חישוביות וסיבוכיות תשפ"ה סמסטר א'
שיעור 5
בעיות NP שלמות

תוכן העניינים

5.1 הגדרת חישוב יעיל

עד כה כל הבעיות החישוביות שעסקנו בהן הניחו שהמשאבים שעומדים לרשות מכונת הטיורינג שפותרת אותן הם **בלתי מוגבלים**. כעת נעבור לעסוק בשאלה מה קורה כאשר אנחנו מגבילים חלק ממשאבים אלו. יש סוגים רבים של משאבים שניתן לעסוק בהם, אבל שני הנפוצים ביותר בתיאוריה של מדעי המחשב הם

- **זמן החיושב**
- **הזיכרון שנדרש לצורך החיושב.**

אחת מהבעיות שבהן נתקלים:
כשמעוניינים למדוד את צריכת המשאבים הללו של אלגוריתם מסויים היא שלא ברור כיצד למדוד אותם.
האם זמן חישוב נמדד בשניות?
אם כן, כיצד ניתן לחשב את זמן החישוב עבור אלגוריתם נתון?
האם עלינו לקודד ולהריץ אותו על מחשב מסויים?

אבל במחשבים שונים האלגוריתם ירוץ זמנים שונים בשל

- יעילות המעבד,
- אופטימיזציות שמתבצעות ברמת המעבד,
- אופטימיזציות בזמן הקומפליצה,

וכיוצא בהן.

אפילו תנאים חיצוניים כמו החום בסביבת המעבד עשויים להשפיע על זמן הריצה. מכאן הרצון למצוא הגדרה **תיאורטית** של זמן ריצה, שאינה תלויה בחומרה זו או אחרת.

מכונת טיורינג היא סביבה טבעית להגדרה כזו:

הגדרה 5.1: זמן הריצה

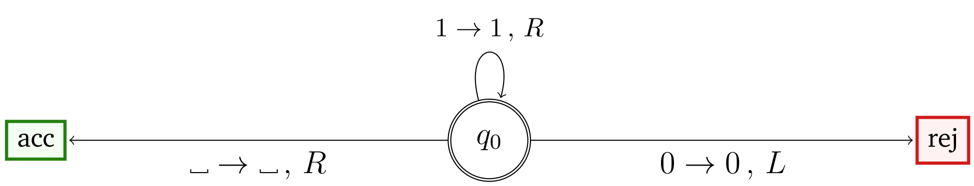
זמן הריצה של מכונת טיורינג M על קלט w הוא מספר צעדי החישוב ש- M מבצעת על w.

נשים לב לכך שמספר צעדי החישוב עשוי להיות אינסופי, אבל כמובן שסיטואציה כזו אינה רלוונטית לנו אם אנו מעוניינים להגביל את זמן הריצה.

בעיה נוספת היא שעל פי רוב, זמן הריצה של אלגוריתם תלוי ב **גודל הקלט** $|w|$ שמוזן אליו.

דוגמה 5.1

נתבונן על מ"ט $M = (Q, q_0, \Sigma, \Gamma, \delta, \text{acc}, \text{rej})$ כאשר $\Sigma = \{0,1\}$ ו- $\Gamma = \{0,1,\sqcup\}$ שמכריעה את השפה $L = \{1^n \big| n \geq 1\}$ של מחרוזות האוניריות.



- על קלט w, המכונה פשוט עוברת סדרתית על כל תווי w.
- אם אחד מהם 0 היא דוחה,
- ואם הגיעה אל $\sqcup$ שבסוף הקלט היא מקבלת.
- ככל שהקלט ארוך יותר, כך M תבצע צעדי חישוב רבים יותר.

ברור כי המספר הצעדים המקסימלי הוא במקרה שבקלט w יש רק תווי 1 ולכן המ"ט תבצעת $n = |w|$ צעדי חישוב. בכל מקרה אחר היא תבצעת $n >$ צעדים.

אנחנו אומרים כי "זמן הריצה של M הוא n" כאשר n אורך הקלט, בגלל שמספר הצעדים המקסימלי הוא n.

באופן כללי, אם מכונה על קלט w מבצעת פחות מ־ $|w|$ צעדי חישוב, המשמעות היא המכונה כלל לא קראה את כל הקלט, וזה אינו מקרה נפוץ במיוחד (אם כי הוא בהחלט קיים). אם כן ברור שמדידת זמן הריצה היא תמיד **ביחס לאורך הקלט**.

הגדרה 5.2: סיבוכיות זמן ריצה

תהי M מ"ט דטרמיניסטית אשר עוצרת על כל קלט. **הזמן הריצה** או **הסיבוכיות זמן** של M היא פונקציה $f:\mathbb{N}\to\mathbb{N}$, כאשר $f(n)$ המספר צעדי חישוב המקסימלי ש- M מבצעת על קלט w של אורך n.

אם $f(n)$ זמן הריצה של M, אומרים כי M רץ בזמן $f(n)$ וש- M היא זמן מכונת טיורינג

אנחנו נייצג את הזמן הריצה **בסימון אסימפטוטי** או **סימון O גדולה.**
למשל, נתונה הפונקציה $f:\mathbb{N}\to\mathbb{N}$ שמוגדרת $f(n)=6n^3+2n^2+20n+45$. בסימון $O(f)$ אנחנו רושמים רק החזקה הכי גדולה ללא המקדם, כלומר

$$f(n)=6n^3+2n^2+20n+45 \quad\Rightarrow\quad f(n)=O\left(n^3\right) \ .$$

הגדרה 5.3: סימון אסימפטוטי

תהיינה f,g פונקציות

$$f:\mathbb{N}\to\mathbb{R}^+ \ , \qquad g:\mathbb{N}\to\mathbb{R}^+$$

כאשר $\mathbb{R}^+$ הממשיים הלא שליליים.
אומרים כי

$$f(n)=O\left(g(n)\right)$$

אם קיימים שלמים c ו- n_0 עבורם לכל $n\geq n_0$ מתקיים

$$f(n)\leq cg(n) \ .$$

אם $f(n)=O\left(g(n)\right)$ אורמים כי $g(n)$ חסם עליון אסימפטוטי של $f(n)$.

דוגמה 5.2

תהי $f:\mathbb{N}\to\mathbb{R}^+$ פונקציה שמוגדרת

$$f(n)=6n^3+2n^2+20n+45 \ .$$

הוכיחו כי $f(n)=O\left(n^3\right)$.

פתרון:
$f(n)=6n^3+2n^2+20n+45$. לכל $n\in\mathbb{N}$ מתקיים $n^2\leq n^3$, $n\leq n^3$, ו- $1\leq n^3$. לכן

$$f(n)=6n^3+2n^2+20n+45\leq 6n^3+2n^3+20n^3+45n^3=73n^3 \ .$$

נגדיר $g(n)=n^3$. מצאנו $n_0=1$ ו- $c=73$ כך שלכל $n\geq n_0$ מתקיים

$$f(n)\leq cg(n) \ .$$

לכן $f(n)=O\left(g(n)\right)=O\left(n^3\right)$.

משפט 5.1:

לכל $a,b,n\in\mathbb{R}$,

$$\log_a n=\frac{\log_b n}{\log_b a} \ .$$

ז"א מעבר מבסיס a לבסיס b משנה את הערך של הלוגריתם עד פקטור של $\frac{1}{\log_b a}$. מכיוון ששינוי של המקדם לא משנה את החסם עליון אסימפטוטי, במידה שההתנהגות האסימפטוטית של פונקציה כלשהי היא $\log_a n$ אנחנו פשוט רושמים $O(\log n)$ ללא הבסיס.

דוגמה 5.3

נתונה הפונקציה $f: \to \mathbb{R}^+$ שמוגדרת

$$f(n) = 3n\log_2 n + 5n\log_2\log_2 n + 2 \ .$$

הוכיחו:

$$f(n) = O\left(n\log n\right) \ .$$

פתרון:

לכל $n \geq 2$ מתקיים

$$\log_2(n) \leq n \ , \qquad 2 \leq n\log_2 n$$

לפיכך

$$\begin{aligned} f(n) =& 3n\log_2(n) + 5n\log_2\left(\log_2(n)\right) + 2 \\ \leq& 3n\log_2(n) + 5n\log_2(n) + 2 \\ \leq& 3n\log_2(n) + 5n\log_2(n) + 2n\log_2(n) \\ =& 10n\log_2(n) \end{aligned}$$

נגדיר $g(n) = n\log_2(n)$. מצאנו $n_0 = 2$ ו- $c = 10$ כך שלכל $n \geq n_0$ מתקיים

$$f(n) \leq cg(n) \ .$$

לכן $f(n) = O\left(g(n)\right) = O\left(n\log n\right)$.

