<u>מרתון חישוביות</u>

מודל חישובי – מכונת טיורינג:

המכונה בנויה כמו אוטומט מהספר מצבים סופי ובנוסף יש לה סרט זיכרון אינסופי המתחיל ממיקום 1.

תזה של צרץ וטיורינג: מכונת טיורינג שקולה למחשב.

איך המכונה עובדת?

התחלה: המכונה המצב התחלתי: q_0. ועל הסרט זיכרון כתוב הקלט x (הכתוב משמאל לימין החל מתא 1)

למכונה יש ראש קורא כותב המתחיל (מצביע) בתא 1.

צעד: המכונה קוראת את התו שנמצא במיקום בזיכרון שהראש מצביע אליו ולפי פונקציית המעברים היא מחליטה: לאיזה מצב לעבור, מה לכתוב במקום התו הזה, לאיפה להזיז את הראש הקורא\כותב (ימינה שמאלה להישאר במקום).

פונקציית המעברים מקבלת: מצב ואת התו שהראש מצביע אליו ובהתאם לשני אילו היא עוברת למצב חדש (או נשארת באותו מצב), כותבת תו אחר באותו מקום בזיכרון או משאירה את אותו תו ומתקדמת עם הראש.

הערה: אם המכונה מצביעה לתא השמאלי ביותר ורוצה ללכת שמאלה – היא נשארת במקום. (אבל אנחנו לא יודעים את זה).

בשונה מאוטומטים המכונה לא עוצרת בסיום קריאת הקלט!!!

לכן למכונה יש מצבים מיוחדים הנקראים מצבי עצירה ברגע שהמכונה מגיע אליהם היא עוצרת.

(בדומה לreturn בתוך פונקציה)

הגדרה פורמלית של מכונת טיורינג:

מ"ט מורכבת כך: $M = (Q, \Sigma, \Gamma, q_0, b, \delta, F)$

Q - קבוצה **סופית** של מצבי המכונה.

Σ – קבוצה **סופית** של התווים המרכיבים את הקלט של המכונה.

Γ – קבוצה **סופית** של התווים שניתן לכתוב אותם בסרט הזיכרון של המכונה. $\Sigma \subsetneq \Gamma$ וגם $b \in \Gamma$

q_0- מצב התחלתי.

b – התו הריק שכתוב בסרט הזיכרון.

δ – פונקציית המעברים.

F – קבוצה סופית של מצבי עצירה. $F \subsetneq Q$

<u>ישנן שני סוגי מכונות:</u>

1. המכונה המחשב פונקציה : מקבלת את הקלוט x כרגיל וכשעוצרת, היא מוציאה פלט – הפלט הוא כל מה שכתוב בסרט הזיכרון מתא 1 ועד התא שבו נמצא הראש (לא כולל)

   דוגמה למכונה המחשבת את הפונקציה $f(x) = x + 1$

   (זה חישוב של 1+ בבינארי)

   אלגוריתם: על קלט x :

   - הזז את כל הקלט צעד אחד ימינה (בתו הראשון שים b)
   - כל עוד לא הגעת לb התקדם ימינה. (קרא את כל הקלט עד הB הראש)
   - חזור לתו האחרון.(צעד אחד שמאלה)

- אם התו הוא 0 כתוב 1 במקומו, התקדם ימינה ועצור.
- כל עוד התו הוא 1 כתוב 0 והתקדם צעד אחד שמאלה.
- אם חזרת להתחלה – כתוב שם 1, קרא שוב את כל הקלט עד ה*b* הראשון ועצור.
- אם לא חזרת להתחלה- קרא שוב את כל הקלט עד ה*b* הראשון ועצור.

ייתכן והמכונה לא תעצור (עבור קלט ספציפי), אם היא לא מגיעה למצב עוצר (לדוגמה תקועה בלולאה אינסופית).

במקרה כזה הפלט לא מוגדר (הפונקציה לא מוגדרת עבור הקלט הזה).

פונקציה מלאה - אם המכונה עוצרת (ואז מחזירה פלט) לכל קלט אז הפונקציה שהיא מחשבת נקראת פונקציה מלאה. (מוגדרת לכל קלט).

מתמטית: אם פונקציה מוגדרת לכל קלט אז היא מלאה (גם אם אין מכונה שמחשב אותה).

- לדוגמה: $f(x) = \frac{1}{x}$ – לא מלאה כי עבור $x = 0$ אין לה תשובה.

פונקציה הניתנת לחישוב – כל פונקציה שניתן לבנות לה מ"ט שתחשב אותה נקראת פונקציה ניתנת לחישוב.

- לדוגמה: $f(M, x) = P$ כך ש P הוא רשימה של כל צעדי המכונה M כאשר היא מקבלת את הקלט x . לא ניתן לחישוב כי אם M לא עוצרת על x אז יש לה אינסוף צעדים עליו ולא ניתן בזמן סופי לכתוב לתוך P את כל צעדי המכונה M.

(מלאה לא תלוי במכונה ורק צריך להיות מוגדר לכל *x* וניתנת לחישוב זה תלוי מכונה שעוצרת).

הערה: יכול להיות לנו כל הקומבינציות של הנ"ל, האחד לא תלוי בשני.

2. מכונה המכריעה שפה : מקבל או לא מקבלת (כמו אוטומטים)
למכונה כזו יש שני מצבי עצירה שונים: q_{accept}, q_{reject}
אם המוכנה מגיעה לq_{accept} אז המילה שניתנה כקלט בהתחלה מתקבלת במכונה.
(גם אם נעשו שינויים בקלט, אנו מתייחסים לקלט שהיה בהתחלה).
אם המוכנה מגיעה לq_{reject} אז המילה שניתנה כקלט בהתחלה לא מתקבלת.

אם המכונה לא עוצרת, אנחנו מחשיבים זאת (מתמטית) כאי קבלה אך לא ניתן לדעת את תשובת המכונה.

הסיבה שלא ניתן לדעת היא – אם המכונה לא עוצרת זה **לא** אומר שהיא לא תעצור בהמשך (אולי החישוב ארוך מאוד).

דוגמה למכונה המכריעה את $L = \{a^n b^n c^n | n \in \mathbb{N}\}$

אלגוריתם: על קלט x:

- בדיקה שהמילה מהצורה $a^n b^n c^n$
  - כל עוד קראת *a* התקדם צעד ימינה.
  - כל עוד קראת *b*, התקדם צעד ימינה.
  - אם לאחר קריאת *b* קראת *a*, - דחה.
  - כל עוד קראת *c* – התקדם צעד ימינה.
  - אם לאחר קריאת *c* קראת *a* או *b* – דחה.
- חזור לתחילת הקלט.
- אם התו הראשון הוא *blank* –קבל.

- כל עוד התו שעומדים עליו הוא לא X
  - אם התו הוא a כתוב X במקומו.
  - כל עוד קוראים a או X התקדם ימינה.
  - אם הגעתם ל$blank$ - דחה
  - ברגע שהגעת לb הראשון (התו) כתוב X במקומות והתקדם ימינה.
  - כל עוד קוראים b או X התקדם ימינה.
  - אם הגעתם ל$blank$ - דחה
  - ברגע שהגעת לc הראשון כתוב X במקומו וחזור שמאלה עד הa הראשון.
  - אם חזרת להתחלה ולא מצאת a:
    - עבור כל המילה מההתחלה אם יש בה רק X-ים (ללא abc קבל)
    - אם מצאת b או c דחה.

מושגים:

- שפה של מכונה : $L(M)$ – קבוצת כל המילים ב Σ^* שהמכונה מקבלת אותן (אם מכניסים מילה מהקבוצה כקלט למכונה אז היא עוצרת ומקבלת אותה).

- קונפיגורציה: תיאור של מצב נוכחי של המכונה באמצע עבודתה (או בהתחלה).
הקונפ' מורכבת מהמצב (בQ) שבו המכונה עומדת כרגע, מה כתוב בכל סרט הזיכרון, איפה עומד הראש קורא כותב.

- מכונה מכריעה של שפה L: מכונה שמקבלת כל מילה ששייכת לL ודוחה כל מילה שלא שייכת לL ו**תמיד עוצרת**.

- מכונה מזהה של שפה L: מכונה שמקבלת כל מילה ששייכת לL אבל עבור מילים שלא שייכות לL היא **יכולה לדחות והיא יכולה לא לעצור.**

- שפה כריעה: שפה שקיימת עבורה מ"ט מכריעה (שתמיד עוצרת).

- שפה מזוהה טיורינג: שפה שקיימת לה מ"ט מזהה.

- הערה: אם שפה היא כריעה אז היא גם מזהה. (כי מותר למכונה מזהה גם לעצור תמיד ולדחות במקרה והקלט לא בשפה).

שקילות מודלים:

ניתן להגדיר מספר מודלים של מ"ט שונות המתנהגות אחרת.

נאמר ש2 מודלים של מ"ט שקולים אם:

- כל פונקציה שהמודל הראשון יכול לחשב גם המודל השני יכול לחשב. ולהפך.
- כל שפה שהמודל הראשון יכול להכריע, גם המודל השני יכול להכריע ולהפך.

כדי להוכיח שקילות: צריך להוכיח שני כיוונים, להראות שאם קיים מודל 1 עבור פונקציה\שפה אז איך ניתן לבנות מי זה את מודל 2 לאותה הפונקציה\שפה.

כדי להוכיח אי שקילות – מספיק להראות דוגמה לשפה אחת\פונקציה שמודל אחד יכול לחשב והשני לא.

תרגילים:

1. הוכיחו או הפריכו: המודל הרגיל של מ"ט שקול למודל של מ"ט שהראש קורא כותב שלה יכול לזוז רק ימינה רק להישאר במקום.

<u>פתרון:</u> נראה שהמודל החדש שקול או חלש לאוטומט סופי ובכך לא שקול למ"ט שמקבלת גם שפות לא רגולריות.
תהא L שפה שיש לה מכונת טיורינג מכריעה מהמודל החדש. נראה אוטומט סופי עבור השפה הנ"ל:
נגדיר את קבוצת המצבים של האוטומט להיות המכפלה של מצבי המכונה עם תווי Γ.
כלומר : $Q_A = Q_M X \Gamma_M$
לכל מצב q ולכל תו x אם המכונה רק נשארת במקום (ולא משנה מה כותבת) – עבור למצב בור.
נסמן בq את המצב בו המכונה בפעם הראשונה זזה ימינה על התלו של הקלט a.
$\delta_A\big((q,x),a\big) = (p,a)\ if\ \delta_M(q,x) = (p,y,R)$
$\delta_A\big((q,x),\varepsilon\big) = (p,y) if\ \delta_M(q,x) = (p,y,S)$
אם המכונה עוברת למצב מקבל אז באוטומט נעבור למצב מקבל שלא ניתן לצאת ממנו.
אם המכונה עוברת למצב דוחה אז באוטומט .... להשלים.

2. הוכיחו או הפריכו המודל הרגיל של מכונת טיורינג שקול למודל הזהה למכונת טיורינג עם אינסוף ראשים קוראים כותבים.
כל אחד מהראשים מתחיל בתא השמאלי ביותר, ובכל צעד מחליטים עבור כל ראש בנפרד האם לזוז ימינה שמאלה או להישאר במקום.
פונקציית המעברים תראה כך : $\delta(q,a) = (p,b,Movment)$
כאשר a וb הם ווקטורים $a,b \in \Gamma^{\infty}$ ו $M \in \{L,S,R\}^{\infty}$

- כלל: אם ניתן לראות את כל המילה של הקלט בבת אחת אז ניתן לקבל כל דבר.
- הערה: אם כל הראשים כותבים למקום אחד, נחליט שהתו ילך לפי הראש הראשון ביותר שעומד מעל אותו תא (מבחינת מספר הראש).

פתרון:

המודלים לא שקולים אלא המודל החדש חזק יותר.

נראה שהמודל החדש יכול להכריע כל שפה וזה יראה שהם שלא שקולים כי המודל הרגיל לא יכול להכריע כל שפה (יש שפות שהן לא כריעות).

תהי L שפה. נראה מ"ט מהמודל החדש המכריעה את L.

הפונקציה תראה כך: $\delta\big(q_0,(\sigma_1',\dots\sigma_i',\sigma_{i+1},\dots)\big) = \big(q_0,(\sigma_1',\dots\sigma_i',\sigma'_{i+1},\dots),(S,S,\dots S,R,R\dots)\big)$

כלומר, לאחר שקראנו i תווים וסימנו אותם ב ׳, קוראים את התו הראשון שלא סומן, מסמנים אותו הראש שמעליו נשאר במקום וכל השאר הראשים שמימינו זזים אחד ימינה.

כאשר:$\sigma_i \in \Sigma$

בנוסף: $\delta\big(q_0,(\sigma_1',\dots\sigma_n',b,b,\dots)\big) = \big(q_h,(\sigma_1',\dots\sigma_n',b,b,\dots),(S,S,\dots S,S,S\dots)\big)$

כך ש: אם $\sigma_i \dots \sigma_n \in L$ אז *h = accept* ואחרת *h= reject*.

<u>מחלקות חישוביות:</u>

מחלקה = קבוצה של שפות.

המחלקות הן:

1. R - קבוצת כל השפות הכריעות. (יש מכונה מכריעה לשפה שתמיד עוצרת ועונה נכון.)
2. RE – קבוצת כל השפות המזוהות. (יש מכונה שעונה נכון אם המילה בשפה ואם המילה לא בשפה אז או שהמכונה דוחה או שלא עוצרת).
3. $coRE$ – קב' כל השפות שהמשלימה שלהן מזוהה. (יש מכונה שעונה נכון אם המילה לא בשפה [דוחה אותה] ואם המילה בשפה המכונה עוצרת ומקבלת או לא עוצרת).

