מטלה 2

כל הנחיות המטלה הקודמת תקפות גם כאן, פרט לתאריך ההגשה.

1. קבעו האם שני המודלים הבאים שקולים. הוכיחו את תשובתכם:
   a. מכונת טיורינג דטרמיניסטית לעומת מכונת טיורינג אי דטרמיניסטית בה מילה מתקבלת אך ורק אם קיימים לפחות שני מסלולים מקבלים.
   b. מכונת טיוריניג דטרמיניסטית לעומת מכונת טיורינג אי דטרמיניסטית בה מילה מתקבלת אך ורק אם בדיוק שני מסלולים מקבלים, וכל השאר דוחים/לא עוצרים.

פתרון:

a. המודלים שקולים.

כיוון ראשון- בהינתן מכונה דטרמיניסטית M, להלן מכונה מהמודל החדש M'.

המכונה M' על קלט x:

- מטילה מטבע.
- מריצה את M על x ועונה כמוה.

הטלת מטבע נותנת שני מסלולים, והרצת M על x תענה כמו M עצמה.

כיוון שני- בהינתן מכונה אי דטרמיניסטית M' המקבלת אך ורק אם קיימים לפחות שני מסלולים מקבלים, נבנה מכונה אי-דטרמיניסטית M'', וכבר הוכחנו בהרצאה 4 כי היא שקולה למכונה דטרמיניסטית.

המכונה M'' על קלט x:

- מגרילה שני מסלולים שונים (נגיד, לרשום על שני סרטים מחרוזת בינארית בהתאם לבחירת δ בכל צעד במסלול).
- מריצה את M' על x בשני המסלולים הנ"ל. אם בשניהם הגענו ל q_{acc}, מקבלת. אחרת, דוחה.

נכונות: אם x מתקבלת ב M' אזי קיימים לפחות שני מסלולים מקבלים. לכן, המכונה M'' תנחש שני מסלולים מקבלים, שבשניהם M' מגיעה למצב מקבל, ולכן גם M' תקבל.

אם x מתקבלת ב M', אין שני מסלולים מקבלים. לכן, הניחוש יפיק את אותו מסלול פעמיים, או שיפיק לפחות מסלול אחד שאינו מגיע למצב מקבל, והמכונה M'' תדחה.

b. המודלים שקולים. נוכיח דרך מ"ט א"ד.

כיוון ראשון: תהי מכונה מהמודל החדש M'. נבנה מכונה א"ד רגילה M''.

המכונה M'' על קלט x:

- מגרילה שלושה מסלולים שונים (כל אחד על סרט לבד).
- מנחשת מספר צעדים (כמה צעדים צריך להריץ בכל מסלול- מקסימלי מבין השלושה).
- מריצה את M' על x בשלוש המסלולים הנ"ל (בכמות הצעדים המנוחשת) אם בדיוק בשניים מהם הגענו ל q_{acc}, מקבלת. אחרת, דוחה.

נכונות: אם M' מקבלת בדיוק בשני מסלולים, אזי המסלול השלישי ידחה, ולכן בדיוק בשניים נגיע למצב מקבל. אם M' מקבלת ביותר משני מסלולים, שלושתם ינוחשו ולכן M'' תדחה. אם היא מקבלת במסלול אחד או דוחה, אזי לפחות שניים מהניחושים יובילו ל q_{rej} (או לא יעצרו), ולכן גם M'' תדחה.

כיוון שני: בהינתן מכונה א"ד רגילה M'' נבנה מכונה א"ד מהמודל החדש. המכונה M' על קלט x:

- מריצה את M'' על x. אם דוחה- דוחה.
- מגרילה 2 ביטים: אם התוצאה היא 0 או 1 (בבינארית), מקבלת. אחרת, דוחה.

נכונות: אם $x \in L(M'')$ אזי קיים מסלול מקבל. לכן המכונה לא תדחה בשלב הראשון. בשלב השני, יש בדיוק שתי תוצאות אפשריות עבורן המכונה תקבל, וכל שאר האפשרויות יובילו לדחייה.

אם $x \notin L(M'')$ אזי לא קיים מסלול מקבל, ולכן גם M'' תדחה כבר בשלב הראשון.

2. תהי $L = \{< M > | \exists w_1, w_2 \in \Sigma^* \text{ such that } M \text{ accepts } w_1 \text{ within } |w_2| \text{steps and } M \text{ accepts } w_2 \text{ within } |w_1| \text{ steps}\}$
בנו לשפה L מכונה דטרמיניסטית ומכונה אי-דטרמיניסטית.
למען הסר ספק: "בנו מכונה" הכוונה להראות אלגוריתם. בפרט, אין לציין את כמות המצבים או את פונקציית המעברים.

פתרון:

המכונה N (אי-דטרמיניסטית) על קלט $< M >$:

- מנחשת שתי מילים w_1, w_2. (מילים שונות)
- מריצה את M על w_1 למשך w_2 צעדים. אם דחתה- דוחה.
- מריצה את M על w_2 למשך w_1 צעדים ועונה כמוה.

נכונות: אם $< M > \in L$ אזי קיימות שתי מילים כנ"ל. לכן, המכונה הא"ד מנחשת אותן, לא דוחה בשלב השני ומקבלת בשלב השלישי.
אם $< M > \notin L$ אזי לא קיימות שתי מילים כנ"ל. לכן, המכונה הא"ד דוחה בשלב השני או בשלב השלישי. (הניחוש לא יכול להצליח).

עבור מכונה דטרמיניסטית, "נרמה" ונשתמש במכונה שיצרנו בהוכחת השקילות מהרצאה 4.

3. להלן כמה שפות. לכל שפה קבעו האם היא כריעה (R), מזוהה טיורינג (RE), או שמשלימתה מזוהה טיוריניג ($coRE$).

א. $L = \{< M > | L(M) = \phi\}$.
ב. $L = \{< M > | L(M) \text{ is a CFL and not regular}\}$.
ג. $L = \{< M_1 >, < M_2 > | M_1 \text{ and } M_2 \text{ are TM and } L(M_1) \cap L(M_2) \neq \phi\}$
ד. יהי $n > 0$ טבעי כלשהו. תהי $L_n = \{< M_1, M_2, \dots M_n > | M_1, M_2, \dots, M_n \text{are TM such that no word of length smaller than } n \text{ is in } L(M_1) \cap L(M_2) \dots L(M_n)\}$
במילים- בהינתן n, זוהי שפת קידודי n מכונות שאין מילה בחיתוך שלהן שאורכה קטן מ-n.

