האוניברסיטה
הפתוחה

מבוא לתורת החישוביות והסיבוכיות

מדריך למידה לספר הקורס

Michael Sipser

Introduction to the Theory of Computation

3rd ed., Cengage Learning, 2013

האוניברסיטה הפתוחה

מבוא לתורת החישוביות והסיבוכיות

מדריך למידה

לספר הקורס :

Michael Sipser
Introduction to the Theory of Computation
3rd ed., Cengage Learning, 2013

20585
מהדורה פנימית
לא להפצה ולא למכירה
מק"ט 20585-5059

אחריות אקדמית : פרופ׳ זאב נוטוב, ד״ר תמיר טסה

כתבו : ד״ר תמיר טסה, פרופ׳ זאב נוטוב

אסיסטנט : דניאל רייכמן

ייעוץ : ד״ר אלעזר בירנבוים

עורכת: חוה ניומן

עדכון 2013 : ד״ר אלעזר בירנבוים

הדפסה דיגיטלית – דצמבר 2013

בית ההוצאה לאור של האוניברסיטה הפתוחה, הקריה ע״ש דורותי דה רוטשילד, דרך האוניברסיטה 1, ת״ד 808, רעננה 4353701.
The Open University of Israel, The Dorothy de Rothschild Campus, 1 University Road, P.O.Box 808, Raanana 4353701.
Printed in Israel.

תוכן העניינים

מבוא 1

1. תזת צ'רץ'-טיורינג (פרק 3 בספר הלימוד) 5

1.1 מכונות טיורינג 5

1.2 גרסאות שונות של מכונות טיורינג 11

1.3 הגדרת מושג האלגוריתם 18

פתרון התרגילים 21

2. כריעות (פרק 4 בספר הלימוד) 25

2.1 שפות כריעות 25

2.2 אי כריעות 29

פתרון התרגילים 34

3. רדוקציות (פרק 5 בספר הלימוד) 38

3.1 בעיות לא כריעות מתורת השפות 38

3.2 בעיה בלתי כריעה פשוטה 45

3.3 רדוקציות מיפוי 48

פתרון התרגילים 50

4. סיבוכיות זמן (פרק 7 בספר הלימוד) 57

4.1 מדידת סיבוכיות 57

4.2 המחלקה P 61

4.3 המחלקה NP 66

4.4 NP-שלמות 70

4.5 בעיות NP-שלמות נוספות 77

פתרון התרגילים 87

5. סיבוכיות מקום (פרק 8 בספר הלימוד) 97
- 5.1 משפט Savitch 98
- 5.2 המחלקה PSPACE 98
- 5.3 PSPACE-שלמות 99
- 5.4 המחלקות L ו-NL 103
- 5.5 NL-שלמות 105
- 5.6 NL שווה ל-coNL 108
- פתרון התרגילים 109

6. משפטי היררכיה (סעיף 9.1 בספר הלימוד) 112
- פתרון התרגילים 120

7. נושאים מתקדמים בתורת הסיבוכיות (פרק 10 בספר הלימוד) 122
- 7.1 אלגוריתמי קירוב 122
- 7.2 אלגוריתמים הסתברותיים 130
- פתרון התרגילים 142

8. נספח – נושאים מתקדמים בתורת החישוביות (פרק 6 בספר הלימוד) 146
- 8.1 משפט הרקורסיה 146
- 8.2 כריעות של תורות לוגיות 150
- פתרון התרגילים 159

מבוא

מדריך למידה זה אמור ללוות את החלקים השני והשלישי בספר

Introduction to the Theory of Computation

מאת Michael Sipser, הוצאת Cengage Learning, מהדורה שלישית, 2013.

החלק הראשון של הספר (אוטומטים ושפות – Automata and Languages) עוסק בחומר אותו כבר למדתם בקורס **אוטומטים ושפות פורמליות**. שני החלקים האחרים בספר

- **תורת החישוביות – Computability Theory** (חלק 2, פרקים 3-6)
- **תורת הסיבוכיות – Complexity Theory** (חלק 3, פרקים 7-10)

מהווים את חומר הלימוד של קורס זה.

באופן כללי, הקורס עוסק בשאלה הבאה: ***מהן יכולות ומגבלות החישוב של מחשבים?*** התשובה לשאלה זו תלויה במידה רבה בהגדרה של המושג "מחשב", כלומר במודל החישובי אותו אנו בוחרים לנתח. בקורס אוטומטים ושפות פורמליות נתקלנו בשני **מודלים חישוביים** יסודיים ובשתי משפחות של **שפות פורמליות** המתאימות למודלים אלו.

המודל החישובי הראשון והבסיסי ביותר הוא זה של **אוטומט סופי** (או ליתר דיוק, אוטומט סופי דטרמיניסטי – Deterministic Finite Automaton ובקיצור DFA). אוטומט סופי הוא מכונה מופשטת המקבלת כקלט מחרוזת סופית מעל אלפבית נתון Σ, ומעבדת את הקלט תו אחר תו. אוטומט כזה יכול להימצא באחד מבין מספר סופי של **מצבים**. אחד מן המצבים הללו מוגדר כ**מצב ההתחלתי**, והוא המצב שבו מצוי האוטומט לפני קריאת התו הראשון בקלט. עם כל תו שנקרא, האוטומט מחליט לאיזה מצב לעבור, על סמך התו שנקרא ועל סמך המצב שבו הוא היה עם קריאת התו. חלק מן המצבים מוגדרים כ**מצבים מקבלים**. אם בתום עיבוד הקלט האוטומט נמצא במצב מקבל, הוא מקבל את הקלט; אחרת – הוא דוחה אותו. אוסף כל המחרוזות ב-Σ^* המתקבלות על ידי האוטומט נקרא **השפה המזוהה על ידי האוטומט**.

יש שפות שעבורן קיים אוטומט סופי המזהה אותן. שפות כאלו נקראות **שפות רגולריות**. לא כל שפה היא רגולרית. למשל, השפה $\{0^n1^n : n \geq 0\}$ איננה רגולרית, כפי שנובע מההסבר שלהלן (אם כי הוא איננו מהווה הוכחה פורמלית): על מנת לזהות שפה כזו, האוטומט צריך "לספור" את האפסים המובילים בקלט ולהשוות מספר זה למספר העוקב של אחדים (ולוודא שלאחר רצף אחדים זה המחרוזת מסתיימת). כיוון שלאוטומט אין אפשרות לרשום כמה אפסים נספרו, הדרך היחידה שבה אוטומט כזה יכול לספור היא על ידי עדכון מצבו עם כל אפס נוסף המזוהה ברישא של הקלט. אך כיוון שאורך הרישא בשפה זו הוא בלתי מוגבל, אין לאוטומט בעל מספר סופי של מצבים אפשרות למלא את המשימה הזו.

<u>שאלה:</u> האם כל שפה שאורך המילים בה בלתי מוגבל היא בהכרח אי-רגולרית?

<u>תשובה:</u> לא. למשל, בהינתן אלפבית סופי Σ, אורך המילים בשפה $L=\Sigma^*$ בלתי מוגבל והיא כמובן רגולרית. נציג דוגמה נוספת, טריוויאלית פחות – השפה $L=\Sigma^*$ היא איחוד של שפת כל המילים שאורכן זוגי, $L_{\text{even}}=\bigcup_{i=0}^{\infty}\Sigma^{2i}$, ושל שפת כל המילים שאורכן אי-זוגי, $L_{\text{odd}}=\bigcup_{i=0}^{\infty}\Sigma^{2i+1}$. בשתי השפות האלה אורך המילים בלתי מוגבל ושתיהן רגולריות.

פגשנו גם גרסה "משודרגת" של אוטומט סופי דטרמיניסטי – אוטומט סופי אי-דטרמיניסטי (Non-deterministic Finite Automaton ובקיצור NFA). אוטומט כזה רשאי לבצע בחירות אי-דטרמיניסטיות: אם הוא מצוי במצב מסוים וקורא תו קלט מסוים, הוא יכול לעדכן את מצבו לאחד מבין קבוצה של מצבים. כלומר, ב-DFA כל זוג סדור של מצב ותו קלט מגדיר באופן חד-משמעי מצב יחיד חדש שהאוטומט צריך לעבור אליו, ואילו ב-NFA מוגדרת קבוצת מצבים חדשים שהאוטומט יכול לעבור לאחד מהם באופן שרירותי (קבוצת מצבים זו יכולה אף להיות ריקה). יתר על כן, ישנם מצבים שאם האוטומט מגיע אליהם במהלך העיבוד של הקלט, הוא רשאי לעדכן את מצבו מבלי שהוא יקרא אף תו קלט נוסף (מעברים כאלו קרויים מעברי-ε או מסעי-ε). האוטומט מקבל קלט נתון, אם קיים תרחיש פעולה של האוטומט שבסופו הקלט מתקבל (כלומר, גם אם קיימות בחירות אי-דטרמיניסטיות של האוטומט שבסופן הקלט נדחה, מספיק שיהיה תרחיש פעולה אחד המוביל לקבלת הקלט כדי לומר שהאוטומט מקבל את הקלט הזה).

האי-דטרמיניזם מאפשר חופש פעולה נוסף שלא קיים במודל הקשיח של אוטומטים דטרמיניסטיים. כלומר, כל DFA הוא מקרה פרטי של NFA. אך, למרבה ההפתעה, כוחם החישובי של NFA-ים אינו עולה על זה של DFA-ים. לכל NFA קיים DFA שקול המזהה בדיוק אותה שפה רגולרית. אך DFA זה עלול להיות מסורבל בהרבה מה-NFA השקול, ולכן לפעמים נוח יותר לעבוד עם אוטומטים אי-דטרמיניסטיים.

המודל החישובי השני שבו נתקלנו הוא **אוטומט מחסנית** (Pushdown Automaton או PDA). מדובר באוטומט סופי אי-דטרמיניסטי המצויד בנוסף למצבים גם במחסנית אחסון בעלת עומק בלתי מוגבל. בכל פעם שהאוטומט קורא תו קלט, הוא רשאי להוסיף ערך כלשהו למחסנית (push) או לקרוא את הערך העליון במחסנית ולסלקו (pop) או אף לעשות את שתי הפעולות (תחילה לקרוא את הערך העליון במחסנית ואז לכתוב ערך חדש במקומו). מחסנית זו מספקת לאוטומט זיכרון בלתי מוגבל. לפיכך, PDA-ים "חזקים" יותר מאוטומטים סופיים במובן שמשפחת השפות שהם יכולים לזהות רחבה יותר ממשפחת השפות הרגולריות. השפות המזוהות על ידי PDA-ים קרויות **שפות חסרות הקשר** (Context Free Languages או CFL). ניתן לאפיין CFL-ים לא רק באמצעות תיאור המודל החישובי המזהה אותן, אלא גם על ידי מערכת חוקים המייצרת אותן. מערכת חוקים כזו נקראת **דקדוק חסר הקשר** (Context Free Grammar או בקיצור CFG).

השפה $\{0^n1^n : n \geq 0\}$, שאיננה רגולרית, היא שפה חסרת הקשר. כלומר, אין DFA המזהה אותה, אך קיים PDA המזהה אותה. אך גם כוחו של PDA נראה מוגבל מכדי לשמש מודל חישובי סביר. קיימות שפות פשוטות שאינן חסרות הקשר (כלומר, שאינן ניתנות לזיהוי על ידי PDA) אבל היינו מצפים שכל מחשב יוכל לזהותן. שפה כזו היא למשל $\{0^n1^n2^n : n \geq 0\}$.

לסיכום, נראה כי שני המודלים שתיארנו לעיל אינם יכולים לשמש מודל מופשט למחשב, שכן הם אינם מסוגלים לזהות שפה פשוטה כמו $\{0^n1^n2^n : n \geq 0\}$, אבל קל מאוד לכתוב תכנית מחשב שתזהה אותה. בקורס זה נציג מודל חזק יותר – **מכונת טיורינג** – המקובל כהפשטה הנכונה למה שאנו מכירים כמחשב. בהמשך, נביא את תזת צ'רץ'-טיורינג, האומרת למעשה כי כל מודל חישובי חזק סביר שקול למכונת טיורינג.

חשוב להדגיש כי "המודל החישובי הנכון" הוא עניין של מוסכמה – אקסיומה, ולא משפט הדורש הוכחה, כפי שבגיאומטריה מוסכם עלינו כי "בין כל שתי נקודות עובר ישר יחיד". עם זאת, למודל החישובי שעליו נרצה להסכים צריכות להיות שתי תכונות:

1. הוא צריך להיות מסוגל לבצע כל מטלה חישובית שנראה לנו סביר שמחשב מסוגל לבצע.
2. הוא צריך להיות פשוט ככל האפשר (אחרת, יהיה קשה לנו לנתח את כוחו החישובי).

1. תזת צ'רץ'-טיורינג (פרק 3 בספר הלימוד)

1.1 מכונות טיורינג

קראו את סעיף 3.1 בספר הלימוד עד לפני התת-סעיף "הגדרה פורמלית של מכונת טיורינג"

המודל של מכונת טיורינג המתואר כאן נראה פשוט מאוד: מכונה בעלת ראש קורא וכותב יחיד, המשתמשת בסרט אינסופי המשמש גם לקריאת הקלט, גם לכתיבה של תוצאות ביניים (כלומר, כזיכרון) וגם לכתיבת הפלט (במקרים שבהם יש פלט). למכונה יש מספר סופי של מצבים. היא עוברת ממצב למצב על פי המצב הנוכחי והתו שנקרא זה עתה מהסרט, כמוגדר בפונקציית המעברים. אחד המצבים הוא המצב המקבל ומצב אחר הוא המצב הדוחה. סרט העבודה, המשמש לקלט וכאמצעי זיכרון, הוא אינסופי; מכונת טיורינג יכולה לנוע על הסרט גם ימינה וגם שמאלה; והיא יכולה לעצור עם תשובה (קבלה או דחייה) עוד בטרם קראה את כל הקלט, אם היא הגיעה למצב המקבל או למצב הדוחה.

למרות פשטותה של המכונה, לא נמצאה עד עתה מטלה חישובית אשר נראה כי מחשב אמיתי מסוגל לבצע, ואילו מכונת טיורינג לא. יתרה מכך, מודלים מורכבים הרבה יותר, שנראים לכאורה חזקים יותר, הוכחו כשקולים למכונת טיורינג במובן הבא: הבעיות שניתנות לפתרון במודלים האחרים ניתנות לפתרון גם על ידי מכונת טיורינג.

כדאי להבהיר כבר עתה שמדובר במכונה תיאורטית בלבד. מעין מודל מחשבתי שמטרתו אחת: להבין מהם הגבולות התיאורטיים של החישוב (זוהי שאלת **החישוביות**). כלומר, האם כל דבר ניתן לחישוב, ואם לא – מה ניתן לחשב ומה לא? בשלב זה אין לנו כל עניין בשאלת היעילות של החישוב מבחינת זמן הריצה או כמות הזיכרון הנדרשת לחישוב, וגם איננו מתעניינים בשאלה עד כמה התכנית המוצעת לפתרון בעיה מסוימת מסורבלת או קשה להבנה. בעניין יעילות החישוב יעסוק החלק השני של קורס זה (**סיבוכיות**), ואילו השאלה השנייה איננה רלוונטית כלל ועיקר לגבי תכנות במחשב תיאורטי (היא רלוונטית לגבי שפות תכנות מעשיות ואז קוראים לתחום העוסק בשאלות כאלו **הנדסת תוכנה**). בשל פשטותה של מכונת טיורינג, ה"תכנות" שלה מסורבל, ולכן בדרך כלל נסתפק בתיאור כללי של אופן הפעולה של המכונה בפתרון בעיה נתונה.

המשיכו לקרוא בספר הלימוד את התת-סעיף "הגדרה פורמלית של מכונת טיורינג".

הגדרה 3.3 מגדירה באופן ברור את המושג **מכונת טיורינג**. לאחר מכן מתואר האופן שבו מכונת טיורינג מעבדת קלט נתון. שימו לב להבדל נוסף בין מכונת טיורינג לאוטומט סופי, עם מחסנית או בלעדיה. בשני המודלים הקודמים, האוטומט עבר על סרט הקלט משמאל לימין, תו אחר תו, ותמיד עצר בסוף ונתן תשובה חיובית או שלילית לגבי קבלת הקלט. לעומת זאת, מכונת טיורינג איננה עוצרת עם סיום קריאת הסרט (אין לכך משמעות כיוון שהיא יכולה לשנות את תוכן הסרט

וגם לעבור עליו בשני הכיוונים) אלא רק כאשר היא מגיעה למצב עצירה (q_{accept} או q_{reject}).
קיימת האפשרות המצערת שהמכונה לעולם לא תגיע למצב עצירה על קלט נתון. אכן, זהו אחד המחירים הכבדים שאנו נדרשים לשלם בשל "שדרוג" האוטומט הסופי ואוטומט המחסנית למודל החישובי החזק יותר. עם זאת, יש לזכור כי זה יכול לקרות גם במחשב אמיתי.

נדגיש מספר נקודות ביחס להגדרת מכונת טיורינג:

1. מספר המצבים במכונת טיורינג הוא סופי.
2. האלפבית של מכונת טיורינג הוא סופי.
3. הסרט של מכונת טיורינג הוא אינסופי לכיוון ימין, אך יש לו קצה שמאלי שהראש הקורא-כותב איננו יכול לנוע שמאלה ממנו.
4. הסרט של מכונת טיורינג מאפשר לזכור ולעבד כמות בלתי מוגבלת של מידע.
5. רוב המכונות המתוארות בפרק זה מטפלות בבעיות הכרעה (שהן בעיות שהתשובה עליהן היא "כן" או "לא"). עם זאת, למכונת טיורינג M יש פלט $M(x)$ לכל קלט x שעליו היא עוצרת – פלט זה הוא תוכן הסרט כאשר המכונה עוצרת. לכן מכונות טיורינג מסוגלות גם לחשב פונקציות ולא רק לפתור בעיות הכרעה.

תרגיל 1.1

כזכור, קונפיגורציה היא שלשה (u,q,v) כאשר: u היא המחרוזת משמאל לראש הקורא-כותב, q הוא מצב המכונה, ו-v היא המחרוזת הנותרת, כולל התו שבו נמצא הראש. רשמו את הקונפיגורציה שמכונת טיורינג תעבור אליה בכל אחד מהמקרים האלה:

א. הקונפיגורציה הנוכחית היא $1101q0001$ ו-$\delta(q,0)=(q',1,R)$.

ב. הקונפיגורציה הנוכחית היא $q00001$ ו-$\delta(q,0)=(q',1,L)$.

ג. הקונפיגורציה הנוכחית היא $00001q$ ו-$\delta(q,0)=(q',1,L)$.

ד. הקונפיגורציה הנוכחית היא $00001q$ ו-$\delta(q,\sqcup)=(q',0,R)$ (הסימן $\sqcup$ מציין כאן רווח).

הגדרה:

שפה נקראת **ניתנת לזיהוי (על ידי מכונת טיורינג)**, או **מזוהה-טיורינג** (**Turing Recognizable)**, אם קיימת מכונת טיורינג המקיימת את התנאי הבא: לכל מילת קלט ששייכת לשפה המכונה עוצרת במצב המקבל; לכל מילה שלא שייכת לשפה, המכונה עוצרת במצב הדוחה, או שאינה עוצרת.

במילים אחרות, מכונת טיורינג מזהה שפה מסוימת אם לכל מילת קלט בשפה המכונה תמיד עוצרת במצב המקבל, אך קלטים שאינם מהשפה מביאים אותה לעצירה ולדחיית הקלט או לאי-עצירה. האפשרות האחרונה אינה רצויה כלל, מכיוון שבמצב כזה לא נדע אם לפנינו תהליך עיבוד

ארוך במיוחד שיסתיים בעוד דקה או אולי בעוד שנה, או שמדובר בקלט שהמכונה לעולם לא תעצור עליו.