תכונות:

1. $R = RE \cap coRE$ – כלומר אם הוכחנו ששפה היא בRE וגם ב$CORE$ אז היא בR.
2. $R \subseteq RE$ וגם $R \subseteq coRE$.
3. יש שפות שהן לא בRE ולא ב$coRE$.

הוכחה:
כמות השפות השונות בעולם היא כל תתי הקבוצות של Σ^* . כלומר: $|P(\Sigma^*)| = א$.
בעצם יש לה סדר ולכן היא בת מנייה.
כמות מכונות הטיורינג השונות שיש בעולם (כמות התוכניות השונות ב$JAVA$ או בC שאפשר לייצר)
היא : בגודל $\aleph_0$. (כי מדובר בכמות תווים סופים המרכיבה את קוד התוכנית).
מסקנה: יש יותר שפות ממכונות ולכן בהכרח יש שפה שאין לה מכונה שמקבלת אותה.
$|RE| = |coRE| = \aleph_0$

תכונות סגור: (סגירות של שפות בתוך המחלקות)

- במחלקה R:
  פעולות אונאריות: אם $L \in R$ אז:
  1. $\bar{L} \in R$
  2. $L^R \in R$
  3. $L^* \in R$

  פעולות בינאריות: אם $L_1, L_2 \in R$ אז:

  1. $L_1 \cup L_2 \in R$
  2. $L_1 \cap L_2 \in R$
  3. $L_1 \cdot L_2 \in R$
  4. $L_1 \setminus L_2 \in R$

- במחלקה RE:
  פעולות אונריות: אם $L \in RE$ אז :
  1. אם סגירות בהכרח למשלים.
  2. $L^R \in RE$
  3. $L^* \in R$

  פעולות בינאריות: אם $L_1, L_2 \in RE$ אז:

  1. $L_1 \cup L_2 \in RE$
  2. $L_1 \cap L_2 \in RE$
  3. $L_1 \cdot L_2 \in RE$
  4. $L_1 \setminus L_2$ לא סגור.

- במחלקה $coRE$ הוא כמו RE.
- **דגש**: בהכלה שום דבר לא סגור.

קידודים:

לכל אובייקט יש מחרוזת של אפסים ואחדות המתארת אותו.

עבור אובייקט O נסמן את הקידוד שלו ב$< O >$.

גם מכונת טיורינג היא אובייקט ולכן ניתן לשלוח למ"ט קלט שהוא קידוד של מ"ט אחרת (או אפילו את הקידוד של המכונה עצמה).

לכל מכונת טיורינג יש קידוד (ואף אינסוף קידודים שיכולים לתאר את אותה מכונה).

הקידוד מכיל בתוכו את תיאור של מצבי המכונה מספר מצבים מצב התחלתי א"ב סרט וקלט ופונקציית המעברים.

אם קידוד המכונה אינו תקין, לדוגמה: *0*. נאמר שבכל זאת הוא מייצג מכונת טיורינג המוסכמת לפי הקורס – חלק אומרים מכונה שלא עוצרת על כלום ויש כאלו שמסכימים על מכונה שמקבלת הכל.

<u>כל הקידודים הלא תקינים מתארים את אותה המכונה.</u>

דוגמה לשפות:

$$L = \{< M > | M \text{ is turing machin and } \varepsilon \in L(M)\}$$

האם השפה הנ"ל כריעה?

אינטואיציה: השפה לא כריעה כיוון שאם M לא עוצרת, לא נוכל לדעת אם היא מקבלת את המילה הריק או לא.

$$L = \{< M > | M \text{ does at most } 3 \text{ steps on input } 010\}$$

אינטואיציה: השפה כריעה כי ניתן לסמלץ את ריצת M על 010 למשך 3 צעדים ולראות האם הגענו לתשובה.

אם כן – נקבל. אם לא – נדחה.

<u>מטלה:</u>

1.
   א. צריך להיעזר בf לא מבקשים משהו ספציפי.. לעשות פונקציה שהיא לא מוגדרת על קלט אחד ולכן היא לא מלאה כי היא לא מוגדרת לקלט אחד..
   ב. לא צריך להראות את הפונקצהי אפשר לעשות את אותו רעיון של שפות שהם לא בR ולא RE, שיקולי ספירה.
   ג. ...
   ד. ...
2.
   א. רעיון ההוכחה דומה לאינסוף ראשים.
   ב. לראות סרטון
   ג. גם אינסוף.
   ד. גם...
   (א ג ד אותו סגנון)

מרתון 2 –

<u>שיטות הוכחה ששפה כן שייכת למחלקה מסוימת:</u>

1. מראים מכונת טיורינג עבור השפה.
   a. אם זה בR – מראים מכונה שתמיד עוצרת
   b. אם זה בRE – מראים מכונת טיורינג שתמיד עוצרת עם הקלט בשפה ואם הוא לא בשפה אז לא חייבת לעצור.
   c. אם זה ב$CORE$ – מראים מכונה עבור השפה המשלימה. (שהיא בRE).

<u>הרצה מבוקרת:</u> לעיתים נצטרך לבצע הרצה מבוקרת כאשר צריך להריץ מכונה על מספר מילים (כמה הרצות) או מספר מכונות על אותה מילה (כמה הרצות).

כלומר ברגע שצריך לבצע יותר מהרצה אחת כאשר כל מילה היא בספק אם היא מתקבלת אז אחת ההרצות עלולה לא לעצור.

הפתרון: להריץ את כל ההרצות ב"מקביל" –

מבצעים לולאה של t שמתחילה מ1 עד ∞ כאשר בכל איטרציה מריצים את כל ההרצות למשך t צעדי חישוב. כלומר לאחר t מעדים עוצרים בכח את פעולת המכונה.

דוגמה: $L=\{<M>|M\ accepts\ 00\ or\ 11\}$

נראה ש$L\in RE$.

פתרון שגוי:

נראה את המכונה N הבאה:

N: על קלט $<M>$ כאשר היא מ"ט:

- הרץ את M על 00 אם קיבלה קבל.
- אם לא, הרץ את M על 11, אם קיבלה קבל.
- אם לא, דחה.

הבעיה בפתרון הזה : אם M מקבלת את 11 ולא עוצרת על 00.

אז הקלט $L\ni<M>$ ולכן לכאורה N צריכה להחזיר כן, אבל למרות זאת היא לא עוצרת. (תקועה בלהמתין לM שתסיים את החישוב על 00).

הפתרון לבעיה הזאת היא להריץ על 2 הקלטים במקביל ולעצור ברגע שהיא מקבלת את אחד מהם.

צעד אחד על 00, צעד אחד על 11

פתרון נכון:

נראה את המכונה N הבאה:

N: על קלט $<M>$ כאשר היא מ"ט:

- עבור t מ1 עד ∞
  - הרץ את M על 00 למשך t צעדים אם קיבלה בזמן המוקצב הזה קבל.
  - אם לא, הרץ את M על 11 למשך t צעדים אם קיבלה בזמן המוקצב הזה קבל.
  - אם שני המילים נדחו – דחה.

שיטות הוכחה ששפה לא שייכת למחלקה מסוימת:

1. הלכסון. (שיטה לא מומלצת)

מתי משתמשים? ניתן להשתמש בה תמיד אך מומלץ כאשר הקלט של השפה הוא מ"ט.

- מניחים בשלילה שהשפה כן שייכת למחלקה (לדוגמה לR)
- מכאן קיימת לה מ"ט בהתאם M.
- משתמשים במכונה הזאת ומנסים לבלבל אותה.

לדוגמה: אם המכונה M מקבלת מכונה אחרת ובודקת האם היא עוצרת.

ניצור מכונת טיורינג שמקבלת מ"ט אחרת ושולחת למכונה M ואם המכונה M עונה שהקלט עוצר אז אנחנו לא נעצור ניכנס ללולאה אינסופית ואם היא עונה שהוא לא עוצרת – נעצור.

2. רדוקציות:

טוב כאשר יש לנו ידע על שפה L' שכבר לא נמצאת במחלקה ואנחנו רוצים להוכיח ששפה L גם לא שייכת למחלקה.

מראים ש L יותר קשה לחישוב מאשר L'. לכן אם L' לא נמצאת במחלקה, כל שכן שL לא תהיה.
איך מראים? אומרים שאם שיש לנו מכונה לL נוכל להשתמש במכונה הזאת כדי לפתור גם את L'.
הדרך הפורמלית לעשות זאת היא על ידי רדוקציה.

רדוקציה – היא פונקציה $f: \Sigma^* \to \Sigma^*$ מלאה (מוגדרת על כל קלט) וניתן לחישוב (יש מכונה שיכולה לחשב את הפונקציה הזאת ותמיד תעצור – מכריעה) המקיימת:
לכל $x \in \Sigma^*$ מתקיים: $x \in L` \leftrightarrow f(x) \in L$.
כלומר התכונה המיוחדת של הפונקציה היא שאם נותנים לה מילה ששייכת לשפה היותר קלה אז היא תחזיר מילה השייכת לשפה השנייה (היותר קשה) ואם נותנים לה מילה שלא שייכת לשפה היותר קלה היא תחזיר מילה שלא שייכת לשפה היותר קשה.

אם קיימת רדוקציה מ$L`$ ל$L$ נסמן זאת ב: $L` \leq L$.
דוגמה:
שפה 1: $L_1 = HP = \{< M, x > |: M \; stops \; on \; x\}$
ידוע ש $L_1 \in RE \backslash R$.
שפה 2: $L_2 = \{< M > | M \; stops \; on \; every \; input\}$

נוכיח ש$L_2 \notin R$.
נראה רדוקציה: $L_1 \leq L_2$
- הערה: תמיד מה שרוצים להוכיח שהוא לא שייך למשהו יהיה בצד ימין

$$f(< M, x >) =< M' >$$
צריך לומר מהו $M`$ כך שאם $< M, x > \in L_1$ אז $< M` > \in L_2$ ואם $< M, x > \notin L_1$ אז $< M` > \notin L_2$
כלומר צריך לכתוב אלגוריתם ל$M`$ שישתמש ב$< M, x >$.

(המלצה של גיל לכתוב את המטרה שלנו)
מטרה:
אם M עוצרת על x אז $M`$ עוצר על כל קלט שלה.
אם M לא עוצרת על x אז $M`$ לא עוצרת על כל קלט קלה. (מספיק קלט אחד שהיא לא תעצור עליו).

אלגוריתם ל$M`$: על קלט w
- הרץ את M על x.
- עצור.

הרדוקציה ניתנת לחישוב כי בסה"כ היא כותבת תיאור של מ"ט (שעלולה לא לעצור).

הרדוקציה לא מריצה אף מכונה. היא רק מתארת מכונה אחת עם מכונה אחרת.

דוגמה לרדוקציה בין שפות שלא ניתנת לחישוב:

$$f(< M, x >) = \begin{cases} \varepsilon & if \; M \; stops \; on \; x \\ 00 \; else & \end{cases}$$

תכונות של רדוקציות:

1. רפלקסיביות: $L \leq L$. הפונקציה תהיה פונקציית הזהות $f(x) = x$.
2. טרנזיטיביות: $L_1 \leq L_2 \; and \; L_2 \leq L_3 so \; L_1 \leq L_3$.
3. הרדוקציה ממש לא חייבת להיות חח"ע ועל. היא יכולה אפילו למפות את כל הקלטים לאותו פלט.
4. אם $L_1 \leq L_2$ אז $\bar{L_1} \leq \bar{L_2}$. (אפילו עם אותה פונקציה).
5. משפט הרדוקציה: אם $L_1 \leq L_2$
   a. אם $L_1 \notin R \; so \; L_2 \notin R$ (או : אם $L_1 \in R \; so \; L_2 \in R$)

b. אם $L_1 \notin RE$ so $L_2 \notin RE$ (או : אם $L_1 \in RE$ so $L_2 \in RE$)
c. אם $L_1 \notin coRE$ so $L_2 \notin coRE$ (או : אם $L_1 \in coRE$ so $L_2 \in coRE$)
6. אם שפה נמצאת ב R אז קיימת רדוקציה ממנה לכל שפה אחרת פרט לΣ^*, Φ כלומר לכל שפה L' : $L \le L'$.
דוגמה: $L_u \le \{a\}$ על ידי: $f(a) =< M', 0 >$ כאשר M' היא המכונה שמקבלת הכל. $f(x) =< M'', 0 >$ לכל $x \neq a$ כאשר M'' היא המכונה שדוחה הכל.

דוגמה:

תרגיל 1: נתונות השפות:

$$L_1 = L_u = \{< M, x > | M\ accepts\ x\}$$

$$L_2 = \{ < M >: |L(M)| \ge 5\}$$

ידוע ש $L_u \in RE \backslash R$. הוכיחו ש$L_2 \notin R$.

נראה רדוקציה:$L_1 \le L_2$.

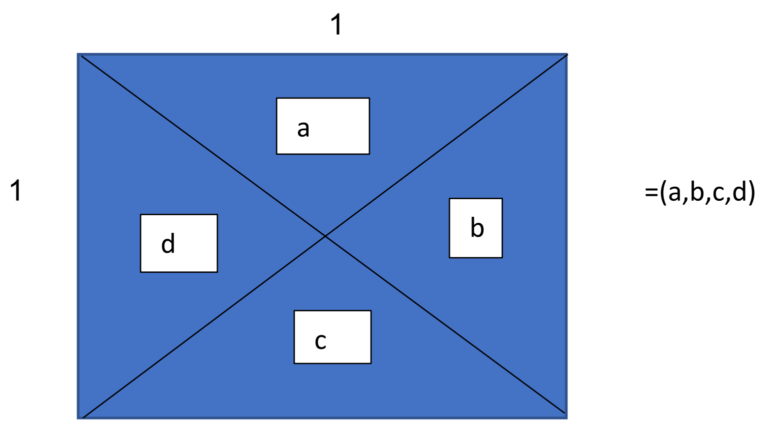
הפונקציה: $f(< M, x >) = < M` >$.