פתרון:

א. השפה שייכת ל $coRE \setminus R$.
נבנה מכונה דוחה לשפה:
עבור $i = 1, \dots$
עבור $j = 1, \dots, i$
הרץ את המילה w_j למשך i צעדים. אם התקבלה – דחה.
נכונות: אם $< M > \in L$, המכונה לא תעצור (צריך לעבור על כל Σ^*).
אם $< M > \notin L$, אזי קיימת מילה שתתקבל ב M. במילה הזו, המכונה תדחה.

יש גם להוכיח כי $L \notin R$. ניעזר ברדוקציה מ $\overline{HP}$.
$$f(< M >, < x >) = (< M_x >)$$
המכונה M_x על קלט y:
- הרץ את M על x.
- קבל.

הרדוקציה מוגדרת לכל קלט, וניתנת לחישוב (פעולת קומפילציה פשוטה. בפרט, אין כאן הרצה של מכונה שעלולה לא לעצור). נראה כי הרדוקציה תקפה:
$$(< M >, < x >) \in \overline{HP} \rightarrow M \text{ doesn't stops on } x \rightarrow L(M_x) = \phi \rightarrow M_x \in L$$

$$(< M >, < x >) \notin \overline{HP} \rightarrow M\ stops\ on\ x \rightarrow L(M_x) = \Sigma^* \rightarrow M_x \notin L$$

ב. נראה כי $L \in \overline{RE \cup coRE}$ בשלושה שלבים (השלב הראשון לא נחוץ. הוא פה נטו לתרגול)

I. נראה כי $L \notin R$ בעזרת משפט רייס!

נגדיר את התכונה $S = \{L \in RE: L\ is\ a\ CFL\ but\ not\ Regular\}$. זאת תכונה של שפה ומתקיים $L = L_S$. נוכיח שהתכונה אינה טריוויאלית:

$S \neq RE$ כי $\phi \notin S$ (השפה הריקה רגולרית). לעומת זאת, $S \neq \phi$ כי $\{a^n b^n\} \in S$ כי זו שפה חסרת הקשר שאינה רגולרית.

לכן, לפי משפט רייס $L \notin R$.

II. נראה כי $L \notin RE$ בעזרת רדוקציה מ $\overline{L_u}$:

$$f(< M >, < x >) = < M_x >$$

המכונה M_x על קלט y:

- אם y מהצורה $0^n 1^n$ עבור n טבעי כלשהו, קבל.
- הרץ את M על x וענה כמוה.

הרדוקציה מוגדרת לכל קלט וניתנת לחישוב (בפרט, היא לא כוללת הרצת מכונה שעלולה לא לעצור). נוכיח שהרדוקציה תקפה:

$$(< M >, < x >) \in \overline{L_u} \rightarrow M\ rejects\ x\ (or\ doesn't\ stops)$$
$$\rightarrow L(M_x) = \{0^n 1^n\}, CFL\ and\ not\ regular \rightarrow < M_x > \in L$$
$$(< M >, < x >) \in L_u \rightarrow M\ accepts\ x \rightarrow L(M_x) = \Sigma^* \rightarrow < M_x > \notin L$$

III. נראה כי $L \notin coRE$ בעזרת רדוקציה מ L_u:

$$f(< M >, < x >) = < M_x >$$

המכונה M_x על קלט y: (מכונה א"ד הפעם)

- אם y מהצורה $0^n 1^n$ עבור n טבעי כלשהו, קבל.
- נחש מספר k.
- הרץ את M על x למשך k צעדים וענה הפוך. אם לא עצרה – קבל.

הרדוקציה מוגדרת לכל קלט וניתנת לחישוב (בפרט, היא לא כוללת הרצת מכונה שעלולה לא לעצור). נוכיח שהרדוקציה תקפה:

$$(< M >, < x >) \in L_u \rightarrow M\ accepts\ x \rightarrow L(M_x) = \{0^n 1^n\} \rightarrow < M_x > \in L$$
$$(< M >, < x >) \notin L_u \rightarrow M\ rejects\ x\ (or\ doesn't\ stops\ for\ any\ k)$$
$$\rightarrow L(M_x) = \Sigma^* \rightarrow < M_x > \notin L$$

(קיים k' מספר צעדים שבו המכונה עוצרת, והניחוש תופס אותו. ואם המכונה לא עוצרת, לכל ניחוש היא לא תעצור, ואז המכונה M_x תקבל כל y)

שני החלקים האחרונים מוכיחים את הטענה הרשומה בראש התרגיל.

ג. נראה כי $L \in RE \setminus R$:

תחילה נבנה מכונה מקבלת לשפה.

המכונה M על הקלט $< M_1 >, < M_2 >$

עבור $i = 1, \ldots$

עבור $j = 1, \ldots, i$

הרץ את המילה w_j למשך i צעדים ב-M_1 וגם ב-M_2. אם שתיהן קיבלו- קבל.

נכונות: $(< M_1 >, < M_2 >) \in L$ ולכן קיימת מילה w_m שמתקבלת בשתיהן. מילה זו דורשת k צעדים, ולכן באיטרציה $\max(m, k)$ המכונה תקבל.

$(< M_1 >, < M_2 >) \notin L$ ולכן האלגוריתם ירוץ לנצח. לא מפריע ב RE.

שנית, נוכיח כי $L \notin R$ ברדוקציה מ HP.

$$f(< M >, < x >) \rightarrow (< M_x >, < M_x >)$$

המכונה $< M_x >$ על קלט y:
- מריצה את M על x.
- מקבלת.

