מרתון חישוביות:

מכונת טיורינג מודל השקול בכוחו למחשב לא הוכח אלא השערה

יש לה קבות מצבים סופית – Q

יש לה סרט זיכרון של קריאה כתיבה. מעין מערך אינסופי המתחיל בתא השמאלי ביותר ואין לו תא ימני ביותר

יש לה קרש קורא כותב המצביע על התא בזיכרון שעליו עומדים כרגע

איך המכונה עובדת?

התחלת החישוב:

- מצב q_0 התחלתי.
- ראש קורא כותב נמצא הכי שמאלה
- על סרט הזיכרון כתובה מילת הקלט נגיד x החלק מהתא השמאלי ביותר ועד שהיא נגמרת לאחריה יש בכל שאר התאים את התו הריק (blank)

מעברים:

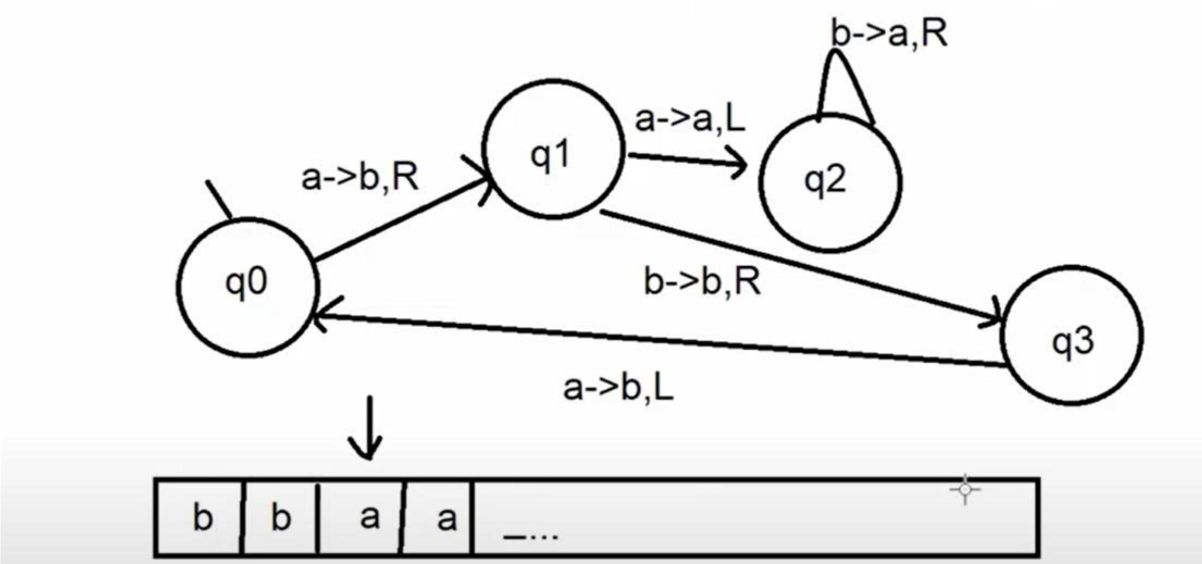
בהתאם **למצב** בו נמצאים **ולאות** שאותה הראש קורא כותב רואה (נמצא עליה) עושים 3 דברים:

- עוברים למצב חדש (אולי נשארים)
- כותבים באותו התא שנמצאים עליו איזשהו תו אחר או שמשאירים את מה שהיה ומזיזים את הראש קורא כותב צעד אחד שמאלה או ימינה או שנשארים במקום.

מצב, מיקום ראש, תוכן הזיכרון יחד נקראים קונפיגורציה.

שלושתם ביחד.

דוגמה:



איך מסתיים החישוב?

בשונה מאוטומט (רגיל או מחסנית שם החישוב מסתיים כשמסיימים לקרוא את המילה) כאן החישוב לא תלוי בסיום קריאת המילה.

אפשרי לא לקרוא את כל המילה אפשרי לקרוא שוב ושוב תווים קודמים ואפשר גם לגלוש מחוץ למילה לתאים שהיו ריקים ולמלא תאים נוספים בתוכן.

החישוב מסתיים כאשר מגיעים למצב מיוחד שנקרא מצב עצירה.

אם המכונה לא מגיעה בחישוב למצב עצירה אז היא תקועה בלולאה אינסופית.

נדבר בדרך כלל על שפות וכמו באוט' המכונה צריכה לקבל קלט X ולומר האם X בשפה או לא.

לכן במקום מצב עצירה אחד יהיו לנו שני מצבים שנקרים accept & reject.

אם מגיעים למצב עצירה נגמר החישוב ולא ממשיכים.

אם מכונה מסיימת את החישוב במצב מקבל – היא קיבלה את המילה שהתקבלה כקלט בהתחלה (אומרת שהמילה בשפה)

ואם המכונה מסיימת במצב דוחה אז היא דוחה את המילה ז"א לא בשפה.

ניתן לחשב על מכונת טיורינג כעל פונקציה (תוכנית מחשב) שמקבלת סטרינג ומחזיקה T או F

במהלך הקורס תצטרכו לכתוב אלגוריתמים שמכונה יכולה לבצע ולא מכונות.

מ"ט היא: $M = \left(Q, \Sigma, \Gamma, \mathrm{q}_0, \text{\_}, \delta, q_{accept}, q_{reject}\right)$

כאשר Q קבוצה סופית של מצבים

קבוצה Σ – קבוצה סופית של אותיות הקלט

קבוצה Γ – קב' סופית שאותם מותר לכתוב בסרט

פונקציה δ - פונקציית מעברים $\delta(q,\sigma) = (p,\gamma,\{R,L,S\})$

q הוא המצב הנוכחי σ היא אות שנמצאת בתא שכרגע מצביעים עליו. P הוא המצב שאליו עוברים γ היא האותר שכותבים בתא שכרגע מצביעים עליו והיא דורסת את σ ו*RLS* לאן מזיזים את הראש.

מתקיים:

$$q_0 q_{reject} q_{accept} \in Q$$

$$\sigma \in \Gamma$$

$$\_ \in \Gamma$$

<u>שפה של מכונה:</u>

קבוצת על המילים בא"ב שאם הינו מכניסים אותן למכונה כקלט היא הייתה מגיעה למצב מקבל.

מילים שעבורן המכונה לא עוצרת (נתקעת בלולאה אינסופית) הן לא חלק מהשפה של המכונה.

M(String x)

If(x.len %2 ==0) return true

Else

While(x.len > 0)

If (x.len <5) X= x.substring(0,x.len-1)

Return true

מה השפה של *M*?

כל הזוגיים וכל אלה שבאורך 1 או 3.

סימון השפה של מכונה *M*:*L(M)*

נסיון לכתוב מכונה לשפה כלשהי:

$$L=\{wcw \mid w\in\{a,b\}^*\}$$

דוג' למילה:

$|a|a|b|a|c|a|a|b|a|$

הרעיון הוא לסמן ב X
לבדוק a להתקדם עד אחרי C
אם יש x מדלגים ושם אין בודקים אם
האותיות מתאימות לזיכרון וככה ממשיכים

יש לנו קונפיגוקציה שזה בעצם המצב של המחשב, אם במהלך החישוב המכונה (דטר') חזרה לקונ' שהיא כבר הייתה בה אז היא תקועה בלולאה אינסופית.

היא בעצם תחזור על אותם קונפ' ושוב את אותם צעדים.

<u>מודלים שונים של מ"ט: (כל המודלים שקולים בכח החישוב למ"ט דטר' אבל שימו לב שכאשר נגיע לחצי השני של הקורס – סיבוכיות – אז המודלים לאו דווקא שקולים בזמן הריצה)</u>

1. מכונה אי-דט' – מכונה שמנחשת את המעבר או לחלופים יש לה מספר אפשריות למעברים על אותו מצב ועל אותה אות (כמו אסל"ד).
2. מ"ט דטר' עם מספר סרטי זיכרון – כל עוד המספר סופי!
3. מ"ט דטר' עם סרט זיכרון אינסופי משני הצדדים
4. מ"ט שאין לה אפשרות להישאר עם הראש קורא כותב במקום חייב להזיז בכל צעד ימינה או שמאלה

<u>מכונה מכריעה מול מכונה מזהה</u>

עבור שפה L אם אנו מצליחים לכתוב לה מכונת טיורינג שמקבלת כל מילה שבשפה ודוחה כל מילה שלא בשפה אז המכונה מכריעה את L.

ואם אנו מצליחים לכתוב לשפה מכונה שמקבלת כל מילה שבשפה אבל עבור מילים שהן לא בשפה היא או דוחה או לא עוצרת אז זאת מכונה מזהה.

דוג': $L = \{w \in \{a, b, c\}^* | w\ starts\ with\ 'a'\}$

מ"ט M מכריעה לL:

על קלט w:

- אם התו הראשון של w הוא a קבל
- אחרת – דחה

מ"ט M מזהה לL:

על קלט w:

- ....
- אם התו הראשון של w הוא b תיכנס ללולאה אינסופית
- אחרת דחה

חלק מהמילים שלא בשפה יכניסו ללולאה אינסופית.

<u>שפה כריעה:</u>

שפה שניתן לבנות לה מכונה מכריעה (ברור שאם אפשר מכריעה אז אפשר גם לבנות מזהה)

<u>שפה מזוהה טיורינג:</u>

שפה שניתן לבנות לה מכונה מזהה.

קבוצת כל השפות שהן קריאות תסומן בR

קבוצת כל השפות שהן מזוהות תסומן RE

שימו לב כי הן לא שפות אלה קבוצה של שפות (קבוצה של קבוצות)

אבחנה: $R \subseteq RE$

אם שפה היא כריעה היא גם מזוהה.

רוב השפות שאנחנו מכירים מאוט' 1,2 ומכל התואר עד כה הן בR.

כל השפות הרג' וחסרות הקשר נמצאות בR כי ניתן לבנות להן אוט' ומכאן גם מ"ט.

בקורס הזה נתאר הרבה שפות שהן לא מקבלות רק מחרוזות אלה גם אובייקטים.

כל אובייקט הוא למעשה מחזורת של 0 ו1 שמייצגת את הקידוד של האובייקט במחשב.

את הקידוד של אובייקט O נסמן ב $<O>$/

לדוגמה:101 = <5>

משחקים של ביטים בעצם.

