזירה כמו M_1 אם M_1 סיימה או ההפך מ-M_2 אם M_2 סיימה.
בעיה: המכונה M פועלת נכון אבל בשביל זה צריכה לקבל קלט שהוא לא w אלא מחרוזת אחרת שתלויה ב-w. צריך להוסיף מעבר של M שמחליף את w בקידוד המתאים לקונפיגורציה ההתחלתית.
הערה: סימולציה של צעד בודד של M_1 דורשת הרבה צעדי ריצה של M - עבור סריקה וחיפוש של סימון ראש M_1, עדכון תא נוכחי, עדכון תא לפני או אחרי (לפי האם ההזזה היא ימינה או שמאלה), ולבסוף חזרה של הראש (האמיתי) לתחילת הסרט. ובאופן דומה עבור דימוי צעד בודד של M_2.
הערה: $\sqcup$ מייצג את $00 \sqcup \sqcup$. $\square$

הערה. בכל מכונת טיורינג אם הראש בתא הראשון ויש פקודה שמאלה אז הראש נשאר במקום. לחילופין, אולי נוח יותר שמשמאל לתא הראשון יהיה תא נוסף עם $\sqcup$ שכשמגיעים אליו תמיד משאירים תו ריק ומזיקים את הראש אחד ימינה.

4 עודד - שיעור 4 - 26.05.2024

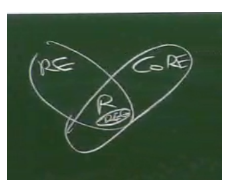
4.1 השפות A_{TM}, A_{DFA}, $Halt_{TM}$

נתחיל בטענה לחימום.

טענה. $REG \subset R$, $REG \neq R$.
כלומר כל השפות שהן שפות של DFA ניתן להכריע באמצעות מכונת טיורינג.

הוכחה. הכלה - אפשר לממש DFA ע"י מכונת טיורינג. הראש של המכונה יזוז רק ימינה. פרטים - תרגיל.
הכלה ממש - נראה שפה $L \in R \setminus REG$. למשל השפה $L = \{a^n b^n : n \in \mathbb{N}\}$. הראינו בעזרת למת הניפוח ש-$L \notin REG$ וקל לבנות מכונת טיורינג M שתכריע אותה. $\square$

מצב העניינים כעת:



איור 10:

נרצה להראות היום שפות שנמצאות למשל ב-$RE \setminus R$ וב-$coRE \setminus R$. נסתכל על השפות הבאות:

הגדרה. שפות מעניינות

$$A_{TM} = \{(\langle M \rangle, w) : \text{machine turing a is M}, w \in \Sigma^*, w \in L(M)\}$$
$$A_{DFA} = \{(\langle A \rangle, w) : \text{DFA a is M}, w \in \Sigma^*, w \in L(M)\}$$

הסימון $\langle M \rangle$ מייצג קידוד של מכונת טיורינג על ידי מחרוזות מעל Σ. הסימון $\langle A \rangle$ מייצג קידוד של DFA.

האם $A_{DFA} \in R$? כן. כדי להוכיח, נצטרך להראות שיש מכונת טיורינג M שמכריעה את A_{DFA}. כלומר, מכונת טיורינג M שפועלת באופן הבא:

$$M(\langle A\rangle, w) = \begin{cases} q_{acc} & if\ w \in L(A) \\ q_{rej} & if\ w \notin L(A) \end{cases}$$

המכונה M תריץ את A על w, צעד אחרי צעד. כלומר בכל שלב תקרא את האות הבאה מ-w, ולפי המצב הנוכחי של A שישמר על הסרט המכונה תבדוק בייצוג של δ לאיזה מצב צריך לשנות אותו. בסוף המעבר על w, M תבדוק האם המצב ששמרה על הסרט הוא אחד מהמצבים ב-F כפי שמופיע בקידוד של A.

מה לגבי A_{TM}? האם $A_{TM} \in R$, האם $A_{TM} \in RE$? כלומר מכונת טיורינג M' שמכריעה (או מקבלת) את A_{TM} צריכה לקבל קלט $(\langle M\rangle, w)$ ולפעול באופן הבא:

$$M'(\langle M\rangle, w) = \begin{cases} q_{acc} & if\ w \in L(M) \\ q_{rej} & if\ w \notin L(M) \end{cases}$$

זו מכונת הכרעה ל-A_{TM}. ואילו מכונה מקבלת ל-A_{TM} פועלת כך:

$$M'(\langle M\rangle, w) = \begin{cases} q_{acc} & if\ w \in L(M) \\ q_{rej}\ or\ \perp \text{(stop no)} & if\ w \notin L(M) \end{cases}$$

טענה. $A_{TM} \in RE$.
כלומר קיימת מ"ט M' שמקבלת את A_{TM}.

הוכחה. הרעיון: נרצה מכונת טיורינג שמדמה את ריצת M על w צעד אחרי צעד ועונה כמוה ככל ש-M על w עוצרת. כלומר, אם $M(w) = q_{acc}$ אז $M'(\langle M\rangle, w) = q_{acc}$. אם $M(w) = q_{rej}$ אז $M'(\langle M\rangle, w) = q_{rej}$ ואם $\perp$, אז $M'(\langle M\rangle, w) = \perp$. למכונה הזו קוראים גם U, **המכונה האוניברסלית**. נראה (אולי) את הפרטים שלה בתרגול. □

נשאלת השאלה האם גם $A_{TM} \in R$?

טענה. $A_{TM} \notin R$. **(משפט הלכסון)**
זו אחת התוצאות המרכזיות של טיורינג.

הוכחה. ההוכחה תתבצע באמצעות "טכניקת הלכסון".
נניח בשלילה שקיימת מ"ט M' שמכריעה את A_{TM}. נסמן ב-D מכונת טיורינג שפועלת באופן הבא:

$$D(\langle M\rangle) = M'(\langle M\rangle, \langle M\rangle)$$

הערה: בהינתן M' אפשר לבנות את המכונה D.

נסמן ב-$\overline{D}$ את המכונה שפועלת באופן הבא:

$$\overline{D}(\langle M\rangle)=\overline{D(\langle M\rangle)}$$

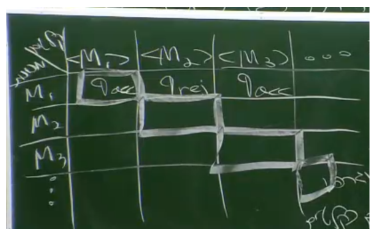
בהינתן D אפשר לבנות את $\overline{D}$ ע"י החלפת q_{acc} ו-q_{rej}.

שאלה: מה תחזיר $\overline{D}(\langle\overline{D}\rangle)$? נבחן את האפשרויות. אם $\overline{D}(\langle\overline{D}\rangle)=q_{acc}$ אז $D(\langle\overline{D}\rangle)=q_{rej}$ אז $M'(\langle\overline{D}\rangle,\langle\overline{D}\rangle)=q_{rej}$ כלומר $\overline{D}\notin L(\overline{D})$ ואז לא ייתכן ש-$\overline{D}(\langle\overline{D}\rangle)=q_{acc}$. הגענו לסתירה, אז אולי ההנחה ש- $\overline{D}(\langle\overline{D}\rangle)=q_{acc}$ הייתה לא טובה.

אם כך, אז אולי $\overline{D}(\langle\overline{D}\rangle)=q_{rej}$. אז $D(\langle\overline{D}\rangle)=q_{acc}$ כלומר $M'(\langle\overline{D}\rangle,\langle\overline{D}\rangle)=q_{acc}$ כלומר $\overline{D}\in L(\overline{D})$ ועל כן $\overline{D}(\langle\overline{D}\rangle)=q_{acc}$. בסתירה!

מסקנה: M' לא קיימת! על כן $A_{TM}\notin R$. □

הוכחה. תיאור ההוכחה באופן ויזואלי שמסביר מדוע קוראים לזה טכניקת הלכסון:



איור 11:

הטבלה מכילה את כל מכונות ההכרעה בסדר לקסיקוגרפי עולה. (גם כמכונות (שורות) וגם כקלטים (עמודות)). בכל תא בטבלה, נרשום מה עונה המכונה של השורה בהינתן הקלט של העמודה. במילים אחרות, הטבלה מתארת את המענה של M', מכונת ההכרעה (הלא קיימת) של A_{TM}.

הערה: יש קלטים עבור M' שהם לא מהצורה הנכונה והם כלל לא מופיעים בטבלה.

האלכסון בטבלה מתאים למכונה D. ואם נהפוך את הערכים באלכסון, נקבל את הפלטים של המכונה $\overline{D}$.

תזכורת מהשבוע הראשון: נסתכל על האלכסון ההפוך. אם יש מכונת טיורינג (תכנית $C++$) שמחשבת בדיוק אותו, אז בהכרח היא מופיעה כשורה בטבלה (וגם כעמודה). שאלנו מה מחזירה $\overline{D}(\langle\overline{D}\rangle)$ כלומר מה קורה במפגש של השורה והעמודה שלה, והגענו לסתירה. לא משנה אם נניח שרשום במפגש q_{acc} או q_{rej}, נגיע לסתירה. □

נרצה לראות עוד דוגמאות לשפות ב-$RE\setminus R$ וכן לשפות ב-$coRE\setminus R$ וב-$\overline{RE\cup coRE}$. נראה שיטות להוכחה ששפות במחלקות אלה, מבלי להשתמש שוב בטכניקת הלכסון.

טענה. $\overline{A_{TM}}\in coRE\setminus R$.

הוכחה. ראינו $A_{TM}\in RE$ ולכן מהגדרה $\overline{A_{TM}}\in coRE$.

נניח בשלילה כי $\overline{A_{TM}}\in R$ אז גם $A_{TM}\in R$ כי R סגורה למשלים. לכן $\overline{A_{TM}}\notin R$. □

שאלה רנדומלית שמישהו שאל: למה מ"ט הן קבוצה בת מנייה? תשובה: כי כל מ"ט ניתנת לקידוד על ידי סדרה סופית של סימנים מתוך א"ב סופי.

הגדרה. שפה מעניינת

$$Halt_{TM} = \{(\langle M\rangle, w) : \text{machine turing a is M}, w \in \Sigma^*, M(w) \in \{q_{acc}, q_{rej}\}\}$$

כלומר השפה של מכונות ומילים כך ש- M עוצרת על w.

טענה. $Halt_{TM} \in RE$.
כלומר יש מ"ט M' שמקבלת את $Halt_{TM}$.

הוכחה. נבנה את M' באופן הבא: נשתמש ב-U המכונה האוניברסלית, אבל נשנה את U באופן הבא: אם U עוצרת אז היא מחזירה q_{acc}. (גם אם במקור הייתה אמורה להחזיר q_{rej}). ואז: אם $M(w)$ ומחזירה q_{acc} או q_{rej} אז M' תעצור ותחזיר q_{acc}, ואם $M(q) =\perp$ אז גם $U(\langle M\rangle, w) =\perp$ ולכן גם $M'(\langle M\rangle, w) =\perp$. □

טענה. $Halt_{TM} \notin R$.

הוכחה. נניח בשלילה כי $Halt_{TM} \in R$ כלומר יש מ"ט M' שמכריעה את $Halt_{TM}$. כלומר M' פועלת כך:

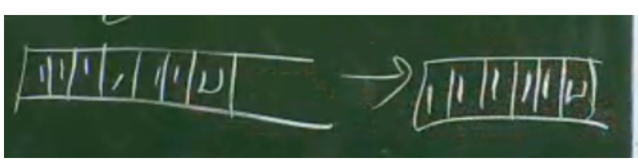
$$M'(\langle M\rangle, w) = \begin{cases} q_{acc} & if\ M(w) \in \{q_{acc}, q_{rej}\} \\ q_{rej} & if\ M(w) =\perp \end{cases}$$

נראה איך בעזרת M' ניתן לבנות מכונת טיורינג M'' שמכריעה את A_{Tm} וזו סתירה כי ראינו ש-$A_{TM} \notin R$. אז M'' תפעל כך: עבור זוג $(\langle M\rangle, w)$ ראשית תריץ עליו את M'. מובטח ש-M' עוצרת (כי היא מכונת הכרעה). אם M' מחזירה q_{rej} אז M'' גם תחזיר q_{rej}, ואם M' החזירה q_{acc} זה אומר ש-M עוצרת על w. לכן אפשר להריץ את U על הזוג $(\langle M\rangle, w)$ ולענות כמוה. קיבלנו מכונה M'' שמכריעה את A_{TM} בסתירה. □

4.2 רדוקציות, פונקציות חשיבות

הגדרה. נגיד שמכונת טיורינג M **מחשבת פונקציה** $f : \Sigma^* \to \Sigma^*$, אם לכל קלט $w \in \Sigma^*$, ריצת M מסתיימת ועל הסרט רשום $f(w)$.
אם לפונקציה f יש מכונת טיורינג שמחשבת אותה, נגידה ש-f היא **פונקציה חשיבה.**

דוגמה. $f(a, b) = a + b$ כאשר $a, b, a + b$ מספרים טבעיים בקידוד אונארי, כלומר $\Sigma = \{1, ","\}$



איור 12:

הגדרה. **(רדוקציה)** יהיו A, B שפות. נגיד שיש **רדוקציה** (נאו-רדוקציית מיפוי) מ-A ל-B (הכיוון חשוב) ונסמן $A \leq_m B$ אם קיימת פונקציה חשיבה f ומתקיים, לכל w: $w \in A \iff f(w) \in B$:

משפט. **(משפט הרדוקציה)** אם $A \leq_m B$ וגם $B \in R$, אז $A \in R$.

מסקנה. אם $A \leq_m B$ וגם $A \notin R$, אז $B \notin R$.