הרדוקציה מוגדרת לכל קלט וניתנת לחישוב (בפרט, אין כאן הרצה של מכונה שעלולה לא לעצור). נוכיח כי הרדוקציה תקפה:

$$(< M >, < x >) \in HP \to M\ halts\ on\ x \to L(< M_x >) = \Sigma^*$$
$$\to L(M_x) \cap L(M_x) = \Sigma^* \neq \phi \to (< M_x >, < M_x >) \in L$$

$$(< M >, < x >) \notin HP \to M\ doesn't\ halts\ on\ x \to L(< M_x >) = \phi$$
$$\to L(M_x) \cap L(M_x) = \phi \to (< M_x >, < M_x >) \notin L$$

ד. $L_n \in coRE \setminus R$
תחילה נראה רדוקציה מ $\overline{L_u}$:

$$f(< M >, < w >) = < M', M', \dots, M' >$$

(יש כאן n מכונות, וכולן זהות).
המכונה M' על הקלט y:
- הרץ את M על w וענה כמוה.

הרדוקציה מוגדרת לכל קלט וניתנת לחישוב. תקפות:

$$(< M >, < w >) \in \overline{L_u} \to M\ rejects\ w\ (or\ doesn't\ halts) \to L(M') = \phi$$
$$\to L(M') \cap L(M') \cap \dots \cap L(M') = \phi \to (< M', M', \dots, M' >) \in L_n$$

$$(< M >, < w >) \notin \overline{L_u} \to M\ accepts\ w \to L(M') = \Sigma^*$$
$$\to L(M') \cap L(M') \cap \dots \cap L(M') = \Sigma^* \to \exists x\ s.t\ |x| < n \wedge x \in L(M') \cap L(M') \dots \cap L(M')$$
$$\to (< M' >, < M' >, < M' >, \dots < M' >) \notin L_n$$

כעת נראה כי $L \in coRE$ על ידי בניית מכונה דוחה.
עבור $i = 1, \dots$
    עבור $j = 1, \dots, |\Sigma^{n-1}|$
        אתחל מונה $c = 0$.
        עבור $k = 1, \dots, n$
            הרץ את המילה ה-j למשך i צעדים על המכונה M_k. אם קיבלה $c++$.
        אם $c = n$ – דחה.

נכונות: אם $(< M_1 >, < M_2 >, \dots, < M_n >) \notin L$ אזי קיימת מילה באורך חסום ב $n-1$ שמתקבלת בכל המכונות, ולכן נמצא אותה בהרצאה המבוקרת לעיל ונדחה.
אם $(< M_1 >, < M_2 >, \dots, < M_n >) \in L$ אזי אין מילה באורך זה בשפת כל המכונות. לכן ההרצה המבוקרת לא תעצור (i רץ לאינסוף). תקין עבור $coRE$.

4. בנו מכונה דטרמיניסטית לשפה משאלה 3א, ומכונה אי דטרמיניסטית לאותה שפה. שימו לב שבשאלה זו יש לבנות מכונה לפי הסיווג שלכם בשאלה 3. כלומר, המכונה מכריעה/מזהה/דוחה בהתאם לתשובה שלכם בשאלה 3.

פתרון:
מכונה דטרמיניסטית לשפה, כבר בנינו בשאלה 3א. עיינו שם.
מכונה אי-דטרמיניסטית:
המכונה N על קלט $< M >$:
- מנחשת מילה w.
- מריצה את M על w ועונה הפוך.

נכונות: אם $< M >\in L$ אזי לכל ניחוש המילה אינה בשפה. לכן N תקבל את $< M >$.
אם $< M >\notin L$ אזי קיימת מילה בשפה של M והניחוש ימצא אותה. ואז N תדחה את $< M >$.

5. סווגו כל אחת מהשפות הבאות למחלקות. אם ניתן, היעזרו במשפט רייס. אם לא ניתן, הסבירו מדוע לא ניתן, וגם הוכיחו בצורה אחרת.
א. $L_1 = \{< M >|\ M\ stops\ to\ every\ input\ after\ more\ than\ 42\ steps\}$
ב. $L_2 = \{< M >|\ 10 \le |L(M)| \le 1000\}$
ג. $L_3 = \{< M >| \exists w \in L(M)\ such\ that\ M\ accepts\ w\ in\ less\ than\ |w| steps\}$
ד. $L_4 = \{< M >| M\ is\ a\ TM\ that\ halts\ on\ at\ most\ one\ input\}$

פתרון:

א. לא, תכונה של המכונה. לא שייכת לאף מחלקה מ-$R, coRE, RE$

ראינו את השפה הבאה, והוכחנו שהיא ב-$\overline{RE \cup coRE}$

$$L_{\Sigma^*} \coloneqq \{< M >: |L(M)| = \Sigma^*\}$$

הרדוקציה $L_{\Sigma^*} \le L_1$ תהיה באופן הבא:

$$f(< M >) =< U >$$

<u>$U(y)$:</u>

1. בצע 42 פעולות במקום.
2. סמלץ את M על y - אם דחה כנס ללולאה אינסופית.

הרדוקציה מלאה כיוון שהיא מוגדרת לכל קידוד מכונה M שנקבל. ניתנת לחישוב כיוון שניתן לקודד כל מכונה, בפרט את U שהגדרנו לעיל.
נרצה כעת להוכיח שהרדוקציה תקפה:

אם $< M >\in L_{\Sigma^*}$ אז M מקבלת כל קלט. אם כך, עבור כל קלט y נבצע תחילה 42 פעולות במקום, נסמלץ את M על y ונקבל. סה"כ עבור כל מילה y המכונה U תעצור אחרי לפחות 42 פעולות. לכן $< U >\in L_1$

אחרת, אם $< M >\notin L_{\Sigma^*}$ אז תהיה לפחות מילה אחת ש-M לא תקבל. אם כך, קיימת מילה 'y כלשהי שהמכונה M לא עוצרת עליה או שהמכונה M דוחה. בשלב מסויים הקלט הזה יתקבל, עבורו נבצע תחילה 42 פעולות המקום, נסמלץ את M על 'y – אם המכונה M לא עוצרת גם U לא עוצרת ולכן ישנו קלט שעבורו המכונה U לא עוצרת עבורו תוך מספר צעדים גדול גדול מ-42.
אחרת, אם המכונה M תעצור ותדחה, U תיכנס ללולאה אינסופית ושוב תהיה לנו מילה שהמכונה U לא תעצור עליה בכלל ובפרט אחרי יותר מ-42 צעדים.
לכן בכל מקרה $< U >\notin L_1$

סה"כ שפה זו לא שייכת לאף אחת מ-3 המחלקות.