לעתים אנחנו רושמים פונקציות כסכום של התנהגויות אסימפטוטיות, לדוגמה הפונקציה

$$f(n) = 6n^2 + 2n$$

ניתנת לרשום בצורה

$$f(n) = O\left(n^2\right) + O\left(n\right) \ .$$

כל O בביטוי זה מייצג מקדם כלשהו מודחק. מכיוון שה- $O\left(n^2\right)$ שולטת על ה- $O\left(n\right)$ ביטוי זה שקול ל- $f(n) = O\left(n^2\right)$.

אנחנו ראינו כי הסימון $f(n) = O(g(n))$ אומר שהפונקציה $f(n)$ שווה אסימפטוטי ל- $g(n)$ לכל היותר. הסימון $f(n) = o\left(g(n)\right)$ אומר שהפונקציה $f(n)$ פחות אסימפטוטי מ- $g(n)$. פורמלי:

הגדרה 5.4:

תהיינה f, g פונקציות

$$f: \mathbb{N} \to \mathbb{R}^+ \ , \qquad g: \mathbb{N} \to \mathbb{R}^+ \ .$$

אומרים כי

$$f(n) = o\left(g(n)\right)$$

אם

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \ .$$

במילים פשוטות, $f(n) = o(g(n))$ אם לכל מספר ממשי $c > 0$ קיים מספר ממשי n_0 כך ש- $f(n) < cg(n)$ לכל $n \geq n_0$.

דוגמה 5.4

א) $\sqrt{n} = o(n)$

ב) $n = o(n \log \log n)$

ג) $n \log \log n = o(n \log n)$

ד) $n \log n = o(n^2)$

ה) $n^2 = o(n^3)$

דוגמה 5.5

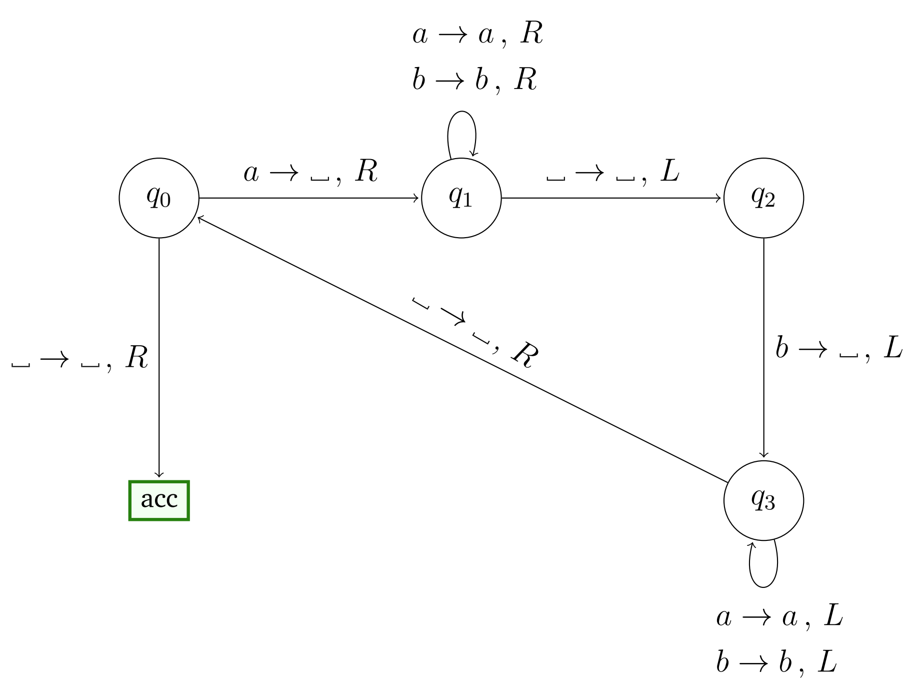
תהי M_1 המ"ט שמכריעה את השפה המילים

$$A = \{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\} \ .$$

חשבו את הזמן הריצה של M_1.

פתרון:

המכונה שמכריעה את הפשה מתוארת בתרשים למטה. המכונה, באופן איטרטיבי, עוברת על הקלט ובכל מעבר מורידה $\mathtt{a}$ מתחילת המילה ומורידה $\mathtt{b}$ מסוף המילה ומחליפה אותן ברווח.
אם לאחר מספר מעברים כאלו הסרט ריק, המכונה מרבלת, וזה יתקיים לכל מילה ורק מילים בשפה $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$.



האלגוריתם של המכונה מפורט להלן:

1) הראש מתחיל בסימן הראשון של הקלט. q_0:

- אם תו הנקרא b, $\leftarrow$ rej.
- אם תו הנקרא ␣ $\leftarrow$ acc.
- אם תו הנקרא a, כותבים עליה ␣, זז לסוף הקטלט ועוברים לשלב 2).

2)
- אם תו הנקרא a או ␣, $\leftarrow$ rej.
- אם תו הנקרא b כותבים עליה ␣, זז לתחילת הקלט ועוברים לשלב 3).

3) חוזרים על שלבי 1) ו- 2) שוב ושוב עד שהמ"ט מגיע למצב acc או rej.

בסבב הראשון, בשלב 1) המספר הצעדים המקסימלי שהמכונה מבצעת הוא $n+1$:
n צעדים כדי לזוז לסוף המילה ועוד צעד נוסף כדי להחזיר את הראש למשבצת האחרונה של הקלט.
בשלב 2) המכונה מבצעת n צעדים:
$n-1$ צעדים כדי לחזור לתו רווח הראשון ועוד צעד אחד כדי לשים את הראש בתו הראשון שאינו תו רווח.

לכן אחרי הסבב הראשון המספר הצעדים המקסימלי הוא $2n+1$.

בסבב השני יש $n-2$ תווים שאינם תוי רווחים.
לכן בסבב השני המכונה תבצע $2n-3$ צעדים לכל היותר.

בסבב השלישי, יהיו $n-4$ סימנים לסרוק והמכונה תבצע $2n-9$ צעדים לכל היותר.

בכללי, בסבב ה- k -ית המכונה מבצעת $2n-4k+5$ צעדים לכל היותר.
בסה"כ יהיו $\frac{n}{2}$ סבבים מקסימלי.

לכן המספר הצעדים המקסימלי הוא

$$\sum_{k=1}^{n/2}(2n+5-4k)=\frac{n^2}{2}+\frac{3n}{2}$$

לפיכך הזמן הריצה של M_1 הוא

$$O\left(n^2\right)+O\left(n\right)=O\left(n^2\right)\ .$$

דוגמה 5.6

תהי M_{L_2} המ"ט שמכריעה את השפת המילים

$$L_2=\{\mathrm{a}^n\ \big|\ n=2^k\ ,k\geq 0\}\ .$$

חשבו את הזמן הריצה של M_{L_2}.

פתרון:

1) סבב $\underline{k=1}$

נתון הקלט למשל

␣	␣	a	a	a	a	a	a	a	a	␣	␣
		$\uparrow$									

נתחיל בתו הראשון ונעבור על סרט הקלט משמאל לימין ומבצעים מחיקה לסירוגין של האות a, כלומר אות אחת נמחק ואות אחת נשאיר וכן הלאה.

␣	␣	$\checkmark$	a	$\checkmark$	a	$\checkmark$	a	$\checkmark$	a	␣	␣
									$\uparrow$		

אחרי סבב הראשון

- אם יש $\checkmark$ בתו האחרון $\Leftarrow$ יש מספר אי-זוגי אותיות a אחרי חילוק ב- $2 \Leftarrow$ אין חזקת 2 אותיות a. $\leftarrow$ rej.
- אם יש a בתו האחרון $\Leftarrow$ יש מספר זוגי אותיות a אחרי חילוק ב- 2 ונמשיך לשלב 2).

2) הראש חוזר לתו הראשון של הקלט

␣	␣	$\checkmark$	a	$\checkmark$	a	$\checkmark$	a	$\checkmark$	a	␣	␣
		$\uparrow$									

3) חוזרים על שלבים 1) ו- 2), עד שהמכונה תגיע למצב rej או למצב שנשאר רק אות אחת של a, ואז המכונה עוברת למצב acc.