הוכחה. (של משפט הרדוקציה): $B \in R$ ולכן קיימת מ"ט M_B שמכריעה את B. $A \leq_m B$ לכן קיימת מ"ט M_f, שהיא רדוקציה מ-A ל-B, כלומר M_f על קלט w מחזירה $f(w)$ ומתקיים $f(w) \in B \iff w \in A$. נבנה מכונה בשם M_A באופן הבא:
עבור קלט w, M_A תריץ את M_f על הקלט w ואז תריץ את M_B על מה שכתוב על הסרט. נותר להוכיח ש-M_A מכריעה את A. עבור $w \in \Sigma^*$ נסמן ב-y את תוכן הסרט בתום ריצת M_f.
אם $w \in A$ אז $y = M_f(w) \in B$, (כי f רדוקציה מ-A ל-B) ולכן $M_B(y) = q_{acc}$ כי M_B מכונת הכרעה של B. כלומר M_A מחזירה q_{acc}.
אם $w \notin A$ אז $y = M_f(w) \notin B$ (כי f רדוקציה מ-A ל-B) ולכן $M_B(y) = q_{rej}$ כי M_B מכונת הכרעה של B. כלומר M_A מחזירה q_{rej}.
כלומר M_A מכונת הכרעה של A. □

4.2.1 שימוש במשפט הרדוקציה

הגדרה. שפה מעניינת:

$$Halt_\varepsilon = \{\langle M \rangle : M(\varepsilon) \in \{q_{acc}, q_{rej}\}\}$$

טענה. $Halt_\varepsilon \in RE \setminus R$.

הוכחה. שייכות ל-RE: נבנה מכונה מקבלת ל-$Halt_\varepsilon$, M' בעזרת U המכונה האוניברסלית. עבור קלט $\langle M \rangle$ המכונה M' תריץ את $U(\langle M \rangle, \varepsilon)$, ותחזיר q_{acc} אם U ענתה q_{acc} או q_{rej}.
נכונות: אם $M \in Halt_\varepsilon$, כלומר $M(\varepsilon)$ עוצרת (הגדרת השפה) אז U תחזיר q_{acc} או q_{rej} ולכן M' תחזיר q_{acc}. אם $M \notin Halt_\varepsilon$, כלומר $M(\varepsilon)$ לא עוצרת (הגדרת השפה) אז U לא תעצור ולכן גם M' לא תעצור. מסקנה: M' מקבלת את $Halt_\varepsilon$.
אי שייכות ל-R: נראה ש- $Halt_{TM} \leq_m Halt_\varepsilon$ וידוע ש-$Halt_{TM} \notin R$ ולכן ע"פ משפט הרדוקציה נסיק ש-$Halt_\varepsilon \notin R$.
כל שנותר הוא להראות רדוקציה $Halt_{TM} \leq_m Halt_\varepsilon$. כלומר להראות שקיימת פונקציה חשיבה f שמקיימת $(\langle M \rangle, w) \in Halt_{TM} \iff \langle M' \rangle \in Halt_\varepsilon$ כאשר $\langle M' \rangle = f(\langle M \rangle, w)$
עבור זוג $(\langle M \rangle, w)$, f תחזיר $\langle M' \rangle$ כאשר M' היא המכונה M שמקבעים לתוכה את w. כלומר M' מתעלמת מהקלט שלה ומריצה את M על w.
ברור ש-f חשיבה - קל לבנות מ"ט M'' שמקבלת קלט $(\langle M \rangle, w)$ ומחזיקה $\langle M' \rangle$ שהוא קידוד של מכונה שפועלת כמו M אבל מתעלמת מהקלט שלה ומריצה את $M(w)$. כעת נראה את נכונות הרדוקציה.
בכיוון אחד, אם $(\langle M \rangle, w) \in Halt_{TM}$, אז M עוצרת על w. לכן M' עוצרת על כל קלט (כי היא רצה כמו M על w), ובפרט על ε. לכן $\langle M' \rangle \in Halt_\varepsilon$.
בכיוון שני, אם $(\langle M \rangle, w) \notin Halt_{TM}$ אז M לא עוצרת על w. לכן M' לא עוצרת על כל קלט (כי היא רצה כמו M על w), ובפרט על ε. לכן $\langle M' \rangle \notin Halt_\varepsilon$. □

5 עודד - שיעור 5 - 02.06.2024

נתחיל בטענת חימום.

טענה. **(משפט הרדוקציה ל-RE ול-$coRE$)**

1. אם $A \leq_m B$ וגם $B \in RE$ אז $A \in RE$.
2. אם $A \leq_m B$ וגם $B \in coRE$ אז $A \in coRE$.

הוכחה.

1. בדומה להוכחה עבור משפט הרדוקציה ל-R. כלומר נתון שיש מכונת רדוקציה M_f עבור הרדוקציה מ-A ל-B וכן יש מכונה שמקבלת (לא בהכרח מכריעה אותה). נסמן אותן M_B. נרכיב את המכונות. כלומר עבור קלט w, המכונה החדשה M_A, תפעל באופן הבא: תחשב $y = M_f(w)$ ואז תריץ את $M_B(y)$ ותענה כמוה. (או לא תעצור אם M_B לא עוצרת).

ברור ש-M_A קיימת - בעזרת שני הרכיבים M_B, M_f, נותר להוכיח שהיא מקבלת את A.

אם $w \in A$ אז $y \in B$ לפי נכונות מכונת הרדוקציה M_f. אז $M_B(y) = q_{acc}$ (לפי נכונות המכונה M_B מקבלת את B). ואם $w \notin A$ אז $y \notin B$ (לפי נכונות M_f). ואז $M_B(y) \in \{q_{rej}, \perp\}$ (לפי נכונות M_B).

2. נסתכל על השפות המשלימות. אם $A \leq_m B$ אז $\overline{A} \leq_m \overline{B}$ (בדיוק אותה רדוקציה f !). אם $B \in coRE$ אז $\overline{B} \in RE$ ואז לפי משפט הרדוקציה ל-RE מתקיים $\overline{A} \in RE$ ולכן $A \in coRE$. □

5.1 דוגמאות לשימוש ברדוקציות כדי לסווג שפות למחלקות חישוביות

נוכל להשתמש ברדוקציות כדי להראות ששפה שייכת (כיוון אחד) או לא שייכת (כיוון שני) למחלקת חישוביות נתונה.

5.1.1 דוגמא ראשונה - REG_{TM}

דוגמה. נגדיר את השפה הבאה:

$$REG_{TM} = \{\langle M\rangle : L(M) \in REG\}$$

למשל כל מכונת טיורניג M שמקבלת מספר סופי של מילים מקיימת $\langle M\rangle \in REG_{TM}$.

למשל M שמחזירה q_{acc} לכל קלט מקיימת $L(M) = \Sigma^* \in REG$ ולכן $\langle M\rangle \in REG_{TM}$.

למשל M שמכריעה את השפה 0^n1^n מקיימת $\langle M\rangle \notin REG_{TM}$.

<u>**טענה:**</u> $REG_{TM} \in \overline{RE \cup coRE}$ (דוגמא ראשונה לשפה שהיא לא RE ולא $coRE$).

<u>**הוכחה:**</u> נראה בנפרד $REG_{TM} \notin RE$ ו-$REG_{TM} \notin coRE$. נראה רדוקציה $Halt_{TM} \leq REG_{TM}$ ונסיק (לפי משפט הרדוקציה ל-RE) ש-$REG_{TM} \notin coRE$. כלומר נראה f חשיבה כך ש:

$$\langle\langle M\rangle, w\rangle \xrightarrow{f} \langle M'\rangle$$
$$\langle\langle M\rangle, w\rangle \in Halt_{TM} \iff \langle M'\rangle \in REG_{TM}$$

f תחזיר $\langle M'\rangle$ כאשר M' פועלת כך:

עבור קלט w', אם $w' \in \{0^n1^n\}$ תחזיר q_{acc}. אחרת תריץ את $M(w)$ ואם היא תעצור תחזיר q_{acc}.

ברור ש-f חשיבה, כי להכריע את 0^n1^n זה קל ע"י מ"ט ולכן בניית M' ניתנת לביצוע בזמן סופי. נראה את נכונות הרדוקציה.

<u>כיוון אחד:</u> אם $\langle\langle M\rangle, w\rangle \in Halt_{TM}$ אז $M(w)$ עוצרת. אז לכל קלט w', M' עוצרת ומחזיקה q_{acc} (בין אם w' מהצורה 0^n1^n ובין אם לא). לכן $L(M') = \Sigma^*$ ובפרט רגולרית. כלומר $\langle M'\rangle \in REG_{TM}$.

כיוון שני: אם $\langle\langle M\rangle, w\rangle \notin Halt_{TM}$ אז $M(w)$ לא עוצרת. אז עבור קלט מהצורה 0^n1^n, M' מחזירה q_{acc} אבל עבור קלט מצורה אחרת M' לא עוצרת. כלומר $L(M') = \{0^n1^n : n \in \mathbb{N}\}$ ובפרט $L(M') \notin REG$ ועל כן $\langle M'\rangle \notin REG_{TM}$.

כלומר סיימנו להראות ש-$Halt_{TM} \leq REG_{TM}$, כלומר $REG_{TM} \notin coRE$.

הערה (ששימשה ותשמש בעתיד): הנחנו ברדוקציה שהקלט לרדוקציה הוא בפורמט הנכון (במקרה זה - קידוד של מכונה ומילה) ויכול להיות כן או לא בשפה. אבל באופן פורמלי רדוקציה צריכה לטפל גם בקלטים שאינם בפורמט הנכון. ועליהם צריכה להחזיר פלט שמשמר את תכונה השייכות. כלומר פלט שאינו בשפה שבצד ימין. אפשר למשל 1. להחזיר פלט בפורמט לא תואם לשפה הימנית (למשל במקרה הזה משהו שאינו קידוד של מ"ט) 2. להחזיר פלט כן בפורמט אבל שמובטח שאינו בשפה של צד ימין (למשל במקרה הזה קידוד של מכונה שמקבלת את השפה 0^n1^n).

נראה עכשיו $\overline{Halt_{TM}} \leq REG_{TM}$ ונסיק כי $REG_{TM} \notin RE$. הסבר להסקה: $Halt_{TM} \in RE \setminus R$ ובפרט $Halt_{TM} \notin coRE$. לכן $\overline{Halt_{TM}} \notin RE$, ולכן לפי משפט הרדוקציה ל-RE, גם $REG_{TM} \notin RE$.

נראה את הרדוקציה f מ-$\overline{Halt_{TM}}$ ל-REG_{TM}. כלומר נראה f חשיבה כך ש:

$$\langle\langle M\rangle, w\rangle \xrightarrow{f} \langle M'\rangle$$
$$\langle\langle M\rangle, w\rangle \in \overline{Halt_{TM}} \iff \langle M'\rangle \in REG_{TM}$$

כלומר $L(M') \in REG \iff M(w) = \perp$.

אז f תחזיר M' שפועלת באופן הבא: על קלט w', תריץ את $M(w)$ (היא אולי תעצור ואולי לא). לאחר מכן תבדוק האם w' מהצורה 0^n1^n. אם כן, תחזיר q_{acc}. אם לא, תחזיר q_{rej}. ברור ש-f חשיבה - קל לבנות את M' בהינתן $\langle\langle M\rangle, w\rangle$. נותר להראות את נכונות הרדוקציה.

כיוון אחד: אם $\langle\langle M\rangle, w\rangle \in \overline{Halt_{TM}}$ כלומר $M(w) = \perp$, כלומר על כל קלט w', $M'(w) = \perp$ כלומר $L(M') = \emptyset$, כלומר $L(M') \in REG$ ולכן $\langle M'\rangle \in REG_{TM}$.

כיוון שני: אם $\langle\langle M\rangle, w\rangle \notin \overline{Halt_{TM}}$ אז $M(w)$ עוצרת ולכן M' מחזירה q_{acc} לכל קלט מהצורה 0^n1^n ומחזירה q_{rej} לכל קלט שלא מהצורה הזו. כלומר $L(M') = \{0^n1^n : n \in \mathbb{N}\}$ ובפרט $L(M') \notin REG$, ולכן $\langle M'\rangle \notin REG_{TM}$.

הערה: ברדוקציה הקודמת צריך לוודא שמטפלים נכון גם בקלטים שאינם מהפורמט המתאים, כלומר קלטים שאינם מהצורה $\langle\langle M\rangle, w\rangle$. נרצה להחזיר פלט שמובטח שהוא כן שייך לשפה בצד ימין.(בניגוד ההערה הקודמת) (כי מילה לא בפורמט היא כן ב-$\overline{Halt_{TM}}$). הבדיקה האם קלט בפורמט הנכון היא קלה.

סיימנו להראות את נכונות הרדוקציה $\overline{Halt_{TM}} \leq REG_{TM}$ וכיוון ש-$\overline{Halt_{TM}} \notin RE$ הוכחנו ש- $REG_{TM} \notin RE$. ולסיכום $REG_{TM} \in \overline{RE \cup coRE}$.

5.2 הכללת רדוקציה - משפט רייס

5.2.1 המשפט והוכחתו

נרצה להכליל את סוג הרדוקציות שראינו היום ובשבוע שעבר.

הגדרה. שפה L היא **טריוויאלית** אם $L = \emptyset$ או $L = \{\langle M\rangle : M \ is \ a \ turing \ machine\}$.

הגדרה. שפה L היא סמנטית אם לכל זוג מכונות טיורינג M_1, M_2 אשר מקיימות $L(M_1) = L(M_2)$, מתקיים $\langle M_1\rangle \in L \iff \langle M_2\rangle \in L$.