ב. תכונה של השפה ולכן לאחר שנראה שהתכונה איננה טריוויאלית אפשר יהיה להשתמש ברייס.
שפה שלא מקיימת את התכונה- השפה הריקה

שפה המקיימת את התכונה- $L' = \{1\{0\}^k : 1 \le k \le 1000\}$

נשתמש ברייס המורחב ונקבל קודם כל, שכיוון שהשפה הריקה לא מקיימת את התכונה, השפה $L_2 \notin coRE$

כדי להראות שהשפה לא ב-RE נבצע את הרדוקציה הבאה $\overline{HP} \le L_2$

$$f(<M,x>) =< M_x >$$

$M_x(y)$:

1. אם המילה y מהצורה $1\{0\}^k$ כאשר $1 \le k \le 1000$ – קבל
2. אחרת, סמלץ את M על x
3. קבל

הרדוקציה מלאה כי לכל מכונה M ומילה x ניתן לייצר את המכונה M_x , ניתנת לחישוב כי כל מכונה ניתנת לקידוד, בפרט M_x.

נוכיח תקפות:

אם $< M,x >\in \overline{HP}$ – אז M לא עוצרת על x , אם כך המילים היחידות שיתקבלו הן המילים בשפה L' (שהגדרנו כשפה המקיימת את התכונה). עבור יתר המילים נסמלץ את M על x, לא נעצור לעולם ולכן הן לא יתקיימו, סה"כ המכונה M_x עונה על תנאי השייכות לשפה L_2

אחרת, אם $< M,x >\notin \overline{HP}$ אז M עוצרת על x. אם כך, עבור המילים ב-L' המכונה M_x תקבל אותן, ועבור יתר המילים, נסמלץ את M על x הסמלוץ יסתיים והקליט יתקבל, סה"כ כל המילים יתקבלו, כלומר $L(M_x) = \Sigma^*$ ולכן המכונה לא תענה על תנאי השייכות לשפה L_2

ג. לא, תכונה של המכונה. תחילה נראה שייכות ל-RE ע"י מכונה מקבלת, לאחר מכן נראה אי שייכות ל-coRE ע"י הצגת רדוקציה מ-HP.

מכונה מקבלת:

$U(< M >)$:

1. נחש מילה w כלשהי.
2. סמלץ את M על w למשך |w| צעדים- אם עצרה לפני וקיבלה, קבל, אחרת- דחה.

נרצה להוכיח את נכונות המכונה (שימו לב אם לא הוכחנו נכונות- לא הוכחנו שיש באמת מכונה מקבלת לשפה).

אם $< M >\in L_3$ אז ישנה מילה w ש-M מקבלת תוך פחות מ-|w| צעדים, באחד ממסלולי החישוב המכונה U תנחש את מילה זו, והמילה תתקבל בשלב השני. סה"כ יהיה לנו לפחות מסלול חישוב אחד מקבל ולכן $< M >\in L(U)$

אחרת, אם $< M >\notin L_3$ אז לכל מילה w ב-Σ^* המכונה M לא תעצור תוך פחות מ-|w| צעדים. לכן, כל מילה שהמכונה U תנחש לא תתקבל על ידי M תוך |w| צעדים ותדחה, סה"כ כל מסלולי החישוב של U על <M> יהיו דוחים ולכן $< M >\notin L(U)$

כעת נראה ש-$L_3 \notin coRE$ ע"י הרדוקציה הבאה $HP \leq L_3$

$$f(< M, x >) = < M_x >$$

<u>$M_x(y)$:</u>

1. סמלץ את M על x.
2. קבל.

הרדוקציה מלאה כי לכל מכונה M ומילה x ניתן לייצר את המכונה M_x , ניתנת לחישוב כי כל מכונה ניתנת לקידוד, בפרט M_x.

נוכיח תקפות:

אם $< M, x >\in HP$ אז M עוצרת על x אחרת מספר סופי של צעדים, יהי זה k. לפי בניית המכונה M_x M תמיד תסיים את הריצה ולכן כל המילים יתקבלו ופרט אלו שבאורך k> לכן,
$< M_x >\in L_3$

אחרת, אם $< M, x >\notin HP$ אז M לא עוצרת על x, ולכן לכל מילת קלט M_x לא תסיים את שלב (1), ולכן אף מילת קלט לא תתקבל, ובפרט לא תתקבל אחרי מספר צעדים מותר. סה"כ $< M_x >\notin L_3$

ד. לא, מדבר על עצירה של המכונה ולא על תכונה. השפה נמצאת ב-coRE ולא ב-RE שייכות ל-coRE נראה ע"י מכונה דוחה. אי שייכות ל-RE נראה על ידי רדוקציה.

מכונה דוחה:

<u>$U(< M >)$</u>:

1. סמלץ בהרצה מבוקרת חזקה (מצגת 3 שקף 20) את כל המילים ב-Σ^* לפי סדר לקסיקוגרפי.
2. כל פעם שנגיע אל i חדש נאתחל את ה-counter ל-0.
3. כל מילה שנקבל נעלה את ה-counter ב-1.
4. אם הגענו ל-2 נדחה.

<u>נכונות:</u>

אם $< M >\in L_4$ אז M עוצרת על לכל היותר מילה אחת, ולכן המונה לא יעבור את המספר 1 ולא יעצור לעולם.

אחרת, אם $< M >\notin L_4$ אז M עוצרת אחרי לפחות שני קלטים יהיו אלו w_n, w_m עבורם נעצור לאחר k_n, k_m צעדים בהתאמה. לכן, כאשר $i = \max\{n, m, k_n, k_m\}$ נמסלץ את שתי המילים האלו לפחות מספר צעדים שיספיק כדי לקבלן, המונה יעלה על 2 והמכונה תדחה.

נראה כעת את הרדוקציה $\overline{HP} \leq L_4$:

$$f(< M, x >) = < M_x >$$

<u>$M_x(y)$:</u>

1. אם $y = \epsilon$ קבל.

2. אחרת, סמלץ את M על x.

3. קבל.

הרדוקציה מלאה כי לכל מכונה M ומילה x ניתן לייצר את המכונה M_x , ניתנת לחישוב כי כל מכונה ניתנת לקידוד, בפרט M_x.