כאשר: $M`$ על קלט w :

- הרץ את M על x.
- אם M קיבלה את x וגם $w = 0,00,1,11,101$ קבל אחרת דחה.
- אחרת דחה.

בעיית הריצוף.

נתון מישור אינסופי (רק הרביע הראשון) ברצוננו לרצף אותו במרצפות 1×1 כך שלכל מרצפת יש סמל בכל אחד מארבעת הצדדים: (סימון : *up down right left*)



ישנו סט סופי של סוגי מרצפות כך שמה שמבדיל בין כל 2 סוגים הוא 4 הסמלים שמסביב.

הערה: הסדר של הסמלים חשוב כי לא ניתן לסובב מרצפת.

- **ניתן לחבר 2 מרצפות רק אם הסמלים זהים בצד. (כמו דומינו).**
- **נתונה המרצפת שצריכה להיות ראשון. צמודה ל$(0,0)$.**

תחת המגבלה של חיבור המרצפות והמרצפת הראשונה. האם ניתן להשלים את ריצוף כל המישור?

$$Tiel = \{< T, c >: exists\ valid\ f: \mathbb{N}^2 \to T\ s.t\ f(0,0) = c\}$$

דוגמה:

$< \{(a,b,c,d)\}(a,b,c,d) > \notin Tile$ – נתקע עם הראשונה ולא נוכל לחבר אליה כי בצד ימין יש את b ואין מרצפת שצד שמאל שלה שווה ל b.

$< \{(a,b,c,d),(a,d,c,b)\}(a,b,c,d) > \notin Tile$ – נוכל להשלים את השורה הראשונה פעם אחד שb מימין ואת b משמאל אבל לא נוכל להמשיך למעלה כי אין מרצפת שהצד שלה הוא a.

$< \{(a,a,a,a)\},(a,a,a,a) > \in Tile$ – זה יכול באופן ברור.

<u>טענה: $Tiel \notin R$</u>.

רדוקציה מ$L\ to\ Tiel$.

כאשר: $L = \{< M, x >: M\ dosn't\ stops\ on\ \varepsilon\}$

נראה ש: $L \leq Tiel$.

הרעיון: כל שורה תייצג קונפיגורציה של חישוב M על x.

- קונפיגורציה מורכבת מ:
  - מה המצב הנוכחי שהמכונה נמצאת בו.
  - איפה הראש קורא כותב.
  - ומה כתוב בכל הסרט זיכרון של המכונה.

ברגע שהמכונה לא תעצור על x אז היא תבצע עוד ועוד קונפיגורציות עד אינסוף ולכן גם נמשיך לרצף עוד ועוד עד אינסוף ו"נסיים את הריצוף". כלומר ניתן לרצף.

אם המכונה תעצור – אז נגיע לשורה שממנה לא ניתן להמשיך את הריצוף לשורה הבאה וניתקע – לא ניתן לרצף.

איך לממש?

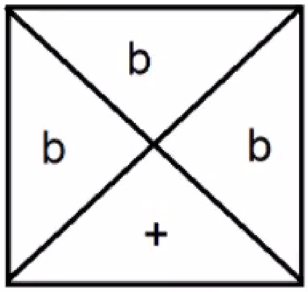
ניקח את מרכיבי המכונה:$M = (Q, \Sigma, \Gamma, q_0, b, \delta, F)$

הסמלים של המרצפות יהיו מהקבוצה הבאה :

$$\{+,*,**\} \cup \Gamma \cup \Gamma \times \{0\} \cup Q \times \Gamma \cup Q \times \Gamma \times \{0\} \cup \mathrm{Q} \times \{\mathrm{L},\mathrm{R}\}$$

קבוצת המרצפות:

$+$ - מתאר את המרצפות שלמטה – המתארות את המצב ההתחלתי.

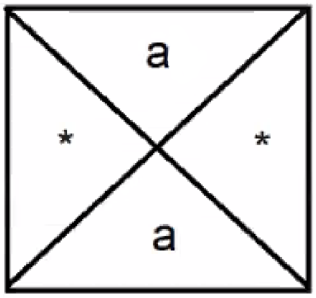


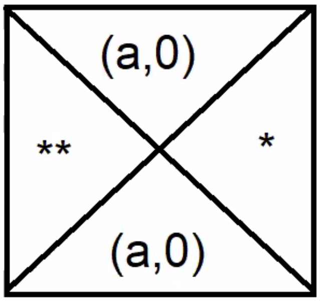
(1)

** - מתאר את המרצפות שנמצאות הכי שמאלה.

* - בא לאפשר חיבור של שורה שלמה של מרצפות.

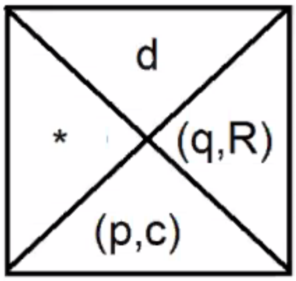
$a \in \Gamma$ כל תו אפשרי בסרט הזיכרון כולל b.

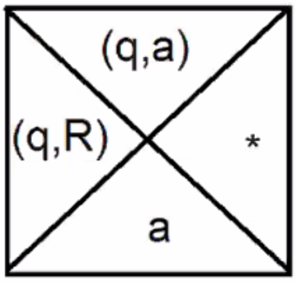


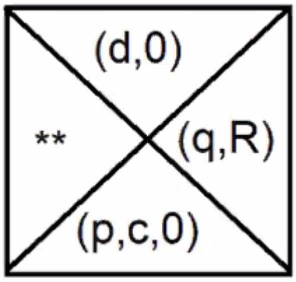


(3) (2)

לכל מעבר: $\delta(p,c) = (q,d,R)$ כאשר $c,d \in \Gamma \text{ and } p,q \in Q$.

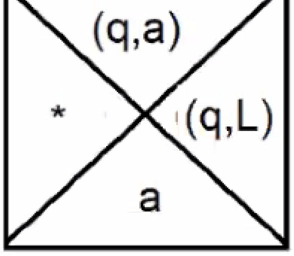


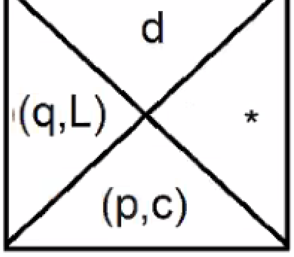


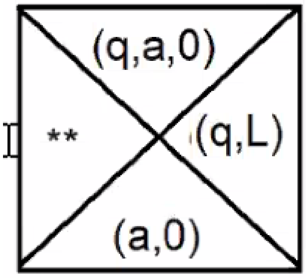


(6) (5) (4)

לכל מעבר: $\delta(p,c) = (q,d,L)$ כאשר $c,d \in \Gamma \text{ and } p,q \in Q$.

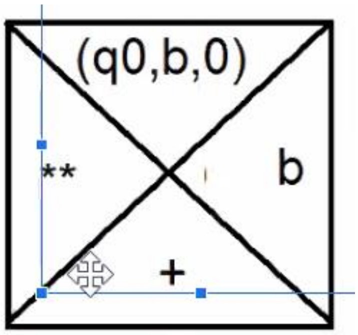






(9) (8) (7)

המרצפת הראשונה השמאלית התחתונה היא:



(10)

מתחילים מהמרצפת הראשונה (10), את השורה הראשונה ניתן למלא במרצפת (1) כי יש לה b משמאל וימין.
בכל שלב, החלק הבעייתי הוא מה לשים מעל המרצפת שהלמעלה שלה הוא המצב שלה הוא מצב ואות (או מצב אות ו0 אם זה הכי שמאלי) כי אם יש רק אותך בחלק העליון אז מרצפת 2 יכולה להשלים עד למעלה (אם זה לא הכי שמאלי) או מרצפת 3 יכולה להשלים דע למעלה אם זה הכי ימני ובצדדים שלהן יש * כך שאפשר גם למלא את השורות.

מעל מרצפת עם מצב ואות למעלה ניתן לשים או את 4 או את 8 ובמקרה שנמצאים הכי בשמאל, את מרצפת 6 או 9.

מרצפות אילו יהיו קיימות רק אם פונקציית המעברים מוגדרת על המצב והאות שנמצאים בחלק העליון של המרצפת הנוכחית.

לכן אם המרצפת הנוכחית באיזשהו שלב עם מצב עצירה, לא נוכל להשלים את הריצוף מעליה מכיוון שלפונקציית המעברים לא מוגדרת להמשיך ממצב עצירה.

אם המכונה לא עוצרת על המילה הריקה אז היא לעולם לא תגיע לעולם למצב עצירה ולכן נוכל להמשיך תמיד את הריצוף ו"לסיים" אותו.

אינטואיציה השפה היא כריעה או לא והיכן היא נמצאת

בדרך כלל השפות יקבלו מכונות טיורינג.

1. נשאל את עצמינו, אם נותנים לנו מילה שהיא בשפה. האם נצליח להוכיח לעצמינו שהיא בשפה? תוך כמות סופית של צעדים. (לא נצטרך להריץ מכונה כי היא עלולה לא לעצור ולא נצטרך לבדוק קבוצה אינסופית של אלמנטים).
   אם הצלחנו להוכיח אז השפה ב RE אחרת היא לא בRE.

2. לאחר מכן נשאל את הכיוון ההפוך, אם הפעם המילה לא בשפה. האם כעת נצליח להוכיח לעצמנו שהיא לא בשפה תוך כמות צעדים סופית.
   אם הצלחנו להוכיח, השפה ב$coRE$ אחרת היא לא ב$coRE$.

דוגמה:

$$L = \{< M >: |L(M)| \geq 2\}$$

אי דטרמיניסטי: ננחש שני קלטים ונריץ את M עליהם והיא תקבל את שניהם.

דטרמיניסטי: נרוץ בהרצה מבוקרת על כל המילים הקיימות: נסדר את המילים לפי האורך ובכל שלב t של הלולאה נרוץ t צעדי חישוב על כל אחת m המילים הראשונות בסדר הלקסיקוגרפי. מתישהו המכונה תקבל בזמן הקצוב לפחות 2 קלטים.

לכן $L_1 \in RE$.

אלגוריתם: קלט $< M >$:

- עבור t מ1 עד ∞:
  - *Counter =0*
  - הרץ את M על כל אחת מt המילים הראשונות למשך t צעדים על כל אחת.
  - על כל מילה ש M מקבל בזמן הקצוב *counter++*
  - אם *counter* גדול שווה ל2 – קבל.

אם נתון כי המכונה לא בשפה אז לא ניתן להוכיח זאת כי צריך לעבור על אינסוף מילים ולהראות אחת אחת שהן לא מתקבלות.

לכן : $L \notin coRE$.

$$L_2 = \{< M >: |L(M)| \leq 2\}$$

אם נתון שהמילה בשפה:

כדי להוכיח זאת צריך להוכיח זאת צריך לרוץ על כל המילים ולהראות שהיא לא מקבלת יותר מ2.

לכן: $L_2 \notin RE$.

אם נתון שהמילה לא בשפה:

כדי להוכיח זאת נרוץ כמו מקודם ונבדוק שהמכונה מקבל 3 מילים לפחות ובזה נפתור.

לכן: $L_2 \in coRE$.

$$L_3 = \{< M >: |L(M) = 2\}$$

אם נתון שהמילה בשפה:

כדי להוכיח זאת צריך להוכיח זאת צריך לרוץ על כל המילים ולהראות שהיא לא מקבלת יותר מ2 וגם לא פחות.

לכן: $L_2 \notin RE$.

אם נתון שהמילה לא בשפה:

ייתכן שהמכונה לא מקבלת כלום ואז נצטרך לרוץ על כל המילים.

לכן: $L_3 \notin coRE$.

$$L_4 = Tiel$$

אם נתון שהמילה בשפה:

צריך לרוץ על כל המישור ולבדוק את כל האפשרויות לריצוף. המישור הוא אינסופי אז אי אפשר לעשות זאת.

לכן: $L_4 \notin coRE$.

אם נתון שהמילה לא בשפה:

כל בדיקה של ריצוף תיתקע לאחד זמן סופי ומספר סוגי המרצפות הוא סופי ולכן נבדוק על אופציה אפשרית ותמיד נתקע.

לכן: $L_4 \in coRE$.

מרתון 3

רדוקציות צריך שפות שהן כבר לא ב$R, RE, coRE$

לכן כדאי לזכור את השפות הבאות:

1. $L_u = \{< M, x > | M\ accepts\ x\}$
2. $L_{hp} = \{< M, x > | M\ halts\ pn\ x\}$

2 השפות הללו הן בRE ולא בR. (הן לא ב$coRE$)

אם השפה שצריך להוכיח שהיא לא בR מכילה תנאי על מה המכונה מקבלת ומה לא אז נעדיף את L_u

ואם התנאי הוא על עצירה שלה אז נעדיף את HP.

אם צריך להוכיח ששפה היא לא בRE אז ניקח את המשלימה אחת מהן. (היא ב$coRE$ ולא בR, RE)

אם צריך להוכיח ששפה היא לא בRE ולא ב$coRE$.