סיכום – שפות שהן בRE אבל לא בR:

1. $A_{TM} = L_U = \{< M, w > | M \text{ is truring machin and } M \text{ accepts } w\}$
2. $HALT_M = HP = \{\dots . | \dots \text{stops on } w\}$
3. $L_D = \{< M > | M \text{ is TM and } M \text{ accepts } < M >\}$
4. $L_{\neq\phi} = \{< M > | L(M) \neq \phi\}$ קובצת המכונות שמקבלות לפחות מילה אחת.

איך מוכיחים שייכות או אי שייכות?

איך מוכיחים ששפה שייכת לR:

- מראים מ"ט, אלגוריתם, שמקבלת את השפה ותמיד עוצרת.
  דוג':$L = \{< M > | M \text{ is TM and } M \text{ dosne'tget into } 10th \text{ cell on input } 0\}$
  נראה ש $L \in R$
  נגיד את המכונה הבאה N: : על קלט <M> כאשר M מ"ט:
  - סמלץ אץ הרצת M על 0 ותוך כדי הרצה בדוק האם M מגיעה לתא ה10. אם כן דחה
  - אם M חזרה על קונפיגורציה שכבר הייתה בה – קבל.
  - אם M עצרה ולא עברה את תא 9 – קבל.

הבחנה: אם מגבילים את המכונה עד תא מסוים אז כמות הקונפ' השונות הופכת להיות סופית כי קונפ' בנויה ממצב מיקום ראש ותוכן זיכרון

כמות המצבים היא סופית אבל הזיכרון הוא אינסופי ולכן יש אינסוף קונ' שונות

אם מגבילים את הזיכרון עד תא מסוים אז זה הופך את האופציות לכמות סופית (בכל תא יש כמות סופית של אופציות מה לכתוב כי הא"ב הוא סופי) לכן סה"כ כמות הקונפ' היא מספר סופי K ולכן אם המכונה מבצעת יותר מK צעדים אז היא כבר בהכרח חזרה על קונפ' שהייתה בה ולכן ניתן יש להסיק שהיא בלולאה אינסופית ולעזור את החישוב.

איך מוכיחים ששפה שייכת לRE?

- מ"ט, אלגוריתם, שמקבלת את השפה ועוצרת אם המילה שנתנו לה בשפה. אם המילה לא בשפה אז היא צריכה או לדחות או לא לעצור.

דוגמה: $L = \{< M > | M \text{ is TM and } M \text{ rejects and stops at least } 4 \text{ inputs}\}$

נראה ש $L \in RE$

מגדיר את המכונה הבאה :N: על הקלט <M> כאשר M מ"ט:

- הוא מדבר על זה שהוא כל פעם מכניס לתוך המכונה מהמילה הראשונה ואז על 2 מילים..... ככה אני לא אתקע אם אני פשוט אבדוק בצרוה מקבילים על כולם.
- עבור i מ1 עד אינסוף
  סמלץ את ריצת M למשך i צעדים על כל אחת מi המילים הראשונות בסדק לקסיקוגרפי אם בתוך הזמן המוקצב היו 4 מילים שM הספיקה לעצור ולדחות אותך – מקבל.

דרך ב' להוכיח ששפה היא בRE ללא ריצה מבוקרת: שימוש במ"ט אי דט'

דוגמה: $L = \{< M > | M\ is\ TM\ and\ M\ rejects\ and\ stops\ at\ least\ 4\ inputs\}$

נראה ש $L \in RE$

- נחש 4 מילים שונות
- עבור i מ1 עד 4 אם M(Wi)= false הגדל את קאונטר ב1
- אם counter =4 – קבל אחרת דחה.

איך מוכיחים ששפה לא בR:

- שיטת לכסון – הנחה בשלילה ובניית מ"ט מבלבלת
- רדוקציה – להראות ששפה אחת יותר קשה לפתרון משפה אחרת, ולכן אם מבקשים להוכיח שL לא בR אז נראה שהיא יותר קשה משפה שאנחנו כבר יודעים שהיא לא בR ומכן ינבע שL לא בR.

הגדרה: רדוקציה

L_1 ניתנת לרדוקציה לL_2 אם קיימת פונקציה $f: \Sigma^* \to \Sigma^*$ כך שלכל מילה x מתקיים:

$x \in L_1$ אם ורק אם $f(x) \in L_2$

מילים אחרות אם נותנים לפונקציה מילה מL_1 אז היא מחזיקה מילה מL_2.

ואם נותנים לפונקציה מילה שהיא לא בL_1 אז היא מחזירה מילה שהיא לא בL_2.

סימון: $L_1 \leq L_2$

אם יש רדוקציה אז כדי לפתור את השאלה האם מילה נמצאת בL1 או לא:

ניקח את המילה ונפעיל עליה את הרדוקציה. לאחר מכן אם אנחנו יודעים לפתור את L_2 נוכל לבדוק אם התוצאה של הפונקציה בL_2 אז נסיק שהמקור היה בL1 ואם התוצאה של הפונקציה לא בL_2 נסיק שהמקור לא היה בL1.

מכאן, משפט הרדוקציה אומר ש:

איפה שL_1 לא נמצאת גם L_2 לא נמצאת.

כלומר : אם $L_1 \notin R\ so\ L_2 \notin R$

ואם $L_1 \notin RE\ so\ L_2 \notin RE$

איפה שL_2 נמצאת גם L_1 נמצאת.

דוגמה: $L = \{< M > | M\ is\ TM\ and\ M\ rejects\ and\ stops\ at\ least\ 4\ inputs\}$

ונראה ש$L \notin R$

נראה רדוקציה – מול איזה שפה נשווה את L? מאחת מהשפות שאנחנו מכירים לרוב $L_U\ or\ L_{hp}$

נראה שיש רדוקציה מL to L_U כלומר: $L_U \leq L$ מכאן אם $L_U \notin R$ so $L \notin R$

$$A_{TM} = L_U = \{< M, w > | M \text{ is truring machin and } M \text{ accepts } w\}$$

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

המטרה: (צריך לכתוב מה $< M' >$ עושה)

1. אם $< M, w >$ כן בL_U אז $< M' >$ כן בL
2. אם $< M, w >$ לא בL_U אז $< M' >$ לא בL

טיפ - M' תלויה בM, w.

טיפ - M' על קלט x:

- הרץ את M על w (במילים אחרות: y = M(w)) עכשיו ננסה להסיק את המטרה, איפה הבעיה?
  הבעיה שM לא עוצרת, אם כן הייתה עוצרת הינו בודק if true או false האם היא מקבלת או לא..
  אבל מתי אני לא יכול לשאול כאשר היא לעוצרת, האם אנחנו משיגים את המטרה המצב הזה?
  אם M' לא עוצרת האם היא מקיימת את השפה השניה? שימו לב ש'M היא לא עוצרת על שום קלט שלה כי על קלט שנכניס היא תעשה את השורה הראשונה שלה אבל M,w הם קבועים זה לא משתנה לכן לא משנה מה נכניס ל'M זה לא יעצור, ולכן היא לא תקבל.
  כאשר
  M לא עצרה על w לא יתקיים לנו התנאי של L_u ולכן $M`$ לא בL .
  התנאי השני.
- אם M קיבלה את w :
  - אם $x = 00$ ir $x = 01$ or $x = 10$ or $x = 11$ – דחה.
  - אחרת קבל.
- אם M דחתה את w:
  - קבל כל x ללא הסתעפות. (מקבלים את כל הקלטים)

מה צריך להוכיח בכל רדוקציה?

1. **מלאה -** הפונק' עובדת על כל קלט.
2. **ניתנת לחישוב –** ניתן לכתוב תוכנית מחשב\מכונה שתחשב את הפונק'.
3. **תקפות –** אם הקלט בשפה אז התוצאה בשפה השניה ואם הוא לא בשפה התוצאה לא בשפה השניה.

הוכחת תקפות:

1. אם M וw נמצאים בL_U אז M עוצרת ומקבלת את wולכן M' תדחה את 00,01,10,11 ולכן $< M' > \in L$
2. אם M וw לא נמצאים בL_U אז M לא עוצרת על w ואת M' לא עוצרת על אף קלט ופרט לא דוחה 4 קלטים או שM עוצרת ודוחה את w ואז M' מקבלת הכל ופרט לא דוחה 4 קלטים. ולכן: $< M' > \notin L$

איך מזהים האם שפה בR והאם שפה בRE?

שואלים את עצמנו: אם מגלים לנו שהקלט בשפה, איך ניתן לשכנע את עצמנו שהוא אכן בשפה?

האם צריך לעבור על קבוצה אינסופית של אפשרויות והאם צריך להריץ מכונה על מילה הרצה מלאה (ואז יתכן שהיא לא תעצור) ?

אם אחד מהדברים הללו צריך לקרות אז השפה לא בRE. אחרת השפה כן בRE

לאחר מכן, רק אם הצלחנו בראשון ז"א בRE שואלים את עצמנו : אם מגלים לנו שהקלט לא בשפה, איך ניתן לשכנע את עצמנו שהוא אכן לא בשפה?

אם אחד מהדברים שלמעלה צריך לקרות אז השפה היא לא בR אחרת אם ב2 השאלות הצלחנו לשכנע את עצמינו אז השפה כן בR.

דוגמה:

$$L = \{< M > | M\ stops\ on\ 2\ inputs\}$$

אם M בשפה אז פשוט ננחש 2 שלטים ונריץ את Mעל שניהם ולפי ההנחה היא תעצור. לכן $L \in RE$

אם M לא בשפה אז כדי להשתכנע צריך לעבור כל הקלטים בעולם ולראות שהיא לא עוצרת על יותר מ1. וזאת קבוצה אינסופית של אפשרויות, לכן $L \notin R$

$$L = \{< M > | L(M) = \phi\}$$

אם M בשפה אז צריך לעבור על כל הקלטים ולהראות שאף אחד לא מתקבל – קבוצה אינסופית לכן $L \notin RE$

מרתון 2 –

המחלקה $coRE$:

כל השפות שהמשלימה שלהן היא בRE

תזכורת: שפה בRE אמ"מ קיימת לה מט מזהה שעל כל מילה בשפה היא עוצרת ואומרת כן ועל כל מילה שלא בשפה עוצרת ואומר לא או שלא עוצרת.

שפת coRE אממ קיימת לה מ"ט מזהה שעל כל מילה שלא בשפה עוצרת ואומרת לא ועל כל מילה שכן בשפה עוצרת ואומרת כן או שלא עוצרת.

תזכורת : שפה של מכונה היא כל המילים שהמכונה עוצרת ואומרת כן עליהן.