נוכיח תקפות:

אם $< M, x > \in \overline{HP}$ אז M לא עוצרת על x, ולכן לכל מילת קלט מלבד מילה אחת, M_x לא תסיים את שלב (2), ולכן רק מילה אחת תתקבל וסה"כ $< M_x > \in L_4$

אם $< M, x > \notin \overline{HP}$ אז M עוצרת על x אחרת מספר סופי של צעדים, לפי בניית המכונה M M_x תמיד תסיים את הריצה ולכן כל המילים יתקבלו ופרט על יותר מ-2 מילים המכונה M_x תעצור ולכן $< M_x > \notin L_4$

6. תהיינה L_1, L_2 שפות המקיימות $L_1 \in RE,\ L_2 \in coRE$. בנוסף, נתון כי $L_2 \leq L_1$. לכל טענה כתבו האם היא נכונה תמיד, שגויה תמיד, או שיש שפות עבורן הטענה נכונה ויש שפות עבורן הטענה שגויה (במקרה זה, יש לציין דוגמה לכל צד).
א. L_2 היא $coRE - complete$.
ב. $L_1 \in R$.
ג. $L_2 \notin R$.

פתרון:

א. תמיד לא נכון. כדי שתהיה רדוקציה כזו, L_1 צריכה להיות שפה קשה ב-coRE. אין כאלו ב-RE. למעשה, היות ו-$L_2 \leq L_1$, מתקיים בהכרח כי $L_2 \in R$. (אין שפה ב $coRE \setminus R$ שניתנת לרדוקציה לשפה ב RE).

ב. יש שפות המקיימות את הטענה, ויש שפות שלא.
לדוגמה: עבור $L_2 = \{0\}$ ו- $L_1 = \{0\}$ שתי השפות ב R, ואכן יש רדוקציה $\{0\} \leq \{0\}$ (רדוקציית הזהות)
עבור $L_2 = \{0\}$ ו-$L_1 = HP$, יש רדוקציה $\{0\} \leq HP$. (משפט: כל שפה לא טריוויאלית ב R ניתנת לרדוקציה לכל שפה אחרת) אבל $L_1 = HP \notin R$.

ג. תמיד לא נכון. כפי שהראנו בסעיף א', $L_2 \in R$.

7. נניח כי $P \neq NP$. ליד כל אחת מהשפות הבאות, קבעו האם השפה שייכת ל-P או לא ידוע.
א. $L_1 = \{< G >| G\ is\ a\ graph\ that\ containes\ a\ n-5\ clique\}$ (כל הגרפים שיש בהם קליקה בגודל $n-5$ קודקודים).
ב. $L_2 = \{< G >\ | G\ is\ 3-colorable\ graph,$
$except\ for\ at\ most\ 2\ vertices\ that\ requires\ a\ forth\ color\}$
ג. $L_3 = \{< G >| G\ is\ a\ graph\ that\ containes\ an\ Euler\ path\}$
ד. $L_4 = \{< G >| G\ is\ a\ graph\ that\ has\ three\ foreign\ domination\ sets\}$
(תזכורת: קבוצת קודקודים $S \subset V$ תיקרא "קבוצה שולטת" אם $\forall v \notin S\ \exists u \in S: uv \in E(G)$. קבוצות קודקודים S_1, S_2 ייקראו זרות אם $S_1 \cap S_2 = \phi$)

פתרון:

א. שייכת ל-P. נשים לב שיש $\theta(n^5) = \binom{n}{5} = \binom{n}{n-5}$ קבוצות שצריך לבדוק ולכן, אם נצליח לבצע בדיקה עבור כל קבוצה בזמן פולינומי- נמצא אלגוריתם פולינומי.

$M(< G >)$:

1. תעבור על כל הקבוצות בגודל $n-5$ בגרף
   1.1. עבור כל קבוצה תעבור על כל שני קודקודים בה
      1.1.1 אם אין צלע בניהם- עבור לקבוצה הבאה
   1.2 אם כל הצלעות קיימות- קבל
2. אם אף קבוצה לא מהווה קליקה- דחה.

נבצע עבור כל קבוצה שנבדקת $O(n^2)$ פעולות, סה"כ יהיה לנו $O(n^5 \cdot n^2) = O(n^7)$ פעולות- שזה בזמן פולינומי.

נכונות:

אם $< G > \in L_1$ אז יש ב-G קבוצת קודקודים בגודל $n-5$ המהווה קליקה בגרף, בשלב מסויים כשנעבור על כל הקבוצות בגודל זה נגיע אל הקבוצה הנכונה- נבדוק ונראה שבאמת בין כל 2 קודקודים בקבוצה יש צלע, ולכן בשלב (1.2) נקבל.

אחרת, אם $< G > \notin L_1$ אז אין ב-G קבוצת קודקודים בגודל $n-5$ המהווה קליקה בגרף, לכן כשנעבור על כל הקבוצות בגודל זה תמיד יהיו זוג קודקודים שאין בניהם צלע- עבור כל קבוצה שנבדוק לא נקבל, ולכן בשלב (2) נדחה.

ב. השפה נמצאת ב-NPH, ולכן איננה ב-P.
נוכיח זאת ע"י רדוקציה משפה שהיא NP שלמה אל L_2.
הרדוקציה: $3-col \leq_p L_2$

$$f(< G >) = < G' >$$

כאשר 'G מוגדר באופן הבא: נוסיף ל-G עוד שני קודקודים ונחבר לכל יתר קודקודי הגרף מלבד זה לזה.
הרדוקציה פולינומית משום שהוספת 2 קודקודים ו- 2n צלעות מצריכה זמן לינארי.
הרדוקציה מלאה כיוון שהיא מוגדרת לכל גרף G וניתנת לחישוב מפני שאת הגרף 'G ניתן לקודד.
תקפות:
אם $< G > \in 3-col$ אז עבור G ישנה צביעה חוקית עם 3 צבעים בלבד, נוכל להשתמש בצביעה זו עבור 'G ולצבוע את שני הקודקודים החדשים בצבע חדש. סה"כ הצביעה כולה תהיה חוקית משום שהצביעה המקורית הייתה חוקית ומשום שלקודקודים הצבועים בצבע הרביעי אין שכנים בצבע זה. לכן, $< G' > \in L_2$
אם $< G > \notin 3-col$ אז נרצה להוכיח שגם $< G' > \notin L_2$. נניח בשלילה ש-$< G' > \in L_2$
אם כך, במקרה הקשה יותר הוא אם לשני הקודקודים החדשים אותו הצבע (אם לשני הקודקודים החדשים היו צבעים שונים, היו יותר הגבלות על צביעת הקודקודים המקוריים).
נניח שלשני הקודקודים החדשים אותו הצבע- כיוון שהצביעה חוקית וכל אחד מהם מחובר לכל קודקוד מקורי מ-G, קודקודי G יוכלו להיצבע רק ב-3 צבעים. מאחר והצביעה על 'G חוקית, נוכל לקחת ממנה את הצביעה החוקית ל-G שתכיל באמת לכל היותר 3 צבעים.