<u>סבב $k = 2$</u>

1')

␣	␣	$\checkmark$	$\checkmark$	$\checkmark$	a	$\checkmark$	$\checkmark$	$\checkmark$	a	␣	␣
									$\uparrow$		

2')

␣	␣	$\checkmark$	$\checkmark$	$\checkmark$	a	$\checkmark$	$\checkmark$	$\checkmark$	a	␣	␣
		$\uparrow$									

<u>סבב $k = 3$</u>

1'')

␣	␣	✓	✓	✓	✓	✓	✓	✓	a	␣	␣
									↑		

2”)

␣	␣	✓	✓	✓	✓	✓	✓	✓	a	␣	␣
		↑									

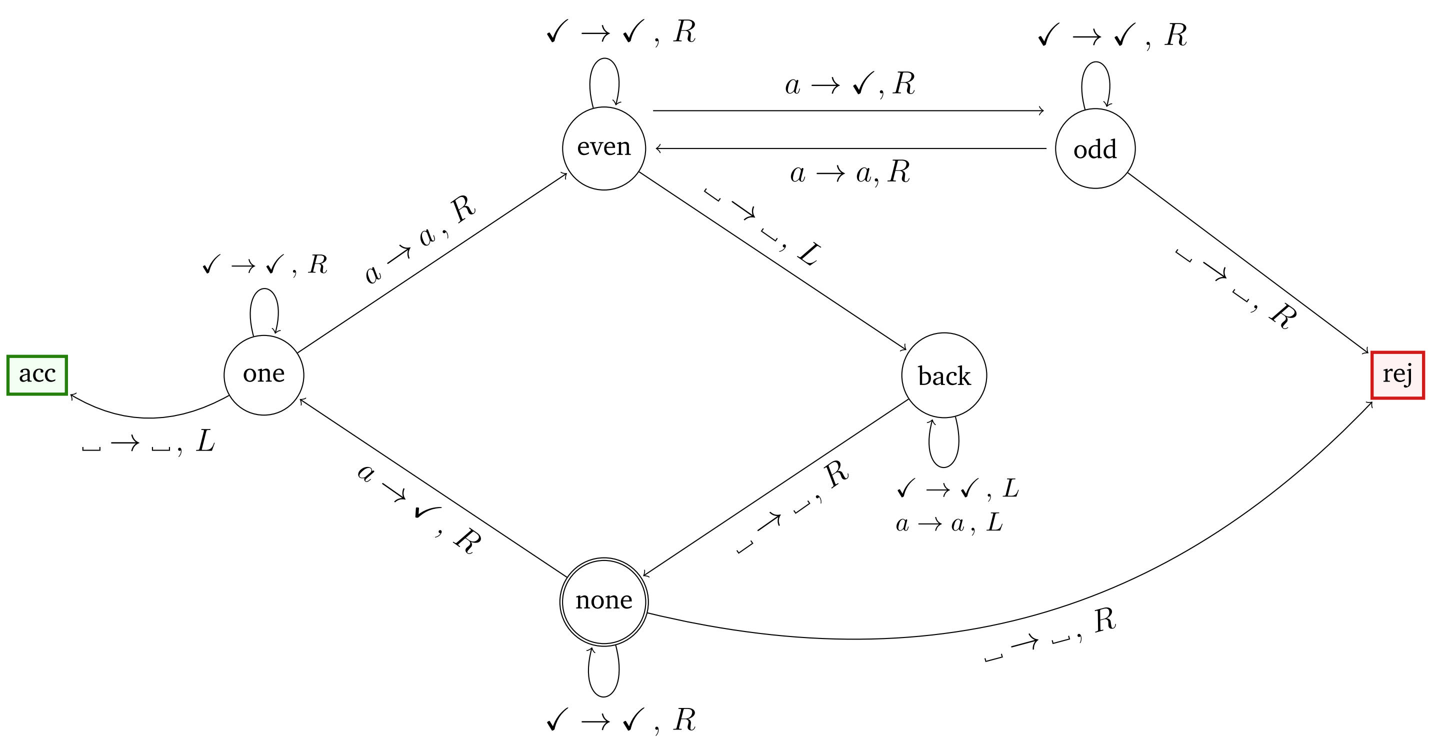
<u>סבב $k=4$</u>

1‴)

␣	␣	✓	✓	✓	✓	✓	✓	✓	a	␣	␣
									↑		

נשאר בדיוק אות a והמכונה ← acc.

המכונת טיורינג אשר מקבלת מילים בשפה זו מתוארת למטה.



בכל סבב, בסריקה מקצה השמאלי לקצה הימני המכונת טיורינג מבצעת $n+1$ צעדים לכל היותר: n צעדים לזוז לסוף הקלט וצעד אחד להחזיר את הראש לתו האחרון של הקלט.
אחרי זה המכונה מבצעת $n+1$ צעדים כדי להחזיר את הראש לתו הראשון של הקלט: n צעדים לתו רווח הראשון וצעד אחד להחזיר את הראש לתחילת הקלט.

המכונה חוזרת על התהליך הזה עד שנשאר אות a אחת בלבד. בכל סבב המכונה מבצעת חילוק ב- 2, לכן ידרשו $\log_2(n)$ סבבים עד שנשאר אות a אחת בלבד.

בשלב האחרון המכונה בודקת האם יש בדיוק אות a אחת, אשר דורש n צעדים בשביל הסריקה מקצה השמאל לקצה הימין, ועוד צעד אחד לתו רווח הראשון אחרי התו האחרון של הקלט.

לפיכך, בסה"כ יהיו לכל היותר

$$2(n+1)\log_2(n)+n+1=2n\log_2(n)+2\log_2 n+n+1$$

צעדים. לכן זמן הריצה הוא

$$O\left(n\log n\right)+O\left(\log n\right)+O\left(n\right)+O\left(1\right)=O\left(n\log n\right)\ .$$

נקודה עדינה שנובעת מהגדרת סיבוכיות זמן ריצה בתור פונקציה של אורך הקלט היא שיש חשיבות **לאופן הייצוג** של הקלט כשאנו מנסים לקבוע אם אלגוריתם הוא יעיל או לא. דוגמה קלאסית היא האלגוריתם הנאיבי לבדיקת ראשוניות:

עבור מספר n ניתן לבדוק אם n ראשוני או לא על ידי מעבר סדרתי על כל המספרים $1<k<n$ ובדיקה האם k מחלק את n.

- אם כן $\leftarrow$ rej.
- אם לכל k הבדיקה נכשלה $\leftarrow$ acc.

אלגוריתם זה מבצע $O(n)$ פעולות חלוקה ולכן הוא לכאורה יעיל, כי "זמן ריצה לינארי הוא יעיל". אך בפועל זה אלגוריתם לא יעיל מאוד.
הסיבה לכך היא שכדי לייצג את המספר n אנחנו צריכים רק ל- $\log n$ ביטים, והאלגוריתמים שלנו לחיבור, כפל וכו' של מספרים פועלים בסיבוכיות $O\left(\log n\right)$ ו- $O\left(\log^2 n\right)$ וכדומה.
כלומר אם n מיוצגבבסיס בינארי אז האלגוריתם שראינו דורש זמן אקספוננציאלי **בגודל ייצוג** n.
לעומת זאת, אם n מיוצג בבסיס אונרי, כלומר בתור 1^n, אז האלגוריתם לבדיקת ראשוניות אכן יהיה $O(n)$.

הגדרה 5.5: מחלקה של סיבוכיות זמן

המחלקת הסיבוכיות זמן מסומנת $\mathrm{TIME}\left(t(n)\right)$ ומוגדרת להיות אוסף של כל השפות אשר ניתנות להכרעה על ידי מכונת טיורינג בזמן $O\left(t(n)\right)$.

דוגמה 5.7

השפה

$$A=\left\{\mathtt{a}^n\mathtt{b}^n \;\middle|\; n\geq 0\right\}$$

המכונת טיורינג M_1 מכריעה את השפה A בזמן $O(n^2)$ לכן $A\in\mathrm{TIME}\left(n^2\right)$.

דוגמה 5.8

קיימת מכונת טיורינג M_2 שמכריעה את השפה A בצורה יותר יעילה, בזמן ריצה $O\left(n\log n\right)$.

האלגוריתם

1) סורקים את סרט הקלט משמאל לימין.

- אם נמצא a לצד ימין של b ← rej.
- אחרת עוברים לשלב 2).

2) סורקים את סרט הקלט משמאל לימין ובודקים אם המספר הכולל של אותיות a ו- b זוגי או אי-זוגי.

- אם אי-זוגי ← rej.
- אחרת עוברים לשלב 3).

3) סורקים את הקלט שוב משמאל לימין.

- מבצעים מחיקה לסירוגין של האות a, (כלומר מתחילים אם האות a הראשונה, אות אחת נמחק ואות אחת נשאר וכן הלאה.)
- אחר כך מבצעים מחיקה לסירוגין של האות (מתחילים אם האות b הראשונה, אות אחת נמחק ואות אחת נשאר וכן הלאה.) b.

4) חוזרים על שלבים 2) ו- 3)

- עד שהמכונה מגיעה למצב rej,
- אחרת אם לא נשאר אף אותיות a או b המכונה עוברת ל- acc.

הסיבוכיות

הזמן הריצה של שלב 1) הוא $O(n)$.