תכונות:

- $RE \cap coRE = R$ כי אם יש מ"ט שתמיד יודעת לומר כן ויש מ"ט שתמיד יודעת לומר לא אז שילוב של שניהם יתן לנו מכונה מכריעה.
- כל שפה שמ"ט יכולה לקבל נמצאת בRE. (או אפילו בR) אבל אם יש מ"ט לשפה בcoRE אז השפה של המכונה לא תואמת לשפה שהיא מזהה.

תכונות של רדוקציה:

1. משפט הרדוקציה: אם $L_1 \leq L_2$ אז: (L2 יותר קשה מL1) :
   - $L_1 \notin R$ אז $L_2 \notin R$
   - $L_1 \notin RE$ אז $L_2 \notin RE$
   - $L_1 \notin coRE$ אז $L_2 \notin coRE$
   - $L_2 \in R$ אז $L_1 \in R$
   - $L_2 \in RE$ אז $L_1 \in RE$
   - $L_2 \in coRE$ אז $L_1 \in coRE$
2. אם $L_1 \leq L_2$ אז $\bar{L_1} \leq \bar{L_2}$ (גג)
3. תמיד מתקיים שרדוקציה לעצמה $L \leq L$
4. טרנזיטיביות: אם $L_1 \leq L_2$ וגם $L_2 \leq L_3$ אז $L_1 \leq L_3$.

משפט רייס:

אם יש שפה מהצורה: $L = \{< M > |$המכונה של השפה עם תנאי$\}$ אם התנאי לא טריוויאלי אז השפה לא בR.

דגשים:

1. הקלט הוא רק מכונה אחת.
2. התנאי מתייחס רק לשפה של המכונה (כלומר למה שהיא מקבלת)

דוגמאות:

$\{< M >: |L(M)| = 5\}$- תנאי על השפה.

$\{< M > | M \ accepts \ 000\}$ – תנאי על השפה.

$\{< M > | \ M \ accepts \ \varepsilon \ in \ less \ then \ 10 \ steps\}$ – צעדים של מכונה זה לא שפה ולכן זה לא תנאי על השפה.

הערה: אם מדברים על צעדים עוצרת וכו' זה תנאי על המכונה ולא על השפה.

3. התנאי לא טריוויאלי. זה אומר שהתנאי שלא כל השפות מקיימות אותו! או תנאי שכולם לא מקבלים אותו.
אם התנאי הוא הוא טריוויאלי כך שכולם מקיימות אותו אז השפה שקיבלנו היא $\Sigma^* \in R$
אם התנאי הוא ..... כך שכולם לא מקיימות אותו קיבלנו את ϕ וזה גם כן בR .

איך מוכיחים באמצעות משפט רייס ששפה היא לא בR?

- מקבלים שפה L ובודקים בלי הוכחה שהתנאי הוא על השפה של המכונה והקלט הוא רק מכונה אחת.
- מגדירים תנאי S שהוא העתק של התנאי של L רק שבמקום לדבר על השפה של המכונה, מדברים על שפות באופן כללי. לדוגמה:
  $\{< M >|\ |L(M)| \leq 7\ \}$ ברור שהקלט רק מכונה והתנאי הוא על השפה.
  נגדיר:$S = \{\ L \in RE\ ||L| \leq 7\}$.
- מוכיחים שהתנאי הוא לא טריוויאלי:
  מוכיחים שS הוא לא ϕ על ידי שנותנים דוגמה לשפה שהיא מקיימת את התנאי.
  מוכיחים שS הוא לא ב RE על ידי שנותנים דוגמה לשפה שהיא לא מקיימת את התנאי.

- מסקנה: השפה שקיבלנו היא לא בR.

- הרחבה: אם לאחר שעשינו את כל התהליך השפה הריקה כן מקיימת את התנאי, $S \in \phi$ אז השפה שקיבלנו לא בRE.
  אם ϕ לא מקיימת את התנאי זה לא מוכיח כלום!

דוגמה מלאה:

$\{< M >|L(M) contains\ only\ even\ len\ word\}$

(רק מילים באורך זוגי)

מקבלים רק מכונה והתנאי הוא על השפה, לכן ניתן להשתמש ברייס.

נגדיר: $S = \{L \in RE\ |L\ contains\ only\ ....\}$ נוכיח שS אינה טריוויאלית.

$S \neq \phi$ כי $\{00\} \in S$.

$S \neq RE$ כי $\{0\} \notin S$.

מסקנה: השפה L לא בR.

השפה הריקה לא מכילה מילים באורך אי זוגי ו0 מילים באורך זוגי ולכן מקיימת את התנאי ולכן לפי רייס המורחב השפה L גם לא בRE.

הערה: אם צריך להוכיח ברד' על תכונה של שפות שלא ניתן להשתמש ברייס עדיף את L_u , אם זה תכונה של המכונה עדיף להשתמש בHP.

הוכחת משפט רייס: (כל תכונה S לא טריוו' של שפות אינה בR)

יהיה S תכונה לא טריוויאלית של שפות. $L_s = \{< M >|L(M) \in S\}$.

צ"ל: ש$L_s \notin R$.

הערה: אם רוצים להוכיח אי שייכות ברד' אז המוכרת בצד שמאל.

נחלק למקרים:

מקרה 1 – השפה הריקה לא בS. $\phi \notin S$

מכיון שS היא לא טריוויאלית אז קיימת שפה L_1 שנמצאת בS. ומכיוון שמדובר רק בשפות שהן בRE אז קיימת מ"ט M_1 שמזהה את L_1.

נשתמש ברדוקציה L_u כלומר: $L_u \leq L_s$.

כאשר: $f(< M, w >) = < M' >$

תזכורת: ברד' צריך להגדיר מהM' עושה וצריך להוכיח שאם M מקבלת אתw אז M' מקיימת את התנאי של Ls ואם M לא מקבלת את w אז M' לא תקיים את התנאי ז"א השפה לא תהיה בS.

M': על קלט x:

- הרץ את M על w
- אם M דחתה את w דחה את x
- אחרת M קיבלת את w הרץ את M_1 על x וענה כמוה.

תקפות:

אם M לא עוצרת על W אז M' לא עוצרת על אף X כי היא תמיד תתקע בשורה הראשונה, ולכן

$L(M') = \phi \notin S.$

אם M דוחה את W אז M' דוחה את כל הX-ים ולכן שוב השפה ריקה.$L(M') = \phi \notin S$

אם M מקבלת את W אז M' מתנהגת כמו M_1 ולכן $L(M') = L(M_1) = L_1 \in S$.

מסקנה ממשפט הרד':$L_u \notin R$ ולכן $L_s \notin R$

מקרה 2 – השפה הריקה כן בS. $\phi \in S$

במקום רד' מL_u נעשה רד' מהמשלים שלהL_u^- .

מכיון שS היא לא טריוויאלית אז קיימת שפה L_2 <u>שלא נמצאת</u> בS. ומכיוון שמדובר רק בשפות שהן בRE אז קיימת מ"ט M_2 שמזהה את L_2.

הרדוקציה : $L_u^- \leq L_s$

נשתמש ברדוקציה L_u^- כלומר: $L_u^- \leq L_s$.

כאשר: $f(< M, w >) = < M' >$

תזכורת: ברד' צריך להגדיר מהM' עושה וצריך להוכיח שאם M מקבלת אתw אז M' מקיימת את התנאי של Ls ואם M לא מקבלת את w אז M' לא תקיים את התנאי ז"א השפה לא תהיה בS.

M': על קלט x:

- הרץ את M על w
- אם M דחתה את w דחה את x
- אחרת M קיבלת את w הרץ את M_2 על x וענה כמוה.

תקפות:

אם M לא עוצרת על W אז M' לא עוצרת על אף X כי היא תמיד תתקע בשורה הראשונה, ולכן $L(M') = \phi \in S$.

אם M דוחה את W אז M' דוחה את כל ה-X-ים ולכן שוב השפה ריקה. ולכן $L(M') = \phi \in S$

אם M מקבלת את W אז M' מתנהגת כמו M_1 ולכן $L(M') = L(M_2) = L_2 \notin S$.

מסקנה ממשפט הרד': $L_u^- \notin RE$ ולכן $L_s \notin RE$

דוגמה לרדוקציות מבחן 2019 א':

3. (30 נקודות) לכל אחת מהשפות הבאות, קבעו האם היא בR? והאם היא בRE? הוכיחו את תשובותיכם.

a. $L_1 = \{< M_1 >< M_2 >| \exists(x \in \sum^*)\ such\ that\ M_1\ accepts\ x\ and\ M_2\ rejects\ x.\}$

בעברית: זוהי שפת קידודי המכונות M_1 ו-M_2 כך שקיים $x \in \sum^*$, כך שM_1 מקבלת את x וM_2 דוחה את x.

b. $L_2 = \{< M >| \exists(x \in \sum^{24})\ such\ that\ M$ does not enter the the |x|'s cell of its tape.$\}$

בעברית: זוהי שפת קידודי המכונות M כך שקיים $x \in \sum^{24}$, כך שM בריצתה על x איננה נכנסת לתא ה-|x| של הסרט שלה.

c. $L_3 = \{< M_1 >< M_2 >< M_3 >| \exists(x \in \sum^*)\ such\ that\ M_1 accepts\ x, M_2\ rejects\ x$

$and\ M_3\ enters\ an\ infinite\ loop\ when\ running\ on\ x.\}$

בעברית: זוהי שפת קידודי המכונות M_1, M_2 ו-M_3 כך שקיים $x \in \sum^*$, כך שM_1 מקבלת את x, M_2 דוחה את x ו M_3 נכנסת ללולאה בריצתה על x.

פתרון:

א. ניתן לעשות שימוש באידטרמיניסטי. (נחש מילה שM_1 מקבלת אותה וM_2 דוחה)
$L_1 \in RE\ but\ L_1 \notin R$

הוכחה:

- $L_1 \in RE$ נגדיר את מ"ט האי דאר' N הבאה:
  N על קלט $< M_1, M_2 >$ כאשר M_1, M_2 הן מ"ט:
  - נחש מילה x.
  - הרץ את M_1 על x. אם דחתה – דחה.
  - הרץ את M_2 על x. אם קיבלה – דחה.
  - קבל.