משפחה מצומצמת יותר של שפות היא המשפחה הבאה:

הגדרה:

שפה נקראת **ניתנת להכרעה (על-ידי מכונת טיורינג)**, או **כריעה-טיורינג (Turing Decidable)** אם קיימת מכונת טיורינג **שתמיד עוצרת**, והיא עוצרת במצב המקבל אם ורק אם מילת הקלט שייכת לשפה (ובמצב הדוחה אם ורק אם המילה לא שייכת לשפה).

כלומר, מכונת טיורינג מכריעה שפה מסוימת אם היא עוצרת ומקבלת כל קלט מהשפה, ועוצרת ודוחה כל קלט שאיננו בשפה (ובפרט, אין קלטים שעליהם המכונה אינה עוצרת).

בהמשך ניתקל בשפות שהן מזוהות-טיורינג אך לא כריעות-טיורינג.

המשיכו לקרוא בספר הלימוד בתת-סעיף "דוגמאות של מכונות טיורינג" עד סוף דוגמה 3.7.

דוגמה 3.7 ממחישה עד כמה התיאור הפורמלי של מכונת טיורינג עלול להיות מסורבל גם במקרה שהשפה שצריך לזהות פשוטה מאוד.

במקרה שלפנינו יש להכריע את שפת כל מחרוזות האפסים שאורכן הוא חזקה של 2 (זוהי שפה שאיננה חסרת הקשר). האלגוריתם מתואר בספר בעמוד 171. נתאר את האלגוריתם תוך ביאור השלבים בו:

1. עבור על הקלט מהקצה השמאלי לקצה הימני ומחק כל אפס שני על ידי החלפתו בסימן x. כלומר, שמור את האפסים שבמקומות האי-זוגיים (הראשון, השלישי וכן הלאה) ומחק את אלו שבמקומות הזוגיים, תוך כדי התעלמות מסימני x המציינים אפסים שנמחקו בסיבובים קודמים. שימו לב: האפס במקום הראשון הפך לסימן רווח ␣. אין להתייחס אליו כאל אפס שנמחק. החלפתו של האפס בסימן הרווח נועדה רק לציין את הקצה השמאלי של הסרט.
2. אם בשלב הקודם נמצא רק אפס יחיד, הרי שהתחלנו עם מספר אפסים שהוא חזקה שלמה של 2, כיוון שבכל סיבוב מספר האפסים קטן בדיוק בחצי. לכן הקלט מתקבל.
3. אם בשלב הקודם מספר האפסים היה גדול מאחד והיה אי-זוגי, הרי שלא יתכן שהתחלנו בחזקה של 2. לכן הקלט נדחה.
4. החזר את הראש הקורא לקצה השמאלי של הסרט על ידי הליכה שמאלה עד למציאת סימן הרווח ששמנו בקצה זה.
5. חזור לשלב 1.

כדי להבין את פעולת המכונה, רצוי להתחיל בהרצה שלה על קלט מדגמי פשוט. בספר מודגמת ריצת המכונה על הקלט 0000, אך כדאי גם להריץ אותה על קלטים קצרים יותר.

תרגיל 1.2

רשמו את סדרת הקונפיגורציות שעוברת המכונה M_2 בהינתן הקלטים הבאים:

א. 0

ב. 00

ג. 000

כעת נסביר את פעולת המכונה, כפי שהיא מתוארת באיור 3.8:

מצב q_1 הוא מצב ההתחלה ואנו לעולם איננו חוזרים אליו. אם התו הראשון הוא אפס, אנו מחליפים אותו בסימן רווח, מזיזים את הראש ימינה ועוברים למצב q_2 להמשך עבודה. לעומת זאת, אם התו הראשון הוא רווח, הרי שמחרוזת הקלט ריקה ולכן אנו עוברים ל- q_{reject} (במקרה זה אין כל משמעות להזזת הראש ימינה, אך תמיד לאחר קריאת תו יש להזיז את הראש ולכן הזזת הראש במקרה זה נועדה רק לצורך "עמידה בתקן"). אם התו הראשון הוא x, אנו פועלים באופן דומה. שימו לב שלא יתכן שהתו הראשון יהיה x (בהנחה שהקלט חוקי), אך מכיוון שעלינו להגדיר את פונקציית המעברים δ לכל צירוף של תו סרט ומצב, אנו מוסיפים גם את הגדרת $\delta(q_1,\text{x})$. גם אם בפועל צירוף כזה הוא בלתי אפשרי.

מצב q_2 הוא המצב שבו נמצאת המכונה על מנת לדלג על x-ים המופיעים מיד לאחר האפס הראשון (זה שהוחלף בסימן רווח) ולהגיע לאפס הבא. שימו לב! אנו נכנסים תמיד למצב q_2 כאשר הראש נמצא על התו השני משמאל. זה קורה באחד משני המקרים הבאים:

- במעבר מ- q_1 ל- q_2. מעבר כזה קורה בדיוק פעם אחת בכל הריצה, אחרי קריאת האפס הראשון והחלפתו בסימן רווח.
- במעבר מ- q_5 ל- q_2. מצב q_5 הוא המצב שבו המכונה נמצאת בזמן ההליכה על הסרט שמאלה עד הקצה (ראו שלב 4 בתיאור האלגוריתם לעיל).

אנו יוצאים ממצב q_2 לאחר שמצאנו את האפס השני ששרד עד כה, אנו מוחקים אותו, זזים ימינה ועוברים למצב q_3. שימו לב שבמעבר הנוכחי על סרט הקלט זיהינו עד כה שני אפסים: האפס הראשון, שהוחלף ברווח, ולעולם לא נמחק; והאפס השני שזיהינו זה עתה והפכנו ל-x. עלינו לזכור אם מספר האפסים שאנו פוגשים במעבר זה הוא זוגי או אי-זוגי.

במצב q_3 אנו נשארים בזמן שאנו מדלגים על x-ים, עד הגיענו לאפס הבא. אפס זה נשאר, וברגע שזיהינו אותו, אנו זזים עם הראש ימינה ועוברים למצב q_4. שימו לב שעד שלב זה זיהינו במעבר הנוכחי מספר אי-זוגי של אפסים. כלומר, q_4 זוכר את העובדה כי מנינו מספר אי-זוגי של אפסים עד כה.

במצב q_4 אנו שוב מדלגים על x-ים. אם הגענו לאפס, אנו מוחקים אותו וחוזרים למצב q_3 המציין מספר זוגי של אפסים. אם הגענו לסוף הקלט, הרי שנותר מספר אי זוגי של אפסים ולכן הקלט נדחה.
בסופו של דבר, אם הגענו לסוף הקלט וזיהינו מספר זוגי של אפסים, שחצי מהם מחקנו, אנו עוברים ממצב q_3 למצב q_5 תוך כדי תחילת הליכה שמאלה. ההליכה שמאלה נמשכת בזמן שהמכונה מצויה במצב q_5. כאשר מזוהה סימן הרווח בקצה השמאלי, אנו עוברים למצב q_2 וזזים תו אחד ימינה כדי להתייצב על התו השני בסרט.
הקלט מתקבל אם במצב q_2 לא נמצא אפס מימין לרווח שרשמנו בריבוע השמאלי ביותר. זה אומר שנותר אפס יחיד (אותו אפס שהוחלף ברווח), וזה מעיד על כך שמספר האפסים ההתחלתי היה חזקה של 2.

<u>שאלה:</u> בהינתן קלט אפסים ארוך, ישנם אפסים שיימחקו בעת המעבר מ- q_2 ל- q_3, וכאלה שיימחקו בעת המעבר מ- q_4 ל- q_3. נסו לאפיין את האפסים שיימחקו בעת המעבר מ- q_2 ל- q_3.

<u>תשובה:</u> אם נתייחס לתא הראשון על הסרט כאל תא מספר 0, לתא הבא כאל תא מספר 1 וכך הלאה, אז האפסים שיימחקו בעת המעבר מ- q_2 ל- q_3 הם אלו המצויים במקומות $1,2,4,8$ וכו', כלומר, האפסים הנמצאים במקומות שהם חזקות של 2.

המשיכו לקרוא בספר הלימוד בתת-סעיף "דוגמאות של מכונות טיורינג" את דוגמה 3.9.

עיינו בתיאור המילולי של האלגוריתם שבעמוד 167 ובאיור 3.10 המתאר את המכונה. זוהי דוגמה מורכבת יותר. כדי לפשט את האיור, לא מופיע כאן המצב q_{reject} ; במקום זה, בכל פעם שאנו מגיעים למצב שממנו אין חץ יציאה המתאים לתו סרט מסוים, יש לפרש מעבר זה כמעבר המוביל ל- q_{reject}.

- אם אנו נתקלים מיד בהתחלה בתו #, אנו עוברים ל- q_8. אם התו הבא הוא סוף הקלט, הרי לפנינו המחרוזת # שהיא חוקית ולכן מתקבלת. אם התו הבא הוא 0 או 1, אין חץ יציאה מתאים מ- q_8. כלומר, במקרה זה המכונה תעבור למצב q_{reject}, לפי המוסכמה שתוארה לעיל. שימו לב שיש מ- q_8 חץ יציאה גם עבור התו x. מסע שכזה איננו אפשרי במקרה שאנו מתארים כעת (המקרה שבו התו הראשון בקלט הוא #), כיוון שההנחה היא שהקלט הוא מעל האלפבית $\Sigma=\{0,1,\#\}$. אך מסע זה רלוונטי במקרים אחרים שבהם נגיע למצב q_8 בשלב מתקדם יותר בריצה (דהיינו, מקרים שבהם התו # מופיע על הסרט אך לא מיד בתחילתו).
- אם התו הראשון הוא 0, אנו זזים ימינה ועוברים למצב q_2. אנו נשארים במצב זה ומדלגים על תווי 0 ו-1 עד שאנו מגיעים ל- #, או אז אנו עוברים למצב q_4 (שימו לב

שאם אין בנמצא על הסרט התו $\#$, אז אנו נגיע לסופו של הסרט ונמצא תו רווח $\sqcup$. במצב זה יידחה הקלט כפי שמשתמע מכך שאין חץ יציאה מ-q_2 המתייחס לתו סרט זה). במצב q_4 אנו מדלגים על כל תווי ה-x שנוצרו במעברים קודמים של המכונה על חלק זה של הסרט עד שאנו מגיעים לתו הראשון שאיננו x. אם תו זה הוא 0, אנו הופכים אותו ל-x ועוברים למצב q_6; אחרת – הקלט נדחה.

- אם התו הראשון הוא 1, אנו זזים ימינה ועוברים למצב q_3. החלק הימני באיור 3.10 מתאר את הטיפול במקרה זה, אשר דומה מאוד לטיפול במקרה הקודם שתואר בחלק השמאלי של האיור.
- מצב q_6 מציין הצלחה בבדיקה עד כה. כעת אנו מתחילים את הטיול בחזרה שמאלה עד לסימן ה-$\#$. כאשר אנו מגיעים אליו, אנו מדלגים עליו ועוברים למצב q_7. כעת אנו ממשיכים בהליכה שמאלה תוך כדי דילוג על 0-ים ו-1-ים שאותם טרם בדקנו, עד שאנו מגיעים ל-x הימני ביותר על החלק השמאלי של הסרט. מקום זה על הסרט הוא המקום האחרון בחלק השמאלי שהושווה בהצלחה אל התו המקביל בחלק הימני. אנו צועדים צעד אחד ימינה, כדי להביא את הראש הקורא לעמוד על התו הבא שטרם נבדק, ואנו עוברים למצב q_1. מכאן, הבדיקה ממשיכה כמקודם.

<u>שאלה</u>: כיצד יידחה קלט מהצורה $wv\#w$ כאשר v היא מחרוזת לא ריקה?

<u>תשובה</u>: אם התו הראשון ב-v הוא 0, אנו נגיע למצב q_4 שבו נדלג על כל ה-x-ים בחלק הימני של הסרט שמציינים תווים שכבר השווינו בהצלחה. אך כיוון שכל התווים ב-w בחלק הימני של הסרט כבר הושוו לתווים זהים בחלק השמאלי, אז בתום מסע הדילוגים הזה נגיע לסימן הרווח. כיוון שאין מ-q_4 חץ יציאה המתאים לתו הרווח, אנו נדחה את הקלט הזה. דחייה דומה מתבצעת במצב q_5 אם התו הראשון ב-v הוא 1.

תרגיל 1.3

רשמו את סדרת הקונפיגורציות שעוברת מכונת טיורינג שלעיל בהינתן הקלטים הבאים:

א. 11

ב. $1\#1$

המשיכו לקרוא בספר הלימוד בתת-סעיף "דוגמאות של מכונות טיורינג" את דוגמה 3.11.

כאן אנו מסתפקים בתיאור מילולי של אלגוריתם המכונה, כשאנחנו יודעים שניתן לתרגם את התיאור המילולי לתיאור פורמלי מדויק (שיהיה, ככל הנראה, מסורבל). הרעיון באלגוריתם זה הוא: לכל אחד מתווי ה-a, מחוק מספר תווי c כמספר תווי ה-b הקיימים. אם בשלב ביניים כלשהו ייגמרו תווי ה-c, אז $k < i \times j$. אם, לעומת זאת, התהליך הסתיים ועדיין נותרו תווי c, אז

$k > i \times j$. לפיכך, אם התהליך הסתיים בהצלחה ובסופו לא נותרו על הסרט תווי c, אז $k = i \times j$, ולכן אנו מקבלים את הקלט.

בספר הלימוד נדונה השאלה כיצד למצוא את הקצה השמאלי של הסרט, ומוצעות שתי שיטות לכך: השיטה שהוצגה בדוגמה 3.7 (דהיינו, החלפת התו הראשון בתו מיוחד) ושיטה נוספת המתוארת כאן והמבוססת על העיקרון שאם פונקציית המעבר דורשת מהראש לנוע שמאלה, אז במקרה שהראש כבר מצוי על התו השמאלי ביותר – הוא פשוט נשאר במקומו.

תרגיל 1.4

נניח שהקלט למכונת טיורינג לעיל היה $a^3b^4c^{12}$. תארו את תוכן הסרט בנקודות הבאות באלגוריתם (מבלי לציין את מיקום הראש הקורא-כותב) לאורך כל הריצה:

- נקודת תצפית 1: בתום המשפט הראשון הרשום בשלב 3 באלגוריתם.
- נקודת תצפית 2: בתום שלב 3 באלגוריתם.
- נקודת תצפית 3: בשלב 4 מיד לאחר החזרת תווי ה-b שנמחקו.

הניחו שאת תווי ה-a מוחקים באמצעות תווי רווח (שכן מעבר לפעולת המחיקה, הם גם צריכים לציין עבורנו את הקצה השמאלי של הסרט), ואילו את תווי ה-b וה-c מוחקים באמצעות x (כאן אסור לנו להשתמש בסימן רווח פן נטעה ונחשוב שהגענו לסוף הסרט מימין).

המשיכו לקרוא בספר הלימוד בתת-סעיף "דוגמאות של מכונות טיורינג" את דוגמה 3.12.

כאן אנו פוגשים בטכניקה של מסמני מקום (place markers). לפני תחילת ההשוואה של שתיים מבין מחרוזות הקלט, על הראש לסמן את ה- # המסמן את התחלת המחרוזת הראשונה ואת זה המסמן את התחלת המחרוזת השנייה. אילו היינו עושים זאת ידנית, באמצעות נייר ועיפרון, היינו מסמנים נקודה מעל שני תווי ה- # האלו. במכונת טיורינג פשוט נחליף את התו # במקומות אלו בתו # עם נקודה מעליו, או בכל תו אחר שנבחר.

1.2 גרסאות שונות של מכונות טיורינג

קראו את סעיף 3.2 בספר הלימוד עד לפני התת-סעיף "מכונות טיורינג מרובות-סרטים".

למודל הבסיסי של מכונת טיורינג שתואר בסעיף הקודם יש גרסאות שונות הנראות כלליות יותר ועל כן, אולי, חזקות יותר. אך כפי שנראה כאן, המודל הבסיסי מספיק חזק, וכל ההכללות המתוארות כאן אינן מסוגלות לבצע משימות שהן מעבר ליכולתו של המודל הבסיסי. כדי להוכיח זאת, מראים כי ניתן לחקות בעזרת המודל הבסיסי של מכונת טיורינג כל פעולה של המכונות ה"משודרגות" המתוארות להלן.

הגרסה הראשונה המתוארת כאן בקצרה היא זו של מכונת טיורינג המתירה לראש להישאר במקומו לאחר טיפול בתו מסוים (במקום לזוז ימינה או שמאלה). במקרה זה קל לראות שניתן לבנות מכונת טיורינג רגילה שעושה אותה עבודה ממש על ידי הוספת מצבים ותנועות ראש ימינה ושמאלה.

תרגיל 1.5

תהי M מכונת טיורינג עם $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$, כלומר, הראש יכול להישאר במקומו. תארו מכונת טיורינג בסיסית M_1 השקולה ל- M, על ידי כך שתראו איך לבצע הדמיה של כל כלל מעבר ב- M מהצורה $\delta(q,a) = (q',b,S)$ על ידי מספר סופי של כללים שיבצעו את אותה פעולה במכונה M_1.

המשיכו לקרוא בספר הלימוד את התת-סעיף "מכונות טיורינג מרובות-סרטים".

בתת-סעיף זה אנו פוגשים מכונות טיורינג שלרשותן עומדים כמה סרטים. מסתבר שניתן לדמות מכונה מרובת-סרטים כזאת באמצעות מכונת טיורינג בסיסית, בעלת סרט יחיד, במחיר של עבודה רבה יותר, מספר גדול יותר של מצבים ושימוש במסמני מקום.

תחילה נרשמים התכנים ההתחלתיים של k הסרטים על סרט בודד, תוך הפרדתם על ידי מפרידי $\#$. כמו כן, מציינים את התו שעליו נמצא הראש הקורא-כותב בכל אחד מהסרטים על ידי נקודה מעליו (כלומר, אם הראש הקורא-כותב מצביע באחד הסרטים על התו a, תו זה ייוצג במכונה יחידת-הסרט על ידי התו החדש $\dot{a}$). דבר זה מודגם יפה באיור **3.14** בספר.

כדי לדמות צעד אחד של המכונה מרובת-הסרטים, המכונה יחידת-הסרט תעבור על כל הסרט שלה כדי לקרוא את התווים הנוכחיים בכל אחד מ- k קטעי הסרט. עם כל קריאת תו היא תעדכן את מצבה כדי לזכור את התווים שנקראו עד כה. למשל, נניח שכל התווים המקוריים הם בינאריים ונניח ש- $k = 3$. אז במכונה החדשה נגדיר את המצבים (14 מצבים סה"כ):

$$q_0, q_1, q_{00}, q_{01}, q_{10}, q_{11}, q_{000}, q_{001}, \cdots, q_{111}$$

כך, אם למשל התווים שעליהם נמצא הסמן בשלושת קטעי הסרט הם 0, 1 ו-1 (כלומר, על הסרט הבודד רשום במקומות המתאימים $\dot{0}$, $\dot{1}$ ו-$\dot{1}$), המכונה תעבור דרך המצבים האלה: $q_0 \rightarrow q_{01} \rightarrow q_{011}$. (סביר להניח שנצטרך עוד מצבי ביניים, למשל כדי לציין מצב שבו זיהינו מעבר מקטע סרט אחד לאחר, אך טרם הגענו לתו המסומן באותו קטע סרט; אולם לא ניכנס לפרטים אלו.) כאשר המכונה תגמור את סריקת הסרט, היא "תזכור" את שלושת התווים הנוכחיים (המיוצגים על ידי מצבה הנוכחי של המכונה, למשל q_{011} בדוגמה לעיל), ואז היא תוכל לדמות את הצעד הבא של המכונה המקורית בעלת שלושת הסרטים. כעת היא תתחיל מעבר שני

על הסרט על מנת לעדכן כל קטע של הסרט בהתאם. גם כאן נצטרך להשתמש במצבים הזוכרים באיזה קטע של הסרט ובאיזה שלב של העדכון אנו נמצאים.