היא תכיל 3 צבעים לכל היותר כי אם היו יותר, גם הצביעה של 'G כולה הייתה מצריכה יותר מ-4 צבעים שכן שני הקודקודים החדשים מחוברים לכל קודקוד מקורי.

במידה ולשני הקודקודים החדשים צבעים שונים, קודקודי הגרף המקורי היו יכולים להיצבע רק בשני צבעים, מה שהיה אומר ש-G הוא 2-צביע ולכן גם 3-צביע.

סה"כ נקבל סתירה לכך ש- $< G >\notin 3-col$

ג. כפי שלמדנו במבנים דיסקרטים, כל שצריך לבדוק הוא את דרגות הקודקודים, וקשירות של הגרף. $EulerPath \in P$.
להלן אלגוריתם דטרמיניסטי פולינומי:
- הרץ BFS לוודא כי הגרף קשיר.
- אתחל מונה $c=0$.
- עבור $i=1,\dots,|V(G)|$
  - אם $\deg(v_i)\equiv 1\ (mod\ 2)$ בצע $c++$.
- אם $c>2$ החזר "אין מסלול אוילר". אחרת החזר "יש מסלול אוילר".

האלגוריתם פולינומי (BFS רץ ב $O(|V|+|E|)$) ובדיקת הדרגות חסומה ב $O(n^2)$.

ד. השפה אינה ב-P (כל עוד $P\neq NP$). נציג אלגוריתם א"ד פולינומי:
המכונה N על הקלט $< G >$:
- מנחשת שלוש קבוצות של קודקודים S_1,S_2,S_3.
- לכל קבוצה S_i מוודאת כי שכני כל הקודקודים ב S_i מכסים את כל הקודקודים $V(G)\setminus S_i$. אחרת דוחה.
- לכל $i\neq j$ מוודאת כי $S_i\cap S_j=\phi$. אחרת, דוחה.
- מקבלת.

8. הוכיחו כי השפות הבאות שייכות ל-NP. לשתי שפות הוכיחו בעזרת מכונה אי-דטרמניסטית, ולשתי שפות הוכיחו בעזרת יחס R_L.

א. $$Subset-Sum=\{(S,k)|S\ is\ a\ set\ of\ numbers.\ \exists S'\subseteq S\ such\ that\ \sum_{s\in S'} s=k\}$$

ב. $$Partition=\{(S)|S\ is\ a\ set\ of\ numbers.\ \exists S'\subset S\ such\ that\ \sum_{\{s\in S'\}} s=\sum_{\{s\in S\setminus S'\}} s\}$$

ג. $$Clique\&IS=\{< G,k_1,k_2 >|G\ is\ a\ graph\ with\ clique\ in\ size\ k_1\ and\ IS\ in\ size\ k_2\}$$

ד. $$CliqueK=\{< G,k_1,k_2 >|G\ is\ a\ graph\ that\ has\ a\ clique\ in\ size\ k,\ and\ k_1\leq k\leq k_2\}$$

(יש עמוד נוסף. אל דאגה, שם נמצאת השאלה הכי קלה של המטלה הארוכה והמייגעת הזו)

פתרון:

א. בעזרת מכונה אי דטרמיניסטית
M(<S,k>)
1. עבור כל איבר ב-S נחש האם הוא נכנס ל-'S או לא.
2. סכום את איברי 'S והשווה ל-k – אם שווים קבל, אחרת דחה.

<u>זמן ריצה</u>: המכונה עובדת בזמן פולינומי משום שגודל |S| ניחושים הוא פולינומי בגודל הקידוד, סכימת O(S) מספרים הוא פולינומי בגודל הקידוד, וגם השוואת שני מספרים (פולינומי בפרט בגודל |<k>|).

<u>נכונות:</u>

אם המילה שייכת לשפה אז יש תת קבוצה $S' \subseteq S$ כך שסכום איברי הקבוצה שווה ל-k

באחד ממסלולי החישוב תנוחש תת הקבוצה הזו, נסכום אותה, נשווה אותה ל-k ונקבל,

לכן, $< S, k > \in L(M)$

אם המילה לא שייכת לשפה אז לכל תת קבוצה $S' \subseteq S$ הסכום שלה יהיה שונה מ-k ולכן כל מסלול יידחה, לכן $< S, k > \notin L(M)$

בעזרת מכונה המוודאת יחס:

<u>סיבוכיות העד:</u> העד y יהיה מחרוזת בינארית באורך |S| שתייצג מי שייך לקבוצה 'S, כך שהתו ה-i במחרוזת יהיה 1 אמ"מ האיבר ה-i בקבוצה שייך ל-'S.

העד לינארי בגודל הקבוצה S ולכן פולינומי בגודל קידוד המילה כולה.

V(<S,k>,y)

1. סכום את כל איברי הקבוצה 'S
2. השווה את הסכום ל-k אם שווים קבל, אחרת דחה.

<u>זמן ריצה:</u> סכימת |S| איברים והשוואה ל-k יצריך מספר פולינומי של פעולות באורך הקלט (שכן הוא קידוד של |S| איברים)

<u>נכונות:</u>

אם המילה שייכת לשפה, אז ישנה קבוצה 'S כלשהי שסכומה שווה ל-k ולכן קיים עד y המייצג את איבריה. יחד עם עד זה, $(< S, k >, y) \in R_{Subset-Sum}$

אם המילה לא שייכת לשפה, לכל תת קבוצה 'S כזו, סכומה יהיה שונה מ-k ולכן לכל עד שייצג איזושהי תת קבוצה, הסכום שלה יהיה שונה מ-k ולכן כל עד לא מוכיח את שייכות המילה לשפה, ולכל עד המכונה V תדחה. סה"כ לעל עד פולינומי נקבל

$(< S, k >, y) \notin R_{Subset-Sum}$

ב. נוכיח כי $Partition \in NP$ בעזרת מכונה א"ד.
המכונה M על הקלט S:
- תנחש תת קבוצה של מספרים $T \subseteq S$.
- תבדוק האם $\sum_{x \in T} x = \sum_{x \in S \setminus T} x$. אם כן תקבל, אחרת תדחה.