  נכונות: אם $< M_1, M_2 > \in L_1$ אז קיימת מילה כנ"ל. ולכן קיים ניחוש לx שהוא הנ"ל ולכן הרצת M_1 תסתיים בקבלה והרצת M_2 תסתיים בדחיה ולכן N תקבל.
  אם $< M_1, M_2 > \notin L_1$ אז לכל ניחוש x או שאחת מהמכונות לא תעצור ואז N לא תעצור ולכן לא תקבל או שM_1 דחה ואז N דחה או M_2 תקבל ואז N דחה.

- $L_1 \notin R$:
  נראה רדוקציה מL_u . כלומר: $L_u \le L_1$
  על ידי: $f(< M, w >) = < M_1, M_2 >$.
  M_1 – על קלט X:
  - הרץ את M על W.
  - אם M דחתה – דחה X.
  - אם M קיבלה – קבל X.

  M_2 - על קלט Y:
  - הרץ את M על W.
  - אם M דחתה – קבל את Y.

- אם M קיבלה – דחה את Y

הרדוקציה מלאה וניתנת לחישוב (כתיבת תיאור 2 מכונות)
תקפות: אם M מקבלת את w אז M_1 מקבלת את כל המילים, M_2 דוחה את כל המילים.
ופרט קיימת מילה אחת שM_1 מקבלת אותה ו M_2 דוחה אותה.
אם M לא עוצרת על w אז שני המכונות לא עוצרות על אף מילה ולכן לא קיימת מילה שמקיימת את התנאי של L_1.
אם M דוחה את w אז M_1 דוחה הכל וM_2 מקבלת הכל, ולכן לא קיימת מילה כך שM_1 מקבלת אותה וM_2 דוחה אותה.

ב. (מילה באורך 24) יש לנו מספר סופי של מילים אבל מה יקרה אם נתקע בלולאה אינסופית כי לא מגיעים לתא ה24? פשוט נבדוק האם נחזור על אותה קונפיגורציה, אם כן היא תקועה ונשחרר.

$L_2 \in R$ נראה את המכונה.
המכונה N על קלט <M> כאשר M מ"ט:
- נחש מילה x בגודל 24
- סמלץ את ריצת M על X כאשר תוך כדי הריצה שמור את כל הקונפיגורציות שM ביקרה בהן.
- אם תוך כדי הריצה M עברה את תא 24 דחה.
- אם M חזרה על קונפיגורציה שכבר הייתה בה ולא עברה עד כה את תא 24 – עצור וקבל.
- אם M עצרה ולא עברה עד כה את תא 24 – קבל.

נכונות:
אם $L_2 \ni < M >$ קיימת מילה כנ"ל ולכן קיים ניחוש לX כך שM על X לא עוברת את תא 24.
אם $< M >$ עוצרת על X אז N תעצור ותקבל (שורה 5).
אם M לא עוצרת על X אז מכיוון שהיא לא עוברת את תא 24, קיימת קבוצה סופית של קונפיגורציות שונות בחישוב של M על X ולכן בהכרח M תחזור על קונפיגורציה שכבר הייתה בה לכן N תעצור ותקבל.
אם $L_2 \notin < M >$ אז לא קיימת מילה כנ"ל ולכן לכל ניחוש של X המכונה M בריצתה על X כן תעבור את ה24 ולכן N תדחה.

ג. $L_3 \notin RE$
להוכיח על ידי אי שייכות – רדוקציה: (אי אפשר רייס כי מקבלים 3 מכונות)
רדוקציה $L_{\bar{U}} \le L_3$ (כי אנחנו יודעים ש $L_u \in RE$ והמשלים לא נמצא)
על ידי $f(< M, w >) =< M_1, M_2, M_3 >$
M_1 על קלט X:
- קבל.

M_2על קלט Y:
- דחה.

M_3 על קלט Z:
- הרץ את M על W
- אם M דחתה – היכנס ללולאה אינסופית.
- אם M קיבלה – קבל.
- אם M.

הרדוקציה מלאה וניתנת לחישוב (כיצתבת תיאור של 3 מכנות)
תקפות: אם M לא עוצרת או לא מקבל את W (המילה בשפה) אז M1 תמיד מקבלת M2 תמיד דוחה וM3 תמיד לא עוצרת ולכן קיימת מילה X שתקיים את תנאי L_3.
אם M מקבלת את W אז:

אז $M1$ תמיד מקבלת $M2$ תמיד דוחה ו$M3$ תמיד מקבלת ולכן לא קיימת מילה המקיימת את תנאי L_3 (כי $M3$ לא נכנסת ללולאה אינסופית).

חלק 2 של הקורס – סיבוכיות

עד כה היה האם שפה היא כריעה או לא מזוהה או לא.

כאן כל השפות יהיו כריעות. כלומר כל המחלקות שנדבר בחלק הזה מוכלות בR

השאלה כעת – האם קיים אלגו יעיל, מ"ט הפותרת את הבעיה בזמן יעיל.

זמן יעיל\מ"ט יעילה – זמן פולינומיאלי: $O(n^c)$ כאשר C הוא קבוע. (אפילו C = מליון)

לא יעיל – זמן אקספוננציאלי: $O(2^n)$ $O(n!)O(n^{\log n})$

הערה : יש הבדל בין מ"ט דטר' לבין מכונה אי-דטר'. צריך לשים לב לזה.

מחלקות הסיביוכיות:

- ***P* –** כל השפות שיש להן מכונה **דטר'** המכריעה אותן בזמן יעיל.
- ***NP* -** כל השפות שיש להן מכונה **אי-דטר'** המכריעה אותן בזמן יעיל.

דוגמא: $L = \{< A, n > | A \text{ is set},\ n \in N,\ \text{and exists } S \subseteq A \text{ such that } \sum_{x \in S} x = n\}$.

סתם דוגמה: $L \ni < [2,9,1,10,4,3,8], 29 >$ כי ניקח את $S = [2,9,10,8]$ הסכום יוצא 29.

פתרון:

- האם $L \in P$? בעיה פתוחה – לא ידוע אלגוריתם ומצד שני אין הוכחה שזה לא נכון.
- $L \in NP$: על קלט $< A, n >$ כאשר A קבוצה וN מספר שלם.
  - נחש קבוצה $S \subseteq A$ בגודל כלשהו.
  - עבור על איברי S וסכום אותם לתוך SUM
  - אם $SUM=N$ – קבל. אחרת – דחה.

סיבוכיות: נסמן $m = | < A, n > |$

ניחוש: עוברים פעם אחת על A ועל כל איבר מחליטים (בצורה אי- דטר') האם לקחת אותו לS או לא.$O(m)$.

מעבר על S שהיא לכל היותר בגודל A – $O(m)$

ההשוואה : $O(m)$

סה"כ: $O(m)$

הערה: ניתן להגיד שבעיה היא פתוחה! זה לא אומר שאין פתרון הוא פשוט לא ידוע.

הערה: כל שפה או אלגו שלמדנו בקורסים קודים זה בP.

תכונות –

1. $P \subseteq NP$
2. $P = NP$? בעיה פתוחה.

קצת לעומק על מ"ט אי-דטר':

מ"ט דטר' - $M = (Q, \Sigma, \Gamma, q_0, blank, \delta, F)$ כאשר $\delta: Q \times \Gamma \to Q \times \Gamma \times \{S, R, L\}$

מ"ט אי-דטר' – $M = (Q, \Sigma, \Gamma, q_0, blank, \delta_1, \delta_2, F)$ כאשר:

$\delta_1: Q \times \Gamma \to Q \times \Gamma \times \{S, R, L\}$

$\delta_2: Q \times \Gamma \to Q \times \Gamma \times \{S, R, L\}$

בכל שלב בחישוב יש אפשרות לבחור ב$\delta_1 \ or \ \delta_2$.

מספיק שקיים מסלול חישוב 1 שמגיע למצב מקבל – אנו אומרים שהקלט (מילה) מתקבל במכונה.

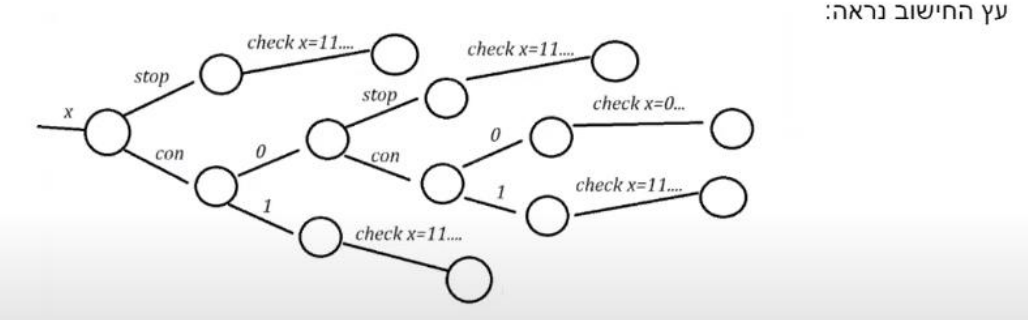
אם כל המסלולים אינם מקבלים אז המילה לא מתקבלת.

דוגמה:

מכונה N על קלט X:

- נחש מילה Y באורך עד 2
- אם $Y = 00$ אז אם X מתחילה בתו 0 – קבל אחרת דחה.
- אחרת, אם X מתחילה ברצף 11 – קבל. אחרת דחה.

עץ החישוב נראה:



הערה: מסלולים מקבלים או דוחים הם לפי המילה.

זמן ריצה = (אם יש מסלול מקבל)= המסלול הכי קצר שהוא מקבל.

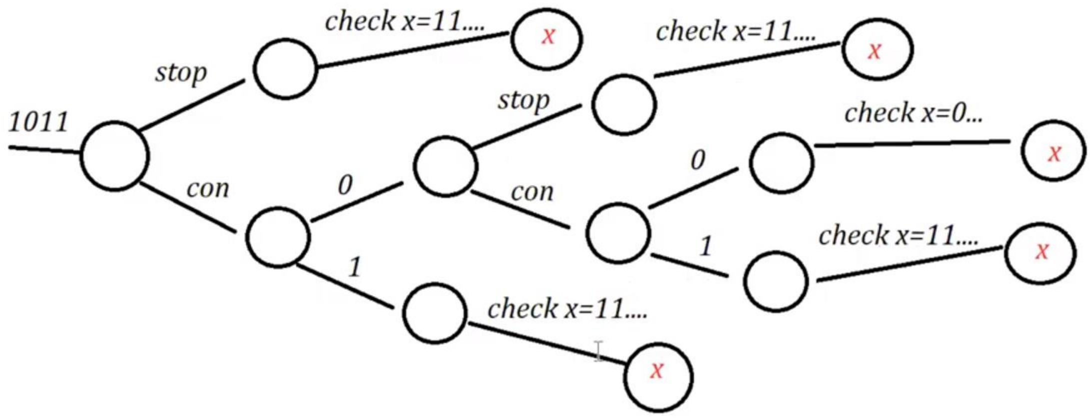
זמן ריצה = (אם אין מסלול מקבל) = אז המסלול הכי ארוך.