נדגיש שיש צורך בקבוצת מצבים כמו זו שתיארנו לכל מצב של המכונה מרובת-הסרטים, מכיוון שעלינו לזכור כל הזמן באיזה מצב של המכונה מרובת-הסרטים אנו נמצאים. למשל, אם נחזור לדוגמה שבה תיארנו מכונה בעלת שלושה סרטים כאשר כל הקלטים בינאריים, אז בהנחה שיש במכונה מרובת-הסרטים הזו s מצבים שונים, נצטרך להגדיר במכונת טיורינג יחידת-הסרט המדמה אותה s קבוצות מצבים שונות מהצורה

$$q_0, q_1, q_{00}, q_{01}, q_{10}, q_{11}, q_{000}, q_{001}, \cdots, q_{111}$$

מכיוון שהמצב של מכונה זו אמור לזכור גם את התווים שנקראו מכל סרטי הקלט וגם את מצבה הנוכחי של המכונה מרובת-הסרטים.

המשיכו לקרוא בספר הלימוד את התת-סעיף "מכונות טיורינג אי-דטרמיניסטיות".

מכונות לא דטרמיניסטיות אמורות להיות מוכרות לכם מלימודיכם הקודמים (על אוטומטים סופיים ואוטומטי-מחסנית). הכוח של מכונות אלה נובע מן ה**הגדרה של קבלת מילה**: מילה מתקבלת אם יש לה אפילו מסלול מקבל אחד בלבד (אף על פי ששאר המסלולים אינם מסתיימים במצב מקבל). לעתים יש למכונות אלה כוח גדול יותר מאשר למודל הדטרמיניסטי המקביל (זה המצב באוטומט-מחסנית), ולעתים לא (זה המצב באוטומטים סופיים). משפט 3.16 קובע שבמכונות טיורינג, אי-דטרמיניזם אינו מוסיף כוח לזהות שפות מעבר למה שניתן בעזרת מכונות דטרמיניסטיות. הוכחת המשפט נעשית, כרגיל, בעזרת סימולציה. מראים שלכל מכונה לא דטרמיניסטית אפשר לבנות מכונה דטרמיניסטית שקולה.

מכונת טיורינג אי-דטרמיניסטית מאפשרת כמה דרכי פעולה בכל קונפיגורציה נתונה. לכן, פונקציית המעברים בה מוגדרת כ-

$$\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\}),$$

במקום

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

במקרה הדטרמיניסטי. שימו לב: הקבוצה $Q \times \Gamma \times \{L, R\}$ היא המכפלה הקרטזית של קבוצת המצבים, Q, קבוצת תווי הסרט, Γ, וקבוצת כיווני התנועה האפשריים – ימין ושמאל $\{L, R\}$. לכן, כל איבר בקבוצה זו הוא שלשה המורכבת ממצב, תו סרט וכיוון תנועה. במכונה דטרמיניסטית, תיארה שלשה כזו את אופן הפעולה של המכונה בהינתן מצב נוכחי ותו סרט נוכחי. הקבוצה $P(Q \times \Gamma \times \{L, R\})$, לעומת זאת, מורכבת מכל התת-קבוצות של $Q \times \Gamma \times \{L, R\}$. לפיכך, כל איבר בקבוצה $P(Q \times \Gamma \times \{L, R\})$ הוא אוסף של שלשות מהצורה שתיארנו לעיל.

במכונה אי-דטרמיניסטית, מגדירה פונקציית המעברים $\delta : Q\times\Gamma \to P(Q\times\Gamma\times\{L,R\})$ את אופן הפעולה של המכונה לכל מצב נוכחי ותו סרט נוכחי. אם למשל

$$\delta(q,a)=\{(q_1,a_1,d_1),(q_2,a_2,d_2),\ldots,(q_k,a_k,d_k)\}$$

אז כאשר המכונה קוראת את התו a ומצבה הוא q, היא יכולה לכתוב את התו a_i, לנוע בכיוון $d_i \in \{L,R\}$, ולעבור למצב q_i, עבור כל אחת מהבחירות האפשריות $1\le i\le k$. שימו לב שייתכן ש- $k=0$.

אופן הפעולה של מכונה כזו עבור קלט נתון יכול להיות מתואר על ידי עץ. שורש העץ הוא הקונפיגורציה ההתחלתית, וכל קדקוד בעץ, המתייחס לקונפיגורציה אפשרית של המכונה בזמן עיבוד הקלט, מוביל לכל הקונפיגורציות שאליהן ניתן להגיע בהתאם להגדרת פונקציית המעברים δ. אם, למשל, אנו בקונפיגורציה שבה המצב הוא q והתו הנוכחי הוא a ו- $\delta(q,a)$ מכילה שני ערכים מהקבוצה $Q\times\Gamma\times\{L,R\}$ (כלומר, שתי דרכי פעולה אפשריות), אז לקדקוד בעץ המתאים לקונפיגורציה זו יהיו שני בנים. אם, לעומת זאת, $\delta(q,a)$ היא הקבוצה הריקה, אז לקדקוד המתאים בעץ אין בנים, כלומר הוא עלה.

אנו אומרים שהמכונה האי-דטרמיניסטית מקבלת קלט נתון, אם קיים מסלול בעץ הקונפיגורציות המתאים לקלט זה המסתיים במצב q_{accept}. המכונה איננה מקבלת קלט נתון, אם **כל** המסלולים בעץ הקונפיגורציות המתאים לקלט זה אינם מסתיימים במצב q_{accept}.

מסתבר שבחירות אי-דטרמיניסטיות אינן מחזקות את הכוח החישובי של מכונות טיורינג (משפט 3.16). הוכחת משפט 3.16 מתארת עבור מכונה אי-דטרמיניסטית נתונה מכונה דטרמיניסטית המריצה את כל המסלולים האפשריים בעץ המוגדר על ידי המכונה האי-דטרמיניסטית. אם קיים בעץ מסלול המסתיים במצב q_{accept}, אז המכונה הדטרמיניסטית תמצא אותו, תעצור ותקבל את הקלט. אם, לעומת זאת, אין בעץ אף לא ענף אחד המסתיים במצב q_{accept}, אז יש שתי אפשרויות:

- כל המסלולים בעץ מסתיימים ב- q_{reject}. במקרה זה העץ הוא סופי, ולכן כשהמכונה הדטרמיניסטית תסיים לסרוק את כולו – היא תוכל לעצור ולדחות את הקלט.
- לפחות אחד מהמסלולים בעץ הוא אינסופי, וכל המסלולים הסופיים (אם יש כאלה) מסתיימים ב- q_{reject}. במקרה זה המכונה הדטרמיניסטית לעולם לא תסיים את ריצתה, מכיוון שהיא תמשיך לסרוק את המסלולים האינסופיים "בתקווה" להגיע למצב q_{accept}.

כפי שמוסבר בספר, יש לעבור על העץ בסריקה רוחבית ולא בסריקת עומק, כדי שלא ניתקע על ענף אינסופי ונפספס ענף אחר המסתיים במצב q_{accept}.

נחזור כעת על תיאור המכונה הדטרמיניסטית שמופיע בהוכחה של משפט 3.16 ונוסיף מעט פרטים והבהרות:

1. למכונה יש שלושה סרטים. הסרט הראשון מכיל את הקלט ואיננו משתנה לאורך כל החישוב. הסרט השני הוא סרט עבודה. בכל פעם שאנו מתחילים לבחון ענף מסוים, אנו מעתיקים את תוכן הסרט הראשון לשני ומתחילים לעבד את הקלט לפי הבחירות המוכתבות על ידי ענף זה בעץ. הסרט השלישי משמש לצורך מעקב מסודר אחר הסריקה הרוחבית של העץ.
2. נניח ש- $\max\{|\delta(q,a)|, q\in Q, a\in\Gamma\}=b$. כלומר, פונקציית המעברים במכונה האי-דטרמיניסטית מאפשרת לכל היותר b דרכי פעולה שונות בכל צעד חישוב. לכן בעץ הקונפיגורציות יש לכל קדקוד לכל היותר b בנים. לכן, לכל קדקוד ניתן להתאים מחרוזת (יחידה) ב- $\{1,2,\cdots,b\}^*$ שנקרא לה "כתובת". שורש העץ הוא בעל הכתובת הריקה ε. אם יש לו 4 בנים, כתובותיהם הן 1,2,3,4 בהתאמה. אם לבן השני יש רק 2 בנים, אז כתובותיהם הן 21 ו-22, וכך הלאה. אם נסרוק את מרחב הכתובות $\{1,2,\cdots,b\}^*$ לפי הסדר הסטנדרטי (מילים קצרות קודמות למילים ארוכות; מילים באותו אורך מסודרות לפי הסדר המילוני), מובטח לנו שנגיע לכל קדקוד בעץ (שהרי במעבר רוחבי, לעולם לא נעבור לקדקוד ברמה k לפני שביקרנו קודם בכל הקדקודים ברמה $k-1$). חלק מהכתובות אינן חוקיות (כמו 23 בדוגמה לעיל), אבל אין זה מהווה בעיה כפי שמיד נראה.
3. במכונת טיורינג שלנו תהיה לולאה חיצונית שתייצר את כל הכתובות ב- $\{1,2,\cdots,b\}^*$ לפי הסדר הסטנדרטי. לכל כתובת כזו, המכונה תעתיק את תוכן הסרט הראשון לסרט השני. כעת, היא תתחיל לעבד את הקלט על פי הבחירות המוכתבות על ידי כתובת הקדקוד הנוכחי.
4. נניח שהמצב הנוכחי בעיבוד הקלט הוא q (זהו המצב של המכונה האי-דטרמיניסטית בשלב זה של העיבוד) ושהתו הנוכחי על הסרט השני הוא a. המכונה שלנו תקרא את התו הבא בכתובת של הקדקוד בעץ הנבחן כעת. נניח שתו זה הוא המספר 4. אז יש כמה אפשרויות:
   - $\delta(q,a)$ כולל לכל היותר שלוש אפשרויות. לפיכך, לפנינו כתובת בלתי חוקית. לכן איננו ממשיכים לבדוק ענף זה בעץ ואנו מסיימים שלב זה בלולאה וחוזרים ללא כל תשובה.
   - $\delta(q,a)$ כולל לפחות ארבע אפשרויות. נניח שהאפשרויות מסודרות בסדר מוסכם על $Q\times\Gamma\times\{L,R\}$. אז המכונה מבצעת צעד אחד בסרט השני לפי אפשרות הפעולה הרביעית. אם הגענו בעקבות כך ל- q_{reject}, אין טעם להמשיך עוד בענף זה בעץ, ואפשר לעבור לענף אחר בעץ במסגרת הלולאה החיצונית. אם הגענו ל- q_{accept} אז אפשר לעצור את המכונה לחלוטין ולקבל את הקלט. אם הגענו לכל מצב אחר,

נמשיך לבצע עוד צעד על הקלט בסרט השני לפי התו הבא בכתובת של הקדקוד בעץ בסרט השלישי, עד שנגיע לסוף הכתובת (תו רווח על הסרט השלישי).

5. שימו לב שמכונה זו אינה יעילה כלל. למשל, בהגיענו לקדקוד בעץ שכתובתו 314525, נבצע שוב את כל העבודה שביצענו בשלב מוקדם יותר שבו ביקרנו באב של קדקוד זה שכתובתו 31452. אך, כפי שכבר אמרנו בעבר, אין אנו מעוניינים (בשלב זה) בדיון ביעילות האלגוריתם אלא רק באפשרות קיומו.

תרגיל 1.6

הוכיחו את מסקנה 3.19. כלומר, אם יש מכונת טיורינג אי-דטרמיניסטית **המכריעה** שפה נתונה (כלומר, לכל קלט אפשרי, עץ הקונפיגורציות המתקבל במכונה זו הוא סופי, משמע, אין בו מסלולים אינסופיים), אז קיימת מכונת טיורינג דטרמיניסטית המכריעה את אותה שפה (כלומר, מכונת טיורינג שעוצרת על כל קלט במצב q_{accept} או q_{reject}). מספיק לתת תיאור מילולי קצר של הבנייה.

המשיכו לקרוא בספר הלימוד את התת-סעיף "מונים".

ההגדרה הפורמלית של **מונה** (enumerator) היא כדלקמן: מונה הוא שביעייה $(Q,\Gamma,\Sigma,\delta,q_0,q_{\text{print}},q_{\text{halt}})$ כאשר:

- Q היא קבוצה סופית של מצבים
- Γ הוא אלפבית סופי של סרט העבודה
- Σ הוא אלפבית סופי של סרט הפלט
- $\delta : Q\times\Gamma \to Q\times\Gamma\times\{L,R\}\times(\Sigma\cup\{\varepsilon\})$ היא פונקציית המעברים
- q_0 הוא מצב ההתחלה
- q_{print} הוא מצב ההדפסה
- q_{halt} הוא מצב העצירה ($q_{\text{halt}} \neq q_{\text{print}}$).

בהתחלת הפעולה, שני הסרטים במכונה ריקים ומצב המכונה הוא q_0. פונקציית המעברים δ מכתיבה את אופן הפעולה: לאיזה מצב לעבור, איזה תו לכתוב במקום התו הנוכחי על סרט העבודה, האם לנוע ימינה או שמאלה על סרט העבודה, ואיזה תו להוסיף לסרט הפלט (אם תו זה הוא ε, אז לא מתבצעת כתיבה לסרט הפלט). אם מגיעים למצב q_{print}, אז הראש הכותב בסרט הפלט יורד שורה (כמו שקורה בלחיצה על כפתור ה-return במעבד תמלילים). אם המכונה מגיעה למצב q_{halt}, היא עוצרת. השפה המופקת על ידי מונה היא קבוצת כל המחרוזות ב-Σ^* שהמונה מדפיס (בשלב כלשהו) על סרט הפלט. שימו לב שהמונה לא בהכרח עוצר, ולכן הוא יכול להפיק גם שפות אינסופיות. כמו כן, המונה יכול לכתוב אותה מחרוזת יותר מפעם אחת.

נציג, לדוגמה, מונה שמדפיס את מילות השפה $A=\{0^k1^k \mid k\geq 0\}$:

- $Q=\{q_1,q_2,q_3,q_4,q_{\text{print}},q_{\text{halt}}\}$
- $\Gamma=\{\text{x},\sqcup\}$
- $\Sigma=\{0,1\}$
- המצב ההתחלתי הוא q_1.

בפונקציית המעברים δ הושמטו מעברים בלתי אפשריים:

$$\delta(q_1,\sqcup)=(q_{\text{print}},\sqcup,\text{R},\varepsilon)$$

$$\delta(q_{\text{print}},\sqcup)=(q_2,\text{x},\text{L},0)$$

$$\delta(q_2,\text{x})=(q_2,\text{x},\text{L},0)$$

$$\delta(q_2,\sqcup)=(q_3,\sqcup,\text{R},\varepsilon)$$

$$\delta(q_3,\text{x})=(q_3,\text{x},\text{R},1)$$

$$\delta(q_3,\sqcup)=(q_4,\sqcup,\text{L},\varepsilon)$$

$$\delta(q_4,\text{x})=(q_{\text{print}},\text{x},\text{R},\varepsilon)$$

הרעיון: בכל שלב (בין הדפסה להדפסה) מגדילים ב-1 את מספר ה-x-ים שרשומים על סרט העבודה. עוברים על ה-x-ים הללו מימין לשמאל, ורושמים 0 על סרט הפלט כנגד כל x בסרט העבודה (מצב q_2). לאחר מכן עוברים על ה-x-ים בסרט העבודה משמאל לימין, ורושמים 1 על סרט הפלט כנגד כל x בסרט העבודה (מצב q_3). בסיום התהליך הזה מדפיסים (במצב q_{print}) את המילה שרשומה בסרט הפלט, ומוסיפים x על סרט העבודה מימין ל-x-ים שכבר רשומים שם (זה מתבצע במעבר מ- q_{print} ל- q_2).

בתחילת פעולת המונה (במצב q_1) נעים ימינה, ואז מדפיסים את המילה הריקה (ששייכת לשפה). בריבוע השמאלי ביותר של סרט העבודה נשאר תמיד רווח. הרווח שרשום בריבוע זה מסייע בעת התנועה שמאלה (במצב q_2) לדעת מתי הגענו לתחילת ה-x-ים על סרט העבודה.

מומלץ שתציירו את המונה ותריצו אותו להדפסת כמה מילים ראשונות מהשפה. כך תבינו היטב את הרעיון של בנייתו.

משפט 3.21 בספר טוען כי שפה היא מזוהה-טיורינג אם ורק אם קיים מונה המדפיס את כל המילים בשפה. נבהיר את הוכחת המשפט:

כיוון אחד: בהינתן מונה, אנו בונים מכונת טיורינג שמדמה את פעולתו. לשם נוחות (ולאור משפט 3.13), אנו בונים מכונה בעלת שלושה סרטים: אחד שישמור את הקלט של המכונה, ואותו לעולם

לא נשנה; שני שישמש כסרט העבודה של המונה, ושלישי שישמש כסרט הפלט של המונה. בכל פעם שהמונה מגיע למצב הדפסה, תשווה המכונה את תוכן הסרט השלישי לראשון. אם הם שווים, היא תעצור ותקבל את הקלט. אחרת, היא תמחק את תוכן הסרט השלישי ותמשיך בהרצת המונה.

אם המונה מגיע למצב עצירה מבלי שאיתרנו שוויון, המכונה תדחה את הקלט. לפיכך, אם המונה עוצר, אז המכונה תכריע את השפה; אם, לעומת זאת, המונה לעולם אינו עוצר, אז המכונה תזהה את השפה אך לא תכריע אותה, כיוון שהיא לא תעצור על קלטים שאינם בשפה (אלא אם כן השפה שהמונה מפיק היא Σ^*).

כיוון שני: בהינתן מכונת טיורינג, נבנה מונה המדמה את פעולתה. יהיו $s_1, s_2, s_3, \ldots$ כל המחרוזות ב-Σ^*, מסודרות בסדר הסטנדרטי. המונה יריץ לולאה אינסופית עבור $i = 1,2,3,\ldots$ ולכל ערך של i הוא יריץ את המכונה על i המחרוזות הראשונות $s_1, s_2, \ldots, s_i$ במשך i צעדים על כל אחת מהן. אם אחד החישובים הללו יצליח (כלומר, איזושהי מחרוזת תתקבל), המונה ידפיס את המחרוזת המתאימה. ברור אפוא שכל מחרוזת המתקבלת על ידי המכונה תודפס על ידי מונה זה, ואפילו אינסוף פעמים, ואילו כל מחרוזת שאיננה מתקבלת על ידי המכונה לעולם לא תודפס על ידי המונה. (שימו לב: אם המחרוזת s_j מתקבלת על ידי המכונה אחרי k צעדים, אז המונה ידפיס אותה בכל שלב בלולאה החל מ-$(i = \max(j,k)$).

המשיכו לקרוא בספר הלימוד את התת-סעיף "שקילות עם מודלים אחרים".