1. ניתן לעשות 2 רדוקציות אחת לכל כיוון.
2. ניתן להשתמש בשפות ידועות נוספות:
   a. $L_\infty = \{< M >: |L(M)| = \infty\}$
   b. $L_\phi = \{< M >: L(M) = \Phi\}$ – בסימן שאלה גיל לא בטוח (?)

~~~~~~

3. **משפט רייס** (דרך להוכיח ששפה לא בR אבל בRE)
   כל שפה מהצורה $L_S = \{< M >: L(M) \in S\}$ כאשר $S \subseteq RE$ הוא תנאי על השפות, היא לא בR.
   התנאים של משפט רייס כדי שנוכל להשתמש בו:
   א. מקבלים רק מכונה (לא מכונה ומילה וכו'..).
   ב. התנאי של השפה מדבר על השפה של המכונה או על מה שהיא מקבלת.
   ג. התנאי אינו טריוויאלי. (לא תנאי שתמיד מתקיים או תמיד מתקיים).

הערה: אם S טריוויאלית אז $L_s \in R$.

דוגמה:

$L = \{< M_1, M_2 >: L(M_1) = L(M_2)\}$- לא ניתן להשתמש ברייס כי הקלט לא תקין.

$L = \{< M >: M\ stops\ on\ 010\ \}$ – לא ניתן להשתמש ברייס כי התנאי הוא על התנהגות המכונה ולא על השפה שלה.

$L = \{< M >: |L(M)| \geq 0\}$ – לא ניתן להשתמש ברייס כי התנאי הוא טריוויאלי: כל שפה היא לפחות בגודל 0.

איך משתמשים ברייס?

1. מגדירים את קבוצת השפות S מתוך השפה נתונה. פשוט מעתיקים את התאני של השפה ובכל מקום שכתוב $L(M)$ כותבים פשוט L.
2. מוכיחים שS אינה טריוויאלית.
   מראים דוגמה לשפה אחת בRE שכן מקיימת ושפה אחת בRE שלא מקיימת את התנאי.
3. מסיקים שהשפה המקורית אינה בR.

דוגמה:

$L_1 = \{< M >: |L(M)| > 10\}$.

נגדיר את: $S = \{L \in RE: |L| > 10\}$ ומכאן $L_S = L_1$.
~~~~~~

נוכיח שS אינה טריוויאלית:

$S \notin \Phi$ כי הגודל של $|\Phi| = 0 < 10$ ולכן $S \neq RE$.

$S \in \{0, 0^2 \dots, 0^{11}\}$ כי גודל השפה הוא 11 ולכן : $S \neq \Phi$.

לכן S אינה טריוויאלית ולכן לפי משפט רייס, $L_1 \notin R$.

$L_2 = \{< M >: M \ accepts \ 0, 00, 1\}$.

נגדיר את: $S = \{L \in RE: 0,00,1 \in L\}$ ומכאן $L_S = L_2$.

נוכיח שS אינה טריוויאלית:

$S \notin \Phi$ כי בשפה הריקה אין את 0 לדוגמה ולכן $S \neq RE$.

$S \in \{0,00,1,11\}$ כי 3 המילים הנ"ל נמצאות בשפה ולכן : $S \neq \Phi$.

לכן S אינה טריוויאלית ולכן לפי משפט רייס, $L_2 \notin R$.

$L_3 = \{< M >: L(M) \in coRE \}$.

נגדיר את: $S = \{L \in RE: L \in coRE\} = R$ ומכאן $L_S = L_3$. כי זה כאילו החיתוך בין RE ל$coRE$.

נוכיח שS אינה טריוויאלית:

$S \notin \mathrm{L_u}$ כי $L_u \in RE \backslash coRE$ $S \neq RE$.

$S \in \Phi$, $RE \cap coRE = R \in \Phi$ ולכן : $S \neq \Phi$.

לכן S אינה טריוויאלית ולכן לפי משפט רייס, $L_3 \notin R$.

$L_4 = \{< M >: L(M) \in R\}$.

בדיוק אותו דבר כמו דוגמה 3.

<u>הוכחת משפט רייס:</u>

נתונה תכונה S לא טריוויאלית של שפות בRE. צ"ל: $L_s = \{< M >: L(M) \in S\}$ לא בR.

נחלק למקרים:

<u>מקרה 1: $\Phi \notin S$.</u>

מכיוון שS לא טריוויאלית אז קיימת שפה אחרת $L \in S$.

מכיוון שכל השפות בS הן RE אז ניתן להגיד $L \in RE$. לכן קיימת לה מכונה מזהה: M_L.

נגיש להוכחה: נראה רדוקציה - $L_u \leq L_s$. על ידי $f(< M, x >) =< M' >$ כאשר:

M' על קלט w:

- הרץ את M על x.
- אם M קיבלה את x הרץ את M_L על w וכנה כמוה.
- אחרת – דחה.

תקפות:

אם M מקבלת את x אז: M' תמיד תגיע לשורה שניה ותקבל רק מה שM_L מקבלת ולכן:

$S \ni L = L(M_L) = L(M')$ ומכאן: $L(M') \in S$ ולכן: $< M >\in L_S$.

אם M לא מקבלת את x, אז אם M לא עוצרת על x אז M' תקוע בשורה הראשונה ולכן לא עוצרת על אף w ולכן: $L(M') = \Phi \notin S$ ולכן $< M' >\notin L_s$.

אם M עוצרת ודוחה את x אז M' תגיע תמיד לשורה שלישית ולכן תדחה הכל שוב: $L(M') = \Phi \notin S$ ולכן $< M' >\notin L_s$.

<u>מקרה 2: $\Phi \in S$.</u>

מכיוון שS לא טריוויאלית אז קיימת שפה אחרת $L \notin S$.

מכיוון שכל השפות שמדברים עליהן RE אז ניתן להגיד $L \in RE$. לכן קיימת לה מכונה מזהה: M_L.

נגיש להוכחה: נראה רדוקציה - $L_u^- \leq L_s$. על ידי $f(< M,x >) =< M' >$ כאשר:

M' על קלט w:

- הרץ את M על x.
- אם M קיבלה את x הרץ את M_L על w וכנה כמוה.
- אחרת – דחה.

תקפות:

אם M מקבלת את x אז: M' תמיד תגיע לשורה שניה ותקבל רק מה שM_L מקבלת ולכן:

$L(M') = L(M_L) = L \notin S$ ומכאן: $L(M') \notin S$ ולכן: $< M >\notin L_s$.

אם M לא מקבלת את x, אז אם M לא עוצרת על x אז M' תקוע בשורה הראשונה ולכן לא עוצרת על אף w ולכן: $L(M') = \Phi \in S$ ולכן $< M' >\in L_s$.

אם M עוצרת ודוחה את x אז M' תגיע תמיד לשורה שלישית ולכן תדחה הכל שוב: $L(M') = \Phi \in S$ ולכן $< M' >\in L_s$.

<u>הרחבה של רייס –</u>

אם בנוסף $\Phi \in S$ אז $L_s \notin RE$.

הסבר על 4 א' – במקום RE של שפות אז יש fRE של פונקציות.

במקום תכונה S של חלק מהשפות בRE אז יש תכונה S של חלק מהפונקציות בfRE.

במקום L_s יש לנו $L_s = \{< M >: f_m \in S\}$. אנלוגי לΦ שהפונקציה לא מוגדרת לכלום אבל כן ניתנת לחישוב אנלוגי לΣ^* זה פונקציה שמחזירה $f(x) = x$.

אפשר לעשות העתק הדבק רק לשנות במקום שיש שפה פונקציה.

סיבוכיות

בחלק זה נתעניין בבעיות שהן כריעות – כל השפות שנראה הן בR.

נגדיר מחלקות חדשות (שכמעט כולן בתוך R) של סיבוכיות הפתרון לחישוב השפה. כלומר, לכל שפה נבדוק האם יש מ"ט שעובדת בזמן ריצה יעיל ומכריעה את השפה.

דגשים:

1. מדידת זמן ריצה היא בהתאם לקלט:
כלומר אם הקלט בגודל n ואנו מבצעים n^2 פעולות אז הסיבוכיות היא $O(n^2)$.
ואם הקלט n^2 ועושים n^2 פעולות אז הסיבוכיות היא $O(n)$.
אם הקלט בגודל $\log n$ ועושים n פעולות אז הסיבוכיות היא $O(2^n)$ <- צריך לתקן לפי גיל.

2. בחלק זה של הקורס יש הבדל בין מ"ט דטדמיניסטית למכונה אי-דטרמיניסטית. בחלק של חישוביות לא היה הבדל כי המודלים שקולים מבחינת כוח החישוב אך לא בהכרח שקולים מבחינת סיבוכיות זמן. כי בעולם שמ"ט דטר' צריכה לעשות לולאה שמחפשת איבר האי-דטר' יכולה פשוט לנחש אותו ב$O(1)$.

3. לא נתמקד בסיבוכיות המדויקת אלה ביעיל או לא יעיל.

יעיל – סיבוכיות פולינומית כאשר החזקה של הפולינום לא משנה. כלומר: $O(n^{1000})$ יחשב יעיל.
לא יעיל – סיבוכיות לא פולינומית, אקספוננציאלית (יותר גדולה מפולינום). $O(2^n)$ לא יעיל.
$O(n^{\log n})$ לא יעיל (למרות שזה לא אקספוננציאלי).

4. את הפעולות סופרים לפי הצעדים של המכונה. כל צעד נחשב פעולה אחת. לכן בדרך כלל יקח למכונה יותר זמן ממחשב לבצע פעולות.
למרות זאת מה שיהיה יעל במ"ט יהיה יעיל גם במחשב ולהיפך.

מחלקות סיבוכיות

1. P – קבוצת כל השפות בR כך שיש להן מכונה **דטרמיניסטית** המכריעה אותן בזמן $O(n^k)$ כאשר k הוא קבוע.
2. NP - קבוצת כל השפות בR כך שיש להן מכונה **אי-דטרמיניסטית** המכריעה אותן בזמן $O(n^k)$ כאשר k הוא קבוע.
3. $coNP$ – כל השפות שהמשלימה שלהן היא בNP.
לדוגמה : $\overline{CLIQUE} \in coNP$

אינטואיטיבית: NP היא כל השפות שניתן לנחש שמילה בשפה ולבדוק זאת בזמן פולינומי.

$coNP$ היא כל השפות שניתן לנחש שמילה לא בשפה ולבדוק בזמן פולינומי.

מה הכוונה בניתן לוודא? הכוונה שנותנים לנו את התשובה (מגלים איפה היא) ואנו צריכים להשתכנע שזה בזמן פולינומי.

דוגמאות:

- $L = \{< G >: G\ is\ graph\ and\ G\ is\ connected\} \in P$
ניתן להכריע האם מילה בשפה על ידי אלגוריתם פולינומי לבדיקת קשירות בגרף (BFS).
- $L = \{a^n b^n : n \in \mathbb{N}\} \in P$
ניתן לספור למנות את כמות הa ואת כמות הb ואז להשוואת.
- $CLIQUE = \{< G, k >: k \in N, and\ G\ contains\ clique\ of\ size\ k\} \in NP$
קליקה בגודל k היא קבוצה של k קודקודים כך שכולם מחוברים לכולם.
אלגוריתם פשוט יהיה לעבור על כל תתי הקבוצות של קודקודים בגודל k ועבור כל קבוצה חברים הקבוצה מחוברים זה לזה. סיבוכיות $O\left(\binom{|V|}{k}k^2|V|^2\right) = O\left(|V|^k k^2 |V|^2\right)$ ואם לדוגמה :
$k = \frac{|V|}{2}$ אז : $O(|V|^{\frac{|V|}{2}+4}) =$ וזה אקספוננציאלי.
(אם K קבוע זה לא יהיה אקספוננציאלי וגם אם $k = n - c$ אז זה לא אקס אבל אם $k = \frac{n}{c}$ אזה זה כן יהיה אקספוננציאלי).

הערה: $\binom{n}{k} = O(n^k)$. וגם $\binom{n}{n-k} = O(n^k)$.
עד כה הראינו אלגוריתם לא טוב.
לא ידוע האם השפה בP או לא.
נראה למה השפה כן בNP:
- עבור הקלט הנ"ל נחשב k קודקודים שונים ואם כולם מחוברים לכולם – קבל. אחרת – דחה.
- עלות הניחוש: $O(k \cdot n)$. בדיקת חיבור: $O(n^2k^2)$ סה"כ פולינומי בגודל הקלט.

תכונות:

1. $P \subseteq NP, coNP$. לכן $P \subseteq coNP \cap NP$
2. $P =^? NP$ בעיה פתוחה לא ידוע הפתרון האם זה נכון או לא. ולכן לא ניתן יהיה לומר על כמעט כל השפות בהמשך הקורס שהן לא בP.
3. $P = coNP \cap NP$? בעיה פתוחה.

רדוקציות פולינומיות

כמו בחישוביות שרדוקציה אמרה מי יותר קשה לחישוב ממי.

בסיבוכיות – הרדוקציה הפולי' תאמר לנו מי יותר קשה לחישוב מבחינת זמן ריצה.

רדוקציה פ' מL_1 לL_2 תסומן : $L_1 \leq_p L_2$.

והיא כמו הרדוקציה של חישוביות פרט לכך שניתן לחשב את פונקציית הרדוקציה בזמן פולינומי.