מתי אומרים שמכונה אי דטר' לא עוצרת?

אם מסלול מקבל אז היא **עוצרת** גם אם יש מסלולים אחרים שהם אינסופיים.

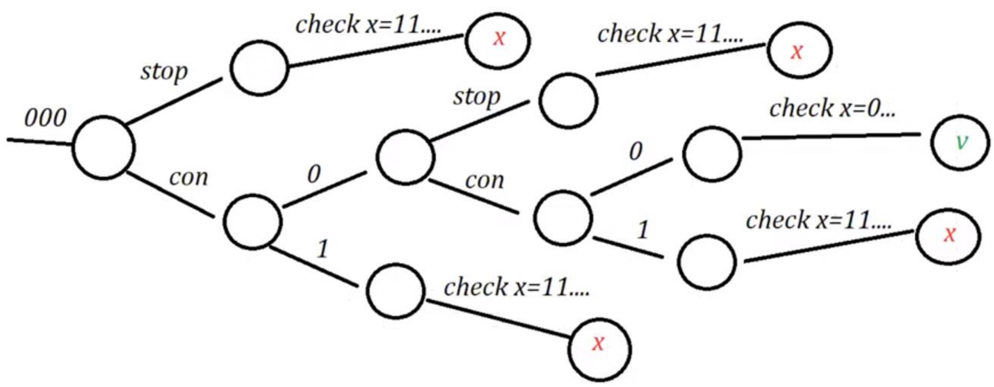
אם אין מסלול מקבל – אז אם יש אפילו מסלול אחד שלא עוצר המכונה לא עוצרת.
ואם כל המסלולים סופיים המכונה עוצרת.

בדוגמה למעלה, דוגמה לחישוב עבור $x = 1011$, שום דבר לא מתקבל.



זמן הריצה שלה יהיה 6, למה? כי זה המסלול הכי ארוך שלנו כי זה לא מקבל.

דוגמה לחישוב עבור $x = 000$:



הערה: במכונה דטר' **שתמיד עוצרת** אם מחליפים q_{accept}, q_{reject} מקבלים את השפה המשלימה.
באי- דטר' **שתמיד עוצרת**, אם מחליפים q_{accept}, q_{reject} לא בהכרח מקבלים את השפה המשלימה.

בדוגמה למעלה: שפת המכונה היא: $L(N) = \{w : w \ starts \ in \ 0 \ or \ 11\}$

אם הופכים בין המצבים המקבלים והדוחים : $L(N) = \Sigma^*$.

רדוקציה פולינומית:

$L_1 \leq_p L_2$ אמ"מ קיימת רדוקציה בין $L_1 \leq L_2$ כאשר פונקציית הרדוקציה ניתנת לחישוב בזמן פולינומי.

דוגמה:$L_1 = \{< G >: G \ is \ undirected \ graph \ and \ connected\}$

$L_2 = \{< G >: G \ is \ undirected \ graph \ and \ has \ exactly \ two \ connected \ components \ with \ same \ size\}$

נראה: $L_1 \leq_p L_2$ על ידי $f(< G >) =< G` >$. כאשר $G`$ יכיל 2 העתקים של G.

סיבוכיות: כמה זמן לוקח כדי לבנות את התוצאה של f , כלומר בהינתן G , כמה זמן ייקח לבנות את $G`$?

העתקת הגרף G – פולינומי בגודל של G.

תקפות –

אם G אז הוא רכיב אחד ולכן ב$G`$ יהיו שני רכיבים בדיוק וכל אחד מהם הוא G ולכן הם באותו גודל.

אם G לא קשיר, אז יש בו לפחות שני רכיבי קשירות ולכן ב$G`$ יהיו לפחות 4 רכיבי קשירות. (ולא 2).

תכונות: אם $L_1 \leq_p L_2$

1. כל התכונות של רדוקציה רגילה תקפות גם כאן.
2. הרדוקציה אומרת שL_2 יותר קשה מL_1 מבחינת יעילות פתרון.
   a. אם $L_1 \notin P \rightarrow L_2 \notin P$
   b. אם $L_2 \in P \rightarrow L_1 \in P$

המחלקה *NPC*:

השפות שהן NP שלמות.

באופן כללי: לכל מחלקה של שפות: $R, RE, coRE, P, NP$ ניתן להגדיר שלמות.

שפה תהיה שלמה במחלקה אם:

1. היא שייכת למחלקה.
2. היא קשה מכל השפות האחרות באותה המחלקה.

שפה *L* היא *NP* שלמה אם:

1. $L \in NP$
2. לכל שפה $A \in NP$ מתקיים ש $A \leq_p L$.

דוגמה: השפה L_u היא RE שלמה.

הוכחה:

1. $L_u \in RE$
2. תהיי L שפה בRE. ולכן קיימת לה מ"ט M_L מזהה.

נראה רדוקציה $L \leq L_u$ על ידי : $f(x) =< M_L, x >$

קשה להוכיח ששפה היא NP שלמה כי צריך להראות שהיא קשה יותר מכל האחרות באופן כללי.

אבל, אם כבר ידועה לנו שפה שהיא NP שלמה ונוכיח שL יותר קשה מהשפה הידועה אז זה יספיק כדי להראות שL היא NP שלמה. (בצירוף ההוכחה ש$L \in NP$).

במילים אחרות, אם אני יותר קשה משפה NP שלמה ואני בNP אז אני NP שלמה.

הערה: כל השפות שהן NP שלמות לא ידוע אם הן בP ואם מישהו יצליח להוכיח אפילו על אחת מהן שהיא בP אז $P = NP$.

שפות שהן *NPC*

1. $SAT = \{<\varphi>: \varphi\ is\ CNF\ formula\ and\ exists\ satisfying\ assignment\ for\ \varphi\}$

מקבלים נוסחה לוגית בצורת *CNF* כאשר בטבלת אמת של הנוסחה יש שורה שיוצאת *TRUE*, כלומר, יש הצבה למשתנים שתתן סה"כ *TRUE*.

צורת *CNF* – זה ביטוי שיש בו משתנים כאשר יש "או" בתוך הסוגריים ובין הסוגריים יש ו"גם".

כלומר : $(a \vee b \vee \dots) \wedge (\dots) \wedge \dots$ כל איבר בסוגריים הוא משתנה או שלילתו.

דוגמה: $\varphi = (x_1 \vee x_2 \vee x_3^{\neg} \vee x_4) \wedge (x_2^{\neg} \vee x_4^{\neg}) \wedge (x_1^{\neg})$

השמה מספקת: $x_1 = False, x_2 = False, x_3 = False, x_4 = True$

אם נציב את הערכים הללו נוכל לקבל *TRUE*.

או בקיצור : $(0,0,0,1)$. הצורה שהמכונה תקבל את ההשמה.

דוגמה: אין השמה מספקת

$\varphi = (x_1) \wedge (x_1^{\neg})$ לא משנה מה נקבל *FALSE*.

זה פסוק לא ספיק כי אין הצבה שתגרום לכל הפסוק להיות *TRUE*.

2. $3SAT\ \{\varphi: \varphi\ is\ 3\ CNF\ formula\ and\ \varphi \in SAT\}$.

כמו *SAT* רק שכל אחד מהסוגריים מכיל בתוכו בדיוק 3 משתנים (או שלילתם), אין הגבלה על מספר הסוגריים.

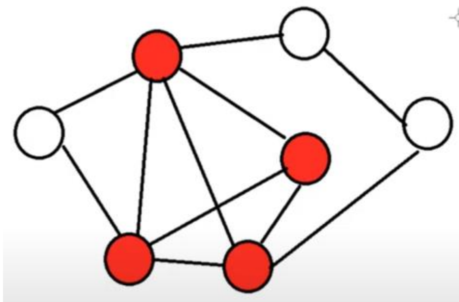
דוגמה: $\varphi = (x_1 \vee x_2 \vee x_3^{\neg}) \wedge (x_1^{\neg} \vee x_4 \vee x_2^{\neg})$

משפט *cook-levin*: הוכיח כי שפות 1,2 הן אכן *NP* שלמות, ההוכחה מסובכת מאוד וזה שייך לסיבוכיות תואר שני.

3. $CLIQUE = \{<G, k>: G\ has\ cliqye\ of\ size\ k\}$.

מקבלים גרף *G* ומספר שלם חיובי *K*. וב*G* יש קליקה בגודל *k* כאשר קליקה הוא תת גרף שלם.

דוגמה:

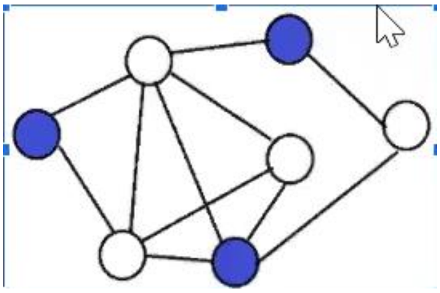


אם זה *G* ו*K* הוא 4 אז יש לנו קליקה כזאת.

4. $IS = \{<G, k>: G\ has\ independeted\ set\ of\ size\ k\}$

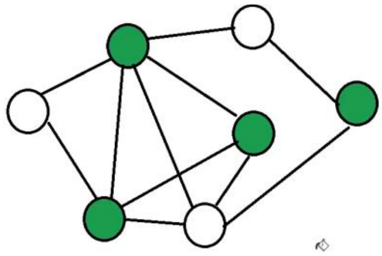
מקבלים גרף *G* ומספר חיובי שלם *k*. ב*G* יש קבוצה בלתי תלויה בגודל *k* כאשר קבוצה בלתי תלויה הוא תת גרף ריק (*k* קודקודים שאף אחד לא מחובר לאף אחד).

דוגמה:



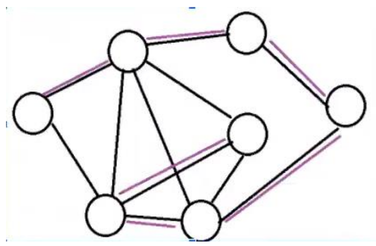
אם *K=3*.