ישנם מודלים מופשטים נוספים למכונות חישוב. כולם, למרות השוני הצורני ביניהם, שקולים למודל של מכונת טיורינג במובן זה שהם יכולים לממש אותה מחלקה של "אלגוריתמים". זה מוביל אותנו לשאלה מהו אלגוריתם.

1.3 הגדרת מושג האלגוריתם

קראו את סעיף 3.3 בספר הלימוד עד לפני התת-סעיף "מינוח לתיאור מכונות טיורינג"

המונח **אלגוריתם** מעולם לא הוגדר באופן פורמלי, גם אם אנו מבינים באופן אינטואיטיבי למה הכוונה. הטיפול המדויק במושג הופיע לראשונה במאמר של אלן טיורינג, שבו הציע את המודל של מכונות טיורינג, ובמאמר של אלונזו צ'רץ' שבו הציע לוגיקה פורמלית הקרויה λ-calculus. שני המודלים האלה היו שונים בתכלית זה מזה, אך הם התגלו כשקולים - שניהם מסוגלים להכריע או לזהות את אותן שפות. מכאן מתקבלת **התזה של צ'רץ'-טיורינג** המזהה בין המושג אלגוריתם ובין המושג מכונת טיורינג.

הבעיה העשירית של הילברט, במינוח "מודרני" שלא היה קיים בשנת 1900 שבה היא נוסחה, מבקשת לבנות מכונת טיורינג **שתכריע** את שפת הפולינומים במקדמים שלמים שיש להם שורש שכל רכיביו הם מספרים שלמים. למשל, x^2-2 או x^2+5y^4+3 הם פולינומים שאין להם שורשים שכל רכיביהם שלמים (לפולינום השני אין אפילו שורשים שכל רכיביהם ממשיים) אבל $4x^3-y^2z+8$ הוא פולינום בעל שורש שכל רכיביו שלמים ($x=-1, y=1, z=4$). (שימו לב: מדובר בפולינומים בכמה משתנים.)

על מנת לנסח את הבעיה בצורה פורמלית, עלינו לתאר פולינומים כאלו בצורה פורמלית, כדי שיוכלו לשמש כקלט למכונת טיורינג. אוסף כל הפולינומים במקדמים שלמים הוא תת-קבוצה של Σ^*, כאשר $\Sigma=\{0,1,\ldots,9,+,-,\_,\hat{},x\}$. למשל, הפולינום בדוגמה האחרונה יכול להירשם כ- $4x\_1\hat{}3-x\_2\hat{}2x\_3+8$ (כאן x_2 הוא y ו- $x\_2\hat{}2$ הוא y^2). שימו לב שיש מחרוזות ב- Σ^* שאינן מתארות פולינום (למשל, $4\_x\hat{}1$).

כפי שמתואר בספר, זוהי שפה שיש אפשרות לזהותה: נבנה מכונת טיורינג שתרוץ על כל ערכי השורשים השלמים האפשריים באופן סדרתי, תציב אותם בפולינום, תחשב את ערכו ותבדוק אם התוצאה היא אפס. אם קיים לפולינום שורש בעל רכיבים שלמים, אז הוא יימצא בסופו של דבר והמכונה תעצור ותקבל את הקלט הזה; אך אם אין לפולינום שורש כזה, החיפוש לעולם לא יסתיים.
בשנת 1970 הושלמה ההוכחה שאין בנמצא מכונת טיורינג שתוכל **להכריע** את השפה הזו. כלומר, הבעיה העשירית של הילברט אינה ניתנת לפתרון.

תרגיל 1.7

פתרו את בעיה 3.10 בספר הלימוד. הסיקו ששפת הפולינומים במשתנה אחד שלהם יש שורש שלם היא כריעה.

קראו בספר הלימוד את התת-סעיף "מינוח לתיאור מכונות טיורינג".

תרגיל 1.8

פתרו את חלק a של תרגיל 3.8 בספר הלימוד.

תרגיל 1.9

א. הוכיחו שכל שפה סופית (כלומר, שפה שיש בה מספר סופי של מילים) היא כריעה-טיורינג.
ב. פתרו את בעיה 3.9 בספר הלימוד.

תרגיל 1.10

פתרו את בעיה 3.13 בספר הלימוד.

תרגיל 1.11

פתרו את החלקים b,a של בעיה 3.15 בספר הלימוד.

תרגיל 1.12

פתרו את החלקים e,d,a של בעיה 3.16 בספר הלימוד.

תרגיל 1.13

פתרו את בעיה 3.17 בספר הלימוד.

פתרון התרגילים

תרגיל 1.1

א. $11011q'001$

ב. $q'10001$

ג. אין מספיק נתונים לענות על השאלה.

ד. $000010q'$

תרגיל 1.2

א. $q_1 0 \rightarrow \sqcup\, q_2 \ \sqcup \rightarrow \sqcup \sqcup q_{\text{accept}}$

ב. $q_1 00 \rightarrow \sqcup\, q_2 0 \rightarrow \sqcup \text{x} q_3 \sqcup \rightarrow \sqcup\, q_5 \text{x} \sqcup \rightarrow q_5 \sqcup \text{x} \sqcup \rightarrow \sqcup\, q_2 \text{x} \sqcup \rightarrow \sqcup \text{x} q_2 \sqcup \rightarrow$

$\rightarrow \sqcup \text{x} \sqcup q_{\text{accept}}$

ג. $q_1 000 \rightarrow \sqcup\, q_2 00 \rightarrow \sqcup \text{x} q_3 0 \rightarrow \sqcup \text{x} 0 q_4 \sqcup \rightarrow \sqcup \text{x} 0 \sqcup q_{\text{reject}}$

תרגיל 1.3

א. $q_1 11 \rightarrow \text{x} q_3 1 \rightarrow \text{x} 1 q_3 \sqcup \rightarrow \text{x} 1 \sqcup q_{\text{reject}}$

ב.

$q_1 1\#1 \rightarrow \text{x} q_3 \#1 \rightarrow \text{x}\# q_5 1 \rightarrow \text{x} q_6 \#\text{x} \rightarrow q_7 \text{x}\#\text{x} \rightarrow \text{x} q_1 \#\text{x} \rightarrow \text{x}\# q_8 \text{x} \rightarrow \text{x}\#\text{x} q_8 \sqcup \rightarrow \text{x}\#\text{x} \sqcup q_{\text{accept}}$

תרגיל 1.4

- בהתחלה: aaa bbbb cccccccccccc
- נקודת תצפית 1: $\sqcup$aa bbbb cccccccccccc
- נקודת תצפית 2: $\sqcup$aa xxxx xxxxcccccccc
- נקודת תצפית 3: $\sqcup$aa bbbb xxxxcccccccc
- נקודת תצפית 1: $\sqcup\sqcup$a bbbb xxxxcccccccc
- נקודת תצפית 2: $\sqcup\sqcup$a xxxx xxxxxxxxcccc
- נקודת תצפית 3: $\sqcup\sqcup$a bbbb xxxxxxxxcccc
- נקודת תצפית 1: $\sqcup\sqcup\sqcup$ bbbb xxxxxxxxcccc
- נקודת תצפית 2: $\sqcup\sqcup\sqcup$ xxxx xxxxxxxxxxxx
- נקודת תצפית 3: $\sqcup\sqcup\sqcup$ bbbb xxxxxxxxxxxx

הקלט במקרה זה מתקבל.

תרגיל 1.5

נניח שב- M יש הכלל $\delta(q,a)=(q',b,S)$. נגדיר מצב חדש $\hat{q}$ ואז נגדיר את שני כללי המעבר הבאים ב- M_1 : $\delta(q,a)=(\hat{q},b,R)$ ו- $\delta(\hat{q},\gamma)=(q',\gamma,L)$ לכל $\gamma\in\Gamma$.

תרגיל 1.6

הבנייה תהיה בדיוק כמו קודם, אלא שכעת נצטרך לבדוק במהלך מעבר על רמה מסוימת בעץ אם כל המסלולים בה הסתיימו (כלומר, אם הגענו בהם למצב q_{accept} או q_{reject} או שהם ענפים המתייחסים לתרחיש בחירות בלתי אפשרי). לצורך כך, כאשר נפתח בסריקה של רמה חדשה בעץ (למשל, הרמה החמישית בעץ הכוללת את כל b^5 הכתובות מ-11111 עד $bbbbb$), אז נאפס משתנה דגל מסוים (שאותו אנו יכולים לשמור בסרט השלישי בקטע שמימין לכתובת). אם במהלך המעבר על b^5 הכתובות ברמה זו נגיע לכתובת חוקית שבסופה הגענו למצב שאיננו q_{accept} או q_{reject}, אז נשנה את ערכו של משתנה הדגל מ-0 ל-1. בתום סריקת b^5 הכתובות ברמה זו, נבחן את ערכו של משתנה הדגל. אם הוא 1, סימן שעלינו להמשיך בעבודה. לצורך כך נאפס אותו ונעבור לרמה הבאה של כתובות (הרמה השישית בדוגמה לעיל). אם, לעומת זאת, משתנה הדגל הוא 0, הרי שכל הענפים בעץ שנבחנו היו בלתי חוקיים או שהסתיימו ב- q_{reject} (לא יתכן שאחד מהם הסתיים ב- q_{accept} כיוון שאז היינו עוצרים מיידית ומקבלים את הקלט). לכן בשלב זה אנו יכולים לעצור ולדחות את הקלט.

אם גובה העץ המתאים למכונה האי-דטרמיניסטית הוא h, אז המכונה הדטרמיניסטית המתוארת לעיל תעצור לכל המאוחר בתום הסריקה של הרמה ה- h. לכן מכונה זו תעצור תמיד; כלומר, היא מכריעה את השפה הנתונה.

תרגיל 1.7

אנו יודעים ששפת הפולינומים בעלי שורש שלם היא שפה מזוהה-טיורינג, כפי שתיארנו עבור המקרה הכללי של כמה משתנים. מכונת טיורינג המתאימה פשוט מנסה את כל המספרים השלמים ($0,1,-1,2,-2,\cdots$) ומציבה אותם בפולינום, והיא עוצרת במצב המקבל אם באחד מהחישובים התקבל הערך אפס. הבעיה הייתה שמכונה כזו לא תעצור אם אין לפולינום שורש.

אך במקרה של פולינום במשתנה אחד, $P(x)=\sum_{i=0}^{k}c_i x^i$, קל לראות שאם לא התגלו שורשים עד לערך $|x|=\dfrac{k\cdot c_{\max}}{|c_k|}$, כאשר $c_{\max}=\max_{0\le i\le k}|c_i|$, אז אין טעם להמשיך בחיפוש ואפשר לעצור ולדחות את הפולינום. אכן, עבור $|x|>\dfrac{k\cdot c_{\max}}{|c_k|}$ מתקיים:

$$\left|\sum_{i=0}^{k-1} c_i x^i\right| \le \sum_{i=0}^{k-1} |c_i|\cdot|x|^i \le c_{\max}\cdot\sum_{i=0}^{k-1}|x|^i \le c_{\max}\cdot k\cdot|x|^{k-1} < |c_k|\cdot|x|^k$$

במעבר הראשון הסתמכנו על אי-שוויון המשולש $|a+b|\le|a|+|b|$. במעבר הלפני אחרון הסתמכנו על כך ש:

$$|x| > \frac{k\cdot c_{\max}}{|c_k|} \ge k \ge 1$$

ולכן $|x|^i \le |x|^{k-1}$ לכל $0\le i\le k-1$. במעבר האחרון הסתמכנו על כך ש:

$$|x| > \frac{k\cdot c_{\max}}{|c_k|}.$$

לכן, לפי האמור לעיל, ולפי אי-שוויון המשולש $|a+b|\ge|a|-|b|$, מתקיים בטווח זה של ערכי x:

$$|P(x)| = \left|\sum_{i=0}^{k} c_i x^i\right| \ge \left|c_k x^k\right| - \left|\sum_{i=0}^{k-1} c_i x^i\right| > 0$$

כלומר, בטווח זה מתקיים $P(x)\ne 0$.

תרגיל 1.8

פתרון תרגיל זה מופיע בספר הלימוד.

תרגיל 1.9

א. אם השפה סופית, נניח בת k מילים, נבנה מכונת טיורינג בת $k+1$ סרטים המכריעה אותה:

הקלט הוא מחרוזת מעל Σ. קלט זה יאוחסן על הסרט הראשון שיזוהה כסרט מספר 0.

1. לכל $i=1,\ldots,k$ כתוב על הסרט ה-i את המילה ה-i בשפה.
2. לכל $i=1,\ldots,k$ השווה את תוכן סרט 0 לתוכן סרט i. אם יש שוויון, קבל את הקלט.
3. דחה את הקלט.

ב. פתרון בעיה זו מופיע בספר הלימוד.

תרגיל 1.10

אם A היא שפה כריעה, אז המונה המתאים ייצר את כל המחרוזות ב-Σ^* לפי הסדר הסטנדרטי, יפעיל את מכונת טיורינג המכריעה את A על כל אחת מהמחרוזות האלו, וידפיס רק את המחרוזות שנמצאו שייכות ל-A.

מצד שני, נניח שקיים מונה המדפיס את כל המילים בשפה A בסדר הסטנדרטי. ללא הגבלת הכלליות, נניח ש-A אינסופית (כיוון שאם A סופית, אז היא בוודאי כריעה. ראו התרגיל הקודם). נתאר מכונת טיורינג המכריעה את A: בהינתן קלט $w\in\Sigma^*$, המכונה תריץ את המונה עד

שתודפס בפעם הראשונה מילה $v \geq w$ (סימן הסדר כאן הוא הסדר הסטנדרטי בין מחרוזות). אם $v = w$, המכונה תקבל את w, אחרת (כלומר $v > w$), היא תדחה אותו. שימו לב שמובטח לנו שנגיע למצב עצירה זה מכיוון שהנחנו שהשפה היא אינסופית.

תרגיל 1.11

פתרון חלק a של בעיה 3.15 מופיע בספר הלימוד.

פתרון חלק b: נניח ש- L_1, L_2 הן שתי שפות מזוהות-טיורינג ו- M_1, M_2 הן מכונות טיורינג המזהות אותן. נבנה מכונת טיורינג אי-דטרמיניסטית M' המזהה את השרשור של שתי השפות הללו:

$$L_{1,2} = \{uv : u \in L_1, v \in L_2\}$$

בהינתן קלט w, המכונה תחתוך אותו באופן שרירותי לשתי מחרוזות $w = w_1 w_2$ (זהו האי-דטרנימיניזם בפעולת המכונה הזו. איננו מכתיבים למכונה היכן לשבור את מחרוזת הקלט לשתיים). אחר כך, תריץ המכונה M' את M_1 על w_1 ואת M_2 על w_2. M' תקבל את w רק אם שתי הרצות אלו יסתיימו במצבים המקבלים של המכונות המקוריות. לבסוף קיימת מכונת טיורינג **דטרמיניסטית** המזהה את השפה לפי משפט 3.16.

תרגיל 1.12

פתרון חלק a של בעיה 3.16 מופיע בספר הלימוד.

פתרון חלק d: אם L היא שפה כריעה ו- M היא מכונה המכריעה אותה, נבנה מכונה M' הזהה באופן פעולתה ל- M, אלא שבכל פעם ש- M מקבלת קלט, M' תדחה אותו, ולהפך – אם M דוחה, אז M' תקבל. קל לראות ש-M' מכריעה את השפה המשלימה של L.

פתרון חלק e: נניח ש- L_1, L_2 הן שתי שפות כריעות ו- M_1, M_2 הן מכונות טיורינג המכריעות אותן. נבנה מכונת טיורינג M' המכריעה את החיתוך של שתי השפות הללו. בהינתן קלט w, תריץ המכונה M' גם את M_1 וגם את M_2 על w. אם שתי המכונות מקבלות את w, אז גם M' תקבל אותו, אחרת (כלומר, לפחות אחת משתי המכונות דחתה קלט זה), היא תדחה אותו.

תרגיל 1.13

פתרון בעיה זו מופיע בספר הלימוד.

2. כריעוּת (פרק 4 בספר הלימוד)

לאחר שבפרק הקודם הזכרנו את המושג אלגוריתם במהלך הדיון במכונות טיורינג, בפרק זה נחקור את המגבלות של החישוב האלגוריתמי. נגלה שיש בעיות שהן מעבר ליכולת החישוב של המודל החישובי המקובל – מכונת טיורינג שעומדים לרשותה סרט וזמן ריצה בלתי מוגבלים.

קראו את המבוא לפרק 4 בספר הלימוד.

2.1 שפות כריעוּת

קראו את התת-סעיף "בעיות כריעוּת העוסקות בשפות רגולריות" בסעיף 4.1 בספר הלימוד.

בתת-סעיף זה אנו דנים בבעיות הכרעה העוסקות באוטומטים סופיים (דהיינו, אוטומטים בעלי מספר סופי של מצבים ואלפבית סופי) ובשפות שהם מזהים. בבעיות הכרעה אנו נדרשים לענות "כן" או "לא". אנו מעדיפים לדבר במונחים של **שפות** במקום במונחים של **בעיות**. בהינתן שפה L (כלומר, תת-קבוצה של Σ^* עבור אלפבית נתון כלשהו Σ), אנו מעוניינים באלגוריתם (מכונת טיורינג) המכריע עבור כל מחרוזת ב-Σ^*, האם היא שייכת ל-L או לא.

אם אתם תוהים מדוע אנו מתמקדים בבעיות הכרעה דווקא, הסיבה היא שבעיות הכרעה הן במובן מסוים הבעיות הפשוטות ביותר. למשל, בעיית ההכרעה "בהינתן אוטומט, האם השפה שהוא מקבל ריקה?", פשוטה יותר מהבעיה המקבילה "בהינתן אוטומט, מצא מחרוזת שהוא מקבל, או קבע שאין מחרוזת כזו", מכיוון שהבעיה השנייה דורשת כתשובה יותר מידע מן הבעיה הראשונה. בפרק הנוכחי אנו רוצים להראות שקיימות **מגבלות** על יכולת החישוב של מכונת טיורינג. נתרכז בבעיות החישוב הפשוטות ביותר, דהיינו, בעיות הכרעה, ונראה שגם ביניהן יש בעיות הנמצאות מעבר ליכולת החישוב של מכונות טיורינג.

נראה לדוגמה כיצד ניתן לתרגם את הבעיה "האם אוטומט מסוים מקבל קלט מסוים?" לבעיה מתאימה של שייכות לשפה. אנו יכולים להסכים על שיטה מסוימת לתאר אוטומט על ידי מחרוזות של תווים. תיאור אוטומט דורש חמישה מרכיבים:

1. קבוצת המצבים Q;
2. האלפבית Σ (ניתן פשוט לרשום את כל אותיות האלפבית);
3. פונקציית המעברים δ (על ידי פירוט ערכי הפונקציה לכל צירוף אפשרי של מצב ותו קלט);
4. מצב ההתחלה;
5. קבוצת המצבים המקבלים.

אם כן, ניתן לתאר אוטומט באמצעות מחרוזת מעל אלפבית הכולל סמלים לייצוג מספרים טבעיים (למספור המצבים), את Σ, ומספר סימני פיסוק (כדי להפריד בין חמשת המרכיבים לעיל, או בין ערכים שונים בתוך מרכיבים אלו). כעת, בהינתן אוטומט B וקלט w עבור אוטומט זה, ניתן להרכיב מחרוזת $\langle B,w\rangle$ שחלקה הראשון הוא הייצוג של האוטומט וחלקה השני הוא הקלט הנדון. השפה $A_{\text{DFA}} = \{\langle B,w\rangle\}$ מכילה את כל הייצוגים דלעיל שבהם B הוא אוטומט ו-w הוא קלט שהאוטומט הזה מקבל. כעת, הבעיה האם B מקבל את w שקולה לשאלה האם המחרוזת $\langle B,w\rangle$ שייכת לשפה A_{DFA}.