תכונות:

1. רפלקסיביות: $L \leq_p L$. הפונקציה תהיה פונקציית הזהות $f(x) = x$.
2. טרנזיטיביות: $L_1 \leq_p L_2\ and\ L_2 \leq_p L_3 so\ L_1 \leq_p L_3$.
3. הרדוקציה ממש לא חייבת להיות חח"ע ועל. היא יכולה אפילו למפות את כל הקלטים לאותו פלט.
4. אם $L_1 \leq_p L_2$ אז $\bar{L_1} \leq_p \bar{L_2}$. (אפילו עם רוצה פונקציה).
5. משפט הרדוקציה: אם $L_1 \leq_p L_2$
   a. אם $L_2 \in P\ so\ L_1 \in P$
   b. אם $L_2 \in NP\ so\ L_1 \in NP$
   c. אם $L_2 \in coNP\ so\ L_1 \in coNP$
6. אם $L \in P$ אז קיימת רדוקציה פולינומית ממנה לכל שפה אחרת פרט ל Φ, Σ^*.
   כלומר לכל שפה L' : $L \leq_p L'$.

(רדוקציה פולי' אומר ש**הפונקציה** מעבירה משפה לשפה בזמן פולינומי!)

דוגמה:

$\{< G >: G\ is\ connected\} \leq_p \{< G >: G\ has\ exactly\ two\ connected\ comp.\}$.

על ידי : $f(< G >) =< G' >$ כאשר G' יהיה כמו G בתוספת קודקוד בודד.

תקפות: אם G קשיר אז בG' יהיו בדיוק שני רכיבי קשירות כיוון שG הוא כולו רכיב 1 + קודקוד בודד.

אם G לא קשיר אז כבר בG יש לפחות 2 רכיבי קשירות ולכן בG' יהיו לפחות 3 רכיבי קשירות.

סיבוכיות הרדוקציה: $O(n)$ עבור העתקת G ועבור תוספת קודקוד זה $O(1)$ פולינומי בגודל הקלט.

$L = \{< A, k >: A\ is\ set\ of\ ints, And\ exist\ S \subseteq A\ s.t\ |S| = k, and\ for\ all\ a, b \in S: a|b\ or\ b|a\}$

לדוגמה - $< \{1,2,4,9\}, 3 >\in L$.

הראו ש$L \leq_p CLIQUE$.

נראה את הרדוקציה : $f(<A,k>) = <G',k'>$

כאשר $G' = (V,E)$ מוגדר באופן הבא: $V = A$ ו $E = \{\{a,b\}: a,b \in A, a \neq b, a|b \text{ or } b|a\}$ ו $k' = k$.

תקפות: אם קיימת $S \subseteq A$ כך ש$|S| = k$ ולכל $a,b \in S$ לפחות אחד מהם מחלק את השני, אז S היא גם קליקה בG' כי לפי בניית G' - אם כל שני איברים בS, לפחות אחד מחלק את השני אז הם מחוברים בצלע ולכן כולם מחוברים לכולם ולכם זו קליקה. מכיוון ש $K=|S|$ יש לנו קליקה בגודל k.

אם קיימת ב G' קליקה בגודל k אז כמו בכיוון הראשון (מהסוף להתחלה) הקליקה היא הקבוצה S המבוקשת.

סיבוכיות הרדוקציה: $O(n^2)$ עבור בניית הגרף כפול $O(n^2)$ כדי לבדוק האם האיברים מחלקים אחד את השני + העתקת k, סה"כ פולינומי.

שלמות במחלקה:

עבור מחלקה C. שפה L תקרא $C-complete$ או במילים "שפה C שלמה" אם:

1. $L \in C$
2. כל שפה אחרת במחלקה ניתנת לרדוקציה לL.
   לכל L' מתקיים $L' \leq L$.

איך מוכחים שפה L היא שלמה?

1. מראים שהשפה שייכת ל$L \in C$.
2. אם יודעים כבר על שפה L' שהיא שלמה בC אז מראים : $L' \leq L$.

בR כל השפות פרט ל Σ^*, Φ הן $R-complete$.

בRE כל השפות שהן $RE \backslash R$ הן גם $RE-complete$.

ב$coRE$ כל השפות שהן ב$coRE \backslash R$ הן גם $coRE-complete$.

ב$RE \cup coRE$ אין שפה שלמה.

במחלקה של כל השפות שהן מחוץ ל$RE \cup coRE$ אין שפה שלמה.

עבור סיבוכיות:

בP כל השפות פרט ל Σ^*, Φ הן $P-complete$.

בNP - ??? נראה שפות כאלו בהמשך.

ב$coNP$ – נראה בהמשך.

שפות מוכרות שהן *NPC*

גרפים:

1. $CLIQUE = \{<G,k>: G \text{ contains clique of size } k\}$
2. $IS = \{<G,k>: G \text{ contains independet sets of size } k\}$ – כלומר יש בגרף קבוצה של k קודקודים שאף אחד מהם לא מחובר לאף אחד באופן ישיר.
3. $VC = \{<G,k>: G \text{ has vertex cover of size } k\}$ – כלומר יש בגרף קבוצה של k קודקודים שמכסה את כל צלעות הגרף. = כל צלע נוגעת בלפחות אחד מקודקודי הקבוצה שנלקחה.
4. $HP = \{<G>: G \text{ has hamilton path}\}$ – יש בגרף מסלול המילטוני = מסלול שעובר **בכל** הקודקודים בגרף בדיוק פעם אחת.
5. $HC = \{<G>: G \text{ hmilton cycle}\}$ – דומה ל4 רק שצריך להגיע לקודקוד שהתחלנו.
6. $k-Col = \{<G>: G \text{ is } k-colorable\}$ – עבור $k > 2$. כל הגרפים שהם k צביעים.

נוסחאות CNF:

7. $SAT = \{< \varphi >: \varphi \text{ is a CNF formula and } \varphi \text{has satisfiying assigment}\}$
דוגמה: $\varphi = (a \vee \bar{b}) \wedge (a \vee b \vee c \vee d) \wedge (\bar{d} \vee \bar{c}) \wedge (\bar{b})$
ההשמה: $a = true, b = false, c = true, d = false$.
כל סוגריים נקראים פסוקית. והשמה שגורמת להכל להיות $true$ נקראת השמה מספקת.
השמה זו מספקת את φ.
8. $3SAT = \{< \varphi >: \varphi \text{ is a 3CNF formula and } \varphi \text{has satisfiying assigment}\}$ – כמו SAT פרט לכך שכל פסוקית בנויה מבדיוק 3 משתנים שונים (או שלילתם).
דוגמה: $\varphi = (a \vee \bar{b} \vee c) \wedge (b \vee c \vee d) \wedge (\bar{d} \vee \bar{c} \vee b) \wedge (\bar{b} \vee a \vee \bar{d})$.

קבוצות:

9. $SS = \{< a_1, \dots a_n, s >: There\ is\ S \subseteq \{a_1 \dots a_n\} s.t\ \Sigma_{x \in S} x = s\}$
10. $SC = \{< A_1, \dots, A_t, n, k >: There\ are\ k\ sets\ from\ A_{1\dots t}\ s.t\ the\ union\ of\ all\ k\ sets\ is\ \{1,2 \dots n\}\}$
דוגמה אצל גיל.

דגש : אם שפה היא בP וגם בNPC אז $P = NP$. בדרך כלל במבחן האינטואיציה צריכה להיות ש $P \neq NPC$.

תרגילים:

2020 א' א'

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

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

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

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

שיטת הוכחה:

אם זה בP מראים אלגוריתם (מ"ט) דטרמיניסטי יעיל.

אם זה בNPC אז מראים אלגוריתם א"ד שמראה שזה בNP ובסוף מראים רדוקציה מאחת השפות הידועות שהיא בNPC לשפה בשאלה.

א. הבעיה בNPC. הוכחה:
   a. הבעיה בNP: נראה מ"ט א"ד פולינומית עבור הבעיה:
      N: על קלט $< G, k >$:
      - ש בדוק: $|V| \geq 3k + 10$. אם לא דחה.
      - נחשב קבוצה S_1 של k קודקודים מתוך G.

- עבור כל 2 קודקודים $u,v \in S_1$ בדוק שהצלע בינהם קיימת. אם לא - דחה.
- נחש S_2 של $2k$ השונים מהקודקודים בS_1 מתוך G.
- עבור כל 2 קודקודים $u,v \in S_2$ בדוק שהצלע בינהם קיימת. אם לא - דחה.
- קבל.

נכונות: ...

סיבוכיות:

- בדיקה ראשונית: חישוב של $3k + 10$ ב$O(k)$. והשוואה מול $|V|$ - $O(|V|)$.
- ניחוש 2 הקבוצות: $O(k \cdot \log|V|)$.
- בדיקת צלע: $O(k^2|V|^2)$.

סה"כ: פולינומי בגודל הקלט.

.b נראה רדוקציה: $CLIQUE \leq_p L_1$.
על ידי: $f(< G,k >) =< G',k' >$
G' יכיל את: G, רכיב נוסף שהוא קליקה בגודל $2k$ ועוד 10 קודקודים בודדים.$k' = k$.
תקפות: אם בG יש קליקה בגודל K אז יש בG לפחות K קוקדקודים ולכן בG' יהיו לפחות $k +$ $2k + 10$ קודקודים.
בנוסף, בG' תהיה הקליקה בגודל k של G ועוד קליקה בגודל $2k$ (מה שהוספנו) ולכן: $< G',k' > \in L_1$.
אם בG אין קליקה בגודל k אז בG' תהיהה רק קליקה אחת בגודל $2k$ ולכן: $< G',k' > \notin L_1$.

סיבוכיות:
$O(n)$ עבור העתקה של G + הוספת 10 קודקודים $O(1)$ + הוספת רכיב בגודל $2k$ $O(k^2)$

ב. הבעיה בP. נראה אלגוריתם דטרמיניסטי לבעיה:
M על קלט: $< G,k >$:

- בדוק שכמות הקודקודים $|V| \leq 3k + 10$. אם לא דחה.
- חלק את G לרכיבי קשירות $C_1, \dots, C_t$.
- בדוק האם קיים רכיב קשירות בגודל לפחות $2k$ וגם רכיב קשירות נוסף בגודל לפחות k.
- אם $k < 11$ עבור על כל רכיבי הקשירות בגודל לפחות $2k$:
  - עבור על כל תתי הקבוצות S בגודל $2k$ מתוך הרכיב ובדוק האם כל $\{u,v\} \in S$ חוברים בצלע.
  - אם מצאת S כזאת. עבור על כל הרכיבין בגודל לפחות k פרט לרכיב הנ"ל:
    - עבור על כל תתי הקבוצות T בגודל k מתוך הרכיב ובדוק כל $\{u,v\} \in T$ חוברים בצלע.
    - אם כן – קבל.
- אם לא נמצאה קליקה בגודל $2k$ או בגודל k – דחה.
- אם $11 \geq k$ קיים רק רכיב אחד C מגודל $2k$ לפחות וקיים רק רכיב אחד D מגודל k לפחות. (ופחות מגודל $2k$).
- עבור על כל תתי הקבוצות בגודל $2k$ מתוך קודקודים C. ובדוק אם כל 2 קודקודים בתת הקבוצה מחוברים בצלע.
- אם עברת על כל תתי הקבוצות ולא נמצאה קליקה – דחה.
- עבור על כל תתי הקבוצות בגודל k מתוך קודקודים D. אם כל 2 קודקודים בתת הקבוצה מחוברים בצלע - קבל.
- אם עברת על כל תתי הקבוצות ולא נמצאה קליקה – דחה.

נכונות: חיפוש שלם.
סיבוכיות: אם $k < 11$ אז מבצעים : $O(1)$ פעולות.
אם $K \geq 11$ אז יש בגרף רק רכיב אחד בגודל $2k + c$ ורכיב אחד בגודל לפחות $k + d$, כאשר: $c,d \leq 10$.
מספר תתי הקבוצות בגודל $2k$ מתוך הרכיב הגדול הוא : $\binom{2k+c}{c} = O(k^c)$.

בדיקת צלעות בין כל 2 קודקודים: $O(k^2)$. סה"כ : $O(k^{c+2})$.
באותו אופן עבור הרכיב בגודל k: $O(k^{d+2})$
סה"כ: $O\left(k^{c+2} + k^{d+2}\right)$ פולינומי בגודל הקלט.

מרתון פתירת מבחנים:

1. (30 נקודות) לכל שפה מהשפות הבאות, הוכיחו

א. $L_1 = \{(G,k)|$ exists a VC of size k in G, and k is 3 modulo 10$\}$
בעברית: זוהי שפת זוגות של (קידוד) גרף ומספר k, כך ש k שווה ל 3 מודולו 10 וקיים בגרף כיסוי בצמתי בגודל k.

ב. $L_2 = \{G(V,E)|$ exists a VC of size k in G, where k is 3 modulo |V|$\}$
בעברית: זוהי שפה של (קידודי) גרף כך שקיים מספר k השווה ל 3 מודולו |V| וקיים בגרף כיסוי בצמתים בגודל k.

ג. $L3 = \{(G(V,E),k)|$There exists a VC of size k, but has no simple cycle with at least $0.7|V|$ vertices.$\}$

בעברית: זוהי שפת של זוגות של (קידודי) גרף $G(V,E)$ ומספר k כך שקיים בגרף כיסוי בצמתים בגודל k וכן לא קיים בו מעגל פשוט בגודל לפחות $0.7|V|$. נזכיר שמעגל פשוט הוא מעגל שבו דרגת כל צומת היא בדיוק 2 (כחלק מהמעגל, לא בהכרח בגרף G עצמו).

א. השפה בNPC.

הערה: אם ידוע לנו שפה שהיא בNPC אנחנו יכולים לעשות שימוש במכונה שלה כדי לבנות את המכונה שלנו לשפה.