(אם השפה של המכונות שווה, אז או ששתיהן ב-L או ששתיהן לא ב-L)

משפט. (משפט רייס $RICE$)
תהי L שפה של קידודים של מכונות טיורינג, שמקיימת:
א. L לא טריוואלית.
ב. L סמנטית.
אז $L \notin R$.

דוגמה. דוגמאות לשפות סמנטיות:

$$L_1 = \{\langle M\rangle : M(n) = q_{acc} \iff n\ is\ even\}$$
$$L_2 = \{\langle M\rangle : M\ halts\ for\ infinite\ inputs\}$$

דוגמאות לשפות לא סמנטיות:

$$L = \{\langle M\rangle : M\ has\ 4\ states\}$$

ייתכנו M_1 עם 4 מצבים, M_2 עם 100 מצבים שייקיימו $L(M_1) = L(M_2)$.

$$L = \{\langle M\rangle : M\ halts\ on\ w\ before\ |w| + 5\ steps\}$$

למשל אפשר לבנות M_1 שעוצרת מייד ועונה q_{rej} ומכונה M_2 שרצה $|w| + 10$ ואז עונה q_{rej}. מתקיים $L(M_1) = L(M_2) = \emptyset$ אבל $\langle M_1\rangle \in L$ אך $\langle M_2\rangle \notin L$.

הערה. **משמעות משפט רייס.**
אינטואיטיבית משפט רייס אומר שלא ניתן להכריע האם מכונה (או תכנית מחשב) מבצעת משימה חישובית מסוימת. למשל אי אפשר להכריע את:

$$L = \{\langle M\rangle : M(n) = q_{acc} \iff n\ is\ prime\}$$
$$L = \{\langle M\rangle : M(w) = q_{acc} \iff L(M) = L(M_1)\ s.t.\ M_1\ is\ a\ T.M.\}$$

דוגמה. נראה שימוש במשפט רייס. L_1 מהדוגמא הקודמת שפה סמנטית. בנוסף L_1 לא טריוויאלית - כדי להוכיח זאת צריך להראות מכונה אחת ששייכת לשפה ומכונה אחת שלא שייכת - ברור שיש M_1 כך ש- $\langle M_1\rangle \in L_1$, וברור שיש $\langle M_2\rangle \notin L_1$, למשל M_2 שתמיד מחזירה q_{acc}.
מסקנה: לפי רייס $L_1 \notin R$.
באופן דומה עבור L_2 ברור שהיא סמנטית. נראה שהיא לא טריוויאלית. M_1 שעוצרת מיד לכל קלט ומחזירה q_{acc} מקיימת $\langle M_1\rangle \in L_2$, ומצד שני M_2 שהולכת כל הזמן ימינה ולא עוצרת לא משנה מה הקלט, השפה שלה ריקה ובפרט היא עוצרת על 0 קלטים ולכן $\langle M_2\rangle \notin L_2$.
הראינו כי L_2 סמנטית ולא טריוויאלית ולכן לפי רייס $L_2 \notin R$.

הוכחה. **(הוכחת משפט רייס)**

תהי L שפה סמנטית ולא טריוויאלית. נראה $Halt_{TM} \leq_m L$ ונסיק כי $L \notin R$ (לפי משפט הרדוקציה עבור R). נסמן ב-$M_\emptyset$ מ"ט שמקיימת $L(M_\emptyset) = \emptyset$ למשל מכונה שתמיד מחזירה q_{rej}. נניח בה"כ $\langle M_\emptyset \rangle \notin L$ (אחרת, נחזור על ההוכחה עם $\overline{L}$ ואז $\langle M_\emptyset \rangle \notin \overline{L}$ נראה ש-$\overline{L} \notin R$ ונסיק שגם $L \notin R$).
$\langle M_\emptyset \rangle \notin L$ וקיימת M_{in} כך ש-$\langle M_{in} \rangle \in L$ (הקיום מובטח כי L אינה טריוויאלית). נגדיר את f הרדוקציה מ-$Halt_{TM}$ ל-L באופן הבא:

$$\langle \langle M \rangle, w \rangle \xrightarrow{f} \langle M' \rangle$$

כאשר M' פועלת כך: בהינתן קלט w' תריץ את $M(w)$ (ייתכן שלא תעצור). ואז תחזיר את $M_{in}(w')$. ברור ש-f חשיבה, נותר להראות נכונות. (כלומר ש-$\langle M' \rangle \in L \iff \langle \langle M \rangle, w \rangle \in \overline{Halt_{TM}}$)
<u>כיוון אחד:</u> אם $\langle \langle M \rangle, w \rangle \in Halt_{TM}$ אז M עוצרת על w ולכן לכל קלט w', $M'(w')$ עונה כמו $M_{in}(w')$ כלומר $L(M') = L(M_{in})$. לכן $\langle M' \rangle \in L$ (כי L סמנטית).
<u>כיוון שני:</u> אם $\langle \langle M \rangle, w \rangle \notin Halt_{TM}$ כלומר $M(w) = \perp$. אז לכל קלט w' מתקיים $M'(w') \perp$. כלומר $L(M') = \emptyset$ כלומר $L(M') = L(M_\emptyset)$ ועל כן $\langle M' \rangle \notin L$ (כי L סמנטית).
סיימנו. $\square$

הערה. הראינו $L \leq_m Halt_{TM}$ והסקנו $L \notin R$. נראה כאילו יכולנו להסיק גם ש-$L \notin coRE$ לפי משפט הרדוקציה עבור $coRE$. אבל היה שלב שהנחנו בה"כ כי $\langle M_\emptyset \rangle \notin L$. אם זה נכון, אז באמת הראינו $L \notin coRE$. אבל אם $\langle M_\emptyset \rangle \in L$ אז הוכחנו ש-$\overline{L} \notin coRE$ כלומר ש-$L \notin RE$.

5.2.2 דוגמא שנייה - השפה Inf_{TM}

דוגמה. נגדיר:

$$Inf_{TM} = \{\langle M \rangle : |L(M)| = \infty\}$$

כלומר ב-$L(M)$ יש אינסוף מילים.
<u>**טענה:**</u> $Inf_{TM} \notin R$.
<u>**הוכחה:**</u> נשתמש במשפט רייס. Inf_{TM} שפה של קידודים של מכונות.
ראשית נראה ש-Inf_{TM} לא טריוויאלית. מ"ט M_1 שמחזירה תמיד q_{acc}, שפתה Σ^* ולכן אינסופית ולכן $\langle M_1 \rangle \in Inf_{TM}$. מכונה M_2 שמחזירה תמיד q_{rej}, שפתה ריקה ובפרט סופית, ולכן $\langle M_2 \rangle \notin Inf_{TM}$.
שנית נראה ש-Inf_{TM} סמנטית. יהיו M' ו-M'' זוג מכונות שמקיימות $L(M') = L(M'')$. בפרט מתקיים ש-$L(M')$ סופית אמ"ם $L(M'')$ סופית. כלומר $\langle M_2 \rangle \in inf_{TM} \iff \langle M_1 \rangle \in inf_{TM}$.
מסקנה: $Inf_{TM} \notin R$ (מקיימת את תנאי משפט רייס).
חשוב: נשים לב ש-$L(M_\emptyset)$ סופי (כי $L(M_\emptyset) = \emptyset$). לכן $\langle M_\emptyset \rangle \notin Inf_{TM}$ ולכן לפי הרחבת משפט רייס, $Inf_{TM} \notin coRE$.

משפט. (הרחבת משפט רייס)
תהי L שפה של קידודים של מכונות טיורינג, שמקיימת:
א. L לא טריוואלית.
ב. L סמנטית.
אם $\langle M_\emptyset \rangle \in L$ אז $L \notin RE$. אם $\langle M_\emptyset \rangle \notin L$ אז $L \notin coRE$.

6 עמרי - שיעור 6 - 09.06.2024

6.1 סיבוכיות, המחלקות P ו-NP

6.1.1 מוטיבציה

עד עכשיו התעניינו בשאלה אילו בעיות הן בכלל פתירות, וניתנות לחישוב בזמן סופי, ונוכחנו כי יש בעיות שאינן פתירות (על ידי מכונת טיורינג). המחלקה R היא השפות\בעיות שקיים אלגוריתם שמכריע אותם בזמן סופי. אבל ברור שיש הבדל עצום בין אלגוריתם שרץ בזמן 2^{2^n} לבין אלגוריתם שרץ בזמן n^2. לדוגמא אם יכולנו לכפול מטריצות ב-n^2 במקום n^3 האימון של $chatGPT$ היה לוקח יומיים במקום שנתיים (או משהו).
כל הדיון מעכשיו ועד סוף הסמסטר יהיה בתוך המחלקה R.

דוגמה. בעיית כפל של 2 מספרים.
קלט: שני מספרים $a, b \in \{0,1\}^n$.
פלט: $a \cdot b$.
אלגוריתם גן: חיבור פשוט $\underbrace{a+a+...+a}_{b}$. נשים לב כי $b \leq 2^n$ ולכן זמן הריצה הוא $O(2^n)$ וזה גרוע.
אלגוריתם יסודי: כפל ארוך $O(n^2)$.
אלגוריתם אוניברסיטה: התמרת פורייה FFT $O(nlog(n))$.

6.1.2 הגדרות

הגדרה. עבור פונקציית זמן (מונוטונית) $T : \mathbb{N} \to \mathbb{N}$ נגדיר את המחלקה:

$$TIME\left(T(n)\right) := \{L \subseteq \Sigma^n : \exists \text{ in L deciding TM 1-tape } O(T(n))\text{time } \}$$

כאשר מדובר על הזמן ב-Worst Case. כלומר המקסימום על פני הקלטים בשפה $w \in \Sigma^n$.

כעת נרצה להגדיר את מחלקת הבעיות עבורן קיים אלגוריתם "יעיל".

הגדרה. (המחלקה P) - זמן פולינומי

$$P = \bigcup_{k=1}^{\infty} TIME(n^k)$$

למה זוהי ההגדרה הנכונה לאלגוריתם יעיל? יש שתי אקסיומות שהיא מספקת, ואין שום מחלקה גבוהה ממנה שמספקת את שתי האקסיומות:

1. ההגדרה בלתי תלויה במימוש\חומרה או בייצוג של הקלט.
2. סגירות להרכבה (של פולינומים) - כמו פונקציות, הרצה של משהו בזמן פולינומי כמות פולינומית של פעמים זה פולינומי. נניח ש-$|f_1(x)| \leq O(|x|^{c_1})$ וש-$|f_2(y)| \leq O(|y|^{c_2})$. אז זמן הריצה הכולל הוא $|f_2(f_1(x))| \leq f_2(|x|^{c_1}) \leq |x|^{c_1 \cdot c_2}$

הגדרה. (המחלקה $EXP/EXPTIME$) - זמן אקספוננציאלי

$$EXP = \bigcup_{k=1}^{\infty} TIME(2^{n^k})$$

הערה. מההגדרות ברור ש-$P \subseteq EXP$.

טענה. (שבוע הבא) $P \neq EXP$.

הגדרה. באופן דומה, נתעניין גם בסיבוכיות מקום. עבור פונקציה מונוטונית $f : \mathbb{N} \to \mathbb{N}$ נגדיר:

$$SPACE\left(f(n)\right) := \{L \subseteq \Sigma^n : \exists \text{ in L deciding TM 1-tape } O(f(n))\text{space }\}$$

ב-Worst Case. כלומר המקסימום על פני הקלטים בשפה $w \in \Sigma^n$ של המקום הימני ביותר שהראש הגיע אליו.

טענה. $TIME\left(f(n)\right) \subseteq SPACE\left(f(n)\right)$

אבל ההפך לא נכון.

הוכחה. גם אם מכונת טיורינג תנוע ימינה בכל צעד, היא תגיע מקסימום לסיבוכיות מרחב שזהה לכמות הזמן שהיא נעה.
לגבי זה שההפך אינו נכון, אפשר לתת כדוגמא את מכונת הטיורינג שמזהה פלינדרומים שראינו. ☐

שבוע הבא נראה ש $SPACE\left(f(n)\right) \subseteq TIME\left(2^{f(n)}\right)$.

דוגמה. בעית מעגל אוילר (מעגל שמבקר בכל קשת בדיוק פעם אחת) בגרף לא מכוון:

$$EulerCycle/EC = \{G = (V, E) : \exists \text{ G In Cycle Euler}\}$$

בעיית מעגל המינטוני (מעגל שמבקר בכל קודקוד בדיוק פעם אחת) בגרף לא מכוון:

$$HamCycle/EC = \{G = (V, E) : \exists \text{ G In Cycle Hamilton}\}$$

טענה. $EC, HC \in EXP$

הוכחה. צריך להראות שקיים אלגוריתם בזמן אקספוננציאלי $2^{(|V|+|E|)^{O(1)}}$. נתמקד ב-HC.
נגדיר מ"ט M_{HAM}, המקבל בתור קלט ייצוג של מטריצת שכנויות (מותר לנו לבחור). נתחיל בקודקוד $v_0 \in V$.
האלגוריתם יעבור סדרתית על כל $|V|!$ המעגלים\תמורות הפוטנציאלים של קודקודים. כאשר בכל מעבר על תמורה ספציפית האלגוריתם ידחה את התמורה אם $v_i, v_{i+1} \notin E$ או אם $v_i = v_0$ עבור $i < n$. אחרת, זה אומר שנמצא מעגל חוקי והאלגוריתם יקבל.
<u>נכונות</u>: קיים מעגל המילטוני אמ"ם קיימת פרמוטציה π_v חוקית שמתחילה ומסתיימת ב-v_0 אמ"ם $M_{HAM}(G) = q_{acc}$.
<u>זמן ריצה</u>: מקירוב סטירלינג $O(|V|!) = 2^{O(|V| log|V|)} < 2^{O(|V|^2)}$ ☐

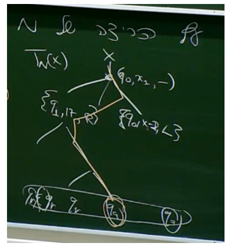
הערה. באופן מעניין, $EC \in P$. כי זה שקול לבדיקה שכל דרגות $v \in V$ זוגיות. לעומת זאת, עבור $HamCycle$, לא ידוע אלגוריתם פולינומי (למעשה, $2^{|V|}$ זה זמן הריצה הכי מהיר עד כה).