בהוכחת משפט 4.1 מתוארת מכונת טיורינג המכריעה שפה זו: ראשית, המכונה בודקת שלפניה קלט חוקי. אם זה אכן המצב, היא מדמה את פעולת האוטומט על הקלט הנתון, ובודקת אם האוטומט מקבל את הקלט הזה או לא. אם כן, המכונה מקבלת את הקלט $\langle B,w\rangle$, אחרת – היא דוחה אותו.

באופן דומה, מוכיחים שהגרסה האי-דטרמיניסטית של בעיה זו כריעה גם היא (משפט 4.2). דהיינו, שפת כל המחרוזות $\langle B,w\rangle$, שבהן B הוא ייצוג של אוטומט אי-דטרמיניסטי ו-w הוא קלט שאוטומט זה מקבל, אף היא כריעה. טענה זו קלה להוכחה מכיוון שמכונת טיורינג המתאימה יכולה תחילה לתרגם את האוטומט האי-דטרמיניסטי הנתון לאוטומט דטרמיניסטי שקול, ואז, כמו קודם, להריץ אוטומט זה על הקלט הנתון ולבדוק אם הוא מתקבל או לא.
במשפט 4.3 מוכחת טענה דומה לגבי ביטויים רגולריים.

בהמשך מופיעות שתי טענות בעלות אופי מעט שונה. בבעיית בדיקת הריקנות עלינו להכריע אם השפה המזוהה על ידי אוטומט נתון היא ריקה או לא. במשפט 4.4 מראים שבעיית בדיקת הריקנות כריעה. טענה זו מאפשרת לנו להמשיך ולהראות (משפט 4.5) שבעיית שקילות האוטומטים כריעה גם כן (שני אוטומטים הם שקולים אם הם מזהים את אותה השפה). נציין שהאלגוריתמים לשתי הבעיות הללו מוצגים ביחידה 4 של הקורס "אוטומטים ושפות פורמליות".

תרגיל 2.1

פתרו את תרגיל 4.1 בספר הלימוד.

תרגיל 2.2

פתרו את תרגיל 4.3 בספר הלימוד.

תרגיל 2.3

פתרו את בעיה 4.26 בספר הלימוד.

תרגיל 2.4

פתרו את בעיה 4.29 בספר הלימוד.

תרגיל 2.5

פתרו את בעיה 4.30 בספר הלימוד.

המשיכו לקרוא בספר הלימוד את התת-סעיף "בעיות כריעוּת העוסקות בשפות חסרות הקשר".

בתת-סעיף זה נערך דיון דומה בנוגע לשפות חסרות הקשר. הבה נתחיל בתזכורת קצרה מהו דקדוק חסר הקשר (CFG), מהי שפה חסרת הקשר (CFL), ומהו אוטומטי מחסנית (PDA). **דקדוק חסר הקשר** הוא קבוצה סופית של משתנים V, אלפבית סופי של אותיות הקרויות **טרמינלים** Σ, ואוסף כללי המרה מהצורה $A \to u_1u_2\cdots u_k \in (V \cup \Sigma)^*$ (A הוא משתנה של הדקדוק). אחד מן המשתנים $S \in V$ הוא משתנה ההתחלה, והשפה הנוצרת על ידי דקדוק כזה היא האוסף ב-Σ^* של כל מחרוזות הטרמינלים שניתנות לקבלה על ידי כללי ההמרה האלו אם מתחילים במשתנה ההתחלה. שפה כזו נקראת **שפה חסרת הקשר**. למשל, הדקדוק G מעל $\Sigma = \{0,\#,1\}$, בעל שני כללי ההמרה

$$S \to 0S1$$
$$S \to \#$$

יוצר את השפה $L(G) = \{0^n\#1^n : n \geq 0\}$. בהינתן $w \in L(G)$, רשימה סדורה של כללי ההמרה שיש להפעיל על מנת לקבל את w נקראת **גזירה** של w.

לכל דקדוק קיים דקדוק שקול בצורה הנורמלית של חומסקי. בצורה נורמלית זו, כל כללי ההמרה הם מהצורה $A \to BC$ כאשר $A,B,C \in V$, $B,C \neq S$ (כלומר, משתנה מומר בשני משתנים שאינם משתנה ההתחלה), או מהצורה $A \to a$ כאשר $A \in V$, $a \in \Sigma$ (כלומר, משתנה מומר בטרמינל יחיד), או כלל ההמרה $S \to \varepsilon$. למשל, דקדוק בצורה הנורמלית של חומסקי השקול לדקדוק לעיל הוא הדקדוק

$$S \to UB,\ \ B \to AV,\ \ A \to UB$$
$$S \to \#,\ \ A \to \#,\ \ U \to 0,\ \ V \to 1$$

מודל חישובי השקול לדקדוק חסר הקשר הוא **אוטומט מחסנית**. אוטומט מחסנית הוא אוטומט אי-דטרמיניסטי שלרשותו עומדת מחסנית בעלת עומק בלתי מוגבל. לכל שפה חסרת הקשר קיים אוטומט מחסנית המזהה אותה, ולהפך – השפה המזוהה על ידי אוטומט מחסנית היא תמיד חסרת הקשר.

כדי להיזכר בנושאים של שפות חסרות הקשר ביתר פירוט, אתם יכולים לעיין בפרקים המתאימים בקורס "אוטומטים ושפות פורמיות".

תרגיל 2.6

הראו שאם G הוא דקדוק חסר הקשר בצורה הנורמלית של חומסקי ו- $w \in L(G)$ היא מילה באורך $n > 0$, אז בכל גזירה של w יש בדיוק $2n-1$ צעדים.

שימו לב שניתן לייצג גם דקדוק חסר הקשר באמצעות מחרוזת על ידי רישום כל הכללים בדקדוק זה אחר זה. במשפט 4.7 מוכיחים ששפת כל הזוגות $\langle G, w \rangle$, כאשר G הוא דקדוק חסר הקשר ו- w היא מילה הנוצרת על ידי דקדוק זה, היא שפה הניתנת להכרעה. מכונת טיורינג המתאימה מבצעת את הפעולות הבאות לצורך ההכרעה הנחוצה:

1. הסבת הדקדוק הנתון לדקדוק שקול בצורה הנורמלית של חומסקי. יש פרוצדורה פשוטה לביצוע הסבה כזו, שבמהלכה מוחלף כל כלל המרה שאיננו בצורה הנורמלית של חומסקי בכלל או בכללים המתאימים לצורה זו.
2. רישום כל הגזירות האפשריות בדקדוק חומסקי זה שמכילות $2n-1$ צעדים, כאשר n הוא אורך המילה w. אם $w = \varepsilon$, כלומר $n = 0$, המכונה תרשום את כל הגזירות שמכילות צעד אחד.
3. בדיקת כל אחת מהגזירות ברשימה. אם אחת מהן נותנת את w, המכונה עוצרת ומקבלת את הקלט, אחרת - היא דוחה אותו.

בהמשך, במשפט 4.8, נעשה שימוש בטכניקת הסימון שפגשנו בעבר, כדי להראות שבעיית בדיקת הריקנות של דקדוקים חסרי הקשר ניתנת להכרעה. ניתן היה לטעות ולחשוב שכריעות בעיית בדיקת הריקנות מובילה לכך שגם בעיית השקילות בין שני דקדוקים חסרי הקשר היא כריעה, אך מסתבר שאין זה נכון. במקרה זה אי אפשר להפעיל אלגוריתם דומה לזה שבהוכחת משפט 4.5. הסיבה לכך היא שאוסף השפות חסרות ההקשר איננו סגור תחת פעולת החיתוך או פעולת המשלים (בניגוד לאוסף השפות הרגולריות).

לבסוף, במשפט 4.9 מוכיחים שכל שפה חסרת הקשר היא כריעה. ההוכחה פשוטה: נניח שהשפה היא L ועלינו להכריע אם $w \in L$. מכונת טיורינג המתאימה תכיל עותק של דקדוק G היוצר את L (כלומר, תהיה במכונה קבוצת מצבים שתכתוב את התיאור של G על הסרט), ואז היא תפעיל את המכונה S ממשפט 4.7 על הקלט $\langle G, w \rangle$. שימו לב כי מכונת טיורינג זו תמיד עוצרת (מדוע?). אם קלט זה יתקבל על ידי S, אז המכונה שלנו תקבל את הקלט w כמילה בשפה. אחרת – היא תדחה אותו.

משפט זה מספק את הקשר ההיררכי האחרון בין שפות רגולריות, שפות חסרות הקשר, שפות כריעות ושפות הניתנות לזיהוי-טיורינג, כפי שמתואר באיור 4.10. שימו לב כי ההכלות המתוארות באיור הן הכלות במובן החזק. למשל, משפחת השפות חסרות-ההקשר היא תת-קבוצה **ממש** של משפחת השפות הכריעות (למה?). העובדה כי מדובר בהכלות חזקות הוכחה כבר עבור כל ההכלות המתוארות באיור, למעט ההכלה האחרונה. בהמשך פרק זה נוכיח שמשפחת השפות הכריעות מוכלת **ממש** במשפחת השפות המזוהות-טיורינג.

תרגיל 2.7

פתרו את תרגיל 4.4 בספר הלימוד.

תרגיל 2.8

פתרו את בעיה 4.18 בספר הלימוד.

2.2 אי-כריעות

קראו את המבוא לסעיף 4.2 בספר הלימוד עד לפני התת-סעיף "שיטת האלכסון".

כאן אנו נתקלים לראשונה בשפה שאיננה כריעה. השפה A_{TM} המורכבת מכל הזוגות $\langle M,w\rangle$, כאשר M היא מכונת טיורינג ו- w היא מילה ש- M מקבלת, איננה ניתנת להכרעה.
(אנו מקודדים את M בעזרת מחרוזת בצורה דומה מאוד לצורה שקודדנו קודם לכן אוטומט סופי.)

שימו לב כי שפה זו ניתנת לזיהוי על ידי מכונת טיורינג אוניברסלית U. מכונת טיורינג אוניברסלית U מקבלת כקלט זוג $\langle M,w\rangle$, ומדמה את פעולת M על w. בכך דומה מכונת טיורינג אוניברסלית למחשב, המסוגל להריץ תכניות שונות. U תעצור ותקבל את הקלט $\langle M,w\rangle$ אם M עוצרת על הקלט w ומקבלת אותו. אך אם M איננה עוצרת על w, אז פעולת ההדמיה של M המבוצעת על ידי המכונה U לא תעצור אף היא, וזו בדיוק הצרה: אין אפשרות לבנות מכונת טיורינג שתוכל לקבוע אם מכונה נתונה M עוצרת על קלט נתון w. מכל מקום, השפה A_{TM} היא דוגמה לשפה לא כריעה הניתנת לזיהוי טיורינג (כלומר, שפה זו נמצאת באיור 4.10 באליפסה החיצונית ביותר המציינת את אוסף השפות הניתנות לזיהוי טיורינג, אך היא נמצאת מחוץ לאליפסה השנייה בגודלה המציינת את אוסף השפות הכריעות).
הדרך שבה מוכיחים את אי-כריעותה של A_{TM} מבוססת על **שיטת האלכסון**, שפותחה על ידי המתמטיקאי **גאורג קנטור** במחצית השנייה של המאה ה-19 על מנת להוכיח שקבוצת המספרים הממשיים "גדולה יותר" מהתת-קבוצה שלה המורכבת מכל המספרים הטבעיים (במובן שאין שום פונקציה חד-חד-ערכית מקבוצת המספרים הממשיים לקבוצת המספרים הטבעיים).

קראו בספר הלימוד את התת-סעיף "שיטת האלכסון".

(הערה: בספר הלימוד נקראת השיטה The Diagonalization Method ולא במונח שאנו משתמשים The Diagonal Method. שני השמות מקובלים בספרות, אך אנו בחרנו במדריך זה בשם "שיטת האלכסון" ולא "שיטת הלְכסון" מכיוון שהוא מדויק יותר.)

אם כן, כאשר מדובר בקבוצות אינסופיות, עלינו להיות מוכנים לתוצאות מפתיעות. למשל, שקבוצת המספרים הטבעיים שקולה בגודלה לתת-קבוצה של המספרים הזוגיים! האינטואיציה הראשונית אומרת שקבוצת המספרים הזוגיים מהווה "מחצית" מקבוצת המספרים הטבעיים. אך ההתאמה החד-חד-ערכית ועל בין שתי הקבוצות מסירה כל ספק: אלו הן שתי קבוצות השקולות בגודלן[1]. דבר דומה קורה עם קבוצת המספרים הרציונליים שנראית, במבט ראשון, גדולה לאין שיעור מהתת-קבוצה של המספרים הטבעיים. אך, שוב, התאמה חד-חד-ערכית ועל מתאימה מראה שניתן "למנות" (to count) את המספרים הרציונליים, במובן זה שניתן לסדרם ברשימה אינסופית, כך שכל אחד מהם מותאם למספר הטבעי המתאר את מקומו ברשימה (למשל, בהתאמה המתוארת באיור 4.16 המספר $\frac{2}{3}$ מופיע שמיני ברשימה ולכן הוא מותאם למספר הטבעי 8).

קבוצות סופיות וקבוצות אינסופיות השקולות בגודלן לקבוצת המספרים הטבעיים $\mathbb{N}$ נקראות **קבוצות בנות-מנייה** (countable). לא כל הקבוצות האינסופיות הן בנות-מנייה. קבוצת המספרים הממשיים, למשל, גדולה יותר מקבוצת המספרים הטבעיים. כלומר, אין אפשרות למנות את המספרים הממשיים (דהיינו, לסדר אותם ברשימה). בהוכחת משפט 4.17 אנו פוגשים את הטיעון המבריק המראה זאת, תוך שימוש בשיטת האלכסון של קנטור. אנו מניחים בשלילה שקיימת רשימה מסודרת המכילה את **כל** המספרים הממשיים. אז, על ידי הסתכלות בספרות המופיעות באלכסון של רשימה זו (כלומר, בספרה ה- k אחרי הנקודה העשרונית של המספר הממשי ה- k ברשימה, כאשר $k=1,2,3,\cdots$), אנו בונים מספר ממשי x שבוודאות אינו מופיע ברשימה זו; x זה אינו מופיע ברשימה מכיוון שלכל מספר טבעי $k=1,2,3,\cdots$ הוא שונה מן המספר הממשי ה- k ברשימה, כפי שנובע מהאופן שבו נבחרה הספרה ה- k אחרי הנקודה העשרונית ב- x.

תרגיל 2.9

פתרו את תרגיל 4.7 בספר הלימוד.

תרגיל 2.10

פתרו את תרגיל 4.8 בספר הלימוד.

במסקנה 4.18 מוכיחים, בכלים דומים, שיש שפות שאינן ניתנות לזיהוי טיורינג. עקרון ההוכחה פשוט ואלגנטי:

- קבוצת מכונות הטיורינג היא בת-מנייה, כיוון שכל מכונת טיורינג ניתנת לתיאור על ידי מחרוזת סופית מעל אלפבית סופי כלשהו Σ, והקבוצה Σ^* (אוסף כל המחרוזות הסופיות מעל Σ) היא בת-מנייה: ניתן לרשום תחילה את כל המחרוזות באורך 1, אח"כ את כל אלו באורך 2, וכך הלאה.

[1] נציין שכאשר מדברים על קבוצות אינסופיות משתמשים במושג "עוצמה" (cardinality או power) ולא "גודל".

- קבוצת כל השפות מעל אלפבית סופי כלשהו Σ, היא למעשה קבוצת כל התת-קבוצות של Σ^*, או במילים אחרות, **קבוצת החזקה** של Σ^* המסומנת $P(\Sigma^*)$ או $2^{(\Sigma^*)}$. ידוע שאם קבוצה היא בת-מנייה, אז קבוצת החזקה שלה איננה בת מנייה[2]. שימו לב ששפה מעל Σ היא פשוט קבוצה חלקית של Σ^*. לכן, קבוצת כל השפות מעל Σ היא קבוצת החזקה של Σ^*, דהיינו $P(\Sigma^*)$. כיוון ש-Σ^* היא בת-מנייה, ניתן לסדר ברשימה את כל אבריה, $\Sigma^* = \{s_1, s_2, s_3, \ldots\}$ ואז לזהות כל שפה עם הסדרה האופיינית שלה שהיא הסדרה הבינארית המסמנת בביט 1 כל מחרוזת מ-Σ^* השייכת לשפה ובביט 0 כל מחרוזת שאינה שייכת לשפה. לפי תרגיל 2.9 לעיל, קבוצת הסדרות הבינאריות באורך אינסופי איננה בת-מנייה. לכן גם קבוצת כל השפות מעל Σ איננה בת-מנייה.
- לאור האמור לעיל, חייבות להיות שפות שאין להן מכונת טיורינג מתאימה המזהה אותן; פשוט, יש הרבה יותר שפות מאשר מכונות טיורינג, וכל מכונת טיורינג מזהה שפה אחת בלבד.

קראו בספר הלימוד את התת-סעיף "שפה לא כריעה".

רעיון ההוכחה של אי-הכריעות של A_{TM} עלול להיות מעט מבלבל, אך ההסבר בספר מצוין, ואיורים 4.19, 4.20 ו-4.21 מבארים היטב את פעולת המכונות השונות המופיעות בהוכחה. הבה נחזור בקצרה על רעיון ההוכחה:

נניח שקיימת מכונת טיורינג H המכריעה את A_{TM}. כלומר H מקבלת את $\langle M, w\rangle$ אם M מקבלת את w ודוחה את $\langle M, w\rangle$ בכל שאר המקרים (כלומר, אם M דוחה את w או לא עוצרת עליה). אנו משתמשים ב-H לבניית מכונת טיורינג D המקבלת כקלט ארגומנט אחד, שהוא תיאור של מכונת טיורינג $\langle M\rangle$ (בניגוד ל-H שקיבלה שני ארגומנטים, $\langle M, w\rangle$). המכונה הזו מוציאה בדיוק את הפלט ההפוך מזה ש-H הייתה מוציאה על הקלט $\langle M, \langle M\rangle\rangle$.

הסתירה צצה כאשר אנו מנסים לברר מה הפלט שתוציא D כאשר נריץ אותה על התיאור שלה עצמה, $\langle D\rangle$:

- אם D מקבלת את $\langle D\rangle$, אז פירוש הדבר ש-H דחתה את $\langle D, \langle D\rangle\rangle$, כלומר ש-$D$ איננה מקבלת את $\langle D\rangle$!

---

[2] למעשה, לכל קבוצה A, העוצמה של קבוצת החזקה של A גדולה ממש מזו של עוצמת הקבוצה A. בפרט, אם A אינסופית בת-מנייה (כלומר, שקולה בעוצמתה לקבוצת המספרים הטבעיים), אז קבוצת החזקה שלה היא בעלת עוצמת הרצף, דהיינו, עוצמת קבוצת המספרים הממשיים.

- אם, לעומת זאת, D דוחה את $\langle D\rangle$, אז פירוש הדבר ש- H קיבלה את $\langle D,\langle D\rangle\rangle$, כלומר ש- D מקבלת את $\langle D\rangle$!

על כן, לא יתכן שהמכונה H קיימת.

קראו בספר הלימוד את התת-סעיף "שפה שאיננה ניתנת לזיהוי-טיורינג".