5. $.VC = \{< G, k >: G \ has \ vertex \ cover \ of \ size \ k\}$
מקבלים גרף G ומספר חיובי שלם k. בG יש כיסוי קודקודים בגודל k כאשר כיסוי קודקודים הוא קבוצת קודקודים שנוגעת בכל צלעות הגרף. (קבוצת קודקודים שאם נוריד אותם ואת כל הצלעות שמחוברות אליהן אז הורדנו את כל הצלעות בגרף).

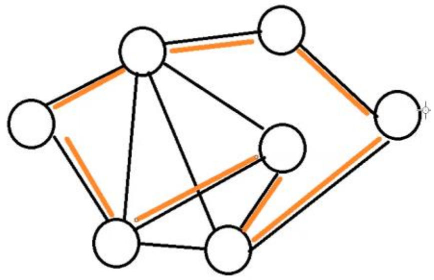


אם נוריד את הקודקודים הללו ירדו כל הצלעות בגרף.

6. $HamP = \{< G >: G \ is \ hamilton \ path\}$
מקבלים גרף G בG יש מסלול המילטוני (מסלול העובר בכל הקודקודים בדיוק פעם אחת).
בצלעות אין חובה לעבור בכולם.
דוגמה:



7. $HamC = \{< G >: G \ is \ hamilton \ cycle\}$
מקבלים גרף G בG יש מעגל המילטוני (כמו מסלול אבל מסתיים בקודקוד שהתחיל – זה הקודקוד היחיד שמותר לחזור עליו פעמיים).
בצלעות אין חובה לעבור בכולם.
דוגמה:



8. $SS = \{< a_1, a_2, a_3 \dots a_k, n >: exists \ S \subseteq \{a_1, a_2 \dots\} \ such \ that \ \Sigma_{x \in S} x = n\}$
בהינתן קבוצת מספרים וסכום n, האם יש תת קבוצה של המספרים שסכומה בדיוק n.

9. $SC = HS = \{< A, C_1, C_2, \dots, C_m, n, k > : A, C_1 \dots C_m \subseteq \{1 \dots n\}, \ \ exists \ k \ sets \ from \ C_1 to C_m s.t \ unuon \ is \ A\}$
בהינתן קבוצה A ועוד m קבוצות $C_1 \dots C_m$ כאשר כל הקבוצות מכילות מספרים בין 1 לn. האם יש k קבוצות מתוך $C_1 \dots C_m$ שהאיחוד של כולן יתן את A.

דוגמה לרדוקציה פולינומית בין השפות:

ע"י: $VC \leq_p SC$: $f(<G,k>) = <E, C_1 = \{v_1u \mid v_1u \in E\}, \ldots, C_m = \{v_mu \mid v_mu \in E\}, |E|, k>$

כאשר לכל צלע יותאם מספר בין 1 ל-$|E|$, כמות הקודקודים היא m.

סיבוכיות: ספירת הצלעות, מעבר על הקודקודים ובכל פעם מעבר על כל הקודקודים האחרים ובדיקה האם יש צלע. העתקת כמות הצלעות, העתקת K.

פולינומי בגודל הקלט.

בשאלות שבהן צריך להוכיח האם שפה היא בP או בNPC:

תחילה נבדוק האם אנו מצליחים לבנות אלגוריתם (מ"ט דט') שמוכיח את P.

ייתכן שלכאורה קשה למצוא אלגוריתם אבל יש משהו בתנאים שהופך את הבעיה לטריוויאלית יחסית.

או שאם יש הגבלה של מספר קבוע אז לפעמים חיפוש שלם לא יהיה אקספוננציאלי.

לדוגמה: אם צריך לעבור על כל תתי הקבוצות בגודל 5, מתוך קבוצה בגודל n. – שהוא: $O(n^5)$.

אם לא הצלחנו את כל מה שנאמר לעיל אז מנסים למצוא אלגוריתם עם מ"ט אי דטר'.

(ניחושים + וידוא שהניחוש יצא נכון) וככה מוכיחים שזה בNP.

לאחר מכן בוחרים את אחת מהשפות השלמות שהצגנו)הכי דומה לזו שבשאלה) ועושים רדוקציה פולינומית ממנה לשפה שבשאלה וזה כדי להוכיח שלמות.

שאלות מבחן:

2020 א' א'

1. (30 נקודות) לכל אחת מהשפות הבאות, קבעו אם היא ב P או אם היא NPC. הוכיחו את תשובותיכם.

א. $= \{(<G(V,E)>, k) \mid G$ is an undirected graph which has two vertex-disjoint cliques C_1, C_2 of respective sizes k,2k. Additionally, $|V| \geq 3k+10\}$

בעברית: זוהי שפת כל הזוגות $(G(V,E),k)$ כאשר G הוא גרף שקיימים בו זוג קליקים זרים בצמתים בגדלים $k, 2k$ בהתאמה. בנוסף, $|V| \geq 3k+10$.

ב. $= \{(<G(V,E)>, k) \mid G$ is an undirected graph which has two vertex-disjoint cliques C_1, C_2 of respective sizes k,2k. Additionally, $|V| \leq 3k+10\}$

בעברית: זוהי שפת כל הזוגות $(G(V,E),k)$ כאשר G הוא גרף שקיימים בו זוג קליקים זרים בצמתים בגדלים $k, 2k$ בהתאמה. בנוסף, $|V| \leq 3k+10$.

שימו לב: הניסוח של סעיף זה כפי שהופיע בבחינה היה מטעה. הכוונה הייתה ששתי הקלילות בגדלים $k, 2k$ נמצאות ברכיבי קשירות שונים בגרף. זה לא ברור מהכתוב. למי שמתעניין, נעיר שבמקרה יצא שגם השפה כמו שהיא כתובה שייכת ל P אך האלגוריתם משתמש ברעיונות שלא למדנו (בפרט אלגוריתם פולינומי לגרסה של שפה שנקראית partition שבה המספרים בקלט

פתרון:

א. $L_1 \in NPC$

$L_1 \in NP$ (קודם נראה שזה שייך לNP עד ידי מכונה אי דטרמיניסטית)

נגדיר מ"ט אי דטר':

N על קלט $< G, k >$ כאשר G גרף וk מספר:

- אם $|V| < 3k + 10$ – דחה.
- נחש 2 קבוצות S_1, S_2 של קודקודים בגדלים $k, 2k$ בהתאמה.
- בדוק שהן זרות, אם מצאת קודקוד שמופיע בשני הקבוצות דחה.
- עבור כל זוג קודקודים בS_1 בדוק שהם מחוברים. אם לא – דחה.
- עבור כל זוג קודקודים בS_2 בדוק שהם מחוברים. אם לא – דחה.
- קבל.

נכונות: אם קיימות קליקות המוזכר לעיל אז קיים ניחוש נכון של S_1, S_2 שיהיו הקליקות הללו וכל הבדיקות יעברו ונקבל.

אחרת, אז או שאין מספיק קודקודים בגרף או שלכל ניחוש, יש קבוצה שאינה קליקה או שהן לא זרות ולכן נדחה.

סיבוכיות: חישוב של $3k + 10$ והשוואה מול $|V|$ - פולינומי.

ניחוש 2 הקבוצות: $O(|V|^2)$

בדיקה שהן זרות: $O(|V|^2)$

בדיקה שכל זוג קודקודים מחוברים – פולינומי כפול 2.

סה"כ: פולינומי בגודל הקלט.

נראה רדוקציה מ$Clique$:

$Clique \leq_p L_1$ על ידי: $f(< G, k >) =< G', k' >$

כאשר: $k' = k$. G' יהיה כמו G עם תוספת רכיב קשירות נוסף שהוא קליקה בגודל $2k$ + 10 קודקודים בודדים.

סיבוכיות: את k מעתיקים – גודל הקלט. העתקת G עם תוספת k2 קודקודים +10 כאשר אנחנו יודעים $k \leq n$ אחרת מראש אפשר לומר שאין קליקה.

ולכן זה חסום בפולינום בגודל הקלט.

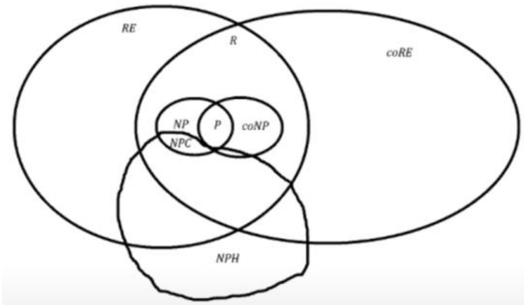
סה"כ פולינומי.

נכונות: אם יש בG קליקה בגודל K אז יש בו לפחות K קודקודים ולכן בG` יש לפחות $k + 2k + 10 = 3k + 10$ קודקודים וכן קליקה בגודל k מG + קליקה בגודל $2k$ אחרת (זאת שהוספנו)

אם אין בG קליקה בגודל k אז בG` יהיה רק קליקה בגודל $2k$ אבל לא תהיה קליקה בגודל k זרה לה.

דגשים כללים בחומר:

- מחלקות של שפות:
  בחישוביות: $R, RE, coRE$
  סיבוכיות: $P, NP, coNP, NPC$
- תמונת מצב של כל המחלקות:



הערה: בתמונת המצב הנחנו מספר הנחות (שמאמינים שכך המצב):

- ..

- $R \subseteq RE$
- $R \subseteq coRE$
- $R = RE \cap coRE$
- אם $L \in RE$ וגם $L \in coRE$ אז $L \in R$.
- לא ניתן לעשות רדוקציה משפה קשה יותר לשפה קלה!
- סדר הקושי:
  - השפות הכי קלות: Φ, Σ^*
  - השפות בR
  - השפות ב RE, באותו אופן השפות ב$coRE$ (אבל לא תמיד ניתן לעשות רדוקציה בין המחלקות).
  - השפות שלא ב$RE, coRE$.

- $P \subseteq NP$
- $NPC \subseteq NP$
- בעיות פתוחות:
  - $P =^? NP$
  - $coNP =^? NP$
  - $cpNP \cap NP =^? P$
  - האם יש שפה בNP ולא בP שהיא לא NPC?

- אם $P = NP$ אז $P = NP = coNP$ כל השפות בP פרט לΦ, Σ^* הן NPC.
  כל שתי שפות בP ניתן לעשות ביניהן רדוקציה בשני הכיוונים (פרט לΦ, Σ^*).

דגשים נוספים:

- מספר תתי הקבוצות בגודל k של קבוצה בגודל n הוא $O(n^k)$.
  לכן, אם k הוא מספר קבוע שלא תלוי בn אז זה בP.