הגדרה. <u>**מכונת טיורינג אי-דטרמיניסטית**</u> NTM היא שביעייה $N = \langle \Sigma, \Gamma, Q, \delta, q_0, q_{acc}, q_{rej} \rangle$
היא מכונת טיורינג כרגיל רק שפונקציית המעברים אי-דטרמיניסטית: $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R\}}$

הערה. נשים לב שעבור קלט מסוים $w \in \Sigma^n$ תיתכנה ריצות <u>שונות</u>.

הגדרה. נאמר ש-N ,NTM <u>**מקבלת**</u> את הקלט $x \in \Sigma^n$ אם קיימת ריצה של N כך ש- $N(x) = q_{acc}$.

הערה. דרך טבעית להסתכל על NTM זה בתור עץ $T_N(x)$ של המצבים האפשריים, וה-NTM מקבלת אם קיים עלה שהוא q_{acc}:


איור 13:

הערה. מכונת טיורינג היא **__מכריעה__** אמ"ם כל ריצה אי דטרמיניסטית מסתיימת. זה אומר אחד משני דברים:

- קיימת ריצה שמגיעה ל-q_{acc} (כלומר גרף המצבים לאו דווקא סופי, אבל יש עלה בעץ הריצה עם q_{acc}). במקרה זה זמן הריצה הוא אורך הריצה **__המינימלית__** (בעץ) שמגיעה ל-q_{acc}.
- כל הריצות מגיעות ל-q_{rej} (כלומר גרף המצבים סופי, ובכל העלים יש q_{rej}). במקרה זה זמן הריצה הוא אורך הריצה **__המקסימלית__** (בעץ) שמגיעה ל-q_{rej}.

הגדרה.

$$NTIME(T(n)) := \{L \subseteq \Sigma^n : \exists \text{ in L decides NTM } O(T(n))\text{time nondeterministic } \}$$

הערה: Nondeterministic Time זה זמן הריצה הגרוע ביותר על פני $x \in \Sigma^n$.

__זמן ריצה לא דטרמיניסטי:__

$$ND\ Time(N) := max_{x\in\Sigma^n} depth\,(T_N(x))$$

נשים לב שבמקרה שהמכונה מקבלת, זה הגובה רק עד ה-q_{acc} הראשון. (השערה מבוססת שלי)

הגדרה. **__המחלקה__** NP - Nondeterministic Polynomial Time

$$NP := \bigcup_{k=1}^{\infty} NTIME(n^k)$$

אוסף כל הבעיות שניתנות לפתרון בזמן פולינומי על ידי מכונת טיורינג לא דטרמיניסטית.

הערה. בתרגול נראה ש-$NP \subseteq EXP$. (כי אפשר להמיר את האי-דטרמיניזם בדטרמיניזם במחיר של זמן - סוג של אוטומט מכפלה). אבל אנו לא יודעים האם $NP = EXP$.

טענה. $HamCycle \in NP$.

הוכחה. צריך למצוא אלגוריתם אי-דטרמיניסטי שבזמן $n^{O(1)}$ מקבל גרף G אמ"ם קיים מעגל המיטוני ב-G. נבנה NTM, $N(G)$ שעובדת כך:

N תעבור על הגרף ("הילוך") בין הקודקודים, החל מ-v_0, כאשר בכל שלב N תחזיק את רשימת הצמתים $v_0, ..., v_n$ שהיא ראתה בטיול עד עכשיו, ותדלג לקודקוד $v_{i+1} \rightarrow v_i$ באופן אי דטרמיניסטי. אם v_{i+1} כבר ברשימה, נחזיר q_{rej}. אחרת, אם עברנו על הכול והגענו ל-v_0 אחרי n צעדים, נקבל. $\square$

הגדרה. הגדרה אלטרנטיבית למחלקה NP - באמצעות מוודא: שפה $L \in NP$ אמ"ם קיימת מכונת טיורינג דטרמיניסטית V ("המוודא\Verifier") שרצה בזמן פולינומי, וקיים קבוע $c \in \mathbb{N}$ כך שלכל $x \in \Sigma^*$:

- אם $x \in L$, אז קיים "עדות\Certificate" y המקיימת $|y| \leq n^c$ כך ש-$V(x,y) = q_{acc}$.
- אם $x \notin L$ אז לכל עדות y המקיימת $|y| \leq n^c$, מתקיים $V(x,y) = q_{rej}$.

מסקנה. $P \subseteq NP \subseteq EXP$

האם $P = NP$ היא שאלת מיליון הדולר של מדעי המחשב.

טענה. שתי ההגדרות של NP שקולות.

הוכחה. נראה כי אם שפה L שייכת ל-NP על פי הגדרה אחת, אז היא שייכת גם על פי ההגדרה השנייה.
$NTM \Leftarrow$ מוודא: נתונה מ"ט אי דטרמיניסטית N שמכריעה את L בזמן פולינומיאלי. הCertificate יהיה המסלול המקבל (כלומר עלה=q_{acc}) בעץ הריצה T_N. נשים לב כי אורך המסלול הזה הוא לכל היותר עומק העץ T_N.
נגדיר מכונת טיורינג דטרמיניסטית V אשר בהינתן קלט (x,y) תחזיר q_{acc} אם y היא ריצה מקבלת של N על x. קל לבדוק זאת על ידי בחינת פונקציית המעברים δ של N. אחרת V תחזיר q_{rej}.
<u>זמן ריצה</u>: $|N(x)| \leq O(n^c)$
<u>נכונות</u>: $x \in L$ אמ"ם קיים מסלול $|y| \leq n^c$ כך שהוא מסתיים ב-q_{acc}, אמ"ם קיים מסלול $|y| \leq n^c$ כך ש- $V(x,y) = q_{acc}$.
מוודא $\Leftarrow NTM$: אנו יודעים ש-$x \in L$ אמ"ם קיים $|y| \leq n^{O(1)}$ כך ש-$V(x,y) = q_{acc}$. נגדיר מכונת טיורינג אי דטרמיניסטית N שתכריע את L על ידי בחירת y באופן אי דטרמיניסטי (אפשר כי $|y| \leq n^{O(1)}$) והרצת V על y. אז קיים y עבורו $V(x,y)$ יקבל אמ"ם N היא NTM שרצה בזמן פולינומי ומזהה את L. $\square$

7 עמרי - שיעור 7 - 16.06.2024

ניזכר בבעיית $Hampath$. ראינו את שקילות שתי ההגדרות ל-NP. אז נראה כעת שניתן להראות גם בעזרת הגדרת המוודא ש-$HamPath$ ב-NP. נגדיר מוודא $V(x,y)$ כך: נעבור קשת קשת בקלט ונוודא שאכן זה מעגל המילטון חוקי (כמובן צריך לפרט יותר, אבל לא לרמת המצבים במכונה, פשוט לתאר אלגוריתם). לבסוף צריך להוכיח נכונות וזמן ריצה.

7.1 רדוקציות פולינומיאליות

שאלת מיליון ה$ היא אם $P \stackrel{?}{=} NP$. האם כל בעיה שניתן לוודא בצורה יעילה, ניתן גם לפתור בצורה יעילה. 50+ שנה עברו ועדיין רחוקים מתשובה, אבל כיום נראה איך אפשר לצמצם את החיפוש לשאלה.

הגדרה. (רדוקציה פולינומיאלית) יהיו A, B שפות. נאמר ש-$A \leq_P B$ אמ"ם $A \leq_m B$ ובנוסף הרדוקציה רצה בזמן פולינומי ($n^{o(1)}$).

טענה. תכונות רדוקציה פולינומיאלית

1. טרנזיטיביות: אם $A \leq_P B$ ו-$B \leq_P C$ אז $A \leq_P C$
2. אם $A \leq_P B$ וגם $B \in P$ אז $A \in P$.

הוכחה. ל-2. תהי M_f המכונה שעושה רדוקציה בזמן ריצה פולינומי (n^{c_1}) מ-A ל-B. ותהי M_B המכונת טיורינג הפולינומית שמכריעה את B בזמן ריצה פולינומי. (n^{c_2}). נגדיר מכונת טיורינג:

$$M_A(x) = M_B(M_f(x))$$

M_A רצה בזמן פולינומי ומכריעה את A.

נכונות:

$$x \in A \iff f(x) \in B \iff M_B(f(x)) = q_{acc} \iff M_A(x) = q_{acc}$$

זמן ריצה:

$$|M_A| = O(M_B(M_f(x))) = O(|x|^{c_1 \cdot c_2}) \in Poly(n)$$

□

דוגמה. בעיית הקליקות. מקבלים k ו-G בגרף לא מכוון ושואלים האם קיימת קליקה בגודל k בגרף (תת גרף מלא בגודל k).

$$Clique := \{G = (V, E), k : \exists k - \text{Clique in } G\}$$

נשים לב שאם k היה קבוע אז הבעיה הייתה ב-P כי זמן הריצה של הפתרון הנאיבי הוא $n^k \approx \binom{n}{k}$. אך הבעיה הכללית היא ב-NP. ואנו לא יודעים אם היא ב-P.
בעייה דומה אבל הפוכה היא $Independent - Set(IS)$ מקבלים גרף לא מכוון G ו-k ושואלים האם קיים תת-גרף בגודל k שאין בו קשתות בכלל.

$$IS := \{G = (V, E), k : \exists k - \text{IS in } G\}$$

טענה. $Clique \leq_P IS$.

הוכחה. צ"ל: קיים אלגוריתם (מכונת טיורינג דטרמיניסטית) M פולינומיאלי, שבהינתן גרף G, ו-$k \in \mathbb{N}$, מחזיר גרף $\tilde{G}$ ו-$\tilde{k}$ כך ש-

$$(G, k) \in Clique \iff (\tilde{G}, \tilde{k}) \in IS$$

נתאר את הרדוקציה. M תחזיר את הגרף המשלים $\overline{G}$ (כלומר תהפוך כל קשת לאנטי קשת וכל אנטי קשת לקשת. ותחזיר $(\overline{G},k)$.

<u>נכונות:</u> ב-G יש קליקה בגודל k אמ"ם ב-$\overline{G}$ יש IS בגודל k.

<u>זמן ריצה:</u> M בסך הכול הופכת $1 \to 0$ ו-$0 \to 1$ במטריצת השכנויות של G ולכן זמן הריצה הוא $O(n^2)$ (ואולי פחות כי n זה אורך הקלט אבל לא קריטי). אנחנו מניחים שמודל ה-RAM של גישה ב-$O(1)$ שקול למכונות טיורינג (עד כדי האטה פולינומית). (אם אני מתאר אלגוריתם במודל ה-RAM אפשר לכתוב את זה במבחנים בתחילת התשובה). $\square$

7.2 קשות ושלמות ל-NP

הגדרה. <u>**(קשות ושלמות ל-**NP**)**</u> נאמר ששפה L היא <u>NP**-קשה**</u> (באנגלית $NP-Hard$) אם לכל שפה $L' \in NP$ מתקיים:

$$L' \leq_P L$$

אם בנוסף $L \in NP$ אז נאמר ש-L היא <u>NP**-שלמה**</u> (באנגלית $NP-Complete$)

הערה. לא ברור שכלל <u>קיימת</u> בעיה NP שלמה.

הערה. המוטביציה להגדרות היא הטענה הבאה: אם L^* היא NP-שלמה ($L^* \in NPC$) וגם $L^* \in P$ אז $P = NP$!! וההיפך גם נכון. כלומר באופן פורמלי:

$$P = NP \iff L^* \in P$$

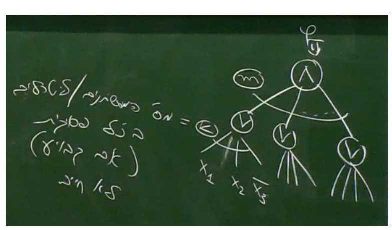
וזיקקנו את כל השאלה לבעיה אחת!

7.3 משפט Cook-Levin

הגדרה. <u>**(נוסחאת** CNF**)**</u> CNF היא נוסחא על n משתנים\ליטרלים בוליאניים (Variable) ו-m פסוקיות (Clauses) מהצורה גימום של איווי, כלומר לדוגמא:

$$\varphi(x_1,...,x_n) = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_4 \vee x_5)$$

והשאלה היא האם קיימת הצבה של המשתנים כך שהפסוקית תהיה אמת. אם בכל פסוקית יש בדיוק k משתנים, הבעיה היא $k-CNF$.
נאמר ש-φ <u>**ספיקה**</u> אם קיימת הצבה $v \in \{T,F\}^n$ כך ש-$\varphi(v) = T$ (כלומר בכל פסוקית יש לפחות ליטרל אחד שהוא T)
ניתן להסתכל על זה בתור עץ:


איור 14:

הערה. CNF $(\wedge_i \vee_j (x_i))$ (גימום של איווי) שקול לבעיית DNF $(\vee_i \wedge_j (x_i))$ (איווי של גימום) בגלל חוקי דה מורגן.

למה. כל פונקציה בוליאנית $f:\{T,F\}^k \to \{T,F\}$ מעל k משתנים ניתנת לכתיבה כ-CNF עם $m = O(2^k)$ פסוקיות.