גם שלב 2) הוא $O(n)$ וגם שלב 3) הוא $O(n)$.

בכל סבב, בשלב 3) המכונה מורידה לפחות חצי האותיות a ולפחות חצי האותיות b.
לכן המכונה תבצע $1+\log_2 n$ סבבים לכל היותר.
לפיכך הזמן הריצה של M_2 יהיה

$$O(n)+2O(n)\left(1+O\left(\log_2 n\right)\right)=2O\left(n\log_2 n\right)+3O(n)=O(n\log n)\ .$$

לפיכך $A\in \text{TIME}\left(n\log n\right)$.

דוגמה 5.9

קיימת מכונת טיורינג M_3 עם שני סרטים שמכריעה את השפה A בצורה יותר יעילה, בזמן ריצה $O\left(n\right)$.
זמן הריצה זה נקרא **זמן ליניארי**.

האלגוריתם

בהתחלה על סרט 1 כתוב את הקלט.
סרט 2 ריק.

1) סורקים את סרט 1 משמאל לימין.

- אם המילה ריקה ← acc.
- אם נמצא a לצד ימין של b ← rej.

- אם תו הנקרא הראשון b $\leftarrow$ rej.
- אחרת עוברים לשלב 2).

2) סורקים את סרטים 1 ו- 2 משמאל לימין והעתיקו את כל האותיות a מסרט 1 לסרט 2 צעד צעד.

- אם התו הראשון אחרי האותיות a הוא ␣ "$\leftarrow$" rej
- אם התו הראשון אחרי האותיות a הוא b עוברים לשלב 3).

3) אם נקרא אות b בסרט 1 נמחק אות a בסרט 2.

אחר כך הראש של סרט 1 זז ימינה צעד אחד והראש של סרט 2 זז שמאלה צעד אחד וחוזרים על השלב הזה.

- אם לא נשארות אף אותיות a בסרט 2 אבל עדיין נשארות אותיות b בסרט 1 אז $\leftarrow$ rej.
- אם כאשר כל האותיות a על סרט 2 נמחקו ולא נשארות אף אותיות b על סרט 1 אז $\leftarrow$ acc.

הסיבוכיות

הזמן הריצה של שלב 1) הוא $O(n)$.

הזמן הריצה של שלב 2) ושלב 3) ביחד הוא $O(n)$.
לפיכך הזמן הריצה של M_3 יהיה

$$O(n) + O(n) = 2O\left(n\right) = O(n)\ .$$

לפיכך $A \in \text{TIME}\left(n\right)$.

ראינו דוגמה של עקרון חשוב בסיבוכיות:

משפט 5.2:

הגדרת זמן הריצה שנתנו היא תלויה במודל של מכונת הטיורינג שאיתו אנחנו עובדים.

משפט 5.3:

תהי $t : \mathbb{N} \to \mathbb{R}^+$ פונקציה $t(n)$.
אם מתקיים

$$t(n) \geq n$$

אז לכל מכונת טיורינג $O\left(t(n)\right)$ רב-סרטי קיימת מ"ט $O\left(t^2(n)\right)$ עם סרט אחד.

הוכחה:
תהי M מ"ט k סרטים שרץ בזמן $O\left(t(n)\right)$.

נבנה מ"ט S עם סרט אחד שרץ בזמן $O\left(t^2(n)\right)$.

רושמים את התוכן של ה- k סרטים והמיקום של הראש של כל אחד הסרטים של M על הסרט היחיד של S.
להלן יש דוגמה לכך של מכונת טיורינג עם 3 סרטים:

<u>המכונת טיורינג M</u>

0	1	0	1	0	␣	␣
			$\uparrow$			

a	a	a	␣	␣	␣	␣
		$\uparrow$				

a	b	b	␣	␣	␣	␣
	$\uparrow$					

<u>המכונת טיורינג S</u>

#	0	1	0	1	0	#	a	a	a	#	a	b	b
				$\uparrow$					$\uparrow$			$\uparrow$	

כדי לסמלץ צעד אחד של M במ"ט S:

1) בהתחלה, S מאתחלת את הסרט שלה על ידי לכתוב את התוכן של כל אחד של ה- k סרטים על הסרט היחיד שלה, עם תו # להפריד בין שני סרטים של M כמתואר בדוגמה למעלה.

2) כדי לסמלץ צעד אחד של M על המכונה S, המכונה S מבצעת סריקה אחת מה- # הראשון בקצה השמאלי ל- # ה- $k+1$ -ית בקצה הימין.

בסריקה זו S זוכרת את הסימנים במיקומים של ה- k ראשים של M באמצעות k תאי זכרון.

3) אחר כך S מבצעת סריקה שנייה של הסרט. בסריקה זו, לפי הפונקצית המעברים S מבצעת

- כתיבה של הסימן החדש בסרט ה-i במיקום של כל ראש של סרט ה-i,
- תזוזה של הראש של סרט ה- i

לכל $1 \leq i \leq k$.

במקרה שכל אחד של הראשים של M זז ימינה לתו רווח בקצה הימין של הסרט שלו, כלומר למשבצת ריקה שטרם לא נקרא, S מוסיפה משבצת עם תו רווח לצד שמאל של ה- # ומזיזה את כל המשבצות מקום אחד ימינה.

האורך של הסרט של S שווה לסכום של הארכים של ה- k סרטים של M.
הזמן הריצה של M הוא $t(n)$.
ז"א M משתמשת ב- $t(n)$ משבצות לכל היותר ב- $t(n)$ צעדים.
לכן בהכרח האורך של הסרט של S הוא $t(n)$ לכל היותר,
לכן סריקה אחת של S דורשת $O\left(t(n)\right)$ צעדים לכל היותר.

כדי לדמות צעד אחד של M, המכונה S מבצעת שתי סריקות.
כל סריקה לוקחת זמן $O\left(t(n)\right)$.
לכן S לוקחת זמן $O\left(t(n)\right)$ כדי לבצע צעד אחד של M.

בשלב האתחול S מבצעת $O(n)$ צעדים.

S מסמלצת כל אחד של ה- $t(n)$ צעדים של M, בזמן $O\left(t(n)\right)$.

לפיכך הזמן הכולל הנדרש של S לבצע $t(n)$ צעדים של M הוא

$$t(n) \times O\left(t(n)\right) = O\left(t^2(n)\right) \ .$$

הסימלוץ הכולו לוקח

$$O(n) + O\left(t^2(n)\right) \ .$$

אנחנו הנחנו כי $t(n) \geq n$ לכן הזמן הריצה של S הוא

$$O\left(t^2(n)\right) \ .$$

■

הגדרה 5.6: זמן הריצה של מ"ט לא דטרמיניסטית

צהי N מכונת טיורינג לא דטרמיניסטית.
הזמן הריצה של N מוגדרת להיות הפונקציה $f : \mathbb{N} \to \mathbb{N}$ כאשר $f(n)$ הוא המספר הצעדים המקסימלי אשר N מתוך כל הענפים של החישוב שלה על קלט של אורך n.

משפט 5.4:

תהי $t(n)$ פונקציה המקיימת $t(n) \geq n$. כל מ"ט $O\left(t(n)\right)$ לא דטרמיניסטית N סרט אחד, שקולה למכונת טיורינג $2^{O(t(n))}$ דטרמיניסטית סרט אחד.

הוכחה:

■

5.2 המחלקה P

הגדרה 5.7: מכונת טיורינג פולינומית

מכונת טיורינג M תיקרא **פולינומית** או **יעילה** אם קיים $c \in \mathbb{N}$ כך ש- M פועלת בסיבוכיות זמן ריצה $O\left(n^c\right)$.

הגדרה 5.8: המחלקה P

המחלקה P היא אוסף השפות שקיימת מכונת טיורינג פולינומית M המקבלת אותן. כלומר:

$$P = \bigcup_k \text{TIME}\left(n^k\right) \ .$$

הגדרה 5.9: המחלקה $POLY$

המחלקה $POLY$ היא אוסף הפונקציות שקיימת מכונת טיורינג פולינומית M המקבלת אותן.

דוגמה 5.10 גרף מכוון

נתון גרף G עם קדקודים s ו- t.
נגדיר בעיה $PATH$ להיות הבעיה לקבוע האם קיים מסלול בין s לבין t.

$$PATH = \left\{ \langle G,s,t \rangle \;\middle|\; \text{גרף מכוון המכיל מסלול מכוון מ- } s \text{ ל- } t \;\; G \right\}$$

הוכיחו כי

$$PATH \in P \ .$$

פתרון:

נבנה אלגוריתם M פולינומי בשביל בעיה $PATH$ כמפורט להלן.

עבור הקלט $\langle G,s,t \rangle$ כאשר G הגרף המכוון עם קדקודים s ו- t:

1) נסמן את הקדקוד s.