המכונה פולינומית: ניחוש תת קבוצה של מספרים (אפשר לנחש את המספרים עצמם או את האינדקסים שלהם, זה גם יספיק כדי לא להסתבך עם ייצוג עשרוני/בינארי/הקסה-דצימאלי) חסום בכמות המספרים בקלט, וסכימת איברי שתי הקבוצות מתבצעת בזמן פולינומי בכמות המספרים בקבוצה.

נכונות:

אם $S \in Partition$ אזי קיימת חלוקה לשתי תתי קבוצות שסכומן זהה. לכן, המכונה M תנחש את אחת הקבוצות, ותזהה שסכומה זהה לסכום הקבוצה המשלימה.

אם $S \notin Partition$ אזי כל חלוקה לשתי קבוצות תגרום לכך שיהיה הפרש בין הסכומים. לכן, כל ניחוש של תת קבוצה לא יפיק סכום זהה לתת הקבוצה המשלימה, והמכונה תדחה.

ג. בעזרת מכונה א"ד

$M(< G, k_1, k_2 >)$:

1. נחש קבוצת קודקודים בגודל k_1
2. בדוק האם היא קליקה- אם לא, דחה
3. נחש קבוצת קודקודים בגודל k_2
4. בדוק האם היא בת"ל- אם לא, דחה
5. קבל

<u>זמן ריצה:</u> המכונה עובדת בזמן פולינומי משום שסיבוכיות זמן ניחוש הוא כגודל הניחוש.
במקרה של בדיקה האם קבוצה קודקודים היא קליקה או בל"ת, יש לעבור על $O(\binom{k}{2})$ קודקודים,
כאשר $k \coloneqq \{k_1, k_2\}$ ו-n הוא מספר הקודקודים בגרף.
סה"כ נקבל זמן ריצה פולינומי.

<u>נכונות:</u>
אם מילה שייכת לשפה, אז המילה עונה על תנאי השייכות לשפה, כלומר ב-G לפחות אחד מהשניים הבאים יקרה:

1. יש קבוצת קודקודים S_1 בגודל k_1 המהווה קליקה בגרף
2. יש קבוצת קודקודים S_2 בגודל k_2 המהווה קבוצה בת"ל בגרף

במקרה (1) באחד ממסלולי החישוב תנוחש הקבוצה S_1 , נעבור על כל זוגות הקודקודים בה, נראה שבאמת יש צלע בין כל זוג, הקבוצה אכן מהווה קליקה, לכן יש לפחות מסלול אחד מקבל.
במקרה (2), אם נוחשה קבוצה S_1 שמהווה קליקה- המילה כבר התקבלה בשלב השני במכונה, אחרת, אם נוחשה קבוצה S_1 שאינה קליקה, באחד ממסלולי החישוב תנוחש הקבוצה S_2 , נעבור על כל זוגות הקודקודים בה, נראה שבאמת אין צלע בין כל זוג, הקבוצה אכן בת"ל, לכן יש לפחות מסלול אחד מקבל.
סה"כ, כך או כך $< G, k >\in L(M)$

אם המילה לא בשפה, אז גם אין בגרף G קליקה בגודל k_1 וגם אין בגרף קבוצה בת"ל בגודל k_2, לכן בכל מסלול חישוב, תמיד תנוחש קבוצת קודקודים בגודל k_1, תמיד יהיה זוג קודקודים שאין בניהם צלע, לכן תמיד נגיע לשלב השלישי, לכל קבוצת קודקודים בגודל k_2 תמיד יהיה זוג קודקודים שתהיה בניהם צלע ולכן כל המסלולים יהיו דוחים. סה"כ
$< G, k >\notin L(M)$

נוכיח כעת בעזרת מכונה המוודאת יחס:
<u>סיבוכיות העד:</u> העד y יהיה מחרוזת בינארית באורך $2|V(G)|$, כך שבחצי הראשון של העד, האינדקס ה-i יהיה 1 אמ"מ הקודקוד ה-i הוא חלק מהקליקה, ובחצי השני של העד, האינדקס ה-$i + |V(G)|$ יהיה 1 אמ"מ הקודקוד ה-i הוא חלק מהקבוצה הב"ת.

העד לינארי במספר הקודקודים בגרף ולכן בהכרח פולינומי בגודל קידוד הגרף, ולכן גם בגודל המילה (גודל קידוד הגרף הוא לכל היותר גודל קידוד המילה כולה).

<u>$V(< G, k_1, k_2 >, y)$</u>

1. צור את הקבוצות S_1, S_2 מהעד.

2. בדוק האם $|S_1| = k_1$
2.1. אם כן, בדוק האם הקבוצה מהווה קליקה.
2.2 אם יש זוג קודקודים שאין בניהם צלע- דחה.

3. בדוק האם $|S_2| = k_2$
3.1 אם כן, בדוק האם הקבוצה מהווה קבוצה בת"ל.
3.2 אם יש זוג קודקודים בקבוצה שיש בניהם צלע- תדחה
3.3 אחרת, קבל

המכונה דטרמיניסטית, עלינו להוכיח נכונות ופולינומיות.

זמן ריצה: המכונה פולינומית כי גם במקרה הכי גרוע אם נעבור פעמיים על כל הצלעות האפשרויות בגרף, עדיין נישאר פולינומיים באורך הקידוד.