הוכחה. נסתכל על טבלת האמת של f. יש בה לכל היותר 2^k שורות. נגדיר DNF על ידי איווי של גימום של כל השורות שתוצאתן $True$.

$$\varphi_{DNF} = \vee_{i:f(a_i)=T}\left(\wedge_{j=1}^{k} a_{i,j}\right)$$

$|\varphi_{DNF}| \le 2^k$ פסוקיות.

אפשר להשתמש בדה מורגן ולקבל CNF $\varphi_{CNF} \equiv \varphi_{DNF}$ שקולה כך ש-$|\varphi_{CNF}| \le O(2^k)$. □

הגדרה. השפה SAT:

$$SAT := \{\varphi : \varphi \text{is a satisfiable CNF on } n \text{vars and } m \text{clauses } \}$$

השפה $k-SAT$:

$$k-SAT := \{\varphi : \varphi \text{is a satisfiable CNF on } n \text{vars and } m \text{clauses } \wedge \forall i \in [m], |C_i| = k\}$$

משפט. **(Cook-Levin)** SAT היא NP-שלמה!

(בתרגול נראה $SAT \le_p 3-SAT$ ועל כן גם $3-SAT$ היא NP-שלמה)

הוכחה. ראשית נראה כי $SAT \in NP$. ה"עד" = הצבה מספקת $v \in \{T,F\}^n$. המוודא $V(\varphi, v)$ יבדוק לכל פסוקית אם קיים ליטרל אמת. זמן הריצה הוא $O(n \cdot m)$.

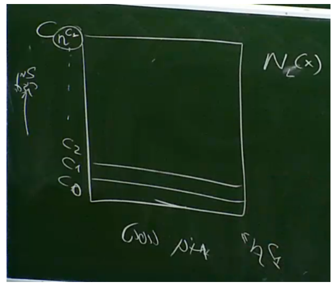
כעת נראה כי SAT היא NP קשה. תהי $L \in NP$ (כלשהי!). צריך להוכיח שקיימת רדוקציה פולינומית M, $L \le_P SAT$. הנתון היחידי הוא שקיימת מכונת טיורינג אי-דטרמיניסטית N_L שמכריעה את L בזמן $O(n^\alpha)$ (זמן אי דטרמיניסטי). נסמן

$$N_L = \langle \Sigma, \Gamma, Q, \delta_N, q_0, q_{acc}, q_{rej} \rangle$$

נסמן $M(x) = \varphi_x$ צריך להוכיח ש-$\varphi_x \in SAT \iff x \in L$.

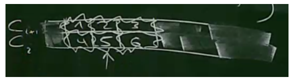
<u>הרעיון המרכזי של ההוכחה:</u> נקודד את הריצה האי-דטרמיניסטית של $N_L(x)$ כנוסחאת CNF φ_x, שהמשתנים שלה הם הבחירות האי-דטרמיניסטית של N_L (העד - ונרצה שיהיה פולינומי).

הבנייה: ריצה של $N_L(x)$ היא פשוט אוסף של קונפיגורציות. כפי שמתואר בטבלה הזאת:



איור 15:

הטבלה הזאת מקודדת את כל האינפורמציה של ריצה א"ד כלשהי. האובסרבציה הקריטית היא שחישוב של מ"ט הוא לוקאלי. כלומר ההבדל בין קונפיגורציה לקונפיגורציה עוקבת נבדל רק על ששת התאים בסביבה של הראש הקורא בתיאור הקונפיגורציות:


איור 16:

<u>סימון:</u> נסמן $c_i \overset{\delta_N}{\to} c_{i+1}$ אם השורה $i+1$ נובעת מהשורה i באופן חוקי על פי δ_N.

<u>טענה:</u>

$$N_L(x) = q_{acc} \iff \wedge_{i=1}^{n^\alpha}\left(\text{"}c_i \overset{\delta_N}{\to} c_{i+1}\text{"}\right) \wedge \left(c_{n^\alpha} = q_{acc} \sqcup ... \sqcup\right)$$

וההוכחה ברורה.

נרצה לקודד את זה כ-CNF (באורך פולינומי).

מה הם המשתנים של ה-CNF?

נסמן $x_{i,j}$ = התוכן של תא i, j במטריצה. (יש $O(n^{2\alpha})$ כאלה כי אם אורך הריצה הוא לכל היותר n^α אז הראש יכול לזוז לכל היותר n^α פעמים ימינה). נשים לב כי $x_{i,j} \in Q \cup \Gamma$. כלומר כדי לקודד את $x_{i,j}$ אנחנו צריכים משהו באורך $log(|Q| + |\Gamma|)$ ביטים (משתנים בוליאנים), וזה מספר קבוע.

במקום לבדוק את התנאי "$c_i \overset{\delta_N}{\to} c_{i+1}$", אנחנו יכולים לבדוק שכל שישייה היא חוקית. כלומר:

$$\left(\text{"}c_i \overset{\delta_N}{\to} c_{i+1}\text{"}\right) \iff \wedge_{6-tuples \in c_i, c_{i+1}}\left(\begin{bmatrix} \square & \square & \square \\ \square & \square & \square \end{bmatrix} = T\right)$$

כמה הצבות יש לשישיה? $(|Q| + |\Gamma|)^6 = O(1)$.

ניזכר בלמה שראינו, כל פונקציה בוליאנית $f : \{T, F\}^k \to \{T, F\}$ מעל k משתנים ניתנת לכתיבה כ-CNF עם $m = O(2^k)$ פסוקיות. אז כמות המשתנים הבוליאנים בנוסחא הנדרשת כדי לתאר שישיה חוקית היא לכל היותר $log(|Q| + |\Gamma|) \cdot (|Q| + |\Gamma|)^6 = O(1)$, על כן ניתן לקודד תנאי של שישיה ב-$2^{O(1)} = O(1)$. בנוסף, יש כמות פולינומית של שישיות בטבלה, ניתן לקודד את כל

$$\wedge_{i=1}^{n^\alpha}\left(\text{"}c_i \overset{\delta_N}{\to} c_{i+1}\text{"}\right)$$

כ-CNF פולינומי, שהוא:

$$\varphi_x = \wedge_{i=1}^{n^\alpha} \wedge_{j=1}^{n^\alpha - 5}\left(\underbrace{\begin{bmatrix} \square & \square & \square \\ \square & \square & \square \end{bmatrix} = T}_{O(1)}\right)$$

וההוספה של התנאי שנגמר ב-q_{acc} מוסיפה עוד עלות פולינומית פשוטה. ומהאופן בו בנינו אותו φ_x ספיקה אמ"ם קיים מילוי חוקי של משתנים לטבלה $N_L(x)$ שמסתיים ב-q_{acc} אמ"ם $x \in L$. $\square$

8 עמרי - שיעור 8 - 23.06.2024

8.1 דוגמאות לשפות NP שלמות נוספות

יש לנו ביד את הבעיות ה-NP שלמה הראשונות. (SAT ו-$3-SAT$).
ניזכר בבעיית הקליקות. בה מקבלים k ו-G בגרף לא מכוון ושואלים האם קיימת קליקה בגודל k בגרף (תת גרף מלא בגודל k).

$$Clique := \{G=(V,E), k : \exists k-\text{Clique in } G\}$$

טענה. $Clique$ היא NP שלמה.

הוכחה. ראינו כבר ש-$Clique \in NP$. (העד: הקליקה: $Q \subseteq V$, $|Q|=k$).
כעת נראה כי $Clique \in NP-hard$. נראה $3-SAT \leq_p Clique$ ונסיק $Clique \in NP-hard$ מטרנזיטיביות של רדוקציות פולינומיאליות.
צריך להראות שקיימת פונקציה חשיבה פולינומית, כך שבהינתן נוסחא φ מחזירה גרף G ו-k כך ש:

$$\varphi \in 3-SAT \iff (G,k) \in Clique$$

כלומר φ ספיקה אמ"ם קיימת קליקה בגודל k ב-G.
אינטואיציה: נרצה שהמספר המקסימלי של פסוקיות c_i אשר ספיקות ע"י הצבה ל-φ יהיה תואם לגודל הקליקה מקסימלית ב-$G = f(\varphi)$.
יש שתי תכונות שמאפיינות הצבה מספקת:

1. בכל פסוקית קיים ליטרל T.
2. עקביות (Consistency): ערכי האמת למשתנים בכל פסוקית הם עקביים.

הבנייה\רדוקציה:

(1) (קודקודים) לכל פסוקית c_i, נייצר בדיוק 7 קודקודים (כי 7 מתוך 8 הצבות\שלישיות מספקות אותה, וכל קודקוד מסמל הצבה כלשהי).
אם מספר הפסוקיות שלנו הוא m, מספר הקודקודים שלנו יהיה $7m$.
(2) (קשתות) נמתח קשת בין כל 2 קודקודים אמ"ם הקודקודים עקביים אחד עם השני, לא תהיה קשת אם t_i ו-t_j חולקות משתנה משותף וערך האמת שלו איננו עקבי.
(3) נגדיר $k := m$.
לדוגמא:
עבור הפסוקית $(x_1 \vee \overline{x_2} \vee x_3)$ נייצר את הקודקודים $(000), (001), (100), (101), (011), (110), (111)$ (נשים לב שאין את הקודקוד (010) כי ההצבה המתאימה לא מספקת את הפסוקית).
עבור הפסוקית $(x_1 \vee x_2 \vee x_4)$ נייצר את הקודקודים $(010), (001), (100), (101), (011), (110), (111)$. (נשים לב שאין את הקודקוד (000) כי ההצבה המתאימה לא מספקת את הפסוקית).
לעולם לא נמתח קשתות בין קודוקודים של אותה פסוקית. במקרה הזה יהיו קשתות בין: $(a_1y_1z_1)$ (של הפסוקית הראשונה) ו-$(a_2y_2z_2)$ של הפסוקית השנייה, אמ"ם $(a_1=a_2) \wedge (y_1=y_2) \wedge (z_1=z_2)$ (חשוב לשים לב שכבר דאגנו לשלילה כשייצרנו את הקודקודים ולכן נדרוש רק שוויון ולא אי שוויון).
זמן הבנייה:
(נוכיח במודל ה-RAM)

- לייצר את $7m$ הקודוקודים זה $O(m)$.
- לייצר את הקשתות זה $O(|V|^2) = O(m^2)$, למתוח כל קשת בין כל t_i, t_j זה $O(1)$.
- סה"כ: $O(n + m^2)$

<u>נכונות:</u>

בכיוון ראשון, נניח כי $\varphi \in 3 - SAT$ ונראה כי $(G, k) \in Clique$. תהי $v^* \in \{0,1\}^n$ הצבה מספקת ל-φ, אז בפרט $v^*|_{c_i} \in \{0,1\}^3$ (ההצבה הלוקאלית לפסוקית c_i) מספקת לכל i. ומכיוון שכל השלשות נובעות מאותה הצבה v^* אז הקודקודים $v^*|_{c_i}$ ב-G עקביים. אז קבוצת הקודקודים $Q := \{v^*|c_i : i \in [m]\}$ היא קליקה בגודל $m = k$.

בכיוון שני, נניח שקיימת קליקה Q ב-G כך ש-$|Q| = k$ ונראה כי קיימת הצבה $v^* \in \{0,1\}^n$ שמספקת את φ. Q חייב להכיל בדיוק שלשה\קודקוד אחד מכל פסוקית. (כי בין קודקודים של אותה פסוקית אין קשתות). נגדיר את v_i^* להיות ערך האמת בשלשה שרירותית כלשהי שמכילה את x_i או $\overline{x_i}$. אנו טוענים כי השמה זו מוגדרת היטב. זאת מכיוון שכל ערך אמת בשלשה אחרת ב-Q חייב להיות עקבי כי זו קליקה והן מחוברות בקשת. $\square$

הגדרה. השפה $SubsetSum(SS)$:

$$SS := \{(A, s) : text\, below\}$$

A מערך של N מספרים (מעל Σ) כך שקיימת תת-קבוצה $B \subseteq [n]$ כך ש-$\sum_{i \in B} A[i] = s$.

דוגמה.

$$A = [2, 13, 5, 7, 10, 3]$$
$$(A, 18) \in SS$$
$$(A, 27) \in SS$$
$$(A, 51) \notin SS$$

טענה. SS היא NP שלמה.

הוכחה. $SS \in NP$. העד הוא תת קבוצה $B \subseteq [N]$ כך ש--$\sum_{i \in B} A[i] = s$. המוודא ייסכום את כל האיברים ב-B (זמן ריצה $O(N)$ במודל ה-RAM).

כעת נראה כי $SS \in NP - Hard$. נראה ש- $3 - SAT \leq_P SS$. צריך להראות שקיימת פונקציה חשיבה פולינומית, כך שבהינתן נוסחא φ מחזירה מערך A וסכום s כך ש:

$$\varphi \in 3 - SAT \iff (A, s) \in SS$$

<u>רעיון הבנייה:</u> בהינתן $\varphi = \wedge_{i=1}^m c_i$ נבנה מערך A כך ש-$N = |A|$ של מספרים <u>בייצוג בינארי</u> כך שהצבה מספקת לפסוקית c_i תתאים לתת קבוצה $B_i \subseteq [N]$ כך ש-$\sum_{j \in B_i} A[j]$ זה מספר שתואם את <u>הספרה ה-i</u> של $s \in \{0,1\}^{O(m)}$. בזכות זה שאנחנו מסתכלים על הספרות בנפרד, זה מאפשר להפוך m תנאים לתנאי אחד.

בנייה: נבנה מערך של מספרים בינאריים. (ניתן לדמיין אותו כמערך דו מימדי כך שכל שורה היא מספר במערך). המערך יהיה בגודל $(m+n)$ ובכל שורה יהיו $(m+n)$ ספרות, כלומר סה"כ $(m+n)^2$ ספרות.

לכל משתנה x יוקדשו שתי שורות, שורת $true$ שורת $false$.