השפה A_{TM} המורכבת מכל הזוגות $\langle M,w\rangle$, כאשר M הוא מכונת טיורינג ו- w היא מילה ש- M מקבלת, היא שפה שאיננה כריעה, כפי שראינו זה עתה. לעומת זאת, היא ניתנת לזיהוי. השפה המשלימה, $\overline{A_{\mathrm{TM}}}$, איננה יכולה להיות אף היא ניתנת לזיהוי טיורינג, כיוון שאילו כך היה הדבר, אז לפי משפט 4.22 היינו מסיקים ש- A_{TM} כריעה. זוהי, אם כן, דוגמה לאחת השפות שאת קיומן הוכחנו במסקנה 4.18. בעוד ההוכחה של מסקנה 4.18 הייתה מבוססת על שיקולי גודל בלבד, ולא היה בה כל רמז מיהן השפות שעשויות להיות לא ניתנות לזיהוי-טיורינג, אנו פוגשים כאן שפה מפורשת כזו (באיור 4.10 שפה זו תהיה מחוץ לכל האליפסות).

לגבי כל שפה L, אחת מארבע האפשרויות הבאות נכונה:

1. L ומשלימתה $\overline{L}$ ניתנות לזיהוי טיורינג.
2. L ניתנת לזיהוי טיורינג אך $\overline{L}$ איננה ניתנת לזיהוי טיורינג.
3. $\overline{L}$ ניתנת לזיהוי טיורינג אך L איננה ניתנת לזיהוי טיורינג.
4. L ומשלימתה $\overline{L}$ אינן ניתנות לזיהוי טיורינג.

בכל השפות הכריעות (ורק הן) מתקיימת האפשרות הראשונה. השפה A_{TM} היא דוגמה לשפה שבה מתקיימת האפשרות השנייה, בעוד משלימתה $\overline{A_{\mathrm{TM}}}$ היא דוגמה לשפה שבה מתקיימת האפשרות השלישית. נשאלת השאלה האם המקרה הרביעי לעיל אפשרי. כלומר, האם קיימת שפה שגם היא וגם משלימתה אינן ניתנות לזיהוי טיורינג? בפרק הבא נראה שיש שפות כאלו.

תרגיל 2.11

פתרו את בעיה 4.12 בספר הלימוד.

תרגיל 2.12

פתרו את בעיה 4.17 בספר הלימוד.

תרגיל 2.13

פתרו את בעיה 4.19 בספר הלימוד.

תרגיל 2.14

פתרו את בעיה 4.22 בספר הלימוד.

תרגיל 2.15

פתרו את בעיה 4.24 בספר הלימוד. עליכם להוכיח ששפה C ניתנת לזיהוי טיורינג אם ורק אם קיימת שפה כריעה של זוגות D כך ש-C היא אוסף כל המילים x שעבורן קיימת מילה y כך שהזוג (x, y) שייך ל-D.

<u>הדרכה:</u> הראו תחילה שאם קיימת שפה כריעה D כזו, אז קיימת מכונת טיורינג שתזהה את C. הכיוון השני פחות פשוט: הניחו שקיימת מכונת טיורינג T המזהה את השפה C והשתמשו בה להגדרת שפה כריעה מתאימה D.

פתרון התרגילים

תרגיל 2.1

פתרון תרגיל 4.1 מופיע בספר הלימוד.

תרגיל 2.2

בהינתן אוטומט A נהפוך בו כל מצב מקבל למצב לא מקבל וכל מצב לא מקבל למצב מקבל. כך נקבל תיאור של אוטומט אחר A'. קל לראות שהשפות המזוהות על ידי A ו- A', $L(A)$ ו- $L(A')$ בהתאמה, משלימות זו את זו במובן ש- $L(A) \cup L(A') = \Sigma^*$ ו- $L(A) \cap L(A') = \varnothing$. לכן, $L(A) = \Sigma^*$ אם ורק אם $L(A') = \varnothing$. לפיכך, כל מה שנותר לעשות הוא להפעיל את מכונת טיורינג המכריעה את בעיית בדיקת הריקנות (משפט 4.4) על A'.

תרגיל 2.3

מכונת טיורינג הבאה מכריעה את השפה A.

"על הקלט $\langle R \rangle$ כאשר R ביטוי רגולרי:

1. בנה אוטומט סופי E המקבל את השפה $\Sigma^* 111 \Sigma^*$.
2. בנה אוטומט סופי B כך ש- $L(B) = L(R) \cap L(E)$.
3. הרץ את מכונת טיורינג T ממשפט 4.4 על הקלט $\langle B \rangle$ כדי להכריע אם $L(B) = \varnothing$.
4. אם T מקבלת, דחה. אם T דוחה, קבל."

תרגיל 2.4

$L(R) \subseteq L(S)$ אם ורק אם $\overline{L(S)} \cap L(R) = \varnothing$. מכונת טיורינג שלהלן מכריעה את השפה A:

"על הקלט $\langle R, S \rangle$ כאשר R ו- S הם ביטויים רגולריים:

1. בנה אוטומט סופי לא דטרמיניסטי B כך ש- $L(B) = L(R)$.
2. בנה אוטומט סופי לא דטרמיניסטי C כך ש- $L(C) = L(S)$.
3. בנה אוטומט סופי דטרמיניסטי D השקול ל- B.
4. בנה אוטומט סופי דטרמיניסטי E השקול ל- C.
5. בנה אוטומט סופי דטרמיניסטי המזהה את השפה $\overline{L(C)}$ (להזכירכם, $\overline{L(C)} = \overline{L(S)}$).
6. בנה אוטומט סופי דטרמיניסטי F המזהה את השפה $\overline{L(C)} \cap L(B)$.
7. בדוק האם $L(F) = \varnothing$ בעזרת המכונה T ממשפט 4.4.
8. אם T קיבלה, קבל. אם T דחתה, דחה."

תרגיל 2.5

פתרון בעיה 4.30 מופיע בספר הלימוד.

תרגיל 2.6

בכל גזירה יש שני סוגים של צעדים: צעדים שבהם מופעלים כללים מהצורה $A \to BC$ וכאלו שבהם מופעלים כללים מהצורה $A \to a$. ללא הגבלת הכלליות, אנו רשאים להניח שכל הכללים מהצורה השנייה מופעלים בסוף הגזירה (כלומר, בהתחלה מפעילים רק כללים שמשאירים אותנו עם משתנים בלבד, ורק בסוף עוברים על כל המשתנים והופכים אותם, זה אחר זה, לטרמינלים). כיוון שבסוף הגזירה אנו מקבלים מילה באורך n, אז מספר הצעדים מהצורה $A \to a$ חייב להיות n. לכן, מיד לפני הפעלת צעדים אלו יש לנו מחרוזת של n משתנים. כיוון שאנו מתחילים תמיד במשתנה אחד, וכל כלל מהצורה $A \to BC$ מגדיל את מספר המשתנים באחד, אז מספר הצעדים בגזירה שבהם מופעלים כללים מצורה זו הוא $n-1$. לפיכך, מספר הצעדים הכולל בגזירה הוא $2n-1$.

תרגיל 2.7

כל מה שצריך לעשות הוא להפעיל את המכונה S ממשפט 4.7 על הקלט $\langle G, \varepsilon \rangle$.

תרגיל 2.8

השפה: $U = \{<P> \mid P \text{ is a PDA that has useless states}\}$

מכונה T המכריעה את U:

"על קלט $<P>$ כאשר P הוא PDA:

1. לכל מצב q של P:
2. שנה את P כך ש-q יהיה המצב המקבל היחיד.
3. בנה CFG שקול ל-PDA שהתקבל.
4. בדוק, בעזרת המכונה R מהוכחת משפט 4.8, האם השפה שה-CFG השקול יוצר ריקה.
5. אם כן, קבל; אם לא, עבור למצב הבא.
6. אם הגענו לכאן, אז עברנו על כל המצבים, ואף אחד מהם לא נמצא חסר תועלת. לכן דחה."

תרגיל 2.9

נניח בשלילה שניתן לרשום את **כל** הסדרות הבינאריות האינסופיות ברשימה מסודרת. על מנת להגיע לסתירה, נבנה סדרה בינארית x שאיננה מופיעה ברשימה זו: הביט ה-k בסדרה x ייבחר כמשלים של הביט ה-k בסדרה ה-k ברשימה. באופן זה, הסדרה x שונה מכל אחת מן הסדרות ברשימה ועל כן הרשימה איננה שלמה, בניגוד להנחתנו.

תרגיל 2.10

תרגיל זה הוא מעין הכללה של הטענה שקבוצת המספרים הרציונליים $\mathbb{Q}$ היא בת-מנייה. ניתן להסתכל על קבוצת המספרים הרציונליים החיוביים כעל תת-קבוצה של $S = \{(i, j) : i, j \in \mathbb{N}\}$.

לכל מספר רציונלי קיים ייצוג יחיד כשבר פשוט מצומצם, $\frac{m}{n}$, כאשר m ו- n זרים ביניהם.

לפיכך, נזהה כל מספר רציונלי $\frac{m}{n}$ עם הזוג הסדור $(m,n) \in S$ המתאים. ההתאמה המתוארת באיור 4.16 מראה שהקבוצה S (ולכן גם קבוצת המספרים הרציונליים) היא בת-מנייה, על ידי סידור מתאים של אבריה.

הקבוצה T בתרגיל שלפנינו היא המקבילה התלת-ממדית של S. גם כאן ניתן להציע סידור דומה של אברי T שייתן התאמה חד-חד-ערכית ועל עם המספרים הטבעיים. נתאר את הסידור באופן מילולי ולא באופן ציורי:

ניתן לפרק את T באופן הבא: $T = \bigcup_{n=3}^{\infty} T_n$ כאשר $T_n = \{(i, j, k) : i, j, k \in N \text{ , } i + j + k = n\}$.

למשל, $(3,1,6) \in T_{10}$. כל T_n היא סופית (מדוע?). לכן אנו יכולים לרשום את האיבר היחיד ב- T_3, אח"כ את שלושת האיברים ב- T_4, אח"כ את ששת האיברים ב- T_5 וכך הלאה. כיוון שכל T_n היא סופית, מובטח לנו שכל אחד מן האיברים ב- T יופיע ברשימה זו.

תרגיל 2.11

נשתמש בשיטת האלכסון על מנת לבנות את השפה D. יהי $\langle M_1 \rangle, \langle M_2 \rangle, \cdots$ הפלט של מונה עבור השפה A. יהי $\Sigma^* = \{s_1, s_2, \cdots\}$ סידור של כל המחרוזות מעל האלפבית Σ. נסתכל כעת על מכונת טיורינג T הבאה: בהינתן קלט $w \in \Sigma^*$, היא תמצא את האינדקס i כך ש- $w = s_i$ ואז תריץ את M_i על w. אם M_i תקבל את w, אז T תדחה אותו, ולהפך.

אנו מבחינים בשני דברים: ראשית, המכונה T תמיד תעצור כיוון שכל אחת מהמכונות M_i עוצרת תמיד; לכן, יש שפה כריעה כלשהי, שנסמנה D, המוכרעת על ידי T. שנית, D איננה מוכרעת על ידי אף אחת מהמכונות M_i, כיוון שכל אחת מהמכונות הללו פועלת באופן שונה מאשר T. (לכל i, M_i פועלת באופן שונה מ- T על s_i).

תרגיל 2.12

פתרון בעיה 4.17 מופיע בספר הלימוד.

תרגיל 2.13

פתרון בעיה 4.19 מופיע בספר הלימוד.

תרגיל 2.14

נניח ש- A,B הן שתי שפות שחיתוכן ריק וש- $\bar{A}$ ו- $\bar{B}$ ניתנות לזיהוי טיורינג על ידי המכונות J ו- K בהתאמה. נגדיר כעת מכונת טיורינג T שהשפה שאותה היא מכריעה מפרידה בין A ל- B.
המכונה T תפעל כדלקמן על קלט נתון: היא תריץ עליו לסירוגין את שתי המכונות J ו- K (כלומר, כל צעד זוגי של T ידמה את הפעולה של J על הקלט, וכל צעד אי-זוגי ידמה את הפעולה של K). אם J תגיע ראשונה למצב מקבל, T תדחה את הקלט. אם K תגיע ראשונה למצב מקבל, T תקבל את הקלט.

המכונה T תעצור תמיד מכיוון ש- $\bar{A} \cup \bar{B} = \Sigma^*$, ולכן כל קלט שהוא חייב להיות ב- $\bar{A}$ או ב- $\bar{B}$, ולכן לפחות אחת מבין שתי המכונות J ו- K חייבת לקבל את הקלט בסופו של דבר. נסמן ב- C את השפה המוכרעת על ידי T. אז, $A \subseteq C$ מכיוון שאם $w \in A$, אז J לא תקבל את w ומכאן ש- K תקבל אותו, ולכן גם T. מצד שני, $B \subseteq \bar{C}$ מכיוון שאם $w \in B$, אז K לא תקבל את w ומכאן ש- J תקבל אותו, ולכן T תדחה אותו. לפיכך, השפה C מפרידה בין A ל- B.

תרגיל 2.15

<u>כיוון אחד</u>: נניח שקיימת שפה כריעה D, כך שלכל מילה x מ- C יש מילה y כך ש- $(x,y) \in D$.
נבנה מכונת טיורינג לא דטרמיניסטית T שתפעל באופן הבא: בהינתן ל- T קלט x, נבחר באופן לא דטרמיניסטי מילה y כלשהי ונשלח את $\langle x,y \rangle$ למכונה S המכריעה את D. אם S תקבל את $\langle x,y \rangle$, אז T תקבל את x. אחרת, T תדחה את x (במסלול הזה). אם x שייכת ל- C, אז יש y כך ש- S תקבל את $\langle x,y \rangle$. לכן יש ל- x מסלול מקבל ב- T. אם x לא שייכת ל- C, אז אין ל- x מסלול מקבל ב- T. לפיכך, T מזהה את C.

<u>כיוון שני</u>: נניח שקיימת מכונת טיורינג (דטרמיניסטית) T המזהה את השפה C. נגדיר את השפה D המורכבת מכל הזוגות $\langle x,y \rangle$, כך ש- T מקבלת את x אחרי $|y|$ צעדים לכל היותר. זוהי שפה כריעה, שכן ניתן לבנות מכונת טיורינג $\hat{T}$ שתריץ את T במשך $|y|$ צעדים על הקלט x; אם T קיבלה את x במהלך צעדים אלו, המכונה $\hat{T}$ תקבל את הקלט $\langle x,y \rangle$, אחרת - היא תדחה אותו. אם $x \in C$, אז T מקבלת את x אחרי מספר כלשהו של צעדים, ולכן $\langle x,y \rangle \in D$ לכל y שאורכו לפחות כמספר זה של צעדים. אם, לעומת זאת, $x \notin C$, אז T לא מקבלת את x, ולכן אין שום y כך ש- $\langle x,y \rangle \in D$.

3. רדוקציות (פרק 5 בספר הלימוד)

קראו את המבוא לפרק 5 בספר הלימוד.

לאחר שבפרק הקודם ראינו ש- A_{TM} איננה כריעה, נפגוש בפרק זה כלי חשוב הקרוי **רדוקציות,** שיאפשר לנו להוכיח את אי-כריעותן של בעיות נוספות.

בלשון לא פורמלית, בעיה A ניתנת לרדוקציה לבעיה B אם בהינתן פתרון ל- B ניתן להשתמש בו על מנת לקבל פתרון ל- A. כלומר, אם יש מכונת טיורינג המכריעה את B, ניתן להשתמש בה לצורך בניית מכונת טיורינג שתכריע את A. במקרה כזה אנו אומרים ש״בעיה A איננה קשה יותר מבעיה B״ או ש״בעיה B קשה לפחות כמו בעיה A״. בפרט, אם B כריעה, כך גם A. ומכאן שאם A איננה כריעה, גם B איננה כריעה.

3.1 בעיות לא כריעות מתורת השפות

קראו את סעיף 5.1 בספר הלימוד עד לפני התת-סעיף ״רדוקציות דרך היסטוריות חישוב״.

ההוכחה שניתנה בפרק הקודם לכך ש- A_{TM} איננה כריעה השתמשה ברעיון מתוחכם למדי. בהינתן שפה כלשהי B, קשה יהיה לתת הוכחה ישירה לכך ש- B איננה כריעה. אבל ניתן להוכיח את אי-כריעותה של B על ידי שימוש בידע הקודם שכבר רכשנו, דהיינו ש- A_{TM} איננה כריעה. ההוכחה מבוססת על רדוקציה וכוללת את השלבים האלה:

1. מניחים, על דרך השלילה, ש-B כריעה. מכאן נובע כי קיימת מכונת טיורינג, R, המכריעה את B.
2. מראים שמקיומה של R נובע קיום מכונת טיורינג, S, המכריעה את A_{TM}.
3. מכיוון ש- A_{TM} אינה כריעה, מכונה S כזו איננה יכולה להתקיים. מכאן אנו מסיקים שהנחת השלילה הייתה שגויה; משמע, B איננה כריעה.

זוהי הוכחה על דרך השלילה המתבססת על רדוקציה של A_{TM} ל-B.

שימו לב כי ההוכחה הנ״ל תקפה גם לגבי כל בעיה אחרת (במקום A_{TM}) שאנו כבר יודעים שהיא איננה כריעה.

בסעיף זה בספר מוכיחים את אי-כריעותן של ארבע בעיות נוספות על ידי רדוקציה מ- A_{TM}, או מבעיה בלתי כריעה אחרת, לכל אחת מהן.

במשפט 5.1 מראים שבעיית ההכרעה האם מכונת טיורינג נתונה עוצרת על קלט נתון איננה כריעה. בעיה זו מסומנת $HALT_{\text{TM}}$. אין לבלבל אותה עם A_{TM}. הקלטים לשתי הבעיות הם מהצורה $\langle M,w\rangle$. אך בעוד שב- A_{TM} עלינו להשיב האם המכונה M עוצרת על הקלט w ומקבלת אותו, ב- $HALT_{\text{TM}}$ עלינו להשיב רק האם המכונה M **עוצרת** על הקלט w (במצב המקבל או במצב הדוחה).

התוצאה הזו אינה מפתיעה לאור העובדה ש- A_{TM} איננה כריעה (משפט 4.11). ראינו ש- A_{TM} ניתנת לזיהוי טיורינג ותיארנו מכונה U, הקרויה מכונת טיורינג אוניברסלית, שמזהה אותה. מה שמנע מ- U להכריע את A_{TM} היו אותם קלטים $\langle M,w\rangle$ שעבורם המכונה M איננה עוצרת על הקלט w. קלטים כאלו יגרמו ל- U לרוץ עד אינסוף. לכן, אילו U הייתה מסוגלת לזהות מראש קלטים בעייתיים כאלו, היא הייתה מסוגלת להימנע מלולאות אינסופיות, ועל כן – להכריע את A_{TM} (שהרי אם M **עוצרת** בריצתה על w, אנחנו יכולים לקבוע אם M עצרה במצב המקבל או במצב הדוחה).

לפיכך, אילו הייתה בידינו מכונת טיורינג R המכריעה את הבעיה $HALT_{\text{TM}}$, היינו מסוגלים לבנות מכונת טיורינג S להכרעת הבעיה A_{TM} (זוהי רדוקציה מ- A_{TM} ל- $HALT_{\text{TM}}$). כיוון ש- A_{TM} איננה כריעה, הגענו לסתירה המיוחלת. (שימו לב ש- S תלויה בשני ארגומנטים: המכונה M והקלט הספציפי w.)