- ידוע ש: $\binom{n}{k} = \binom{n}{n-k}$
  לכן, אם צריך לעבור על כל תתי הקבוצות בגודל $n-k$ כאשר k קבוע זה גם בP.

- סיבוכיות היא ביחס לגודל הקלט.
  לכן אם הקלט בגודל 2^m ואנו מבצעים $O(2^m)$ פעולות אז זה נחשב $O(n)$.
  וגם מצד שני, אם הקלט הוא רק מספר: $< m >$ כאשר $m \in \mathbb{N}$. אז בדרך כלל מספר ייצג במחשב בייצוג בינארי בינארי התופס $\log_2 m$ ביטים (תאי זיכרון).
  לדוגמה $< 6 > = 110$

לכן, אם עבור הקלט הזה נבצע לולאה: $for\ i = 1\ to\ m$ אז זה יהיה אקספוננציאלי כי: $m = 2^{\log_2 m} = 2^{\text{הקלט}}$ ולכן הסיבוכיות היא $O(2^n)$.

תרגילים:

מבחן 2020 א' א'

$L_3 = \{\psi(x1, \ldots, x_n) | \psi$ is a CNF formula that has (at least) two satisfying assignments ϕ_1, ϕ_2 which differ on at least 20 variables$\}$

א. בעברית: זוהי שפת כל הזוגות פסוקי ה CNF, שקיימות להן שתי השמות מספקות (הן לא חייבות להיות ההשמות המספקות היחידות) הנבדלות על לפחות 20 משתנים.

פתרון:

אינטואיציה: אפילו השמה מספקת אחת קשה למצוא כל שכן 2 השמות והתנאים לא מגבילים את המרחב.

$L_3 \in NPC$.

נראה תחילה ש $L_3 \in NP$ על ידי מ"ט אי-דטר' המקבלת אותה.

N: על קלט $< \psi >$ כאשר ψ היא נוסחת *CNF* בצע:

- נחש 2 השמות לכל משתני ψ: ϕ_1, ϕ_2.
- הצב כל אחת מההשמות בψ ובדוק שמתקבל *TRUE*. אם לא דחה.
- הגדת $count = 0$
- עבור על ההשמה ϕ_1 ובדוק עבור על משנה האם הוא בערך שונה ממה שהוא קיבל בהשמה ϕ_2 אם כן *count++*
- אם $count \geq 20$ קבל, אחרת – דחה.

סיבוכיות: ניחוש ההשמות - $O(n)$ כאשר n הוא כמות המשתנים החסום בגודל הקלט.

כעת נראה שהשפה שלמה, נראה רדוקציה משפה שהיא שלמה: $SAT \leq_p L_3$ ומכיוון ש*SAT* היא *NP* קשה נובע שגם L_3 היא *NP* קשה.

רדוקציה: $f(< \psi >) =< \psi' >$

כאשר: $\psi' = \psi \wedge (y_1 \vee y_2 \vee \ldots \vee y_{21})$

- בעצם הy_{21} יהיה תמיד $true$ ולכן עם כל היתר ניתן לשחק איך שרוצים, לכן נקבל את ההשמה הראשונה שהיא *true* לפי *SAT* וגם מיקס של עוד 20 משתנים לפחות שיתן לנו *true* .
אם ψ מההתחלה הוא $false$ אז אנחנו אמורים לדחות בהתאם.

סיבוכיות: העתקת הקלט + תוספת של קבועה (של 20 משתנים), חסום בגודל הקלט

תקפות: אם יש לψ השמה מספקת, אז ניקח את אותה השמה ביחד עם $(y_1 = y_2 = \cdots = y_{21} = True)$ ונקבל השמה מספקת ל ψ' וגם אותה השמה כאשר $y_1 = \cdots = y_{20} = false\ and\ y_{21} = true$ נקבל עוד השמה מספקת לψ' , השונה ב20 משתנים מהקודמת.

אם אין לψ השמה מספקת אז לכל בחירת השמה למשתני ψ נקבל $false$ בחלק הזה ולכן הכל יהיה $false$ תמיד.

2. (30 נקודות) לכל אחת מהשפות הבאות, קבעו האם היא ב R והאם היא ב RE. הוכיחו את תשובתכם. רמז: ישנה שפה אחת לכל אחד משלושת הסוגים.

א. $L_1 = \{< M > | \exists < M' >, \text{where} \ | < M' > | < | < M > |, \text{and } L(M) \subseteq L(M')\}$

בעברית: L_1 היא אוסף כל קידודי המכונות $< M >$ כך שקיימת מכונה בעלת קידוד קצר מזה של M, שהשפה שלה מכילה את השפה של M.

ב. $L_2 = \{< M > | \exists < M' >, \text{where} \ | < M' > | > | < M > |, \text{and } (\Sigma^2)^* \cap L(M) \subset (\Sigma^2)^* \cap L(M')\}$

בעברית: L_2 היא אוסף כל קידודי המכונות $< M >$ כך שקיימת מכונה בעלת קידוד הארוך מזה של M, שקבוצת המילים באורך זוגי בשפה שלה מכילה **ממש** את קבוצת המילים באורך זוגי בשפה של M.

ג. $L_2 = \{< M > | \exists < M' >, \text{where} \ | < \mathrm{M} > | - | < M' > | \in \{2, 3\} \text{ and } L(M) \cap L(M') \neq \phi\}$

בעברית: L_2 היא אוסף כל קידודי המכונות $< M >$ כך שקיימת מכונה בעלת קידוד הקצר מזה של M ב **2** או **3**, שהשפה שלה נחתכת עם השפה של M.

ס' א:

הערה :הקדמה: לכל מכונה בעולם יש קידוד! ולהיפך, לכל מילה בא"ב מתאימים מכונה.
דוגמה:

1. ε – מתארת מכונה שלא עוצרת על אף קלט.
2. 0 – מכונה שעבור המילה 111 מבצעת 100 צעדים ואז מקבלת. ועבור שאר המילים דוחה.
3. 1 – מכונה שמקבלת הכל.

...

(אתם לא מחליטים מה מייצג מה אלא רק צריכים לדעת שכל מ"ט מיוצגת על ידי רצף בינארי וכל רצף בינארי מייצג מ"ט וייתכן שיהיו שני רצפים המייצגים את אותה המכונה אם לא נאמר אחרת).

נעבור לפתרון השאלה.

קיים קידוד באורך k המייצג את המכונה M' שמקבלת הכל, כלומר $L(M') = \Sigma^*$.

ולכן כל מ"ט שאורך הקידוד שלה גדול מk תתקבל.

מכאן, המשלימה של L_1 היא שפה סופית ולכן היא בR ומכאן גם המשלימה שלה – L_1 היא בR.

- בעצם הבעיה שלי היא על מכונות שהגודל שלהן הוא פחות k שהוא מספר קבוע, כל מה שמעל k מתקבל אבל מה אם מה שמתחת? הבחירה שלנו ב Σ^* היא מאוד כללית ויכולה להכיל הכל אבל האם יש עוד מכונות עם קידוד קצת יותר אשר יגמרו למכונות עם קידוד קטן מk להתקבל? אנחנו לא יודעים, אבל מה שכן זה נותן לנו מספר סופי של מצבים.

ס' ב:

השפה לא בRE (וכל שכן שהיא לא בR)

נראה רדוקציה מ $\overline{HP}$ (המשלימה של HP) כלומר $\overline{HP} \leq L_2$: $f(<N,x>) =<M>$

כאשר M : על הקלט w:

- הרץ את N על x.
- קבל.

נכונות: אם N לא עוצרת על x אז M לא עוצרת על אף קלט ולכן $L(M) = \Phi$. מכיוון שהקידוד של $<M>$ הוא באורך סופי וקיימות אינסוף שפות כריעות המורכבות ממילים באורך זוגי, בהכרח קיימת מ"ט M' עם קידוד ארוך יותר משל M שמקבלת לפחות מילה אחת באורך זוגי ולכן התנאי מתקיים.

אם N עוצרת על x אז M מקבלת הכל. ולכן: $\Sigma^{2^*} \cap L(M) = \Sigma^{2^*}$ ולא קיימת מ"ט M' כך ש:

$\Sigma^{2^*} \subset \Sigma^{2^*} \cap L(M')$.

ס' ג:

השפה בRE ולא R.

הוכחת שייכות ל RE : נראה מ"ט אי-דטר' המזהה את L_3:

N: על קלט $<M>$ כאשר M היא מכונה, בצע:

- חשב: $s = |<M>|$
- נחש מילה y באורך $s-2 \; or \; s-3$.
- סמן את המכונה המתקבלת מהקידוד y כ: M'.
- נחש מילה x
- הרץ את M על x, אם דחתה – דחה.
- הרץ את M' על x, אם דחתה – דחה.
- אחרת – קבל .

הוכחת אי שייכות לR – נראה רדוקציה: $HP \leq L_3$ על ידי: $f(<N,x>) =<M>$.

כאשר M : על קלט w :

- הרץ את N על x . (אם N לא עוצרת על x גם M לא תעצור ולכן החיתוך יהיה ריק ושניהם לא יעצרו)
- קבל.

אם N עוצרת על x אז M מקבלת הכל ולכן $L(M) = \Sigma^*$ ומכיוון שיש אינסוף מכונות שמקבלות לפחות מילה אחת אז בהכרח ניתן ליצור קידוד של מכונה ש"קרוב" באורכו לקידודים הללו ולכן החיתוך יהיה לא ריק.

אם N לא עוצרת על x אז M לא עוצרת על אף קלט ולכן : $L(M) = \Phi$ ובפרט החיתוך יצא ריק.

3. (31 נקודות)

א. (9 נקודות) בסעיף זה, נניח שחוקר מוכשר הצליח למצוא אלגוריתם דטרמיניסטי ל SC, ומימש אותו באמצעות מכונת טיורינג M_{SC}. תארו (במילים) מימוש של מכונת טיורינג דטרמיניסטית יעילה שבהינתן קלט $(n, k, C_1, \ldots, C_t)$ מחזירה כיסוי בקבוצות של $[n]$ על ידי k קבוצות, או מחליטה שאין כזה. אין צורך להוכיח את נכונות האלגוריתם או לנתח את הסיבוכיות שלו.

ב. (3 נקודות) בהנחה שהצלחתם לפתח את האלגוריתם בסעיף הקודם (גם אם לא הצלחתם), הראו כיצד להשתמש באלגוריתם שלכם כדי לבנות אלגוריתם יעיל, שבהנתן קלט $(n, C_1, \ldots, C_t)$ מחזירה את הכיסוי בקבוצות הקטן ביותר האפשרי עבור $[n]$.