לכל פסוקית יוקדשו שתי שורות $padding$.

בשורה ה-$true$ של x_i, אם x_i מופיע ב-c_i נשים 1 בעמודה של c_i.

בשורה ה-$false$ של x_i, אם $\overline{x_i}$ מופיע ב-c_i נשים 1 בעמודה של c_i.

בשורות ה-$padding$ של c_i, יהיה 1 בעמודה המתאימה ל-c_i. (כדי להשלים את הסכום ל-3).

בנוסף, בשתי השורות, $true$ ו-$false$ של x_i, נכתוב 1 בעמודה המתאימה ל-x_i. (כך לבסוף נוודא שאנחנו בוחרים ערך אחד בלבד (אמת או שקר) ל-x_i).

נגדיר: $s = \underbrace{1...1}_{\text{ntimes}}\ \underbrace{3...3}_{\text{mtimes}}$

לדוגמא עבור הנוסחא $(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$:

$\heartsuit$	x_1	x_2	x_3	x_4	c_1	c_2
$x_1(t)$	1	0	0	0	1	1
$x_1(f)$	1	0	0	0	0	0
$x_2(t)$	0	1	0	0	0	1
$x_2(f)$	0	1	0	0	1	0
$x_3(t)$	0	0	1	0	1	0
$x_3(f)$	0	0	1	0	0	0
$x_4(t)$	0	0	0	1	0	1
$x_4(f)$	0	0	0	1	0	0
$c_1(p1)$	0	0	0	0	1	0
$c_1(p2)$	0	0	0	0	1	0
$c_2(p1)$	0	0	0	0	0	1
$c_2(p2)$	0	0	0	0	0	1
s	1	1	1	1	3	3

הוכחת הנכונות נותרת כתרגיל. $\square$

9 עמרי - שיעור 9 - 30.06.2024

9.1 משפטי היררכיה

בשיעורים הבאים נראה שיחסים בין מחלקות זיכרון פועלים באופן שונה מהותית מיחסים בין זמן. כעת נראה את משפטי ההיררכיה שבעיקרון אומרים שאם יש לנו יותר משאבים (זיכרון, זמן וכולי) יש לנו יותר כוח חישובי. ראינו כבר הפרדה בין $n^{O(1)}$ (P) לבין 2^n (EXP). המשפט הזה יראה לנו הפרדה יותר עדינה נגיד בין $nlogn$ ל-n.

הגדרה. **<u>פונקציות חשיבות בזמן</u>** (Time-Constructible Functions).
פונקציית זמן $f:\mathbb{N}\to\mathbb{N}$ היא TCF אם:

1. היא מונוטונית: $f(n+1)\geq f(n)$
2. $f(n)\geq n$
3. $1^n\to 1^{f(n)}$ (כלומר בהינתן הקלט n בייצוג אונארי, קיים אלגוריתם שמחשב את $f(n)$ בייצוג אונארי **(בתרגול ובשאר הקורס זה בינארי)** בזמן $O(f(n))$. (במילים אחרות קיים אלגוריתם יעיל שסופר עד $f(n)$)

דוגמה. דוגמאות לפונקציות TCF: $2^{2^n},2^{log(n)},n^{1.5},n$

הערה. נשים לב כי אמנם n בייצוג אונארי היא פונקציית זמן חשיבה, אך הייצוג הבינארי של n אינו חשיב בזמן $O(f(n))$. כלומר n כן רלוונטי למשפט ההירכיה בזמן (היא הגבול התחתון של המשפט) אבל היא לא חשיבה בזמן בייצוג הבינארי שמדברים עליו בדרך כלל.

משפט. (משפט ההירכיה בזמן) (Time Heirarchy Theorem).
לכל פונקציות זמן <u>חשיבות</u> (TCF) $f(n),g(n):\mathbb{N}\to\mathbb{N}$ כך ש- $f(n)\cdot log\left(f(n)\right)=o\left(g(n)\right)$ מתקיים:

$$TIME\left(f(n)\right)\subsetneq TIME\left(g(n)\right)$$

(במילים אחרות, אם הפער האסימפטוטי בין $f(n)$ ל-$g(n)$ הוא לוגריתמי, אז יודעים למצוא בעיה שניתן להכריע על ידי מכונת טיורינג שרצה בזמן $g(n)$ אבל לא בזמן $f(n)$).

דוגמה. מהמשפט נובע: $TIME\left(n\right)\subsetneq TIME\left(n^2\right)\subsetneq TIME\left(n^3\right)$.

הוכחה. הרעיון המרכזי דומה מאוד ללכסון (כמו בבעיית העצירה, שקיימת $L\notin R$).
<u>תזכורת:</u> (מ"ט אוניברסלית): קיימת מ"ט אוניברסלית U כך שלכל מ"ט M וקלט x, $U(\langle M,x\rangle)=M(x)$, וזמן הריצה מקיים: $|U(\langle M,x\rangle)|=$ $|\langle M\rangle|^{O(1)}\cdot|M(x)|\cdot log|M(x)|$.
יהיו f ו-g כמו בתנאי המשפט.
נגדיר מכונת טיורינג D הפועלת כך: בהינתן קלט $\langle M\rangle$ כך ש-$n=|\langle M\rangle|$, D תשמור מונה שישמור את כמות הצעדים עד $1^{(g(n))}$ - (אפשרי כי g היא TCF). D תסמלץ את ריצת $U(\langle M,M\rangle)$ במשך $O(f(n))$ (למשל $g(n)$) צעדים.
אם $M(M)$ לא עצרה אחרי לכל היותר $O(f(n))$ צעדים, D תחזיר q_{rej}.
אם $M(M)$ החזירה q_{rej}, D תחזיר q_{acc}.
אם $M(M)$ החזירה q_{acc}, D תחזיר q_{rej}.
נשים לב כי השפה $L(D)$ היא שפת הקידודים של כל מכונות הטיורינג שמחזירות q_{rej} על קלט שהוא הקידוד של עצמן (זה האלכסון) תוך $O\left(f(n)\right)$ צעדים.
נשים לב כי $L(D)\in TIME\left(g(n)\right)$ למה $O\left(g(n)\right)$? המכונה האוניברסלית מוסיפה עלות של $|\langle M\rangle|^{O(1)}\cdot log(f(n))\cdot f(n)$ ומהנתון $log\left(f(n)\right)=o\left(g(n)\right)$. אבל $|\langle M\rangle|^{O(1)}=O(n^{O(1)})$ והיינו רוצים לגרום לאוברהד הזה להיות זניח ביחס ל-n. כלומר ש- $\frac{n^{O(1)}}{n}\underset{n\to\infty}{\to}0$.
הטריק (Padding Argument\ריפוד) הוא פשוט לקודד את הקלט $\langle M\rangle$ עם ריפוד בגודל n^n ("לפני" שמכניסים אותו למכונה). עכשיו מהגדרה עדיין $n=|\langle M\rangle|$ ועל כן החלק עצמו שמשמש לקידוד שואף ל-0 כאשר n שואף לאינסוף.
כעת נניח בשלילה כי $L(D)\in TIME(f(n))$. אז אפשר להכריע בזמן $f(n)$ את $L(D)$. על כן אפשר להכריע בזמן $f(n)$ גם את $\overline{L(D)}$ (פשוט לענות הפוך). תהא $\overline{D}$ השפה שמכריעה את $\overline{L(D)}$ אז זמן הריצה שלה הוא $O\left(f(n)\right)$. נחלק כעת למקרים:

- אם $\overline{D}\in\overline{L(D)}$ אז $\overline{D}\notin L(D)$ לכן $D\left(\overline{D}\right)=q_{rej}$ ולכן שני דברים - מהגדרת $\overline{D}$, $\overline{D}\left(\overline{D}\right)=q_{acc}$ ומהגדרת D, $\overline{D}\left(\overline{D}\right)=q_{rej}$. סתירה.

- אם $\overline{D} \notin \overline{L(D)}$ אז $\overline{D} \in L(D)$ לכן $D\left(\overline{D}\right) = q_{acc}$ ולכן שני דברים - מהגדרת $\overline{D}$, $\overline{D}\left(\overline{D}\right) = q_{rej}$ ומהגדרת D, $\overline{D}\left(\overline{D}\right) = q_{acc}$ (כי זמן הריצה מובטח). סתירה.

על כן $L(D) \notin TIME\left(f(n)\right)$ וראינו כבר כי $L(D) \in TIME\left(g(n)\right)$ וסיימנו. □

מסקנה.

$$TIME(n) \subsetneq TIME(nlog^2(n)) \subsetneq P \subsetneq SUBEXP$$

הגדרה. **<u>פונקציות חשיבות בזיכרון</u>** (Space-Constructible Functions).
פונקציית זמן $f : \mathbb{N} \to \mathbb{N}$ היא TCF אם $1^n \to 1^{f(n)}$ (כלומר בהינתן הקלט n בייצוג אונארי, קיים אלגוריתם שמחשב את $f(n)$ בייצוג אונארי תוך שימוש ב-$O(f(n))$ זיכרון).

משפט. (משפט ההיררכיה בזיכרון) (Space Heirarchy Theorem).
לכל פונקציות זיכרון <u>חשיבות</u> (SCF) $f(n), g(n) : \mathbb{N} \to \mathbb{N}$ כך ש- $f(n) = o\left(g(n)\right)$ מתקיים:

$$SPACE\left(f(n)\right) \subsetneq SPACE\left(g(n)\right)$$

(במילים אחרות, אם קיים פער אסימפטוטי **כלשהו** בין $f(n)$ ל-$g(n)$, אז יודעים למצוא בעיה שניתן להכריע על ידי מכונת טיורינג שרצה תוך שימוש ב-$g(n)$ זיכרון אבל לא ב-$f(n)$ זיכרון).

הוכחה. אותה הוכחה, רק שהאוברהד של <u>הזיכרון</u> של מכונת טיורינג אוניברסלית הוא $O_M(1)$ (אוברהד שתלוי רק בקידוד - עושים Padding כדי לתקן). (ניתן להשתמש מחדש בזיכרון). □

הערה.

1. הבעיה היותר "טבעית" $L_f = \{M : M(\langle M \rangle) = q_{acc} \text{ in } O(f(n))\text{time } \} \notin TIME\left(f(n)\right)$. הוכחה: לכסון.
2. משפט ההיררכיה בזמן הלא דטרמיניסטי, גורס שעבור פונקציות זמן <u>חשיבות</u> (TCF) $f(n), g(n) : \mathbb{N} \to \mathbb{N}$ כך ש- $f(n) = o\left(g(n)\right)$ מתקיים:

$$NTIME\left(f(n)\right) \subsetneq NTIME\left(g(n)\right)$$

9.2 סיבוכיות מקום

תזכורת:

$$SPACE\left(f(n)\right) := \{L \subseteq \Sigma^n : \exists \text{ in L deciding DTM } O(f(n))\text{space } \}$$

$$NSPACE\left(f(n)\right) := \{L \subseteq \Sigma^n : \exists \text{ in L deciding NTM } O(f(n))\text{space }\}$$

הגדרה. <u>**(המחלקה** $PSPACE$**)**</u> - זיכרון פולינומי

$$PSPACE = \bigcup_{k=0}^{\infty} SPACE(n^k)$$

הגדרה. <u>**(המחלקה** $NPSPACE$**)**</u>

$$NPSPACE = \bigcup_{k=0}^{\infty} NSPACE(n^k)$$

טענה.

$$P \subseteq NP \subseteq PSPACE \subseteq NPSPACE \subseteq EXP$$

הוכחה. $P \subseteq NP$ טריוויאלי. $PSPACE \subseteq NPSPACE$ טריוויאלי. היום נוכיח את $NP \subseteq PSPACE$ ואת $NPSPACE \subseteq EXP$.

<u>$NP \subseteq PSPACE$:</u>

מספיק להראות ש-$3-SAT \in PSPACE$. (הסבר: כל רדוקציה פולינומיאלית בזמן היא פולינומיאלית גם במקום (כי בכל צעד זמן המכונה יכולה להגדיל את הזיכרון ב-1 לכל היותר)).

יהא סדר לקסיקוגרפי על כל ההשמות $v \in \{T, F\}^n$. נסמן ב-$g : \{0,1\}^n \to \{0,1\}^n$ את הפונקציה שמקבלת השמה ומחזירה את ההשמה הבאה בסדר הזה.

נגדיר מ"ט (דטרמיניסטית) M, בהינתן נוסחאת $3-CNF$ φ עם n משתנים ו-m פסוקיות.

- תגדיר $v_0 \leftarrow 0^n$, $v = v_0$.
- לכל $0 \leq i \leq 2^n$ היא תבצע:
  - כתוב על הסרט את v (ההצבה).
  - נשערך את $\varphi(v) \in \{T, F\}$ ($O(m)$ מקום). אם זה T אז נעצור ונקבל. אחרת נמשיך.
  - הצב $v = g(v)$.
- אם שום השמה לא T, לא נקבל.

הנכונות ברורה כי עברנו על כל ההשמות.

סיבוכיות מקום: $O(m+n)$ כדי לשמור את φ, $O(n)$ כדי לשמור מונה (v). סה"כ זמן ליניארי $O(m+n) \in PSPACE$.