הוכחת NP: ידוע כי $VC \in NPC$ ולכן קיימת מכונה א"ד פולינומית M_{VC} עבור VC.

נגדיר מכונה א"ד $L1$:

M_1על קלט $< G,k >$:

- בדוק את השארית של K מודולו 10 אם היא שונה מ3 – דחה.
- הרץ על M_{VC} על הקלט וענה כמוה.

סיבוכיות:

בדיקת שארית החלוקה של k נניח - $O(k)$ – חסון בגודל הקלט.

הרץ M_{VC} פולינומית כיוון שהמכונה רצה בזמן פולינומי.

נכונות:

אם הקלט בשפה אז הבדיקה של שארית K תעבור ולפי נכונות המכונה של VC נקבל.

אם הקלט לא בשפה אז או ששארית החלוקה של K ב10 אינה 3 ואז נדחה בכל מצב בשורה הראשונה או שאין כיסוי בגודל K בגרף ואז לא קיים מסלול מקבל של מכונה M_{VC} ולכן לא קיים מסלול מקבל עבור המכונה שלנו.

הוכחת NPH:

נראה רדוקציה פולינומית: $VC \leq_p L_1$. על ידי $f(< G,k >) =< G',k' >$

כאשר: G' יהיה G + m רכיבים נוספים של 2 קודקודים מחוברת בצלע אחת בכל רכיב.

$m = \big(3-(k \bmod 10)\big) mod\ 10$ במילים: m הוא כמות הרכיבים שיש להוסיף כדי להגיע למצב שיש כיסוי בגודל שמתחלק ב10 עם שארית 3.

$k' = k + m$.

הרדוקציה מלאה וניתנת לחישוב.

סיבוכיות:

חישוב: m הוא ב$O(k)$ חסום בגודל הקלט. + העתקת הגרף – חסום בגודל הקלט.

סה"כ פולינומי בגודל הקלט.

תקפות:

אם בG יש VC בגודל K אז בגרף שנוצר יש VC בגודל K (אותה קבוצה) + M קודקודים , אחד מכל רכיב חדש שהוספנו. בנוסף $M+K$ מתחלק ב10 עם שארית 3 לפי הגדרת m.

אם בG אין VC בגודל K אז בגרף הנוצר אנו חייבים לקחת m קודקודים לכיסוי, אחד מכל אבל אז לא קיים כיסוי בגודל k בשאר הגרף – G ולכן סה"כ אין כיסוי בG' בגודל $m+k$.

ב. השפה בP.

הערה: שאומרים קיים אני יכול לבחור אותו אני לא מקבל אותו! לכן אני יכול לקבוע איזה k זה. אם יש לך קיים והוא לא נתון כקלט אתה בוחר אותו כרצונך.

הערה: VC הוא הפוך מקליק – ככל שאני מגדיל יותר קל למצוא.

מכיוון שכל כיסוי קודקודים הוא בגודל $|V|$ לכל היותר אז האפשרות היחידה ש$k = 3 mod V$ כאשר $k = 3$ ולכן הבעיה למעשה היא האם יש בגרף כיסוי קודקודים בגודל 3.

נראה מ"ט דטרמיניסטית פולינומית עבור השפה:

M_2 : על קלט $< G >$ בצע:

- עבור כל תת קבוצה S של קודקודים בגודל 3 בG :
  - עבור על כל צלע $e \in E$.
    - בדוק שלפחות אחת מקצוות הצלע שייכת לS. אם לא – עבור לקבוצה הבאה.
  - אם כל הצלעות קיימו את התנאי – קבל.
- דחה.

סיבוכיות:

מספר תתי הקבוצות בגודל 3 הוא $|V|^3 = \binom{|V|}{3}$.

עבור כל קבוצה עוברים על כל הצלעות ולכן הסיבוכיות היא $O(|V|^3|E|)$ במ"ט הזמן החיפוש יעלה ריבוע לכל היותר של הזמן הנ"ל – פולינומי בגודל הקלט.

נכונות:

מכיוון שכל כיסוי קודקודים הוא בגודל $|V|$ לכל היותר אז האפשרות היחידה ש$k = 3 mod V$ כאשר $k = 3$ ולכן הבעיה למעשה היא האם יש בגרף כיסוי קודקודים בגודל 3.

מכאן, שאם יש כיסוי בגודל 3 אז מכיוון שעוברים על כל תתי הקבוצות ובודקים האם הן כיסוי – נמצא את הקבוצה המבוקשת.

אם אין כיסוי בגודל 3 (אז אין גם בגדלים קטנים יותר) אז נעבור על כל תתי הקבוצות בגודל 3, לא נמצא ולכן נדחה.

ג. הערה: אין אלגוריתם פולינומי שמוצא מעגל הכי גדול בגרף!

השפה בNPH.

האינטואיציה שהיא לא בNP היא שכדי לדעת שאין מעגל בגודל $0.7|V|$ צריך לעבור על כל תתי הקבוצות בגודל $0.7|V|$ ועל כל הסידורים האפשריים שלהם ולהיווכח שאין מעגל.

ולא ניתן לעשות זאת על ידי מ"ט א"ד בזמן פולינומי או על ידי מאמת. כי אין ניחוש\קלט למאמת בגודל פולינומי שיכול להוכיח זאת .

מאמת: מכונה דטר' שמקבל את התשובה כקלט ורק בודקת אותו.

איזה רמז ניתן למאמת שאין מעגל?

נראה רדוקציה: $VC \leq_p L_3$ על ידי $f(<G,k>) =< G', k' >$.

$G = G' +$ $|V|$ קודקודים בודדים נוספים.

$k' = k$.

הרדוקציה מלאה וניתנת לחישוב.

סיבוכיות:

העתקת הגרף G + הוספת $|V(G)|$ קודקודים בודדים – פולינומי בגודל הקלט.

העתקת k – חסום בגודל הקלט.

תקפות:

אם יש בG כיסוי בגודל k אז גם בG' יש כיסוי בגודל k כי הוספנו רק קודקודים בודדים ולא צלעות.

בנוסף, אין בG' מעגל בגודל $0.7|V(G')|$ כי חצי ממנו הם קודקודים בודדים ולכן המעגל הגדול ביותר יהיה בגודל לכל היותר $0.5|V(G')|$.

אם אין בG בגודל k אז מכיוון שכל סיכוי של G' הוא כיסוי של G אז לא קיים כיסוי בגודל k בG'.

2. (30 נקודות) לכל אחת מהשפות הבאות, קבעו האם היא ב R והאם היא ב RE. הוכיחו את תשובתכם.

א.

$$L_1 = \{< M1 > | \exists \text{ infinitely many } < M_2 >' s \text{ such that } L(M_1) = L(M_2)\}$$

בעברית: זוהי שפת כל קידודי המכונות $< M_1 >$ שקיימים עבורן אינסוף קידודי מכונות $< M_2 >$ כך שהשפה של M_1 שווה לשפה של M_2.

ב. $L_2 = \{< M1 >, < M2 > | \text{ There exist at least 2 words } x_1, x_2$ $\text{that each } x_i \text{ is in } L(M_i) \text{ but not in the language of the other } M_j (\text{ where } j = 3 - i)\}$

בעברית: זוהי שפת כל זוגות קידודי מכונות $< M_1 >, < M_2 >$ כך שקיים זוג מילים x_1, x_2 כך שלכל $i \in \{1,2\}$ $x_i \in L(M_i) \setminus L(M_{3-i})$.

ג. $L_3 = \{< M1 >, < M2 > | \text{ There exist at least 2 words } x_1, x_2$

$\text{such that each } x_i \text{ is in } L(M_i) \text{ and is rejected by the other } M_j (\text{ where } j = 3 - i)\}$

בעברית: זוהי שפת כל זוגות קידודי מכונות $< M_1 >, < M_2 >$ כך שקיים זוג מילים x_1, x_2 כך שלכל $i \in \{1,2\}$ $x_i \in L(M_i)$ וגם M_{3-i} דוחה את x.

א. השפה שייכת לR.

לכל שפה בRE יש אינסוף מכונות שמזהות אותה.

לכל מכונה כזאת יש קידוד שונה. לכן בהינתן $< M_1 >$, השפה $L(M_1)$ שייכת לRE ולכן בהכרח יש עוד אינסוף קידודי מכונות המקלות את אותה השפה.

מכאן: בL_1 יש את כל קידודי המכונות ולכן $L_1 = \Sigma^* \in R$.

ב. הערה: לקיים בדרך כלל ניתן לנחש! הניחוש מתוך קבוצה אינסופית זה בRE ומוציא מR.

הערה: גם אם אנחנו מניחים שהמכונות מקיימות את התנאי – חשוב לבדוק האם יש הפעלה של מכונה שיכולה לא לעצור!

השפה לא בRE (בפרט לא בR)

נראה רדוקציה: $HP^C \le L_2$ על ידי $f(<M,x>) =< M_1, M_2 >$

כאשר:

M_1: על קלט w:

- אם $w=0$ – קבל.
- אם $w=1$ – דחה.
- אחרת – דחה.

M_2: על קלט y :

- אם $y=1$ - קבל.
- אם $y=0$:
  - הרץ את M על x.
  - אם M עצרה – קבל את 0.
- אחרת – דחה.

הסבר: יש לי מילה שמתקבל במקום אחד ולא בשני.. זה 1 אבל 0 אני בודק רק אם זה בסדר בHP^c .

הרדוקציה מלאה וניתנת לחישוב: כתיבת תיאור של 2 מכונות.

תקפות:

אם M לא עוצרת על x (כלומר הקידוד של $<M,x>\in HP^c$) אז M_1 תקבל את 0 ולא את 1. וM_2 מקבלת את 1 ולא את אפס (כי היא לא עוצרת על 0). ולכן: $< M_1, M_2 >\in L_2$.

אם M עוצרת על x (כלומר הקידוד $<M,x>\notin HP^c$) אז M_1 מקבלת את 0 ולא את 1. M_2 מקבלת גם את 1 וגם את 0. (עבור שטר המילים שתיהן דוחות. לכן $< M_1, M_2 >\notin L_2$.

ג. הערה: אם רשום לי בפרוש **דוחה** אני יכול להיות בRE כי זה בפירוש. אבל זה לא בR כי מכונה גם יכולה להיתקע.

השפה ב RE ולא בR.

נוכיח שהשפה בRE:

נגדיר את המכונה הא"ד הבאה (בכח החישוב א"ד שקול לדטרמיניסטי)

N_3 על קלט $< M_1, M_2 >$:

- נחש 2 מילים: x_1x_2.
- הרץ את M_1 על x_1 – אם דחתה דחה.
- הרץ את M_1 על x_2 – אם קיבלה דחה.
- הרץ את M_2 על x_1 – אם קיבלה דחה.
- הרץ את M_2 על x_2 – אם דחתה דחה.
- קבל.

נכונות:

אם $< M_1, M_2 >\in L_3$ אז קיים ניחוש של 2 המילים המובטחות כך ש M_1 תקבל את x_1 ותדחה את x_2 וגם M_2 תקבל את x_2 ותדחה את x_1 ולכן כל הבדיקות יעבור ונקבלץ

אם $< M_1, M_2 >\notin L_3$ אז לא קיימות 2 מילים כנ"ל ולכן כל ניחוש של 2 מילים יגרום לאחד התנאים ליפול או שאחת המכונות לא תעצור ואז גם N_3 לא תעצור.

נוכיח שהשפה לא בR.

נראה רדוקציה מ$HP \leq L_3$ על ידי $f(<M,x>) =< M_1, M_2 >$ כאשר:

כאשר:

M_1: על קלט w:

- אם $w = 0$ – קבל.
- אם $w = 1$ – דחה.
- אחרת – דחה.

M_2: על קלט y :

- אם $y = 0$ - דחה.
- אם $y = 1$:
  - הרץ את M על x.
  - אם M עצרה – קבל.
- אחרת – דחה.

הסבר: יש לי מילה שמתקבל במקום אחד ולא בשני.. זה 1 אבל 0 אני בודק רק אם זה בסדר בHP^c .

הרדוקציה מלאה וניתנת לחישוב: כתיבת תיאור של 2 מכונות.

תקפות:

אם M עוצרת על x (כלומר הקידוד של $HP \ni< M,x >$) אז M_1 תקבל את 0 ודוחה 1. וM_2 מקבלת את 1 ודוחה את אפס . ולכן: $< M_1, M_2 >\in L_2$.

אם M עוצרת על x (כלומר הקידוד $HP \notin< M,x >$) אז M_1 מקבלת את 0 ודוחה את 1. M_2 דוחה את 0 ולא עוצרת על 1. (עבור שטר המילים שתיהן דוחות. לכן $< M_1, M_2 >\notin L_2$.)

הערה:

מכונה א"ד מוכיחה ששפה היא בR אם :

1. כל ניחוש הוא מקבוצה סופית. הניחוש הוא מטווח סופי של איברים.
2. לא מריצים מכונה שעלולה לא לעצור.

3. (22 נקודות)

א. (8 נקודות) בהרצאה למדנו (ללא הוכחה) ש $P \subsetneq R$. בסעיף זה מותר להשתמש במשפט זה. הוכח או הפרך: קיימות זוג שפות L_1, L_2 כך ש $L_1 \leq L_2$ אבל לא $L_1 \leq_p L_2$.

ב. (6 נקודות) בסעיף זה נניח כי $P \neq NP$. הוכח או הפרך: קיימת תת קבוצה אינסופית $H \subseteq NP \setminus P$, כך ש $\forall L_1, L_2 \in H, L_1 \leq_p L_2$.