2) חוזרים על שלב 3) עד שלא נשארים קדקודים לא מסומנים בקצוות של אף צלע של G:

3) סורקים את כל הצלעות של G.

- אם נמצע צלע (a,b) מקדקוד מסומן לקדקוד לא מסומן, נסמן את הקדקוד b.

4) אם קדקוד t מסומן, $\leftarrow$ acc.
אחרת $\leftarrow$ rej.

שלב 1) מבוצע פעם אחת בלבד, ושלב 4) מבוצע פעם אחת בלבד.

אם ל- G יש n קדקודים אז שלב 3) מבוצע n פעמים לכל היותר.

לכן מספר הצעדים המבוצעים הוא $n+1+1$ לכל היותר.

לכן $M = O(n)$.

דוגמה 5.11

תהי $RELPRIME$ הבעיה לבדור אם שני שלמים x,y זרים.

$$RELPRIME = \left\{ \{x,y\} \in \mathbb{N} \;\middle|\; \gcd(x,y) = 1 \right\}$$

הוכיחו כי $RELPRIME \in P$.

פתרון:

נבנה אלגוריתם שמתבסס על האלגוריתם אוקלידס למצוא את ה- gcd. נסמן את האלגוריתם שמבצע האלגוריתם אוקלידס ב- E.
הקלט הוא הצמד שלמים $\{x,y\}$ בבסיס בינארי:

1) משימים $x \leftarrow x \mod y$

2) מחליפים x ו- y.

3) מחזירים x.

4) חוזרים על השלבים 1) - 3) עד שנקבל $y = 0$.

```python
x = 625;
y = 75;

a=x;
b=y;

while y!=0:
    x = x % y

    y1 = x
    x = y
    y = y1

print("gcd(%d,%d) = %d" % (a,b,x))
```

האלגוריתם R פותר את הבעיה $RELPRIME$ על ידי שימוש של E כתת-אלגוריתם:
הקלט של R הוא הצמד שלמים $\{x,y\}$ בבסיס בינארי:

1) מריצים E על $\{x,y\}$.

2) אם הערך חזרה של E הוא 1 אז $\leftarrow$ acc
אחרת $\leftarrow$ rej.

אם E רץ בזמן פולינומי אז גם R ירוץ בזמן פולינומי אז מספיק לבדוק את הסיבוכיות זמן הריצה של E בלבד.

ללא הגבלת כלליות נניח ש- $x > y$. (המקרה של $x = y$ לא מעניין אותנו כי התשובה $\gcd(x,x) = x$ טריוויאלית.)
משפט החילוק של אוקלידס אומר שלכל x,y שלמים קיימים שלמים q,r עבורם

$$x = qy + r \ ,$$

כאשר $y < x$ ו- $r < y$. השלם $q = \left\lfloor \frac{x}{y} \right\rfloor$ נקרא המנה ו- $r = x \mod y$ נקרא השארית.

אחרי ביצוע של שלב 1), שבו אנחנו משימים $x = x \mod y$, אז יהיה $x < y$.

אחרי ביצוע של שלב 2), שבו מחליפים x ו- y, אז יהיה $x > y$.

כעת יש שתי אפשרויות:

- אם אחרי שלב 2) יוצא כי $\frac{x}{2} \geq y$ אז $\frac{x}{2} \geq y > x \mod y$.

  לכן $\frac{x}{2} \geq x \mod y$.

  מכאן אנחנו רואים כי x יקטן לפחות בחצי.

- מצד שני נניח שאחרי שלב 2) יוצא כי $\frac{x}{2} < y$.

  אז $\frac{x}{2} \geq y > x \mod y$.

  לכן $\frac{x}{2} \geq x \mod y$.

מכיוון ש- $x = qy + (x \mod y)$, וגם $x < 2y$ אז בהכרח $q < 2$ ולכן $x = y + (x \mod y)$ ולכן $x - y = x \mod y$.

לפיכך
$$x \mod y = x - y < \frac{x}{2} \ .$$
לכן גם במקרה זה x יקטן לפחות בחצי.

x ו- y מתחלפים כל פעם ששלב 2) מתבצע, לכן אחרי 2 סבבים של האלגוריתם, הערך של x יקטן לפחות בחצי וגם הערך של y יקטן בחצי.

לפי זה מספר הפעמים המקסימלי ששלבים 1) ו- 2) מתבצעים הוא המינימום בין מספר פעמים שאפשר לחלק x ב- 2 לבין מספר פעמים שאפשר לחלק y בחצי. ז"א המימימום בין $2\log_2 x$ לבין $2\log_2 y$. כלומר, מספר הסבבים המקסימלי של האלגוריתם הוא
$$\min\left(2\log_2 x, 2\log_2 y\right) \ .$$
מכיוון ש- x ו- y נתונים בבסיב בינארי אז
$$\log_2 x = n_x - 1 \ , \qquad \log_2 y = n_y - 1$$
כאשר n_x אורך המספר x בבסיס בינארי ו- n_y אורך המספר y.

לכן מספר הצעדים המקסימלי של האלגוריתם E הוא
$$\min(n_x - 1, n_y - 1) \ .$$
זמן הריצה מוגדר להיות המספר הצעדים המקסימלי של האלגוריתם, לכן
$$E = O(n)$$
כאשר n אורך הקלט.

5.3 המושג של אלגוריתם אימות

הגדרה 5.10: מעגל המילטוני

כלומר מעגל המילטוני (Hamiltonian cycle) בגרף מכוון $G = (V, E)$ הוא מעגל אשר עובר כל קדקוד בדיוק פעם אחת.

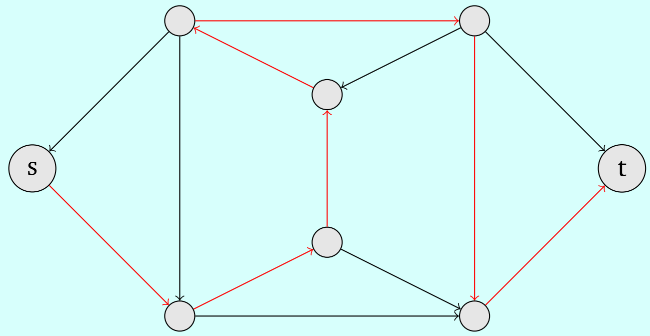
גרף המכיל מעגל המילטוני מכונה **גרף המילטוני** (Hamiltonian). אחרת, הגרף מכונה **לא המילטוני** (non-Hamiltonian).

הגדרה 5.11: הבעיית מעגל המילטוני

הבעיית המעגל ההמילטוני the hamiltonian cycle problem היא הבעיה:

" האם לגרף G יש מעגל המילטוני?"

ניתן להגדיר כשפה פורמלית:
$$HAMPATH = \left\{\langle G, s, t\rangle \ \middle|\ \text{G הוא גרף מכוון המכיל מעגל המילטוני מ- s ל- t.}\right\}$$
התרשים למטה מראה דוגמה של מעגל המילטוני בגרף מכוון.



נשאל שאלה: מהו הסיבוכיות זמן הריצה של הבעיה $HAMPATH$? כעת זה לא ידוע האם הבעיה הזו ניתנת לפתור בזמן פולינומי, כלומר האם $HAMPATH \in P$.

בכל זאת, נניח שחבר מספר לך כי גרף נתון G הוא המילטוני. אפשר לאמת את הטענה ע"י כך שנמנה את הקדקודים על פי סדרם לאורך המעגל ההמילטוני, ונבדוק אם קדקודיו הם תמורה של קדקודי V של G, ואם כל אחת מן הקשתות העוקבות לאורך המעגל אכן קיימת בגרף. בהמשך אנחנו נוכיח כי האלגוריתם האימות הזה ניתן לממש כך שירוץ בזמן $O\left(n^2\right)$, כאשר n הוא אורך הקידוד של G. ז"א הוכחה שגרף מכיל מעגל המילטוני ניתנת לאימות בזמן פולינומיאלי.

אף על פי שלא בהכרח יש לנו אלגוריתם שרץ בזמן פולינומיאלי שבדוק האם גרף מכיל מעגל המילטוני, בכל זאת במקרה שמעגל המילטוני היה נגלה, קל למדי לאמת את המעגל על ידי האלגוריתם לעיל.

הדוגמה הזאת היא מקרה שבה הביעה של לאמת פתרון לבעיה נתונה קלה יותר מהבעיה של למצוא פתרון לבעיה זו.

הגדרה 5.12: מספר פריק

משפר שלם x נקרא **פריק** (composite) אם קיימים שלמים $p>1, q>1$ כך ש-

$$x = pq\ .$$

במילים אחרות, x פריק אם ורק אם x לא ראשוני.