<u>$NPSPACE \subseteq EXP$:</u>

לשם פשטות, נתחיל ב-$PSPACE \subseteq EXP$. זה נובע מטענה כללית יותר: הטענה היא שלכל $f(n)$:

$$SPACE\left(f(n)\right) \subseteq TIME\left(2^{f(n)}\right)$$

הוכחה: ההבחנה המרכזית היא שאם מכונת טיורינג M עוצרת היא לא תחזור על אותה קונפיגורציה פעמיים. המסקנה היא שזמן הריצה של $M(x)$ חסום מלמעלה במספר הקונפיגורציות של M. כמה קונפיגורציות יכולות להיות למכונת טיורינג שרצה בסיבוכיות מקום $O\left(f(n)\right)$?
תשובה: $|\Gamma|^{O(f(n))} \cdot f(n) \cdot |Q| = 2^{O(f(n))}$. על כן:

$$runtime(M) \leq \#conf \leq 2^{O(f(n))}$$

אחלה. אבל האי-שוויון השמאלי נכון רק למכונת טיורינג דטרמיניסטית.
נשים לב כי גם עבור מ"ט אי דטרמיניסטית מתקיים $\#conf \leq 2^{O(f(n))}$, על מנת לחשב את זמן הריצה, מספיק לבדוק האם קיים מסלול מקבל מקונפי ההתחלה. נוכל לבצע חיפוש BFS על גרף הקונפיגורציות. חיפוש BFS דורש זמן לינארי בגודל הגרף $O(|E|+|V|)$, ומתקיים $|V| = \#conf, |E| \leq |V|^2$ ועל כן סה"כ זמן זה הוא $2^{O(f(n))}$, כנדרש. □

10 עודד - שיעור 10 - 07.07.2024

10.1 שוויון PSPACE=NPSPACE

משפט. משפט $Savitch$. לכל $f(n)$ שחשיבה במקום $O\left(f(n)\right)$:

$$NSPACE\left(f(n)\right) \subseteq SPACE\left(f^2(n)\right)$$

כלומר, כל שפה שניתן להכריע במקום $O\left(f(n)\right)$ על ידי מכונת טיורינג לא דטרמיניסטית, ניתן גם להכריע אותה במקום $O\left(f^2(n)\right)$ על ידי מכונת טיורינג דטרמיניסטית.

דוגמה.

1. $NSPACE(n^3) \subseteq SPACE(n^6)$
2. לכל $k \in \mathbb{N}$ קבוע מתקיים $NSPACE(n^k) \subseteq SPACE(n^{2k})$

מסקנה.

$$PSPACE = NPSPACE$$

הוכחה. ברור ש-$PSPACE \subseteq NPSPACE$, כי מכונת טיורינג דטרמיניסטית היא מקרה פרטי של מ"ט א"ד ולכן כל שפה שניתנת להכרעה במקום פולינומי דטרמיניסטי ניתנת להכרעה גם במקום פולינומי לא דטרמיניסטי. נראה שגם $NPSPACE \subseteq PSPACE$. תהי $L \in$

$.NPSPACE$

תזכורת: $NPSPACE = \bigcup_{k=1}^{\infty} NSPACE\left(n^k\right)$. אז קיים k קבוע כך ש- $L \in NSPACE(n^k)$. לפי משפט $Savitch$, $L \in SPACE(n^{2k})$ ולכן $L \in PSPACE$. תזכורת: $PSPACE = \bigcup_{k=1}^{\infty} SPACE\left(n^k\right)$.

על כן $PSPACE = NPSPACE$ $\square$

הערה. אם היינו מראים גרסה של $Savitch$ למחלקות זמן, כלומר משהו מהצורה $NTIME\left(f(n)\right) \subseteq TIME\left(f^c(n)\right)$ כאשר c קבוע כלשהו, זה היה מוכיח באופן דומה ש-$P = NP$.

הגדרה. (<u>גרף קונפיגורציות</u>): נגדיר את גרף הקונפיגורציות $G_{M,w}$, עבור $w \in \Sigma^*$. הצמתים: כל הקונפיגורציות האפשריות (במגבלת המקום). קשת מכוונת: לפי מעבר δ של M. כלומר $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}\rangle$. אם שתי קונפיגורציות c_1, c_2 הן עוקבות לפי δ אז תהיה קשת מכוונת מהצומת שמתאים ל-c_1 לצומת שמתאים ל-c_2.

הערה.

1. עבור מכונת טיורינג דטרמיניסטית, בגרף הקונפיגורציות לכל צומת יש קשת יוצאת אחת ויחידה. (בהתאם לפעולה המתאים שקובעת δ). קשתות נכנסות - יתכן יותר מאחת.

2. עבור מכונת טיורינג א"ד, בגרף הקונפיגורציות לכל צומת תיתכן יותר מקשת יוצאת אחת, בהתאם לפונקציה δ.

הגדרה. (<u>צמתים חשובים בגרף הקונפיגורציות</u>):

$c_0 = q_0 w$ היא <u>הקונפיגורציה ההתחלתית</u>, כלומר הראש בהתחלה, המצב הוא q_0 והסרט מכיל את w.

c_{acc} היא <u>הקונפיגורציה המקבלת</u>.

הערה. ייתכן שיש יותר מקונפיגורציה מקבלת יחידה. כל קונפיגורציה שבה המצב q_{acc}, לא חשוב מיקום הראש, ולא חשוב תוכן הסרט. אבל נוכל להניח בה"כ שיש קונפיגורציה מקבלת יחידה. נראה איך: נשנה את המצב q_{acc} למצב $q_{almost\, acc}$ שהוא גורם למכונה למחוק את כל הסרט, להחזיר את הראש לתחילת הסרט ואז לעבור למצב q_{acc}.

הגדרה. $w \in L$ אמ"ם קיימת ריצה מקבלת של $M(w)$, אמ"ם קיים ב-$G_{M.w}$ מסלול מכוון מ-$c_0 = q_0 w$ אל c_{acc}.

הוכחה. **(למשפט** $Savitch$**)** נתונה שפה $L \in NSPACE\left(f(n)\right)$, כלומר קיימת מכונת טיורינג א"ד M שמכריעה את L תוך הסתפקות במקום $O\left(f(n)\right)$. נראה ש-$L \in SPACE\left(f^2(n)\right)$ כלומר שקיימת מכונת טיורינג דטרמיניסטית M' שמכריעה את L תוך הסתפקות במקום $O\left(f^2(n)\right)$.

הצעה להוכחה (שגויה): בהינתן w (ו-M) נייצר את $G_{M,w}$. ואז נריץ על $G_{M,w}$ אלגוריתם שבודק האם יש מסלול מכוון מ-c_0 ל-c_{acc}. למשל אפשר על ידי הרצת BFS/DFS החל מ-c_0 ולבדוק האם מגיעים ל-c_{acc}. קיבלנו אלגוריתם (מכונה) דטרמיסניטית שעונה האם $w \in L$. כלומר מכונה דט' M' כך ש-$L(M') = L$.

הבעיה: M' הנ"ל לא מסתפקת במקום $O\left(f^2(n)\right)$. בפרט M' רושמת על הסרט את כל $G_{M,w}$. כמה מקום זה דורש? מספר הצמתים הוא לכל היותר מספר הקונפיגורציות, כלומר $\#conf = |Q| \cdot |\Gamma|^{O(f(n))} \cdot O\left(f(n)\right)$. קיבלנו מספר צמתים אקספוננציאלי ב-$f(n)$. כדי לייצג שם של קונפ' בודדת צריך מחזורת באורך:

$$\begin{aligned} O\left(log(\#conf)\right) &= O\left(log(Q) + f(n) \cdot log(\Gamma) + log(f(n))\right) \\ &= O\left(f(n)\right) \end{aligned}$$

אפשר להגיע לאותה מסקנה מתוך ניתוח ישיר: מה האינפורנציה שצריך לתאר בקונפיגורציה (בעיקר - תוכן הסרט שדורש מקום $O\left(f(n)\right)$).

המטרה לשעה הבאה: להצליח לענות על שאלת המסלול מ-c_0 ל-c_{acc} ב-$G_{M,w}$ על ידי מ"ט דט' M' כאשר אי אפשר אפילו לכתוב את כל $G_{M,w}$.

הרעיון: נראה איך לענות על השאלה הבאה. בהינתן (v_1, v_2, k) האם יש מסלול באורך לכל היותר k מ-v_1 ל-v_2. (בלי לכתוב את הגרף המלא).

נקרא לפרוצדורה הזאת $reach(v_1, v_2, k)$ כאשר עונים עליה על ידי הפסאודו קוד הבא:

אם $k=0$ נחזיר כן אמ"ם $v_1=v_2$. אם $k=1$ נחזיר כן אמ"ם $v_1=v_2$ או v_2 עוקב של v_1 כלומר יש קשת מכוונת (v_1,v_2). לכל צומת v_3 נבדוק האם $reach(v_1,v_3,\lceil\frac{k}{2}\rceil)$ וגם האם $reach(v_3,v_2,\lfloor\frac{k}{2}\rfloor)$. אם התשובה היא "כן" לשניהם, נעצור ונחזיר "כן". אם התשובה היא "לא," לפחות לאחד מהם, נעבור לצומת v_3 המועמד הבא. ואם עברנו על כל הצמתים המועמדים להיות v_3 ותמיד התקבל "לא" באחת או יותר משתני הקריאות, נחזיר "לא".

כעת נתאר מימוש חסכוני במקום לפרוצדורה $reach$. זה סוג של סטאק. לכל שלב (עומק) של הרקורסיה תהיה שלשה של צומת התחלה, צומת סיום, חסם על האורך. ב"קריאה רקורסיבית" נקצה שלשה חדש אבל נמחזר את המקום כך שלכל עומק ברקורסיה יש רק שלשה אחת. כלומר, השלשה השנייה תשמש עבור כל הבדיקות $reach(v_1,v_3,\lceil\frac{k}{2}\rceil)$, $reach(v_3,v_2,\lfloor\frac{k}{2}\rfloor)$ לכל צומת v_3. כדי לענות על השאלה האם $v_1=v_2$ אפשר לעשות השוואת מחרוזות ללא צורך במקום נוסף. כדי לבדוק האם $(v_1,v_2)\in E(G_{M,w})$ נצטרך לבדוק האם v_2 קונפיגורציה עוקבת של v_1, כלומר האם הן שוות בכל מקום, למעט באיזור של הראש, ושם השינויים תואמים לפונקצייה δ. שוב - ניתן לבצע ללא צורך במקום נוסף. צריך להשתכנע שאפשר לעבור על כל הקונפיגורציות בסדר לקסיגורפי עולה (או סדר אחר) כדי שנוכל לבדוק את כל ההצבות האפשריות ל-v_3.

כמה מקום נצטרך סה"כ כדי לענות על $reach(c_0,c_{acc},t)$ כאשר $t=\#conf$? המקום עבור שלשה $\times$ מספר מקסימלי של שלשות.

לשלשה בודדת: קונפ' ראשונה $O\left(f(n)\right)$, קונפ' שנייה $O\left(f(n)\right)$. מספר בין 0 ל-$\#conf$, $O\left(f(n)\right)$. סה"כ $O\left(f(n)\right)$.

מספר השלשות: בדיוק עומק הרקורסיה המקסימלי $O\left(log(\#conf)\right)=O\left(f(n)\right)$.

סה"כ: $O\left(f^2(n)\right)$ לכל הריצה של M'. □

10.2 בעיות חיפוש

עד היום חקרנו בעיקר שפות, כלומר בעיות הכרעה (האם?). נרצה להסתכל גם על בעיות חיפוש (איך?).

דוגמה.

$$SAT:=\{\varphi:\varphi\text{is a satisfiable CNF }\}$$

נסתכל על השאלה - <u>מהי</u> ההצבה המספקת עבור φ.

הגדרה. נגיד שלשפה L יש **<u>רדוקציה עצמית</u>** אם קיים אלגוריתם שפותר את בעיית החיפוש בזמן פולינומי, כאשר יש לו גישה ל-"אורקל" שעונה על בעיית ההכרעה.
כלומר בכל צעד ריצה האלגוריתם יכול, בנוסף לכל חישוב רגיל, גם לשאול האם קלט מסוים שייך או לא שייך לשפה L. הערה: בניית הקלט נספרת בזמן ריצת האלגוריתם. המענה על השאילתה לאורקל ניתן ב-$O(1)$.

טענה. לשפה SAT יש רדוקציה עצמית.

הוכחה. נראה שקיים אלגוריתם כך שבהינתן נוסחא φ, אם היא ספיקה, האלגוריתם מוצא הצבה מספקת ל-φ. האלגוריתם רץ בזמן פולינומי ויש לו גישה לאורקל לשפה SAT.

האלגוריתם:

(א) נבדוק האם φ ספיקה. אם לא נעצור ונחזיר F. אם כן נאתחל $i=1$.

(ב) נצרף ל-φ את $\wedge x_i$ ונבדוק האם ספיקה. אם כן, נמשיך. אם לא, נחליף את $\wedge x_i$ בתוספת $\wedge\overline{x_i}$

(ג) $i++$. אם $i<n$ (מספר המשתנים) חזור לב'.

(ד) נסיק הצבה מספקת לפי התוספות. כלומר לכל i: $x_i=T$ אם הוספנו $\wedge x_i$, לעומת $x_i=F$ אם הוספנו $\wedge\overline{x_i}$.

נכונות: אם φ ספיקה, אז בתום כל צעד. הנוסחא החדשה נשארת ספיקה. ולכן ההצבה שהסקנו בסוף היא מספקת.

זמן הריצה: $O(1)$ לכל צעד ריצה. יש מספר קבוע של צעדים לכל ערך i ו-i רץ מ-1 עד מס' המשתנים ב-φ, כלומר סה"כ פולינומי באורך הקלט φ. $\square$

מסקנה. אם $P = NP$, כלומר אם אפשר להכריע הזמן פולינומי את SAT, אז אפשר בזמן פולינומי למצוא הצבה מספקת לנוסחא.

הסבר: באלגוריתם שהראינו קודם נחליף כל קריאה לאורקל של SAT באלגוריתם הפולינומי של SAT. קיבלנו אלגוריתם שמבצע מספר פולינומי של קריאות לאלגוריתם פולינומי. נשים לב שהקלט לאורקל הולך וגדל אבל נשאר תמיד באורך פולינומי בקלט המקורי φ. ולכן כל הקריאות לאלג' שפותר את SAT לוקחת זמן פולינומי. סה"כ זמן ריצה פולינומי.