ג. (8 נקודות) תהי L_1 שפה כלשהי. אזי קיימת שפה L_2 כך ש לא מתקיים $L_2 \leq_p L_1$.

4. (28 נקודות)

א. (12 נקודות) תהי $S \subseteq RE$ תכונה של שפות ב RE, המוגדרת כך:
$S = \{L \in RE || L \cap (\{0,1\}^2)^*| = \infty \text{ and } |(\{0,1\}^2)^* \setminus L| = \infty\}$
בעברית: S היא קבוצת כל השפות ב RE המכילות אינסוף מילים באורך זוגי, וכן קיימות אינסוף מילים באורך זוגי שאינן בשפה. הוכיחו ש $L_S \notin RE$.

א. הוכחה: לפי הנתון, ש$P \subset R$ נובע שקיימת $L \in R$ כאשר $L \notin P$.

ניקח: $L_1 = L \in R \backslash P$ ו$L_2 = \{0\} \in P$.

לפי המשפט: לכל שפה $L \in R$ מתקיים $L \leq L'$ לכל $L' \neq \Phi, \Sigma^*$ נובע שקיימת רדוקציה $L_1 \leq L_2$.

לפי משפט הרדוקציה עבור סיבוכיות (פולינומית): אם $L \leq_p L'$ וגם $L' \in P$ אז $L \in P$ נובע שלא קיימת רדוקציה $L_1 \leq_p L_2$ כי אחרת מכיוון ש$L_2 \in P$ אז $L_1 = L \in P$ וזו סתירה.

ב. אם $P \neq NP$ אז כל שפה שהיא NPC היא לא בP .

לכן מספיק להוכיח שיש אינסוף שפות שהן NPC.

ניקח את: $H = \{kSAT : k \geq 3 . k \in \mathbb{N}\}$.

נוכיח שכולן NPC. באינדוקציה על k:

בסיס - $k = 3$ מתקיים: ש $3SAT \in NPC$ לפי משפט קוק לוין.

צעד – נניח ש$kSAT \in NPC$ צריך להוכיח $k + 1\ SAT \in NPC$.

נראה רדוקציה $(k + 1)SAT \leq_p kSAT$ ולפי הנחת האינדוקציה $kSAT \in NPC$ לכן $k + 1SAT \in NPC$.

הרדוקציה: $f(\phi) =< \phi' >$.

כאשר ϕ' מוגדרת כך: עבור כל פסוקית $C_i \in \phi$ נוסיף לϕ' שני פסוקיות : $(C_i \vee y_i) \wedge (C_i \vee \overline{y_i})$.

הערה: $2SAT \in P$

סיבוכיות: שכפול הקלט פעמיים והוספת משתנה אחד עבור כל פסוקית. פולינומי.

תקפות:

אם $\phi \in kSAT$ אז יש השמה מספקת לϕ ולכן כל פסוקית תסתפק ולמכאן לא משנה איזה ערך נבחר למשתנים החדשים: $y_1, \dots, y_m$ (כאשר m = מספר הפסוקיות) נקבל שהשמה זו מספקת את ϕ'.

אם $\phi' \in k + 1\ SAT$ אז יש השמה מספקת לכל פסוקיות ϕ' ומכיוון שכל 2 פסוקיות הן מהצורה $(C_i \vee y_i) \wedge (C_i \vee \overline{y_i})$ אז בהכרח ההשמה מספקת ברט את C_i (כי לא ניתן לספק גם את y_i וגם את $\overline{y_i}$) ומכאן השמה זו מספקת גם את ϕ.

בנוסף, $kSAT \in NP$.

האלגוריתם: קלט ϕ.

- בדור שי בכל פסוקית של ϕ בדיוק k ליטרלים.
- הרץ את M_{SAT} על ϕ וענה כמוה.

סיבוכיות: בדיקת הפסוקיות – פולינומי בגודל הקלט.

הרצת M_{sat} פולינומי.

נכונות:

לפי הבדיקה ונכונות המכונה של SAT .

הערה: כדי להוכיח אפשר להגיד שייך גורר שייך ואז לא שייך גורר לא שייך או במקום לא שייך לעשות שייך מהכיוון ההפוך.

נשים לב שלפי הגדרת NPC , כל שפה בNP (שזה כולל NPC אחרות) ניתנות לרדוקציה לשפה בNPC.
ולכן קיימת רדוקציה בין כל 2 שפות בH.

ג. נוכיח משיקולי ספירה.

בהינתן L_1. כמות השפות שיש להן רדוקציה לL_1 היא בת מנייה לכל היותר $\aleph_0$ כיוון שכל רדוקציה היא מ"ט אחרת ומספר המכונות שקול לכמות הקידודים השונים בΣ^* השווה ל$\aleph_0 = |\Sigma^*|$.

כמות השפות שיש היא: $|P(\Sigma^*)| = \aleph$ ולכן בהכרח קיימת שפה L_2 כך שלא קיימת רדוקציה ממנה לL_1.

4. (28 נקודות)

א. (12 נקודות) תהי $S \subseteq RE$ תכונה של שפות ב RE, המוגדרת כך:
$S = \{L \in RE \mid |L \cap (\{0,1\}^2)^*| = \infty \text{ and } |(\{0,1\}^2)^* \setminus L| = \infty\}$
בעברית: S היא קבוצת כל השפות ב RE המכילות אינסוף מילים באורך זוגי, וכן קיימות אינסוף מילים באורך זוגי שאינן בשפה. הוכיחו ש $L_S \notin RE$.

ב. (7 נקודות) עבור פונקציה $f : \Sigma^* \leftarrow \Sigma^*$, נסמן ב $S(f)$ את קבוצת המילים ב Σ^* עליהן f מוגדרת. תנו דוגמה לסדרה אינסופית של פונקציות לא מלאות מ $\{0,1\}^*$ ל $\{0,1\}^*$ $f_1, f_2, \ldots,$ כך שכולן ניתנות לחישוב וכן לכל $i \geq 1$ מתקיים: $S(f_i) \subsetneq S(f_{i+1})$. הוכיחו בקצרה את תשובתכם.

ג. (9 נקודות) בנו אלגוריתם (פורמלית מכונת טיורינג M) פולינומי עם גישת אוב לשפה IS (independent set) שעל קלט (G, k) מחזיר "לא קיים" אם לא קיימת קבוצה בלתי תלויה בגודל k ב G, ואחרת מחזירה קבוצה כזו. תזכורת: גישת אוב ל IS מאפשרת למכונה M לשאול קופסא "קסומה" שאלות מהסוג "האם מילה y שייכת ל IS?" ולקבל תשובה "כן" או "לא" בצעד אחד (שנחשב כצעד אחד לסבוכיות זמן הריצה של M).

א. נראה רדוקציה מ$L_s \leq H\bar{P}$. על ידי : $f(<M, x>) = <M'>$.

($H\bar{P}$ בעצם M לא עוצרת על x והאי עצירה בעצם משאירה אותי במצב תקין)

כאשר M' : קלט w:

- אם $|w|$ אי זוגי דחה.
- $|w|$ מתחלק ב4 עם שארית 2 – קבל.
- $|w|$ מתחלק ב4 :
  - הרץ את M על x .
  - קבל.

הרדוקציה מלאה וניתנת לחישוב: תיאור של מ"ט שעושה בדיקות של השוואה ומרציה מכונה אחרת.

תקפות:

אם M לא עוצרת על x (כן ב$H\bar{P}$) אז M' תקבל אינסוף מילים באורך זוגי (אלה שאורכן מתחלק ב4 עם שארית 2) ולא תקבל אינסוף מילים באורך זוגי (אלה שאורכן מתחלק ב4 ללא שארית) כי היא לא תעצור עליהן. ולכן $<M'> \in L_s$.

אם M עוצרת על x אז M' תקבל את כל המילים באורך זוגי ולכן לא יהיו אינסוף מילין באורך זוגי שM' לא תקבל ולכן: $<M'> \notin L_s$.

לפי משפט הרדוקציה $HP^c \notin RE$ אז $L_s \notin RE$.

ב. דוגמה: $F = \{f_i : i \in \mathbb{N}\}$ כך ש: $f_i(x) = x$ רק עבור i המילים הראשונות בסדר לקסיקוגרפים בΣ^* ואחרת, הפונקציה לא מוגדרת.

מתקיים ש $S(f_i) \subset S(f_{i+1})$ לפי בניית הפונקציות.

כל אחת מהן ניתנת לחישוב על ידי האלגוריתם הבא לf_i:

M_{f_i} על קלט x :

- אם x הוא אחת מi המילים הראשונות בסדר לקסיקוגרפי של Σ^*.

- החזר את x.
- אחרת :
  - כנס ללולאה אינסופית.

ג. הערה: גישת אוב נגיד שיש מכונת קסם דטרמיניסטית שמחזירה אם הקלט בשפה או לא.

הערה: בהינתן בסיבוכיות טובה שניתן לפתור את בעיה ההכרעה תנו תשובה לבעיית חיפוש.

הדרך להתמודד עם זה – לעבור אובייקט אובייקט ולשאול את השאלות, לבדוק האם זה שייך או לא לתשובה עצמה. נניח לעבור קודקוד ומסירים ובודקים עם המכונת אוב.

אלגוריתם: על קלט $< G, k >$:

- שאל את הרוב האם $< G, k > \in IS$ אם לא – החזר "לא קיים".
- הגדר $G' = G$.
- עבור $v \in V(G)$ בצע:
  - שאל את האוב האם $< G\big(V - v, E - E(v)\big), k >$. (להוריד קודקוד ואת הצלעות שלו)
  - אם כן, הגדר: $G' = G' - v$.
- החזר את הקודקודים שנשארו בG' כתשובה.

סיבוכיות:

שאלת האוב $O(1)$. + העתקת הקלט לסרט השאלה: חסום בגודל הקלט.

עבור כל קודקוד מייצרים גרף חדש שמורידים ממנו את הקודקוד: פולינומי בגודל הקלט.

סה"כ פולינומי בגודל הקלט.

מבחן 2020 א' א'

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

א. $= \{(< G(V,E) >, k) | G$ 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. $= \{(< G(V,E) >, k) | G$ 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

נתונים באונארי).

$L_3 = \{\psi(x1, \ldots, x_n) | \psi$ 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.

א. ..
ב. ..
ג. השפה שייכת לNPC .

השפה שייכת ל NP: נראה את המ"ט א"ד פולינומית הבאה:

M_3 על קלט $< \phi >$:

- נחשב 2 השמות $z_1 z_2$ לכל משתני ϕ.
- בדוק שיש לפחות 20 משתנים ממשתני ϕ כך שערך האמת שלהם בz_1 שונה במערך האמת שלהם בz_2. אם לא – דחה.
- הצב את z_1 בϕ ובדוק את ערך האמת אם $False$ – דחה .
- הצב את z_2 בϕ ובדוק את ערך האמת אם $False$ – דחה .
- קבל.

סיבוכיות:

ניחוש 2 ההשמות - $O(n)$ על כל אחת.

מעבר והשוואה בין 2 השמות: $O(n)$ כפול זמן עדכון קאונטר (פולינומי).

הצבה ובדיקת ערך האמת עבר כל השמה : פולינומי בגודל הקלט.

נכונות:

אם $\phi \in L_3$ אז יש 2 השמות כנ"ל ולכן ננחש אותם ונבדוק ונקבל.

אם $\phi \notin L_3$ אז לכל ניחוש, נקבל 2 השמות שאחת מהן לא מספקת ונדחה או שנקבל 2 השמות שלא שונות אחת מהשניה בלפחות 20 משתנים ונדחה.

נראה עכשיו שהיא בNPH. נראה רדוקציה: $SAT \leq_p L_3$ על ידי $f(< \phi >) =< \phi' >$.

$$\phi' = \phi \wedge (y_1 \vee \overline{y_2}) \wedge \ldots . (y_{20} \vee \overline{y_{20}})$$

הרדוקציה מלאה וניתנת לחישוב.

סיבוכיות: העתקת הקלט + הוספה של 20 – פסוקיות עם 2 ליטרליים (משתנה אחד **חדש**) בכל פסוקית.

סה"כ 20 משנים חדשים שלא היו ב ϕ. חסום בגודל הקלט.

תקפות:

אם $\phi \in SAT$ אז יש השמה מספקת z למשתני ϕ ומכאן : $(z, y_1 = True, .. y_{20} = True)$ היא השמה מספקת לϕ' וגם $(z, y_1 = False, .. y_{20} = False)$ היא השמה מספקת בϕ' ומכאן קיבלנו 2 השמות מספקות עם 20 משנים שקיבלו ערך אמת שונה בכל השמה. ולכן $< \phi' > \in L_3$

הערה: לפחות אומר שאני צריך להוכיח על התנאי התחתון שאנחנו מקבלים.

אם $\phi \notin SAT$ אז כל השמה של z. למשתני ϕ נקבל ש: $\phi(x) = False$ ומכאן : לכל השמה w למשתני ϕ' נקבל .... $\phi'(w) = False \wedge$ ולכן לא קיימת בכלל השמה מספקת לϕ' . ולכן $< \phi' > \notin L_3$.

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

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.

א. הערה: הקידוד הלא תקין הוא שפה שלא מקבל כלום. שפה ריקה.

השפה היא בR.

נשים לב שקידוד המכונה שמקבלת את Σ^* היא באורך הקצר ביותר הוא באורך k.