במשפט 5.2 מוכיחים שבעיית בדיקת הריקנות של מכונת טיורינג נתונה איננה כריעה. גם פה מראים רדוקציה מ- A_{TM} לבעיה שבה אנו עוסקים, דהיינו E_{TM}. נניח בשלילה שקיימת מכונת טיורינג R שמסוגלת להכריע האם השפה המזוהה על ידי מכונה נתונה M היא ריקה, כלומר האם $L(M)=\varnothing$. מכונה כזו תוכל לשמש לבניית מכונה אחרת S להכרעת A_{TM}. אם הקלט ל- S הוא $\langle M,w\rangle$, אז S תבנה מהתיאור של M ושל w תיאור של מכונה אחרת, M_1, הדוחה כל קלט שאיננו w, אך פועלת בדיוק כמו M אם הקלט הוא w. כעת, נפעיל את המכונה R על תיאור המכונה M_1. אם R תקבע ש- $L(M_1)=\varnothing$, הרי ש- M לא מקבלת את w. אחרת, אם $L(M_1)\neq\varnothing$, אז M מקבלת את w. כלומר, אם E_{TM} כריעה, כך גם A_{TM}.

שימו לב שהמכונה M_1 תלויה גם ב- M וגם ב- w. M_1 מתקבלת מ- M על-ידי הוספה של מצבים שבודקים האם הקלט x (של M_1) שווה ל- w. בדיקה זו מתבצעת בעזרת $|w|$ מצבים שבודקים האם האות הראשונה של x זהה לאות הראשונה של w, האם האות השנייה של x זהה לאות השנייה של w, וכך הלאה עד האות ה- $|w|$. מצב אחד נוסף נדרש כדי לבדוק שהמילה x הסתיימה לאחר $|w|$ האותיות הראשונות (שהיו זהות לאלה של w). כמה מצבים נוספים דרושים

להחזרת הראש הקורא-כותב לתחילת המילה ומעבר למצב ההתחלתי של M (כל זה במקרה שנמצא ש- $x = w$).

נדגיש: המכונה S בונה את המכונה M_1 (על סמך M ו- w), **אך היא אינה מריצה אותה**. הטענה היחידה על M_1 היא: אם $\langle M_1 \rangle$ שייכת ל- E_{TM} אז M לא מקבלת את w, ואם $\langle M_1 \rangle$ לא שייכת ל- E_{TM}, אז M כן מקבלת את w. לא מריצים את M_1 על אף מילה, רק בונים אותה ושולחים את התיאור שלה למכריע ההיפותטי של E_{TM} (המכונה R). לפי מה שיענה המכריע הזה נדע האם M מקבלת את w או לא.

משפט 5.3 עוסק בבעיה האם השפה שמכונת טיורינג נתונה מזהה היא שפה רגולרית. נניח שקיימת מכונת טיורינג R המכריעה את $REGULAR_{TM}$, ונבנה מכונה S שתכריע את A_{TM}. אם הקלט ל- S הוא $\langle M, w \rangle$, אז S תבצע את הפעולות הבאות:

1. היא תבנה תיאור של מכונת טיורינג חדשה M_2 שלגביה יתקיים:
   - אם M מקבלת את w, אז M_2 תקבל את השפה $\{0,1\}^*$.
   - אם M איננה מקבלת את w, אזי M_2 תקבל רק מחרוזות בשפה $\{0^n1^n : n \geq 0\}$ (כזכור, שפה זו **איננה** רגולרית).

   הרעיון מאחורי הבנייה פשוט: M_2 תמיד תקבל קלטים מהצורה $\{0^n1^n : n \geq 0\}$; היא תקבל את כל הקלטים האחרים אם ורק אם M מקבלת את w.
2. כעת, תריץ S את המכונה R על $\langle M_2 \rangle$. אם R תקבע ש- $L(M_2)$ רגולרית, אז $L(M_2) = \{0,1\}^*$; לפי הגדרת M_2, M מקבלת את w במקרה זה. אם, לעומת זאת, תקבע R ש- $L(M_2)$ איננה רגולרית, אז $L(M_2) = \{0^n1^n : n \geq 0\}$, ואז, כפי שנובע מהאופן שבו תכננו את M_2, M איננה מקבלת את w.

גם פה המכונה S בונה את המכונה M_2 (על סמך המכונה M והמילה w) **אך לא מריצה אותה**. את $\langle M_2 \rangle$ שולחים כקלט למכונה R שאמורה להכריע האם השפה ש- M_2 מקבלת רגולרית.

לבסוף, משפט 5.4 עוסק בבעיית השקילות של שתי מכונות טיורינג, EQ_{TM}. ההוכחה שבעיה זו איננה כריעה מתבצעת על ידי רדוקציה פשוטה מ- E_{TM} אליה. מכיוון שאנו יודעים כי E_{TM} איננה כריעה, אזי גם EQ_{TM} איננה כריעה.

במשפטים 5.2 ו-5.3 הראינו ששתי בעיות הנוגעות לתכונות של שפות המזוהות על ידי מכונות טיורינג אינן כריעות. בבעיה E_{TM} התכונה שנבדקה היא ריקנות השפה, כלומר, בהינתן מכונת טיורינג M האם $L(M)=\varnothing$; בבעיה $REGULAR_{\text{TM}}$ יש צורך להכריע האם $L(M)$ רגולרית.

תוצאות אלו הן מקרים פרטיים של משפט כללי יותר.

משפט רייס (Rice Theorem) : תהי Π תכונה אי-טריוויאלית של קידודים של מכונות טיורינג (דהיינו, Π היא תת-קבוצה של כל הקידודים החוקיים של מכונות טיורינג שאיננה הקבוצה הריקה מחד, ואיננה כוללת את כל הקידודים של מכונות טיורינג מאידך). נניח שהשייכות ל- Π נקבעת רק על סמך השפה המזוהה על ידי המכונה (כלומר, אם $L(M_1)=L(M_2)$, אז $\langle M_1\rangle\in\Pi$ אם ורק אם $\langle M_2\rangle\in\Pi$). אז בעיית השייכות ל- Π איננה כריעה.

שאלה : משפט רייס מדבר במפורש על תכונות של מכונות הנקבעות רק על ידי השפה המזוהה על ידי המכונה. כך, למשל, אם Π היא קבוצת כל מכונות הטיורינג שבהן יש מספר זוגי של מצבים, משפט רייס אינו חל עליה.

א. מדוע?

ב. הראו שתכונה זו כריעה.

תרגיל 3.1

הוכיחו את משפט רייס.

הדרכה: הניחו שקיימת מכונת טיורינג להכרעת בעיית השייכות ל- Π והראו כיצד ניתן להשתמש במכונה זו על מנת לבנות מכונת טיורינג להכרעת A_{TM} .

תרגיל 3.2

הראו שכל אחד משני התנאים על Π במשפט רייס הוא תנאי הכרחי.

תרגיל 3.3

פתרו את בעיה 5.30 בספר הלימוד.

תרגיל 3.4

פתרו את בעיה 5.31 בספר הלימוד.

קראו בספר הלימוד את התת-סעיף "רדוקציות באמצעות היסטוריות חישוב".

היסטוריית חישוב של מכונת טיורינג M על קלט w היא רשימת הקונפיגורציות ש- M עוברת בזמן העיבוד של w עד שהיא עוצרת ומקבלת את w או דוחה אותו. אם M איננה עוצרת על w ,

אז לא קיימת היסטוריית חישוב של M על w. שימוש בהיסטוריות חישוב ברדוקציות מאפשר לנו להוכיח את אי-כריעותן של בעיות שקשה להראות את אי-כריעותן בדרך אחרת.

<u>שאלה</u>: אם אותה קונפיגורציה מופיעה במהלך החישוב פעמיים, אז M לא עוצרת על w. מדוע?

<u>תשובה</u>: בהינתן קונפיגורציה אנו יודעים בדיוק מה צופן העתיד: כלומר, פונקציית המעברים מכתיבה את הקונפיגורציה הבאה, ולפיכך גם את כל הקונפיגורציות הבאות אחריה. על כן, אם אותה קונפיגורציה מופיעה בשלבים m ו-$m+k$, אז אנו מסיקים שני דברים:

- בכל השלבים הללו לא הייתה קונפיגורציה עוצרת (דהיינו, לא הגענו ל-q_{accept} או q_{reject}).
- השלבים $m+k$ עד $m+2k$ יהיו זהים לשלבים מ-m עד $m+k$ וכך הלאה.

אם כן, נגזר על המכונה לשחזר את הרצף הזה של k קונפיגורציות שוב ושוב ולעולם לא לעצור. (שימו לב: אנחנו מדברים על מכונות דטרמיניסטיות.)

<u>שאלה</u>: האם תיתכן אי עצירה ללא חזרה על קונפיגורציה?
<u>תשובה</u>: כמובן. חישבו על מכונת טיורינג שתמיד נעה עם הראש ימינה מבלי לשנות את מצבה. מכונה כזו לעולם לא תעצור, אך כל קונפיגורציה תהיה שונה מכל קודמותיה מכיוון שהראש תמיד יהיה במקום שונה.

בהגדרה 5.6 אנו פוגשים מודל חישובי חדש – **אוטומט חסום ליניארית** (Linear Bounded Automaton או LBA). זוהי למעשה מכונת טיורינג בעלת סרט סופי שאורכו שווה לאורך הקלט. לאוטומט כזה, כפי שקל לראות, יש מספר סופי של קונפיגורציות אפשריות: qng^n, כאשר $q=|Q|$, $g=|\Gamma|$ ו-n הוא אורך הסרט (למה 5.8). לפיכך, בעיית הקבלה עבור אוטומט חסום ליניארית היא כריעה (משפט 5.9): אם רצנו qng^n צעדים ולא עצרנו, משמע שקיימת לפחות קונפיגורציה אחת שחזרה על עצמה פעמיים, ועל כן אנו מזהים פה בוודאות לולאה אינסופית.

תרגיל 3.5

הראו שבהוכחת משפט 5.9, מספיק למכונת טיורינג L לדמות את פעולת M על w במשך $(q-2)\cdot n\cdot g^n+1$ צעדים בלבד.

במשפט 5.10 מוכיחים שבעיית בדיקת הריקנות של אוטומט חסום ליניארית, E_{LBA}, איננה כריעה. ההוכחה היא על ידי רדוקציה באמצעות היסטוריות חישוב. הבה נחזור על רעיון ההוכחה:

- ניתן לייצג כל קונפיגורציה על ידי מחרוזת המתארת את תוכן הסרט, את המצב הנוכחי ואת מקום הראש (למשל, $110q_2 0001$ היא קונפיגורציה שבה המצב הוא q_2, תוכן הסרט הוא 1100001, והראש מצביע על התו הרביעי משמאל). לפיכך, גם כל היסטוריית

חישוב ניתנת לתיאור על ידי מחרוזות שבה סימן מיוחד, למשל #, ישמש כמפריד בין תיאורי שתי קונפיגורציות עוקבות.

- בהינתן מכונת טיורינג M וקלט w, ניתן לבנות אוטומט חסום ליניארית, B, שיקבל קלט x אם ורק אם x הוא היסטוריית חישוב מקבלת של M על w. עיקר העבודה הוא להבין איך בונים אוטומט כזה ומדוע זה אכן אוטומט חסום ליניארית (כלומר, מדוע פעולת B לעולם לא תוביל אותו לגלישה מעבר לקטע הסרט שעליו נכתב הקלט x).
- אם קיימת מכונת טיורינג R שמכריעה את E_{LBA} (זוהי הנחת השלילה), נריץ אותה על תיאור האוטומט החסום ליניארית B שבנינו זה עתה. אם R תקבל את $\langle B\rangle$, הרי ש-$L(B)=\varnothing$, כלומר, אין שום היסטוריית חישוב מקבלת של M על w. במצב כזה ניתן להכריע ש-M אינה מקבלת את w. אם, לעומת זאת, R תדחה את $\langle B\rangle$, הרי ש-$L(B)\neq\varnothing$, כלומר, קיימת היסטוריית חישוב מקבלת של M על w (ומכיוון ש-M דטרמיניסטית, הרי קיימת רק היסטוריית חישוב אחת כזו). במצב זה ניתן להכריע ש-M מקבלת את w.

<u>שאלה:</u> מדוע המכונה B המתוארת ברעיון ההוכחה בספר היא אוטומט חסום ליניארית?

<u>תשובה:</u> הפעולה הראשונה שמבצעת המכונה B היא השוואת הקונפיגורציה הראשונה בקלט למה שהיא אמורה להיות, דהיינו $q_0w_1w_2\cdots w_n$. הפעולה הבאה היא סריקת הקונפיגורציה האחרונה בקלט ובדיקה שהמצב המופיע בה הוא q_{accept}. עד כאן, שתי פעולות אלו ניתנות לביצוע מבלי לגלוש מעבר לקצות הסרט. הפעולה הבאה, והמורכבת ביותר, היא לוודא שניתן לקבל כל קונפיגורציה בקלט מתוך הקונפיגורציה שמופיעה לפניה בקלט, לפי פונקציית המעברים של M. כפי שניתן לראות מהתיאור של מימוש בדיקה זו בספר, גם פעולה זו ניתנת לביצוע מבלי שהראש יגלוש מעבר לתחום על הסרט שעליו רשומות שתי הקונפיגורציות הנבדקות.

לבסוף, במשפט 5.13 אנו פוגשים תוצאה מעט מפתיעה. במשפט 4.8 הוכחנו שבעיית בדיקת הריקנות של דקדוקים חסרי הקשר, E_{CFG}, היא כריעה. כאן אנו מוכיחים שהבעיה המקבילה ALL_{CFG} שבה עלינו להכריע האם דקדוק נתון יוצר את **כל** המילים ב-Σ^* איננה כריעה. שימו לב ששתי השפות, E_{CFG} ו-ALL_{CFG}, אינן משלימות זו את זו (השפה המשלימה של E_{CFG} היא שפת כל הדקדוקים שיוצרים לפחות מילה אחת).

כמו בהוכחות קודמות, אנו מראים שאם ניתן להכריע את ALL_{CFG}, אז ניתן להכריע גם את A_{TM}.

יהי $\langle M,w\rangle$ קלט ל-A_{TM}. לצורך ההוכחה, אנו בונים מכונת טיורינג שעושה את הפעולות האלה:

1. בונה דקדוק G שיוצר את כל המחרוזות האפשריות **למעט** מחרוזות המתארות היסטוריות חישוב מקבלות של M על w (כאמור, יש לכל היותר אחת כזו).

2. קוראת למכונת טיורינג שמכריעה את ALL_{CFG} עם הקלט $\langle G\rangle$. אם מכונה זו תקבל את הקלט $\langle G\rangle$ כדקדוק שמייצר את כל המחרוזות האפשריות, הרי של- M אין היסטוריית חישוב מקבלת על w. לכן יש לדחות את $\langle M,w\rangle$. אם, לעומת זאת, מכונה זו תדחה את $\langle G\rangle$, הרי ש- G לא מייצר לפחות מחרוזת אחת. האופן שבו G נבנה מבטיח שמחרוזת זו היא היסטוריית חישוב מקבלת של M על w, ולכן יש לקבל את $\langle M,w\rangle$.

עיקר העבודה הוא הבנייה של הדקדוק G מתוך התיאור של M ו- w. לצורך כך מתארים אוטומט מחסנית D שמקבל את כל המחרוזות שאינן היסטוריות חישוב מקבלות של M על w. כיוון שקיימת פרוצדורה פשוטה לתרגום אוטומט מחסנית לדקדוק חסר הקשר שקול, בנייה זו מספיקה.

הרעיון הוא שמחרוזת מהווה היסטוריית חישוב מקבלת של M על w אם ורק אם היא מקיימת את ארבעה התנאים הבאים:

1. היא ייצוג חוקי של היסטוריית חישוב (כלומר, רשימת קונפיגורציות חוקיות מעל האלפבית Γ וקבוצת המצבים Q המופרדות על ידי סימני #).
2. הקונפיגורציה הראשונה היא $q_0w_1w_2\cdots w_n$.
3. כל קונפיגורציה C_{i+1} נובעת מזו שלפניה, C_i, בהתאם לפונקציית המעברים δ של M.
4. הקונפיגורציה האחרונה C_l היא קונפיגורציה מקבלת.

על כן, נתכנן את D כך שהוא יקבל כל מחרוזת שאיננה מקיימת את התנאים לעיל. לשם כך נוח להשתמש בתכונת האי-דטרמיניסטיות של אוטומטי מחסנית. D יתפצל בהתחלה באופן אי-דטרמיניסטי לארבעה מסלולי בדיקה שונים, וכל אחד מהם יבדוק את אחד התנאים שלעיל. אם הקלט לא יעמוד בתנאי הנבדק, אז האוטומט יקבל אותו. כיוון שאוטומט מחסנית מקבל כל קלט עבורו קיים מסלול קבלה אפשרי, הרי שבסופו של דבר יקבל D את כל המחרוזות **שאינן** היסטוריות חישוב מקבלות של M על w.

בדיקת התנאים 1, 2 ו-4 קלה מאוד. לגבי תנאי 3 ישנה הבעיה הטכנית של השוואת תוכן הסרט בשתי הקונפיגורציות העוקבות. השוואה זו בעייתית כיוון שתוכן הסרט בקונפיגורציה C_i מוכנס למחסנית ואז נשלף לצורך השוואתו עם תוכן הסרט ב- C_{i+1}. אך אוטומט מחסנית, בניגוד לאוטומט חסום ליניארית או מכונת טיורינג, יכול לעבור על הקלט רק פעם אחת משמאל לימין. לכן איך נשווה את C_i הנשלף מן המחסנית בסדר תווים הפוך ל- C_{i+1} המופיע על הסרט בסדר הטבעי? זו הסיבה שאנו בוחרים לייצג את היסטוריית החישוב באמצעות מחרוזת, שבה כל הקונפיגורציות במקומות הזוגיים מופיעות במהופך.

תרגיל 3.6

פתרו את תרגיל 5.1 בספר הלימוד.

תרגיל 3.7

פתרו את חלק a של בעיה 5.14 בספר הלימוד.

תרגיל 3.8

תארו אוטומט סופי דו-ראשי 2DFA לזיהוי השפה $\{a^n b^n c^n : n \geq 0\}$ (מספיק תיאור ברמת המימוש).

תרגיל 3.9

פתרו את בעיה 5.26 בספר הלימוד.

תרגיל 3.10

פתרו את בעיה 5.27 בספר הלימוד.

במדריך זה, הקטעים שהרקע שלהם אפור מיועדים להעשרה, ואינם כלולים בחומר של הבחינה.

3.2 בעיה בלתי כריעה פשוטה

קראו את סעיף 5.2 בספר הלימוד עד לפני הוכחת משפט 5.15

בסעיף זה אנו פוגשים בעיה פשוטה לניסוח, בעיית התאמת המילים של פוסט (על שם המתמטיקאי אמיל פוסט), או PCP, שאיננה כריעה. זו דוגמה מעניינת שממנה אנו למדים שבעיות אי-כריעוֹת אינן קשורות בהכרח לאוטומטים, לשפות רגולריות או חסרות הקשר או למכונות טיורינג.

הקלט לבעיה זו הוא אוסף אבני דומינו, $\left[\frac{u_1}{v_1}\right],\cdots,\left[\frac{u_k}{v_k}\right]$, כאשר כל המילים הכתובות על אבני דומינו אלו הן מילים לא ריקות מעל אלפבית סופי נתון, Σ (כלומר $u_i, v_i \in \Sigma^*$, $u_i, v_i \neq \varepsilon$ לכל $1 \leq i \leq k$). בבעיה עלינו להכריע האם קיימת סדרה סופית של אינדקסים $1 \leq i_1, i_2, \cdots, i_l \leq k$ כך שאם נסתכל על סדרת האבנים המוגדרת על ידי אינדקסים אלו

$$\left[\frac{u_{i_1}}{v_{i_1}}\right],\cdots,\left[\frac{u_{i_l}}{v_{i_l}}\right]$$

אז המילה המתקבלת בחלק העליון של האבנים זהה למילה המתקבלת בחלק התחתון, כלומר

$$u_{i_1}\cdots u_{i_l} = v_{i_1}\cdots v_{i_l}$$

חשוב לציין שסדרת האינדקסים יכולה לכלול חזרות (כלומר, יתכן ש- $i_s = i_r$ עבור $1 \leq s < r \leq l$). בפרט, סדרת האינדקסים יכולה להיות בכל אורך l.