טענה. לשפה $3-COL$ יש רדוקציה עצמית.

$$3-COL := \{G : text\ below\}$$

G גרף לא מכוון וקיימת צביעה חוקית ב-3 צבעים לצמתי G. כלומר צביעה כך שעבור כל קשת, 2 קודקודיה בצבעים שונים.

הוכחה. נראה שקיים אלגוריתם פולינומי עם גישה לאורקל של השפה $3-COL$ שמוצא צביעה חוקית ב-3 צבעים לגרף הקלט G.
האלגוריתם:
(א) נבדוק האם $G \in 3-COL$. אם לא נעצור ונחזיר F. אם כן, נוסיף משולש לגרף עם 3 צמתים חדשים (נסמנם R,G,B). נאתחל $i = 1$.
(ב) נחבר את הצומת v_i ל-R, G ונבדוק האם הגרף עדיין 3-צביע. אם לא - ננתק את v_i מהם ונחבר ל-R, B ונבדוק האם הגרף עדיין 3-צביע. אם לא - ננתק את v_i מהם ונחבר ל-G, B.
(ג) אם $i < n$ (מספר הצמתים), נצבע $i++$ ונחזור ל-ב'.
(ד) נסיק צביעה לגרף: לכל צומת v_i ניתן צבע R, G, B לפי הצומת מהמשולש R, G, B אליו הוא לא מחובר.
בכל שלב החיבור של v_i לזוג צמתים במשולש נבחר להיות כזה שמאפשר צביעה חוקית ב-3 צבעים ולכן בשלב **(ד)** הצבעים שנסיק הם צביעה חוקית של הגרף המקורי. מספר צעדי הריצה הוא פולינומי (ליניארי) במספר צמתי גרף הקלט G. $\square$

11 עמרי - שיעור 11 - 14.07.2024

11.1 אקראיות בחישוב (Randomized Complexity) (לא בחומר למבחן)

עד כה, המודל הכללי ביותר של אלגוריתם\חישוב הוא מכונת טיורינג (והתזה של Curch-Turing מצדיקה את המודל).
אספקט אחד פיזיקלי\מציאותי שלא בא לידי ביטוי במכונת טיורינג הוא היכולת של אלגוריתם לבצע בחירות אקראיות.
השאלה היא מהי החשיבות של אקראיות באלגוריתמים\חישוב?

- על מנת לסמלץ תהליכים חישוביים שהם מטבעם אקראיים (למשל, מחיר מנייה, מוטציות גנטיות, ועוד).
- אקראיות עשויה לייעל\להאיץ פתרון לבעיות חישובית ולכל הפחות לפשט אלגוריתמים. (למשל - מציאת חציון ב-Quick Sort בזמן ליניארי, בדיקת ראשוניות).

שאלה פתוחה (שאלת מיליון הדולר של תת-התחום הזה) - האם אקראיות היא הכרחית על מנת לפתור בעיות מסוימות ביעילות? (נפרמל זאת בהמשך).
רוב הקהילה מאמינה שהתשובה היא לא אבל זה לא מוכח. קיימות בעיות\שפות רבות עבורן ידוע אלגוריתם אקראי יעיל (בזמן פולינומי) אבל לא ידוע אלגוריתם דטרמיניסטי יעיל.

דוגמה. בדיקת שוויון של פולינומים (PIT) .

האינפוט זה פולינום מדרגה d עם n משתנים מעל $\mathbb{F}^n$, (כאשר $\mathbb{F}$ שדה סופי) לדוגמא :

$$P(x_1, ..., x_n) = 3x_1x_2^3x_3 - 15x_9^2x_5^4x_2 + 21x_8^3x_4^3$$

אז

$$L_{PIT} = \{ \text{ n var, deg d Polynomes over} \mathbb{F}^n : \exists x \in \mathbb{F}^n, P(x) \neq 0\}$$

אלגוריתם דטרמיניסטי נאיבי שמכריע את השפה : נעבור על כל ההצבות האפשריות ל-$P(x_1, ..., x_n)$. זמן הריצה או $O(|\mathbb{F}|^n)$ וזה אקספוננציאלי.
למעשה לא ידוע אלגוריתם דטרמיניסטי שהוא תת אקספוננציאלי.
אבל קיים אלגוריתם אקראי יעיל לבעיית ה-PIT שמבוסס על הלמה המתמטית הבאה :

למה. (Schwarz-Zippel Lemma) : אם $P(x_1, ..., x_n)$ הוא פולינום מדרגה d מעל שדה סופי $\mathbb{F}$ (עבור $d \leq |\mathbb{F}|$), אז :

$$\underset{x_1,...,x_n \leftarrow \mathbb{F}^n}{Pr} \left(P(x_1, .., x_n) = 0\right) \leq \frac{d}{|\mathbb{F}|}$$

אינטאיציה להוכחה : עבור $n = 1$, כלומר פולינום עם משתנה אחד מדרגה d. אם הפולינום הוא לא זהותית 0, אז יש לפולינום לכל היותר d נקודות בהן הוא 0 (מהמשפט היסודי של האלגברה).
מסקנה מהלמה של SZ היא אלגוריתם יעיל ל-PIT : נניח שאנחנו רוצים הסתברות שגיאה $\varepsilon \leq$ (קטן כרצוננו).
אלגוריתם : בהינתן $P(x)$ מעל $\mathbb{F}^n$, האלגוריתם יגריל $k = O\left(log(\frac{1}{\varepsilon})\right)$ איברים אקראיים (n-יות) באופן אחיד $\overline{x^{(1)}}, ..., \overline{x^{(k)}} \leftarrow \mathbb{F}^n$ ויקבל אמ"ם קיים $j \in [k]$ כך ש-$P\left(\overline{x^{(j)}}\right) \neq 0$.
נכונות : אם $P \notin L_{PIT}$ אז האלגוריתם תמיד ידחה. אם $P \in L_{PIT}$ אז

$$\begin{aligned}
Pr\left(rejects\right) &= Pr\left(\wedge_{j=1}^{k}\left(P\left(\overline{x^{(j)}}\right) = 0\right)\right) \\
&\leq \left(\frac{d}{|\mathbb{F}|}\right)^k \\
\underset{d<\frac{|\mathbb{F}|}{2}}{} &\leq O\left(2^{-k}\right) \\
&\leq 2^{-O\left(log(\frac{1}{\varepsilon})\right)} \\
&= \varepsilon
\end{aligned}$$

זמן ריצה : $Poly(n)$ עבור ε קבוע.

הערה. נשים לב כי ייצוג הפולינומים ה"רגיל" הוא אקספוננציאלי בn, d אז הבעיה היא ליניארית בקלט בשני המקרים. אבל הבעיה עדיין רלוונטית כי קיים ייצוג פולינומים שהוא פולינומי בקלט, מה שנקרא ייצוג מעגל. (Circut).

הערה. גם לשערך את הפולינום לוקח זמן אקספוננציאלי אבל אפשר להתגבר על זה בעזרת Hashing.

הגדרה. **מכונת טיורינג הסתברותית (PTM)** M היא מכונת טיורינג (סטנדרטית) אשר בכל צעד משתמשת בפונקציית מעברים הסתברותית, כלומר:

$$\delta(q,\sigma) \leftarrow S \subseteq 2^{\lambda \times Q \times \{L,R\}}$$

(ההתפלגות לאו דווקא אחידה, אבל מסתבר שכל התפלגות לא אחידה שקולה להתפלגות אחידה ולכן נניח שהיא כן).
ריצה של PTM שקולה להתפלגות על ריצות של מכונות טיורינג דטרמיניסטיות.
זמן ריצה של PTM הוא $worst-case$. כלומר:

$$|M| = max_{R \in randomness} max_{x \in \Sigma^*, |x|=n} |M_R(x)|$$

הגדרה. **מכונת טירוינג PPT** היא מכונת טיורינג הסתברותית M שזמן הריצה שלה פולינומי בקלט. בפרט המכונה יכולה להטיל רק מספר פולינומי של מטבעות.

הגדרה. **המחלקות RP, $coRP$ (ממדלות טעות אקראית חד-צדדית (False Negative\False Positive):**

1. $L \in RP \iff$ קיימת מ"ט PPT, M כך ש-

$$x \in L \implies Pr_R\left[M_R(x) = q_{acc}\right] \geq \frac{1}{2}$$
$$x \notin L \implies Pr_R\left[M_R(x) = q_{rej}\right] = 1$$

כלומר אין False Positive.

2. $L \in coRP \iff$ קיימת מ"ט PPT, M כך ש-

$$x \in L \implies Pr_R\left[M_R(x) = q_{acc}\right] = 1$$
$$x \notin L \implies Pr_R\left[M_R(x) = q_{rej}\right] \geq \frac{1}{2}$$

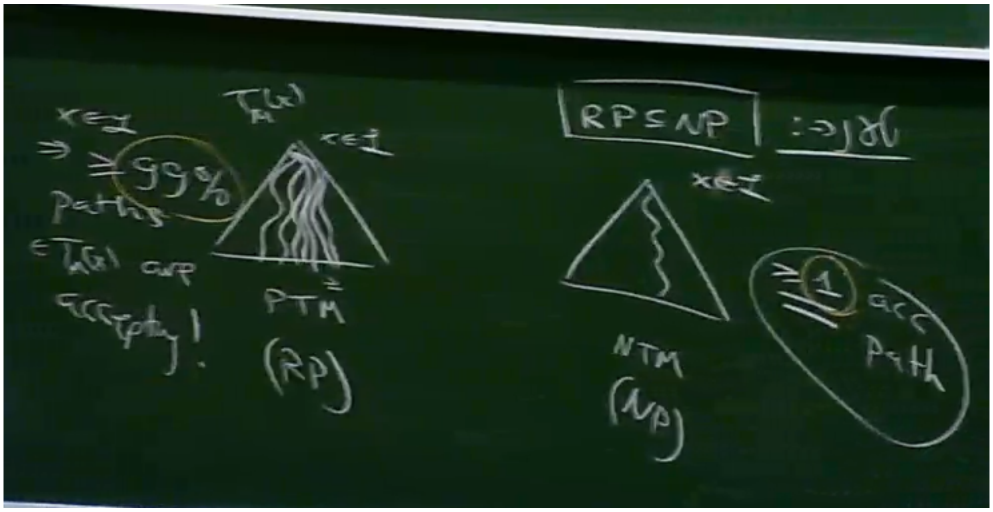
כלומר אין False Negative.

הערה. בה"כ ניתן לחסום את השגיאה של אלגוריתמי $RP/coRP$ ב-ε (במקום $\frac{1}{2}$) כל עוד $\varepsilon \geq \frac{1}{n^c}$ (כי ניתן פשוט לחזור על האלגוריתם המקורי שוב ושוב מספר פולינומי של פעמים (ההרצות בלתי תלויות)):

$$Pr\left[\wedge_{i=1}^{k=n^c} M(x) = q_{acc}\right] = 2^{-k} = 2^{-log(\frac{1}{\varepsilon})} = \varepsilon$$

טענה. $RP \subseteq NP$

הוכחה. נביט בתרשים הבא:



איור 17 :

□

הגדרה. **המחלקה BPP:** Bounded-Error-Probablistic-Polinomial-Time

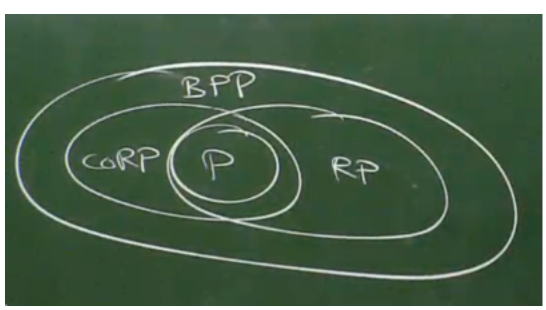
$L \in BPP \iff$ קיימת מ"ט PPT, M כך ש

$$x \in L \implies Pr_R\left[M_R(x) = q_{acc}\right] \geq \frac{2}{3}$$
$$x \notin L \implies Pr_R\left[M_R(x) = q_{rej}\right] \geq \frac{2}{3}$$

הערה. בה"כ ניתן לחסום את השגיאה ב-ε (במקום $\frac{2}{3}$) כל עוד $\varepsilon \geq \frac{1}{n^c}$ (כי ניתן פשוט לחזור על האלגוריתם המקורי שוב ושוב מספר פולינומי של פעמים ולקחת את מה שהרוב קובע) ואפשר להוכיח את זה עם חסם צ'רנוף.

טענה. $RP \cup coRP \subseteq BPP$

לסיכום העולם נראה כך :



איור 18 :

ושאלת מיליון הדולר היא: $P \stackrel{?}{=} BPP$. מאמינים שכן אבל לא ידוע (זו השאלה הכי חשובה בחישוביות הסתברותית).

הגדרה. **המחלקה ZPP:** Zero-Error-Probablistic-Polinomial-Time

$L \in ZPP \iff$ קיימת מ"ט הסתברותית (לאו דווקא PPT), M כך ש

$$x \in L \implies Pr_R\left[M_R(x) = q_{acc}\right] = 1$$
$$x \notin L \implies Pr_R\left[M_R(x) = q_{rej}\right] = 1$$
$$\mathbb{E}_R\left[|M_R(x)|\right] \leq n^{O(1)}$$

כלומר תוחלת זמן הריצה היא פולינומית, ולא ב-Worst Case כמו במחלקות הקודמות. מחלקה זו נקראת גם אלגוריתמי לאס ווגאס.

טענה. $ZPP = RP \cap coRP$