נסמן בS את התנאי של השפה L_1

מכאן: $L_1 = \{< M >: | < M > | \leq k \text{ and } S\} \cup \{< M >: |< M >| > k \text{ and } S\}$

נשים לב שהשפה $\{< M >: |< M >| > k \text{ and } S\}$ היא סופית ולכן כריעה.

נשים לב שעבור כל מכונה בשפה $\{< M >: |< M >| > k \text{ and } S\}$ נבחר את המכונה Σ^* עם הקידוד באורך k. ולכן התנאי בהכרח יתקיים ומכאן השפה הנ"ל היא שפת כל קידודי המכונות באורך גדול מk ולכן היא כריעה: כי תור רק לבדוק שאורך הקידוד של המכונה שקיבלנו גדול מk.

מכאן לפי סגירות של איחוד בR מתקיים: $L_1 \in R$.

ב. השפה לא בRE.

נראה רדוקציה : $HP^c \leq L_2$ על ידי $f(< M, x >) =< N >$ כאשר:

N על קלט x:

- הרץ את M על x.
- קבל.

הרדוקציה מלאה וניתנת לחישוב.

תקפות: אם M לא עוצרת על x אז $L(N) = \Phi$ ולכן לא יהיו בה מילים באורך זוגי ומכאן קיימת מ"ט עם קידוד ארוך יותר שיש לה יותר מ 0 מילים באורך זוג בשפה. ולכן $< N >\in L_2$.

אם M עוצרת על x אז $L(N) = \Sigma^*$ ולכן לא קיימת מ"ט אחרת שמכילה ממש את כל המילים באורך זוגי ש N. (כי N מקבלת את כולן). ולכן: $< N >\notin L_2$.

ג. השפה ב$R\backslash RE$.

השפה בRE נראה מכונה א"ד עבור השפה:

N_3 על קלט $< M >$:

- נחש קידוד של מ"ט $< M' >$ כך שהקידוד של $< M' >$ קצר ב2 או 3 מהקידוד של $< M >$.
- נחשב מילה x.
- הרץ את M על x אם דחתה – דחה.
- הרץ את M' על x אם דחתה – דחה.

- קבל.

נכונות:

אם $<M> \in L_3$ אז קיימת M' כנ"ל ולכן קיים ניחוש שינחש אותה. ובנוסף החיתוך בין השפות של M וM' לא ריק ולכן קיימת מילה x ששתיהן מקבלות ולכן קיים ניחוש של המילה הזו ומכאן נקבל.

אם $<M> \notin L_3$ אז לא קיימת מ"ט M' עם קידוד הקצר ב2 או 3 שיש לה חיתוך עם השפה של M ולכן לכל מכונה בטווח הנ"ל שננחש ולכל מילה x שננחש, לפחות אחת מהן תדחה ולכן נדחה.

השפה לא בR. רדוקציה: $HP \leq L_3$ על ידי $f(<M,x>) = <M'>$.

M' על קלט w:

- הרץ את M על x.
- קבל.

קיימות אינסוף מכונות M' שמקיימות את האלגוריתם הנ"ל ולכן בהכרח יש קידוד שאם ניקח 2 או 3 פחות ממנו, קיים בין היתר קידוד של מכונה שמקבלת משהו ואז החיתוך לא 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, שבהינתן 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 $L_1 \notin NP$.

א. $SC = Set\ Cover$ לקחת K קבוצות מכל ה-C-ים שהן נותנות לנו את n.

אלגוריתם A: קלט $(n,k,C_1 \ldots C_t)$.

- הרץ את M_{SC} על הקלט – אם דחה, החזר "לא קיים".
- קבע $C = \{C_1, \ldots . C_t\}$
- עבור i מ1 עד t:
  - הורד את הקבוצה הi מהקלט. $C - C_i$.
  - הרץ את M_{Sc} על שאר קבוצות הקלט: (n,k,C).
    - אם קיבלה $C := C - C_i$.
- החזר את C.

ב. למצוא קבוצה k מינימלי.

אלגוריתם B: קלט $(n,k,C_1 \ldots C_t)$.

- עבור k מ1 עד t:
  - הרץ את A על $(n, k, C_1 \dots C_t)$.
    - אם חזרה קבוצה – החזר אותה.
- החזר "לא קיים".

ג. הוכחה: נניח ש$L_2 \leq_p L_1$ ונניח ש$L_2 \in NP$.

לפי ההנחות קיימת מכונה דטרמיניסטית פולינומית: M_f המחשבת את הרדוקציה.

וקיימת מכונה א"ד פולינומית: M_2 עבור L_2.

מגדיר מכונה א"ד עבור L_1:

M_1 על קלט x:

- הרץ את M_f על x וקבל את $y = f(x)$.
- הרץ את M_2 על y וכנה כמוה.

הערה: הרדוקציה חייבת להיות דטרמיניסטית.

סיבוכיות: הרצת M_f פולינומית בגודל x.

הרצת M_2 פולינומית בגודל $f(x)$. (כאשר גודל $f(x)$ הוא פולינומי בגודל x כי M_f פולינומית)

הערה: מכונה פולינומית יכולה להחזיר רק פלט פולינומי. ולא יכולה להחזיר לא פולינומי.

ולפי הרכבת פולינומים, סיבוכיות M_1 פולינומית בגודל x.

נכונות:

לפי נכונות M_f, M_2 ולפי תקפות הרדוקציה:

$x \in L_1 \leftrightarrow f(x) \in L_2$.

ד. לא נכון.

לפי משפט: כל שפה בP ניתנת לרדוקציה לכל שפה אחרת שאינה Φ, Σ^*.

ולכן L_1 יכולה להיות ב P וגם L_2 יכולה לא להיות ב NP (יכולה אפילו להיות ב$RE \backslash R$)

ה. (8 נקודות) נגדיר מחלקה חדשה:

$NP_{eq} = \{L|$There exists a non deterministic 2-tape polynomial turing machine M such that L(M)=L. Furthermore, for every x, all paths in the 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} דרך מ"ט חד סרטית.

ה. אם יש מכונה א"ד אני יכול ליצור מכונה כזאתי.

הוכחה: תהי $L \in NP$ צ"ל: $L \in NP_{eq}$.

לפי ההנחה, קיימת מכונה א"ד פולינומית N_L כך ש $L(N_L) = L$.

בנוסף ידוע כי זמן ריצת N_L הוא $p(n)$ פולינומי.

הערה: הפולינום ידוע לנו לכל מכונה. אני יודע את הסיבוכיות של המכונה N_L.

נגדיר את המכונה הבאה:

N_{eq} : על קלט x:

- חשב את $p = p(|x|)$.
- האתחל בסרט השני $counter = p(|x|)$.
- סמלץ את ריצת N_L על x כאשר בכל צעד שלה, הורד את ה$counter$ ב1.
- אם N_L קיבלה או דחתה כאשר $counter > 0$
  - אז בצע עוד $counter$ צעדי סרק וקבל\דחה בהתאם.

המכונה N_{eq} פולינומית כי N_L פולינומית ומספר הצעדים בכל מסלול שווה לזמן חישוב $p(|x|)$ + זמן אתחול $counter$ + זמן הסמלות וצעדי הסר' לפי ה$counter$.

בנוסף: $L = L(N_L) = L(N_{eq})$ ולכן $L \in NP_{eq}$.

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
$L = \{< M > |L(M) = \phi$, and M makes at least |x| 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.

א. פתרון:

1. $L_S^2 \in R$ אמ"מ S היא תכונה טריוויאלית.
2. הוכחה:

כיוון 1: נניח ש S טריוויאלית. ולכן: כל $L \in RE$ מקיימת את S או כל $L \in RE$ לא מקיימת את S.

במקרה הראשון נקבל ש: L_S^2 מכילה את כל הקידודים האפשריים של זוג מכונות ולכן עבור כל קלט פשוט נקבל. $L_S^2 \in R$.

במקרה השני נקבל ש: L_S^2 לא מכילה אף קידודים האפשרי של זוג מכונות ולכן עבור כל קלט פשוט נדחה. $L_S^2 \in R$.

כיוון 2: נניח שS לא טריוויאלית. ולכן: צ"ל: $L_S^2 \notin R$.

מקרה א: $S \notin \Phi$ ולכן קיימת שפה אחרת $L' \in S$ ובפרט $L' \in RE$ ולכן יש לה מכונה מזהה $M_L{}'$.

נראה רדוקציה: $HP \le L_S^2$: $f(<M,x>) =< M_1, M_1 >$ כאשר:

M_1 על קלט w:

- הרץ את M על x.
- הרץ את M_L' על w וענה כמוה.

תקפות:

אם M עוצרת על אז $L(M_1) = L(M_L') = L' \in S$.

אם M לא עוצרת על x אז $L(M_1) = \Phi \notin S$.

מקרה ב: $\Phi \in S$ ולכן קיימת שפה אחרת $L' \notin S$ ולמרות זאת $L' \in RE$ ולכן יש לה מכונה מזהה $M_L{}'$.

נראה רדוקציה: $HP^c \le L_S^2$: $f(<M,x>) =< M_1, M_1 >$ כאשר:

M_1 על קלט w:

- הרץ את M על x.
- הרץ את M_L' על w וענה כמוה.

אם M עוצרת על אז . $L(M_1) = \Phi \notin S$.

אם M לא עוצרת על x אז $L(M_1) = L(M_L') = L' \in S$.

ב. נראה שמשפט רייס לא עוזר להוכיח כיוון שתנאי השפה הוא לא תכונה של שפות.

נגדיר את שתי המכונות הבאות:

M_1 : על קלט x:

- דחה.

M_2: על קלט x:

- כנה ללולאה אינסופית.

מכאן $L(M_1) = L(M_2) = \Phi$.

אבל $< M_1 > \notin L$ כי על קלט 000 היא עושה רק צעד 1 ולא 3.

אבל $< M_2 > \in L$ כי על כל קלט היא עושה אינסוף צעדים.

הקשר בין מחלקות:

$P \subseteq NP \subseteq R \subseteq RE$

$coNP \subseteq R$

$NPC \subseteq NP \subseteq R$

$NPC \subseteq NPH$

אבל NPH לא מוכלת ב$R, RE, coRE$.

<u>בעיות פתוחות:</u>

$P =^? NP$.

אם $P \neq NP$ אז כל שפה שהיא בNPC היא לא בP.

אם יש שפה NPC שהיא P אז $P = NP$ וגם $P \subset NPH$.

<u>רדוקציות בNPC</u>

- תמיד עדיף לא לשנות כלום אלא רק להעתיק ולהוסיף.
- אם זה גרף: מוסיפים קודקודים \ צלעות לרכיבים אחרים או שמחברים אותם לגרף המקורי כל עוד זה לא יכול לפגוע במה שרוצים למצוא (= או שכעת נמצא ביתר קלות או שיהיה יותר קשה).
- אם זה פסוק CNF מוסיפים פסוקיות חדשות או משתנים חדשים לכל פסוקית (או לחלק מהן) כל עוד לא ניתן לבחור אותו לערך מסוים ואז לגרום לפסוק להתקיים מיידית או לא להתקיים.
- שימו לב לטריקים:
  - שייתכן שהתנאים מגבילים כל-כך שזה בP.

2. לכל אחת מהשפות הבאות, קבעו האם היא ב P ב NPC או ב NPH. בכל סעיף הוכיחו את הטענה הכי חזקה שתוכלו. אם בסעיף מסויים הוכחתם רק שהשפה ב NPH, הסבירו מה הקושי בלהראות שהשפה ב NP (ההסבר אינו מהווה הוכחת אי שייכות ל NP, אלא מדוע גישה פשוטה שניסיתם לא עובדת). ניתן להניח שכל שפה שלמדנו בתרגול או בהרצאה שהיא NPC שהיא אכן NPC ללא צורך בהוכחה. הדבר נכון גם לגבי משפט Cook שנלמד בהרצאה האחרונה. הסעיפים המסומנים ב * קשים יחסית (ובגדול מעל רמת בחינה, פרט לסעיפים קשים השווים במצטבר יחסית מעט נקודות).

א. $3-SAT_{13} = \{\phi(x_1,\ldots,x_n) | \phi \text{ is a 3-CNF formula with at least 13 satisfying assignments.}\}$

ב. $-SAT_{>1/2} = \{\phi(x_1,\ldots,x_n) | \phi \text{ is a 4-CNF formula such that more than } 1/2 \text{ of its assignments are satisfying}\}$

ג. $VC_{100} = \{(<G>, k) | k \geq |V| - 100, \text{ and } (G,k) \in VC\}$

ד. $PART_{1/4} = \{(x_1,\ldots,x_n) | \exists I \subseteq [n], \sum_{i \in I} x_i = 1/4 \sum_{i \in [n]}\}$.

ד. שיטה לפתרון של זה.

~~נראה רדוקציה מ: $SS \leq_p Part_{1/4}$ על ידי $f(<x_1, \ldots, x_n, k>) = <4kx_1/s, \ldots, 4kx_n/s,>$~~

הערה: הרעיון הוא לעשות נירמול שאותה קבוצה שסכומה K היא גם תהיה זאתי.

~~נניח שקיימת קבוצה $I \subseteq [n]$ כל ש: $\Sigma_{i \in I} y_i = \frac{1}{4} \Sigma_{i \in [n]} y_i$ ומכאן $\Sigma_{i \in I}(4kx_i) = \frac{1}{4} \Sigma_{i \in [n]}(4kx_i)$~~

נניח מכונה פולינומית ל $PART_{\frac{1}{4}}$. נוכיח שקיימת מ"ט לSS.

(ג) ניתן לעשות חיפוש שלם כי $\binom{n}{100} = \binom{n}{n-100}$ לכן זה יהיה בP

(ב)