הגדרה 5.13: הבעיית $COMPOSITES$

הבעיית $COMPOSITES$ היא הבעיה:

" האם השלם x פריק? "

ניתן להגדיר כשפה פורמלית:

$$COMPOSITES = \left\{x \;\middle|\; \text{קיימים שלמים } p,q>1 \text{ כך ש- } x=pq \right\}$$

קל למדי לאמת כי שלם x פריק: בהינתן הפתרון ש- $x \mid p$, אלגוריתם אימות הוא אלגוריתם אשר מחלק x ב- p ובודק שאין שארית והמנה המתקבל הוא שלם וגדול מ- 1. כעת ניתן הגדרה פורמלית של אלגוריתם אימות

הגדרה 5.14: אלגוריתם אימות

אלגוריתם אימות של שפה A הוא אלגוריתם V כך ש-:

$$A = \left\{w \;\middle|\; V \text{ מקבל } \langle w,c\rangle \text{ על פי } c\right\}$$

במילים, **אלגוריתם אימות** הוא אלרגוריתם V אשר מאמת כי הקלט w שייך לשפה A על פי התנאי c,

שנקרא **אישור** (certification).

אנחנו מגדירים את זמן הריצה של V על פי האורך של w. לכן **אלגוריתם אימות זמן-פולינומיאלי** רץ בזמן פולינומיאלי $O\left(n^k\right)$ כאשר n האורך של w.

5.4 המחלקה NP

הגדרה 5.15: מחלקת הסיבוכיות NP

היא מחלקת השפות שניתן לאמתן באמצעות אלגוריתם זמן-פולינומיאלי.

השם NP מנובע מהכנוי אנגלי nondeterministic polynomial time. הכנוי מנובע מכך שרוב הבעיות של המחלקה NP ניתנות לאימות על ידי מכונות טיורינג אי-דטרמיניסטיות, למשל הבעייה $HAMPATH$ כמוסבר בדוגמה הבאה.

דוגמה 5.12

הוכיחו כי

$$HAMPATH \in NP \ .$$

פתרון:

כזכור הזמן הריצה של מ"א לא דטרמיניסטית מוגדר לפי הזמן הריצה של הענף הכי האורך (הגדרה 5.6 שלעיל). נבנה מ"ט אי-דטרמיניסטית N_1 אשר מכריעה את $HAMPATH$ -בזמן-פולינומיאלי אי דטרמיניסטי.

יהיו m מספר הקדקודים של G ו- n מספר הקשתות של G:

$$m = |V| \ , \qquad n = |E| \ .$$

$N_1 =$ על הקלט $\langle G, s, t\rangle$ כאשר G גרף מכוון ו- s, t קדקודים של G:

1) רושמים רשימה של m מספרים, $p_1, p_2, \ldots, p_m$ כאשר m מספר הקדקודים ב- G.
כל מספר נבחר בצורה אי-דטרמיניסטית מ- 1 עד m.

2) בודקים אם יש חזרות ברשימה זו.
אם יש חזרות $\leftarrow$ rej.

3) בודקים אם $s = p_1$ ו- $t = p_m$.
אם לא $\leftarrow$ rej.

4) לכל $1 \leq i \leq m-1$ בודקים אם הקשת (p_i, p_{i+1}) שייך לקבוצת הקשתות E של G.

- אם אף קשת לא שייכת ל- E $\leftarrow$ rej.
- אם כל הקשתות שייכות ל- E $\leftarrow$ acc.

כעת נבדוק עת הסיבוכיות של האלגוריתם הזה.

- שלב 1) דורש m צעדים ולכן מתבצע בזמן פולינומיאלי.
- שלב 2) דורש m צעדים לכל היותר, ולכן מתבצע בזמן פולינומיאלי.
- שלב 3) דורש m צעדים לכל היותר, ולכן מתבצע בזמן פולינומיאלי.
- עבור שלב 4) לכל קשת (p_i, p_{i+1}), המ"ט N_1 בודקת אם יש קשת תואמת בקבוצת הקשתות E של G.
  לכן ידרשו n צעדים לכל היותר לכל i.
  לכן שלב 4) דורש $n(m-1)$ צעדים לכל היותר בסה"כ.

לכן הסיבוכיות זמן הריצה של N_1 היא

$$O(m) + O(m) + O\left(n(m-1)\right) = O\left(n(m-1)\right)$$

משפט 5.5: $A \in NP$ אם"ם A ניתנת לאימות ע"י N_{TM}

שפה A כלשהי שייכת למחלקה NP אם ורק אם A ניתנת לאימות על ידי מכונות טיורינג אי-דטרמיניסטית זמן-פולינומיאלית.

רעיון ההוכחה:
הרעיון הוא להראות כיצד להמיר אלגוריתם אימות זמן-פולינומיאלי למכונת טיורינג אי-דטרמיניסטית זמן-פולינומיאלי ולהפך.
במילים פשוטות אלגוריתם אימות זמן-פולינומיאלי V שקול חישובי למ"ט אי-דטרמיניסטית זמן-פולינומיאלי N_{TM}:
N מדמה V על ידי ניחוש של האישור c.
V מדמה N_{TM} באמצעות המסלול של N_{TM} אשר מקבל את השפה בתור האישור.

הוכחה: ראשית נוכיח שאם $A \in NP$ אז A ניתנת לאימות ע"י N_{TM}.

$A \in NP$ לכן קיים אלגוריתם אימות זמן פולינומיאלי V של A.

נבנה מ"ט N שרץ בזמן $O\left(n^k\right)$ עבור k כלשהו.

$N =$ " על הקלט w של אורך n:

1) בצורה אי-דטרמיניסטית בוחרים מחרוזת c באורך n^k לכל היותר.

נשים לב שחייב להיות חסם עליון n^k על האורך של c עבור k כלשהו, בגלל ההנחה שלנו ש- V עצמו הוא אלגוריתם זמן-פולינומיאלי.

2) מריצים V על $\langle w, c\rangle$:

3)
- אם V מקבל אז $N \leftarrow$ acc.
- אחרת $N \leftarrow$ rej."

נוכיח שאם A ניתנת לאימות ע"י מ"ט אי-דטרמיניסטית זמן-פולינומיאלית אז $A \in NP$.

נניח ש- A ניתנת לאימות ע"י מ"ט אי-דטרמיניסטית זמן-פולינומיאלית.
נבנה אלגוריתם אימות זמן פולינומיאלי כמפורט להלן:

בהינתן קלט w ומכונת טיורינג אי-דטרמיניסטית N אשר מאמתת כי $w \in A$ בזמן-פולינומיאלי. נסמן ב- n את האורך של הקלט w.

ראשית הוכחנו שכל מכונת טיורינג אי-דטרמיניסטית שקולה חישובית למכונת טיורינג דטרמיניסטית 3-סרטים: סרט הכספת, סרט העבודה וסרט הבחירות.
על סרט הבחירות המכונת טיורינג דטרמיניסטית רושמת כל הסדרות של החביורת בסדר לקסיקוגרפי.
בנוסף מובטח לנו כי האורך של סרט הבחירות חסום מלמעלה על ידי n^k עבור k כלשהו מסיבה לכך שהנחנו ש- N רצה בזמן פולינומיאלי.

יהי c אחת הסדרות של הבחירות. שוב, אורך הסרט הבחיורת חסום מלעלה על יד n^k לכן גם c חסום מלמעלה על ידי n^k.

$V =$ " על הקלט $\langle w, c\rangle$ כאשר w ו- c מחרוזות:

1) מריצים N על הקלט w.

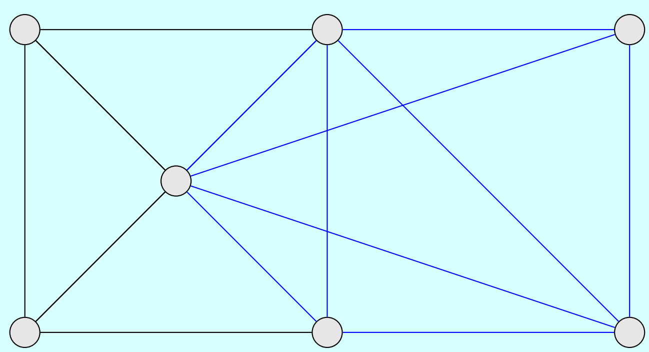
V מתייחס לכל תו של c כתיאור של בחירה האי-דטרמיניסטית לבצע בכל צעד.

2)
- אם המסלול הזה של החישוב של $N \leftarrow \text{acc}$ אז V מקבל את $\langle w, c\rangle$.
- אם המסלול הזה של החישוב של $N \leftarrow \text{rej}$ אז V דוחה את $\langle w, c\rangle$."