נכונות:
אם המילה שייכת לשפה אזי מתקיים:

1. יש קבוצת קודקודים S_1 בגודל k_1 המהווה קליקה בגרף
2. יש קבוצת קודקודים S_2 בגודל k_2 המהווה קבוצה בת"ל בגרף

לכן העד y מייצג את S_1 הקודקודים המהווים $k_1 - clique$ בגרף, ואת S_2 הקודקודים המהווים $k_2 - IS$. כשנעבור על גודל הקבוצה S_1 שתיווצר נראה שהיא אכן בגודל המתאים, נוודא ונראה שיש צלע בין שני קודקודים בה. כשנעבור על S_2 נוודא שהיא בגודל המתאים, נוודא ונראה שאין צלע בין כל שני קודקודים בה, ולכן נקבל.

סה"כ, נקבל שקיים עד פולינומי בגודל הקלט כך ש: $(< G, k_1, k_2 > y) \in R_{Clique\&IS}$
אחרת, אין קבוצה בגודל k_1 המהווה קליקה בגרף ואין קבוצה בגודל k_2 המהווה קבוצה בת"ל בגרף, לכן ייקרו לפחות אחד מהשלושה הבאים:

1. העד ייצג קבוצה בגודל k_1
2. העד ייצג קבוצה בגודל k_2
3. העד ייצג קבוצה בגודל אחר

בכל המקרים נדחה.

סה"כ בכל מקרה המסלול ייגמר במצב דוחה, ולכן $(< G, k_1, k_2 >, y) \notin R_{Clique\&IS}$.

ד. בעזרת מ"ט א"ד:

$M(< G, k_1, k_2 >)$:

1. בדוק האם $k_1 \leq k_2$ – אם לא, דחה
2. נחש מספר בין k_1 ל-k_2
3. נחש קבוצת קודקודים בגודל זה.
4. בין כל שני קודקודים שנבחרו בדוק האם יש צלע, אם אין דחה.
5. אם בין כל שניים הייתה צלע, קבל.

זמן ריצה: הבדיקה בשלב 1 מצריכה $O(k_1 + k_2)$ – פולינומי באורך קידוד הקלט.
ניחוש מספר, הסיבוכיות היא כגודל המספר, ולכן מצריך $O(k_2)$ צעדים- פולינומי גם כן.

מעבר על כל זוג קודקודים מצריך $O(k_2^2)$- פולינומי באורך הקידוד גם כן.
נשים לב כי $k_2 < |V(G)|$ ולכן הכל פולינומי בקלט.

<u>נכונות:</u>
אם $< G, k_1, k_2 > \in CliqueK$, מתקיים $k_1 \le k_2$, ולכן נעבור את שלב (1), ובנוסף יש קליקה בגודל $k_1 \le k \le k_2$, לכן באחד ממסלולי החישוב ינוחש המספר k, ותנוחש קבוצת הקודקודים הנכונה- בין כל שני קודקודים תהיה צלע ולכן נקבל. סה"כ יהיה לנו לפחות מסלול אחד מקבל ולכן $< G, k_1, k_2 > \in L(M)$

אם $< G, k_1, k_2 > \notin CliqueK$, אם שמתקיים $k_1 > k_2$ אז המסלול היחידי שיהיה הוא מסלול דוחה. או שעבור כל $k_1 \le k \le k_2$, כל קבוצה בגודל k לא תהווה קליקה. לכן בכל אחד ממסלולי החישוב עבור כל קבוצת הקודקודים שננחש יהיו שני קודקודים שלא תהיה צלע ביניהם ולכן תמיד נדחה. סה"כ כל המסלולים הם מסלולים דוחים ולכן $< G, k_1, k_2 > \notin L(M)$

בעזרת מכונה המוודאת יחס:

<u>סיבוכיות העד:</u> העד y יהיה מחרוזת בינארית באורך $|V(G)|$, שייצג (באופן שכבר הסברנו) קבוצת קודקודים כלשהי, האידיאל הוא שהקבוצה שתיוצג היא קליקה בגודל המתאים בגרף.

<u>$V(< \mathrm{G}, \mathrm{k}_1, k_2 >, y)$:</u>

1. וודא שגודל הקבוצה המיוצגת הוא לפחות k_1 ולכל היותר k_2- אם לא, דחה.
2. בדוק האם כל שני קודקודים בקבוצה יש צלע- אם לא, דחה.
3. קבל.

<u>זמן ריצה</u>: וידוא גודל הוא לינארי (השוואת תווים). בדיקה האם בין כל שני קודקודים בקבוצה יש צלע הוא $O(V(G)^2)$ ולכן פולינומי בגודל הקלט.

<u>נכונות:</u>

אם $< G, k_1, k_2 > \in CliqueK$, יש איזושהי קבוצה בגודל המתאים, ולכן קיימת מחרוזת שתייצג את קבוצה זו. המכונה המוודאת תוודא כי הקבוצה בגודל הנכון ואכן מהווה קליקה, ולכן $(< G, k_1, k_2 >, y) \in R_{CliqueR}$.

אם $< G, k_1, k_2 > \notin CliqueK$, לא קיימת קליקה כנדרש, ולכן כל עד y יגרום למוודא לדחות.

9. להלן ציטוט מספר של חנוך דאום:

בית כנסת היה עבורי גם מקום העבודה של אבא. אהבתי את ההתנהלות שלו. אבא עשה את הדברים בחן. באחד החגים התגלע ויכוח אם צריך להגיד קדיש או לא. דיון הלכתי. שליימה אמר שוודאי שצריך. כל שנה אומרים. נחמיה אמר מה פתאום, אני זוכר במאה אחוז שלא אומרים כאן קדיש. הם באו לאבא שלי ושאלו אותו: "יחזקאל, מה המנהג?"

"זה המנהג", הוא ענה להם.

"מה המנהג?" הם לא הבינו, "להגיד או לא להגיד קדיש?"

"להתווכח", הוא ענה. "המנהג הוא בכל שנה להתווכח על זה".

אציין רק שהעמדה של נחמיה בזמן אמת נראתה לי פחות אמינה, כי בעוד שליימה זכר שאומרים קדיש, נחמיה טוען שהוא זוכר שלא אומרים, ותהיתי איך אפשר לזכור שלא אומרים משהו? זה קצת כמו

השאלה היא:
מה זה קשור לחומר?

פתרון:

זה בדיוק המתח של NP לעומת coNP. איך אפשר להמציא עד שמשהו לא קיים? צריך לעבור על כולם!
וכן בסיפור - איך אפשר לזכור שמשהו לא קרה?