ג. (6 נקודות) הוכח או הפרך: יהיו $L_1, L_2 \subseteq \Sigma^*$. אזי אם $L_1 \leq_p L_2$ ו $L_2 \in NP$ גם $L_1 \in NP$. שימו לב שהמשפט לא זהה למשפט הרדוקציה עבור רדוקציות פולינומיות שלמדנו בכיתה.

ד. (5 נקודות) הוכח או הפרך: יהיו $L_1, L_2 \subseteq \Sigma^*$. אזי אם $L_1 \leq_p L_2$ ו $L_2 \notin NP$ אז גם $L_1 \notin NP$.

ס' א:

אלגוריתם: על קלט $< n, k, C_1 \ldots C_t >$ בצע:

- הרץ את M_{sc} על הקלט $< n, k, C_1 \ldots C_t >$ אם דחתה –
  - החזר Φ.
  - אחרת, הגדר $Ans = \Phi$
- סמן: $X = < C_1 \ldots C_t >$
- עבור $i = 1\ to\ t$ בצע:
  - הרץ את M_{sc} על $< n, k, X \backslash C_i >$ כאילו X ללא C_i.
  - אם קיבלה (ז"א שC_i לא נצרכת על מנת לכסות את הכל) –
    - הגדרת את $X = X \backslash \{C_i\}$.
  - אם דחתה –
    - $Ans = Ans \cup \{C_i\}$
- החזר את Ans .

ס' ב:

נניח שלמכונה בס' א' קוראים M.

אלגוריתם למציאת כיסוי מינימלי:

על קלט $< n, C_1 \ldots C_t >$ בצע:

- עבור k מ1 עד t בצע:
  - הרץ את M על $< n, k, C_1 \ldots C_t >$
  - אם חזר משהו שונה מΦ אז החזר אותו.
- החזר Φ.

ס' ג:

הוכחה: לפי הנתון $L_1 \leq_p L_2$ נובע שקיימת פונקציה f כך שלכל $x \in \Sigma^*$ מתקיים:

$f(x) \in L_2 \leftrightarrow x \in L_1$. ומכאן קיימת מ"ט דטרמיניסטית פולינומית M_f המחשבת את f.

בנוסף, לפי הנתון $L_2 \in NP$ נובע שקיימת מכונה א"ד פולי N_2 המכריעה את L_2.

צ"ל: $L_1 \in NP$.

נגדיר את המ"ט הבאה: N_1 א"ד : על קלט x:

- הרץ את M_f על x וקבל את $y = f(x)$.
- הרץ את N_2 על y וענה כמוהו.

סיבוכיות: פולינומית בx ופולינומית ב$f(x)$ והרכבה של פולינומים היא פולינומית.

נכונות: $x \in L_1$ אם ורק אם $y = f(x) \in L_2$ אם ורק אם y התקבלה בN_2 אם ורק אם N_1 קיבלה את x.

ס' ד:

לא בהכרח נכון. דוגמה נגדית: $HP \leq_p \Phi$ על ידי : $f(x) = < M, 0 >$ כאשר M היא מ"ט שלא עוצרת על אף קלט.

סיבוכיות הרדוקציה: $O(1)$ – לא תלוי בקלט x תמיד מחזירים את אותה מכונה ואותה מילה.

$HP \notin NP$ אבל $\Phi \in NP$ כי $\Phi \in P$ ומתקיים: $P \subseteq NP$.

ס' ה:

ה. (8 נקודות) נגדיר מחלקה חדשה:

$_q = \{L|$There exists a non deterministic 2-tape polynomial turing machine M such
t L(M)=L. Furthermore, for every x, all paths
he computation tree of M on x are of the same length.$\}$

בעברית: זו מחלקת כל השפות שיש להן מכונת טיורניג א"ד פולינומית דו-סרטית המקבלת אותן, ובנוסף לכל מילה, כל המסלולים בעץ החישוב של M על x הם באותו אורך. הוכיחו ש $NP \subseteq NP_{eq}$. הערה: δ במ"ט דו-סרטית מוגדרת כמו במ"ט דטרמיניסטית, אלא שיש שתי בחירות אפשריות. המכונה היא דו סרטית רק כדי להקל על פתרון השאלה - מי שמעדיף, שיעבוד עם הגדרה של NP_{eq} לאורך מ"ט חד סרטית.

פתרון:

תהי $L \in NP$ ולכן קיימת עבור L מ"ט N_L א"ד פולי' כך שלכל x זמן הריצה של N_L חסום ב$c \cdot |x|^t$. (אורך המסלול הארוך ביותר).

- בעצם החזקת t כי זה החזקה של הפולינום ביחס לקלט x. שזה שקול $O(n^t)$ אני לא יודע כמה נרוץ.

נבנה מ"ט חדשה א"ד פולי' עם 2 סרטים אשר תתנהג בדיוק כמו N_L בסרט הראשון כאשר בסרט השני יהיה מונה צעדים המתחיל מ$c \cdot |x|^t$ (אשר יחושבת לפני הצעד הראשון של N_L) ולאחר כל צעד של N_L המונה יקטן ב1, ברגע שN_L עצרה אז נמשיך לעשות צעדי סקר תוך כדי הורדת המונה ל0 ונעבור למצב העצירה המתאים (מקבל או דוחה, בהתאם לפעולת N_L).

לא ייתכן שהמונה יגיע ל0 לפני N_L תעצור כי הוא מאותחל למספר הצעדים המקסימלי שN_L מבצעת על בכל x מסלול חישוב שלה.

סיבוכיות: סיבוכיות המכונה N_L כפול סיבוכיות עדכון המונה (פולי' בגודל הקלט).

כל המסלולים בדיוק באורך של $c \cdot |x|^t$ + זמן עדכון המונה הכולל מתחילתו ועד 0.

4. (**19** נקודות)

א. (**12** נקודות) בסעיף זה נחקור הרחבה של משפט Rice עבור זוג מכונות.
עבור תכונה של שפות ב RE, נגדיר
$L_S^{\times 2} = \{(< M_1 >, < M_2 >) | L(M_1) \in S, L(M_2) \in S\}$.

א.1 (**4** נקודות) נסחו גרסה של משפט Rice הטוענת עבור אילו תכונות S השפה $L_S^{\times 2}$ שייכת ל R.

א.2 (**7** נקודות) הוכיחו בקצרה את המשפט מהסעיף הקודם.

ב. (**8** נקודות) הוכח או הפרך: ניתן להוכיח שהשפה הבאה אינה ב R באמצעות משפט Rice (המקורי, לא המשפט מסעיף א). כ
$L = \{< M > | L(M) = \phi, \text{ and M makes at least } |x| \text{ steps on every input x}\}$
כלומר נסחו את השפה כ L_S עבור S מתאימה, או הוכיחו שלא קיימת S כזו:

בעברית: L היא שפת כל קידודי המכונות $< M >$, כך ש M אינה מקבלת אף מילה, ועל כל מילה מבצעת לפחות $|x|$ צעדי חישוב.

ס' א:

א.1 – תהי S תכונה לא טריוויאלית של שפות ב RE , אזי: $L_S^{\times 2} \notin R$.

ואם S טריוויאלית אז $L_S^{\times 2} \in R$.

א.2 – אם S טריוויאלית. אז אם $S = RE$ אז כל הזוגות של המכונות יקיימו את תנאי $L_S^{\times 2}$ ולכן $L_S^{\times 2} = \Sigma^*$.

ואם $S = \Phi$ אז אף זוג לא יקיים את התנאי $L_S^{\times 2}$ ולכן: $L_S^{\times 2} = \Phi \in R$.

אם S לא טריוויאלית, אם $\Phi \notin S$ אז קיימת $L_1 \in S$ ולכן קיימת מ"ט M_{L_1} המזהה את L_1 כך ש:

$L(M_{L_1}) = L_1$ נראה רדוקציה מHP ל$L_S^{\times 2}$. כלומר $HP \le L_S^{\times 2}$ על ידי : $f(< M, x >) =< M_1, M_2 >$.

M_1, M_2 על קלט w:

- הרץ את M על x
- הרץ את M_{L_1} על w וענה כמוה.

אם M עוצרת על x , M_1, M_2 מתנהגות כמו M_{L_1} ולכם השפה שלהן היא $L_1 \in S$ ולכן הן מקיימות את התנאי $L_S^{\times 2}$.

אם M לא עוצרת על x , אז M_1, M_2 לא עוצרת כלל ולכן השפה שלהן היא $\Phi \notin S$ ולכן הן לא מקיימות את התנאי $L_S^{\times 2}$.

אם $\Phi \in S$ אז קיימת $L_1 \in S$ ולכן קיימת מ"ט M_{L_1} המזהה את L_1 כך ש:

$L(M_{L_1}) = L_1$ נראה רדוקציה מ$HP^{\neg}$ ל$L_S^{\times 2}$. כלומר $HP^{\neg} \le L_S^{\times 2}$ על ידי : $f(< M, x >) =< M_1, M_2 >$.

M_1, M_2 על קלט w:

- הרץ את M על x
- הרץ את M_{L_1} על w וענה כמוה.

אם M עוצרת על x , M_1, M_2 מתנהגות כמו M_{L_1} ולכם השפה שלהן היא $L_1 \notin S$ ולכן הן לא מקיימות את התנאי $L_S^{\times 2}$.

אם M לא עוצרת על x , אז M_1, M_2 לא עוצרת כלל ולכן השפה שלהן היא $\Phi \in S$ ולכן הן מקיימות את התנאי $L_S^{\times 2}$.

מסקנה ממשפט הרדוקציה - $L_S^{\times 2} \in R$.

ס' ב:

לא ניתן להשתמש במשפט רייס עבור השפה הנ"ל.

הוכחה: נראה שיש שני מכונות עם אותה שפה כך שאחת מהן מקיימת את תנאי L ואחת לא.

M_1 על קלט x:

- עבור על כל הקלט (עד ה*blank*) ודחה.

M_2 על קלט x:

- דחה.

נשים לב $L(M_1) = L(M_2) = \Phi$ אבל M_1 מבצעת לפחות $|x|$ צעדים על כל x ולכן $< M_1 > \in L$ לעומת זאת, M_2 לא מבצעת $|x|$ צעדים אלא רק צעד אחד ולכן $< M_2 > \notin L$.