■

הגדרה 5.16: k-קליקה

קליקה בגרף בלתי-מכוון הוא תת-גרף שבו כל זוג קדקודים קשורים על ידי קשת.
k- קליקה היא קליקה שבו יש k קדקודים.
התרשים למטה מראה דוגמה של 5- קליקה.



דוגמה 5.13 בעיית הקלקיה

בעיית הליקה היא הבעיה לקבוע האם גרף מכיל k- קלקה עבור k מסוים:

$$CLIQUE = \left\{ \langle G,k\rangle \;\middle|\; \text{-}k \text{ גרף בלתי-מכוון שמכיל } G \right\}$$

הוכיחו כי $CLIQUE \in NP$.

הוכחה: עבור הגרף $G(V,E)$ יהי $m = |V|$ מססר הקדקודים ו- $n = |E|$ מספר הקשתות.

האלגוריתם הבא הוא מאמת V של $CLUQUE$:

$V =$ " על הקלט $\langle\langle G,k\rangle, c\rangle$:

1) בודקים האם c קבוצה של k קדקודים שבגרף G.

- אם לא ← rej.
- אם כן ממשיכים לשלב 2).

2) בודקים אם G מכיל את כל הקשתות אשר מקשרות בין כל הקדקודים ב- c.

- אם לא ← rej.
- אם כן ← acc."

- שלב 1) דורש k צעדים לכל היותר.
- שלב 2) ב- k- קליקה יש $\frac{1}{2}k(k-1)=\binom{k}{2}$ קשתות בסה"כ. לכן בשלב 2) האלגוריתם צריך לבדוק אם כל אחד של הקשתות מוכלת ב- G. ז"א לכל קשת של c האלגוריתם סורק את קבוצת הקשתות E אחת אחת ובודק אם יש קשת תואמת. לכן שלב 2) דורש $n\times\frac{1}{2}k(k-1)$ לכל היותר.

לפיכך הסיבוכיות זמן הריצה היא

$$O(k)+O\left(nk(k-1)\right)=O\left(n^3\right)\ .$$

כלומר האלגוריתם המאמתת רץ בזמן פולינומיאלי לכן $CLIQUE\in NP$. ■

הגדרה 5.17: בעיית סכום התת קבוצה $SUBSET-SUM$

נתונה קבוצת שלמים

$$S=\{x_1,\ldots,x_k\}$$

ושלם t. בבעיית סכום התת-קבוצה, $SUBSET-SUM$, נתונים קבוצה נוצרת סופית $S\subset\mathbb{N}$ של שלמים וערך מטרה $t\in\mathbb{N}$ שלם. אנחנו שואלים אם קיימת תת-קבוצה $S\subseteq Y$ שאיבריה מסתכמים לערך t.
נגדיר את הבעיה כשפה:

$$SUBSET-SUM=\left\{\langle S,t\rangle\ \middle|\ \text{קיימת תת-קבוצה } Y\subseteq S \text{ כך שמתקיים } \sum_{y\in Y}y=t\right\}$$

לדוגמה, אם

$$S=\{1,16,64,256,1040,1041,1093,1284\}$$

ו- $t=3754$ אזי התת-קבוצה

$$Y=\{1,16,64,256,1040,1093,1284\}$$

היא פתרון.

דוגמה 5.14

הוכיחו:

$$SUBSET-SUM\in NP\ .$$

הוכחה: אנחנו נבנה מ"ט זמן-פולינומיאלי M אשר מאמת פתרון כלשהו לבעיית סכום התת-קבוצה.

M מ"ט דטרמיניסטית 3 סרטים:

- על סרט S רשומים האיברים של הקבוצה S בבסיס אונרי עם תו "#" להפריד בין איברים.
- על סרט c רשומים האיברים של הקבוצה c בבסיס אונרי תו "#" להפריד בין איברים.

- על סרט t רשום המספר t בבסיס אונרי.

לדוגמה, אם

$$S=\{1,2,3,4\}\ ,\qquad c=\{2,3,4\}\ ,\quad t=9\ .$$

אז התכנים של הסרטים יהיו

S	␣	1	#	1	1	#	1	1	1	#	1	1	1	1	␣
		↑													
c	␣	1	1	#	1	1	1	#	1	1	1	1	␣	␣	␣
		↑													
t	␣	1	1	1	1	1	1	1	1	1	␣	␣	␣	␣	␣
		↑													

האלגוריתם של M מתואר להלן.

$M=$ " על הקלט $\langle\langle S,t\rangle\ ,c\rangle$:
בשלב הראשון אנחנו בודקים אם S מכילה את כל השלמים שב- c.

שלב 1) הראש S והראש c זזים ימינה צעד אחד במקביל.

- אם ראש c קורא ␣ וראש S קורא $1 \leftarrow$ rej.
- אם ראש c קורא 1 וראש S קורא ␣ $\leftarrow$ rej.
- אם ראש-c קורא # וראש- S קורא 1,
  או אם ראש-c קורא 1 וראש- S קורא #:
  - ראש c חוזר לתחילת המחרוזת
  - ראש S זז למשבצת הבאה אחרי ה- #.
- אם ראש-c קורא # וראש- S קורא # אז הראש S והראש c זזים שניהם משבצת אחת ימינה וממשיכים לשלב 2).

שלב 2)
- אם ראש-c קורא ␣ וראש- S קורא ␣, מחזירים ראש S וראש c לתחילת המחרוזת ועוברים לשלב 3).
- אחרת חוזרים על שלב 1)

בשלבים 3) ו- 4) אנחנו בודקים אם הסכום של האיברים של c שווה ל- t.

שלב 3) בשלב זה אנחנו מחברים את המספרים על סרט c:

עבור כל תו # בסרט c, כותבים עליו 1 ומוירדים תו 1 תואם מקצה הימין של הסרט, ומחזירים את הראש לתחילת הסרט c.

שלב 4) בשלב זה אנחנו בודקים שהמספרים על הסרים c ו- t שווים.
הראשים של c ושל t זזים ימינה צעד צעד במקביל.

- אם ראש c רורא ␣ וראש t קורא $1 \leftarrow$ rej.
- אם ראש c רורא 1 וראש t קורא ␣ $\leftarrow$ rej.
- אם ראש c רורא ␣ וראש t קורא ␣ $\leftarrow$ acc."

כעת נבדוק את הסיבוכיות של האלגוריתם. נסמן ב- n האורך המקסימלי מבין הסרטים S,c,t.

- שלבים 1) ו- 2) דורשים n^2 צעדים לכל היותר.
- שלב 3) דורש $2n$ שלבים לכל היותר.

- שלב 4) דורש n שלבים לכל היותר.

לכן

$$M=O\left(n^2\right)+O\left(2n\right)+O\left(n\right)=O\left(n^2\right)$$

לכן $SUBSET-EQ\in NP$. ■

ההוכחה חלופית: נבנה מ"ט אי-דטרמיניסטית N כמפורט להלן:

" $N=$ על הקלט $\langle S,t\rangle$:

1) נבחר בצורה אי-דטרמיניסטית תת-קבוצה c של השלמים S.

2) בודקים אם הסכום של האיברים של c שווה ל- t:

- אם $\sum\limits_{y\in c} y=t$ אז $N\leftarrow$ acc.
- אם $\sum\limits_{y\in c} y=t$ אז $N\leftarrow$ rej. "

5.5 NP-שלמות

עד כה אנחנו ראינו את הגדרות של המחלקות P ו- NP:

$P=$ מחלקת השפות שכריעות בזמן פולינומיאלי ,

$NP=$ מחלקת השפות שניתנות לאימות בזמן פולינומיאלי .

- ראינו שתי דוגמאות של שפות, $HAMPATH$ ו- $CLIQUE$ ששייכות ל- NP אך לא ידוע האם הן שייכות גם ל- P.
- שאלה מרכזית במדעי המחשב היא שאם $P=NP$, כלומר:

  האם כל שפה ששייכת ל- NP גם שייכת ל- P,

  וכל שפה ששייכת ל- P גם שייכת ל- NP? ננסח את השאלה כביטוי פורמלי. האם מתקיים

$$L\in NP\Leftrightarrow L\in P\ .$$

5.6 הבעיה של ספיקות

5.6.1 תזכורת: משתנים בוליאניים

סימן	פעולה
$\wedge$	AND
$\vee$	OR
$\neg$	NOT
$\oplus$	XOR

$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$\neg 0 = 1$	$\bar{0} = 1$
$0 \wedge 1 = 0$	$0 \vee 1 = 1$	$\neg 1 = 0$	$\bar{1} = 0$
$1 \wedge 0 = 0$	$1 \vee 0 = 1$		
$1 \wedge 1 = 1$	$1 \vee 1 = 1$		

הגדרה 5.18: שוויון

יהיו p, q משתנים בוליאניים.

$p \to q = 0$ אם $p = 1$ ו- $q = 0$.

אחרת $p \to q = 1$.

הגדרה 5.19: גרירה

יהיו
p, q משתנים בוליאניים.

$p \leftrightarrow q = 1$ אם $p = q = 0$ או אם $p = q = 1$, כלומר אם ל- p ו- q אותם ערכים.

אחרת
$$p \leftrightarrow q = 0 \ .$$

$0 \oplus 0 = 0$	$0 \leftrightarrow 0 = 1$	$0 \to 0 = 1$
$0 \oplus 1 = 0$	$0 \leftrightarrow 1 = 0$	$0 \to 1 = 1$
$1 \oplus 0 = 1$	$1 \leftrightarrow 0 = 0$	$1 \to 0 = 0$
$1 \oplus 1 = 0$	$1 \leftrightarrow 1 = 1$	$1 \to 1 = 1$

5.6.2 הגדרה של נוסחה ספיקה

נוסחה בווליניאית היא ביטוי במונחי משתנים בווליאניים ופעולות בוליאניות. למשל
$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z}) \ .$$

הגדרה 5.20: נוסחה בוליאנית ספיקה

אומרים כי נוסחה בוליאנית ϕ ספיקה אם קיימת השמת ערכי אמת הגורמת לכך שהערך שמייצגת הנוסחה יהיה 1.

דוגמה 5.15

הנוסחה
$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$$

ספיקה מסיבה לכך שקיימת השמה

$$x=0,\quad y=1,\quad z=0$$

עבורה

$$\phi=1\ .$$

אומרים כי ההשמה $x=0,y=1,z=0$ מספקת את ϕ.

דוגמה 5.16

נתונה הנוסחה

$$\phi=((x_1\leftarrow x_2)\vee\neg((\neg x_1\leftrightarrow x_3)\vee x_4))\wedge\neg x_2$$

מצאו השמה מספקת ל- ϕ.

פתרון:

ההשמה

$$\langle x_1=0,x_2=0,x_3=1,x_4=1\rangle$$

היא השמה מספקת. שכן:

$$\begin{aligned}\phi=&((0\leftrightarrow 0)\vee\neg((\neg 0\leftrightarrow 1)\wedge 1))\wedge\neg 0\\ =&(1\vee\neg(1\wedge 1))\wedge 1\\ =&(1\vee 0)\wedge 1\\ =&1\ .\end{aligned}$$

ולכן נוסחה ϕ זו שייכת ל- SAT.

הגדרה 5.21: הבעיית הספיקות SAT

הבעיית הספיקות שואלת אם נוסחה בוליאנית נתונה היא ספיקה. במונחי שפות פורמלית:

$$SAT=\left\{\langle\phi\rangle\ \middle|\ \phi\ \text{היא נוסחה בוליאנית ספיקה}\right\}$$

האלגוריתם הנאיבי הקובע אם נוסחה בוליאנית כלשהי היא ספיקה אינו רץ בזמן פולינומיאלי. עבור נוסחה ϕ המכילה n משתנים קיימות 2^n השמות אפשריות. אם האורך של $\langle\phi\rangle$ פולינומיאלי ב- n, אזי בדיקה כל ההשמות דורשות זמן על-פולינומיאלי. כפי שמוכיח המשפט שלהלן, לא קיים ככל הנראה אלגוריתם זמן-פולינומיאלי עבור בעיה זו.

5.7 הגדרה של רדוקציה

הגדרה 5.22: פונקציה הניתנת לחישוב

פונקציה $f:\Sigma^*\to\Sigma^*$ ניתנת לחישוב אם קיימת מ"ט M, עבורה על הקלט w M עוצרת עם $f(w)$ על הסרט שלה.

הגדרה 5.23: פונקציה שניתנת לרדוקציה

השפה A ניתנת לרדוקציה לשפה B, נסמן $A\leq_m B$, אם קיימת פונקציה שנתנת לחישוב $f:\Sigma^*\to\Sigma^*$ כך שלכל

$$w \in A \Leftrightarrow f(w) \in B \ .$$

הפונרציה f נקראת הרדוקציה של A ל- B.

5.8 הגדרה של רדוקציה זמן-פולינומיאלית

הגדרה 5.24: פונקיצה הניתנת לחישוב זמן-פולינומיאלי

פונקציה $f:\Sigma^* \to \Sigma^*$ ניתנת לחישוב זמן-פולינומיאלי אם קיימת מ"ט זמן-פולינומיאלית M, עבורה על הקלט w, M עוצרת עם $f(w)$ על הסרט שלה.

הגדרה 5.25: פונקציה שניתנת לרדוקציה זמן-פולינומיאלית

השפה A ניתנת לרדוקציה זמן-פולינומיאלית לשפה B, נסמן $A \leq_P B$, אם קיימת פונקציה שנתנת לחישוב זמן-פולינומיאלית $f:\Sigma^* \to \Sigma^*$ כך שלכל

$$w \in A \Leftrightarrow f(w) \in B \ .$$

הפונרציה f נקראת הרדוקציה זמן-פולינומיאלית של A ל- B.

משפט 5.6: אם $A \leq_P B$ ו- $B \in P$ אז $A \in P$

אם $A \leq_P B$ ו- $B \in P$ אז $A \in P$.

הוכחה: $B \in P$ לכן קיימת מ"ט M_B זמן-פולינומיאלית שמכריעה את B.
$A \leq_P B$ לכן קיימת רדוקציה זמן פולינומיאלית f מ-A ל- B.

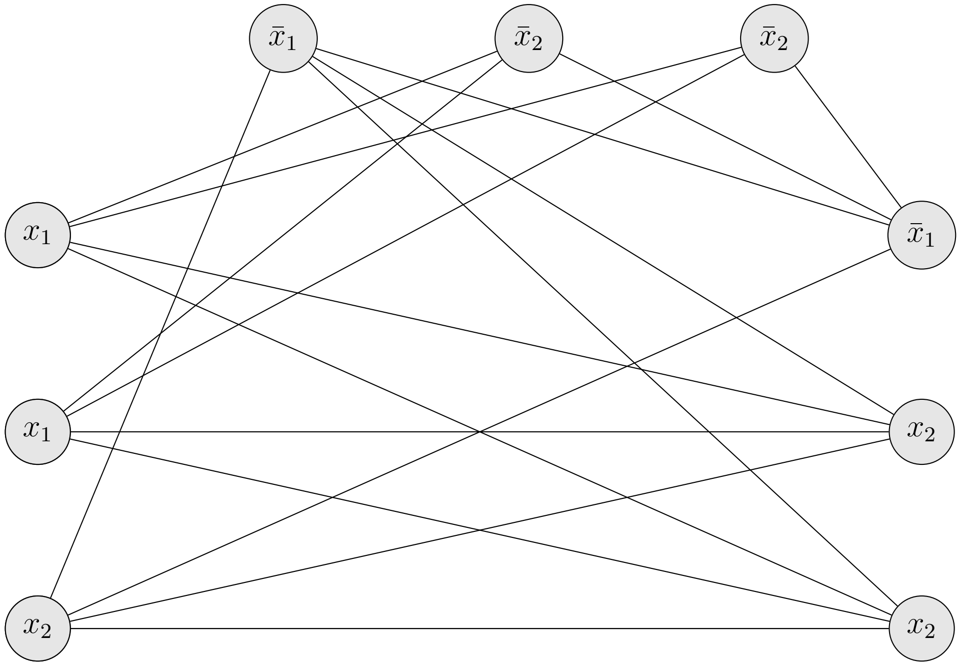
נבנה מ"ט M_A שמכריעה את A:

$M_A =$ " על הקלט w:

1) מחשבים את $f(w)$.
2) מריצים M_B על הקלט $f(w)$.
3) מחזירים את הפלט של M_B.

מכיוון ש- f רדוקציה של B ל-A אז $f(w) \in B$ אם ורק אם $w \in A$.
לכן מסיבה לכך שקיבלנו את התשובה ש- $f(w) \in B$ אז קיבלנו את התשובה כי $w \in A$.
הוכחנו כי M_A מכריעה את A. כעת נוכיח כי $M_A \in P$.
f רדוקציה זמן פולינומיאלי $\Leftarrow$ שלב 1) מתבצע בזמן פולינומיאלי.
M_B מ"ט זמן פולינומיאלית ו- f ניתנת לחישוב זמן-פולינומיאלית $\Leftarrow$ שלב 2) מתבצע בזמן פולינומיאלי (מכיוון שהרכבה של שני פולינומים היא פולינום).

■


$\bar{x}_1$
$\bar{x}_2$
$\bar{x}_2$
x_1
x_1
x_2
$\bar{x}_1$
x_2
x_2