<u>שאלה:</u> בעיית ה-Limited PCP (LPCP) שואלת שאלה דומה לבעיית ה-PCP אך נתון בה חסם מסוים L, כך שמחפשים התאמה (match) באורך $l \leq L$. האם בעיה זו כריעה?

<u>תשובה:</u> כמובן, כיוון שיש רק $\sum_{l=1}^{L} k^l$ סדרות אבנים אפשריות באורך קטן או שווה ל- L. אפשר פשוט לבדוק כל אחת מהן ולראות אם קיימת ביניהן סדרה המספקת התאמה.

ההוכחה של אי-כריעות בעיית ה-PCP (משפט 5.15) היא על ידי בניית רדוקציה מ- A_{TM} ל-PCP באמצעות שימוש בהיסטוריות חישוב מקבלות. כפי שמובהר ברעיון ההוכחה, ישנם שני פרטים טכניים שיש לשים לב אליהם: ראשית, אנו מניחים שהמכונה M מהקלט $\langle M, w\rangle$ של הבעיה A_{TM} איננה מנסה להזיז את הראש הקורא-כותב שמאלה מקצה הסרט. זו הנחה שקל להצדיק אותה, כיוון שאין כל בעיה לתרגם את הקלט $\langle M, w\rangle$ לקלט אחר $\langle M', w'\rangle$ כך ש- M' מקבלת את w' אם ורק אם M מקבלת את w, ו- M' לעולם איננה מנסה לבצע הזזה אסורה שמאלה. שנית, אנו מראים רדוקציה לבעיה MPCP, הקרובה מאוד ל-PCP. מאוחר יותר נצטרך להשלים את הרדוקציה מ- A_{TM} ל-PCP.

תרגיל 3.11

תארו את האופן שבו יש לבנות את $\langle M', w'\rangle$ האמורה לעיל מתוך $\langle M, w\rangle$.

קראו את הוכחת משפט 5.15 בספר הלימוד.

תהי R מכונה להכרעת PCP. אנו מתארים מכונה S שבהינתן קלט $\langle M, w\rangle$ היא בונה אוסף P' של אבני דומינו שיש בו התאמה בבעיית ה-MPCP אם ורק אם M מקבלת את w. בשלב האחרון של ההוכחה מתורגם אוסף זה של אבני דומינו לאוסף אחר P של אבני דומינו שיש בו התאמה בבעיית ה-PCP אם ורק אם M מקבלת את w. כך הושלמה הרדוקציה של A_{TM} ל-PCP, ועל כן אנו מסיקים ש-PCP איננה כריעה.

חלק 1 בבנייה מגדיר את אבן הדומינו הראשונה ב- P', זו שההתאמה חייבת להתחיל בה.

חלקים 2,3,4,5 מגדירים אבני דומינו שיאפשרו לנו לבנות התאמות שבהן בשורה העליונה תופיע הקונפיגורציה לפני מעבר אחד במכונה M, ובשורה התחתונה תופיע הקונפיגורציה בעקבות אותו מעבר. לצורך כך, בחלקים 2,3 עוברים על כל כללי המעבר של המכונה M, ומוסיפים אבן דומינו המתארת את חלק הקונפיגורציה - לפני (למעלה) ואחרי (למטה) ליד הראש הקורא-כותב. אח״כ, בחלק 4 מוסיפים אבני דומינו שיעזרו לבנות את כל יתר חלקי הקונפיגורציה המתאימים לאזורים בסרט שלא השתנו במעבר זה. לבסוף, בחלק 5 מוסיפים שתי אבנים - האחת מאפשרת לשים את

תו הסימון המיוחד # שמשמש כמפריד בין קונפיגורציות בהיסטוריית החישוב, והשנייה מאפשרת לטפל במקרים שבהם הראש נע ימינה באזור של תווי הרווח.

האבנים שהוגדרו עד כה מאפשרות לנו לבנות את ההתאמה שלב אחר שלב על ידי הדמיה של פעולת המכונה M על הקלט w. החלק העליון בהתאמה יתאר את היסטוריית החישוב עד לפני מעבר מסוים, ואילו החלק התחתון יתאר את אותה היסטוריה ממש אך עד אחרי אותו מעבר. שימו לב: בבעיית ה-PCP אין הכרח להשתמש בכל אבני הדומינו ומותר להשתמש בכל אבן מספר בלתי מוגבל של פעמים! לפיכך, אבני דומינו המתאימות לכללי מעבר שאינם באים לידי ביטוי כאשר M פועלת על הקלט w לא ייטלו חלק בהתאמה שאנו בונים פה.
החלק התחתון מקדים תמיד את החלק העליון בקונפיגורציה אחת. בחלקים 6 ו-7 אנו מוסיפים אבנים שמאפשרות לחלק העליון "להשיג" את החלק התחתון אם מגיעים ל- q_{accept}. אם המכונה לעולם אינה מגיעה למצב זה (בין שהיא נכנסת ללולאה אינסופית ובין שהיא מגיעה ל- q_{reject}), החלק העליון לא יוכל להשיג את החלק התחתון על מנת לקבל התאמה.

כעת, נותר לתרגם את P' לאוסף אבנים אחר P עבור בעיית ה-PCP. זה נעשה על ידי טריק פשוט שמחייב אותנו לחפש התאמות שמתחילות באבן הדומינו הראשונה: בכל שאר האבנים יש תו * בתחילת החלק העליון ותו אחר בתחילת החלק התחתון; רק באבן הראשונה שני החלקים מתחילים באותו תו - התו *.

<u>שאלה:</u> מדוע אין אפשרות להסתכל על P' עצמו כעל קלט ל-PCP?
<u>תשובה:</u> משום שכל אחת מהאבנים שהוגדרו בחלק 4 מהווה התאמה בעצמה. על כן, בעוד שאבני P' מגדירות התאמה בבעיית MPCP רק אם M מקבלת את w, אבני P' מגדירות התאמה בבעיית PCP תמיד, ללא כל קשר ל-$\langle M, w \rangle$.

תרגיל 3.12
פתרו את תרגיל 5.3 בספר הלימוד.

תרגיל 3.13
פתרו את תרגיל 5.8 בספר הלימוד.

תרגיל 3.14
פתרו את בעיה 5.33 בספר הלימוד.

3.3 רדוקציות מיפוי

קראו את סעיף 5.3 בספר הלימוד.

המושג הראשון שאנו פוגשים כאן הוא המושג **פונקציה ניתנת לחישוב** או **פונקציה חשיבה** (computable function). עד כה דיברנו על מכונות טיורינג כמכשיר לזיהוי או להכרעת שפה, ולא הייתה חשיבות לתוכן הסרט בסוף הפעולה של המכונה. כעת אנו מדברים על מכונות טיורינג כמכונות לחישוב שיש להן פלט – תוכן הסרט בסוף הפעולה. (בתוכן הסרט הכוונה למחרוזת בין התו השמאלי ביותר על הסרט (כולל) ובין התו הימני ביותר שאיננו רווח (כולל)).

תרגיל 3.15

הראו שכפל מספרים טבעיים הוא פונקציה חשיבה. כלומר, תארו מכונת טיורינג המקבלת כקלט שני מספרים טבעיים m ו- n בייצוג האונארי הבא $1^m\#1^n$ ועוצרת כאשר על הסרט רשום 1^{mn} .

המושג המרכזי בסעיף זה הוא **רדוקציית מיפוי** (הגדרה 5.20). התוצאה המרכזית והחשובה היא משפט 5.22 (ומסקנה 5.23). אם שפה B כריעה ו- $A \le_m B$ (כלומר, קיימת רדוקציית מיפוי מ- A ל- B), אז גם A כריעה. המכונה להכרעת A מיישמת תחילה את פעולת המכונה שעושה את המיפוי ומחליפה את הקלט שעל הסרט בהתאם לפונקציית הרדוקציה; אח"כ מיושמת פעולת המכונה המכריעה את השפה B על הקלט החדש (שהוא תוכן הסרט לאחר הרדוקציה). לפי הגדרת רדוקציית המיפוי, התשובה שתתקבל לגבי שייכות הקלט החדש לשפה B היא גם התשובה הנכונה לגבי שייכות הקלט המקורי לשפה A .

שימו לב: בהגדרת רדוקציות מיפוי צריכים להתקיים שלושה תנאים:

1. הפונקציה f ברדוקציה צריכה להיות **חשיבה**.
2. $x \in A$ גורר כי $f(x) \in B$.
3. $f(x) \in B$ גורר כי $x \in A$.

כאשר בונים רדוקציה, חשוב מאוד לבדוק כי **כל** שלושת התנאים הללו מתקיימים. טעות נפוצה היא בניית רדוקציות שבהן רק שניים (או פחות) מהתנאים דלעיל מתקיימים.

<u>שאלה:</u> הראו שאם $A \le_m B$, אז גם $\overline{A} \le_m \overline{B}$.

<u>תשובה</u>: זה מידי, על ידי שימוש באותה פונקציית מיפוי f . אכן, ל- f התכונה ש- $x \in A$ אם ורק אם $f(x) \in B$. לכן - $x \notin A$ אם ורק אם $f(x) \notin B$. כלומר - $x \in \overline{A}$ אם ורק אם $f(x) \in \overline{B}$.

תרגיל 3.16

פתרו את תרגיל 5.5 בספר הלימוד.

תרגיל 3.17

פתרו את תרגיל 5.6 בספר הלימוד.

תרגיל 3.18

פתרו את בעיה 5.34 בספר הלימוד.

התוצאה החשובה הבאה היא משפט 5.28 (ומסקנה 5.29). אם שפה B ניתנת לזיהוי טיורינג ו- $A \le_m B$, אז גם A ניתנת לזיהוי טיורינג.

<u>שאלה :</u> מדוע אם $A_{\text{TM}} \le_m \overline{B}$ אז B איננה ניתנת לזיהוי טיורינג?

<u>תשובה :</u> אם $A_{\text{TM}} \le_m \overline{B}$, אז $\overline{A_{\text{TM}}} \le_m \overline{\overline{B}} = B$. כיוון ש- $\overline{A_{\text{TM}}}$ איננה ניתנת לזיהוי טיורינג (מסקנה 4.23) , אז לפי מסקנה 5.29, גם B לא.

האבחנה האחרונה הזו משמשת בהוכחת משפט 5.30 כדי להראות ש- EQ_{TM} (שפת כל הזוגות $\langle M_1, M_2 \rangle$ כך ש- $L(M_1) = L(M_2)$) איננה ניתנת לזיהוי טיורינג וגם משלימתה לא. כדי להוכיח את החלק הראשון, אנו מראים רדוקציה מ- A_{TM} ל- $\overline{EQ_{\text{TM}}}$, כלומר, פונקציה חשיבה הלוקחת זוג מהצורה $\langle M, w \rangle$ ומייצרת זוג מהצורה $\langle M_1, M_2 \rangle$ כך ש- $L(M_1) \ne L(M_2)$ אם ורק אם M מקבלת את w . בחלק השני מראים רדוקציה מ- A_{TM} ל- EQ_{TM} .

תרגיל 3.19

פתרו את תרגיל 5.7 בספר הלימוד.

תרגיל 3.20

פתרו את בעיה 5.19 בספר הלימוד.

פתרון התרגילים

תרגיל 3.1

נניח בשלילה שבעיית השייכות ל- Π כריעה על ידי מכונת טיורינג T. תהי S מכונת טיורינג המזהה את השפה הריקה. ללא הגבלת הכלליות, נניח ש- $\langle S\rangle \notin \Pi$ (כיוון שאם $\langle S\rangle \in \Pi$, נסתכל על הקבוצה המשלימה $\overline{\Pi}$, שגם היא מקיימת את תנאי המשפט, והשייכות אליה ניתנת להכרעה על ידי מכונת טיורינג $\overline{T}$ הפועלת בדיוק כמו T אך מוציאה פלט הפוך). כמו כן, כיוון ש- Π איננה הקבוצה הריקה, קיימת מכונת טיורינג N כך ש- $\langle N\rangle \in \Pi$. נראה כיצד ניתן לבנות מהנחת הכריעות של בעיית השייכות ל- Π מכונת טיורינג להכרעת הבעיה A_{TM} :

"בהינתן הקלט $\langle M, w\rangle$:

1. בנה מכונת טיורינג M_w שתקבל קלט נתון x אם ורק אם M מקבלת את w ו- N מקבלת את x.
2. הרץ את T על $\langle M_w\rangle$. אם T מקבלת את $\langle M_w\rangle$, קבל את הקלט $\langle M, w\rangle$. אחרת – דחה אותו."

בחלק הראשון אנו בונים תיאור של מכונת טיורינג חדשה, M_w. מכונה זו מריצה את M על w ; אם M עוצרת ומקבלת את w, אז M_w ממשיכה ומריצה את N על x (שהוא הקלט של M_w). אם שתי ההרצות האלו מסתיימות במצב מקבל, אז M_w תעצור ותקבל את x. אחרת, M_w תדחה את הקלט או תיכנס ללולאה אינסופית.

הבה נבחן כעת את המכונה החדשה M_w :

- אם M מקבלת את w, אז M_w תקבל את הקלט שלה x אם ורק אם N תקבל את x. לכן, במקרה זה $L(M_w) = L(N)$, ולפיכך $\langle M_w\rangle \in \Pi$. על כן, T תקבל את $\langle M_w\rangle$.
- אם M איננה מקבלת את w, אז M_w לעולם לא תקבל את הקלט שלה x. כלומר, במקרה זה $L(M_w) = L(S) = \varnothing$ ולפיכך $\langle M_w\rangle \notin \Pi$, כי $\langle S\rangle \notin \Pi$. על כן, T לא תקבל את $\langle M_w\rangle$.

אם כן, אופן הפעולה של המכונה שתוארה לעיל מבטיח שהיא מכריעה את A_{TM}. כיוון שמכונה כזו איננה קיימת, אנו מסיקים שגם T איננה יכולה להתקיים.

תרגיל 3.2

תהי Π קבוצת כל מכונות טיורינג עם חמישה מצבים. זוהי קבוצה לא טריוויאלית שאיננה מקיימת את התכונה שאם $L(M_1) = L(M_2)$, אז $\langle M_1\rangle \in \Pi$ אם ורק אם $\langle M_2\rangle \in \Pi$ (מדוע?). קל לראות שבעיית השייכות ל- Π ניתנת להכרעה.

תהי Π הקבוצה הריקה. קבוצה זו מקיימת את התנאי שאם $L(M_1) = L(M_2)$, אז $\langle M_1 \rangle \in \Pi$ אם ורק אם $\langle M_2 \rangle \in \Pi$. בעיית השייכות לקבוצה זו כריעה על ידי מכונת טיורינג שתמיד דוחה את הקלט שלה.

תרגיל 3.3

תהי L_{TM} שפת כל הזוגות $\langle M, w \rangle$, כאשר M היא מכונת טיורינג ו-w הוא קלט למכונה זו הגורם לה לנסות להזיז את הראש הקורא-כותב שמאלה כאשר הוא נמצא על התא השמאלי ביותר. נניח בשלילה שקיימת מכונת טיורינג R להכרעת L_{TM}, ונבנה מכונת טיורינג S להכרעת A_{TM} (מכיוון שמכונה כזו לא תיתכן, נסיק שגם R לא תיתכן, ומכאן ש-L_{TM} איננה כריעה). S תפעל באופן הבא:

"בהינתן הקלט $\langle M, w \rangle$:

1. בנה מכונת טיורינג M' שתפעל כך:
   - קודם כל היא תעתיק את כל הקלט שלה תא אחד ימינה, ואז תשים בתא השמאלי ביותר סימן מיוחד לסימון תחילת הסרט.
   - לאחר מכן, היא תזיז את הראש לתא השני כדי שיצביע על התו הראשון בקלט.
   - כעת היא תפעל בדיוק כמו M, למעט הבדל אחד: אם אי פעם נגיע למצב שבו התו הנוכחי הוא תו הסימון המיוחד, המכונה תזיז את הראש תא אחד ימינה ותישאר באותו מצב (תיקון זה מבטיח ש-M' תמיד תפעל כמו M).
   - אם אי פעם נגיע לכניסה למצב המקבל, M' תזיז את הראש הקורא עד לתו הסימון המיוחד בקצה השמאלי של הסרט ואז תנסה לזוז שמאלה שוב.
2. הרץ את R על $\langle M', w \rangle$. אם R מקבלת, אז קבל את $\langle M, w \rangle$. אחרת – דחה."

אכן, S תקבל את הקלט $\langle M, w \rangle$ אם ורק אם R תקבל את הקלט $\langle M', w \rangle$, כלומר אם ורק אם M' תנסה לזוז שמאלה מקצה הסרט בעת העיבוד של w; האופן שבו הוגדרה פעולת M' מבטיח שדבר זה יקרה אם ורק אם M מקבלת את w.

תרגיל 3.4

עלינו להכריע את השפה LM_{TM} המורכבת מכל הזוגות $\langle M, w \rangle$ שבהם M היא מכונת טיורינג ו-w הוא קלט למכונה כך שבזמן הריצה של M על w הראש הקורא-כותב נע שמאלה לפחות פעם אחת.

אנו טוענים שאם נריץ את M על w במשך $|Q_M|+|w|+1$ צעדים, כאשר $|w|$ הוא אורך הקלט ו-$|Q_M|$ הוא מספר המצבים ב-M, ובכל הצעדים הללו היו רק תנועות ראש ימינה, אנו יכולים לעצור את הסימולציה של M על w ולקבוע בוודאות ש-M לעולם לא תנוע שמאלה בעת

העיבוד של w ; כלומר, במקרה זה אנו נדחה את הקלט $\langle M,w\rangle$. אם, לעומת זאת, באחד מ- $|w|+|Q_M|+1$ צעדים אלו הייתה תנועת ראש שמאלה, אנו יכולים לעצור ולקבל את הקלט $\langle M,w\rangle$. החלק השני ברור מאליו, אך החלק הראשון דורש הסבר:

נניח שבמשך $|w|+|Q_M|+1$ הצעדים הראשונים הייתה רק תנועת ראש ימינה. אז לאחר $|w|$ הצעדים הראשונים הגענו לקטע בסרט המלא בסימני רווח. כעת המשכנו וביצענו $|Q_M|+1$ צעדים, כאשר בכולם התו הנקרא היה רווח. מכיוון שיש למכונה רק $|Q_M|$ מצבים אפשריים, אז לפי עקרון שובך היונים יש לפחות שני צעדים, נאמר i ו- j, כך ש-

$$|w|+1 \le i < j \le |w|+|Q_M|+1$$

שבהם הייתה המכונה באותו מצב. לכן, כיוון שתנועת הראש נקבעת רק על פי הערך שמחזירה פונקציית המעברים δ, וזו תלויה רק במצב הנוכחי ובתו הנקרא, אז כפי שתנועת הראש הייתה ימינה בצעדים i עד j, כך יהיה גם בהמשך, עד אינסוף, כאשר סדרת המצבים שדרכם תעבור המכונה בצעדים אלו תחזור על עצמה באופן מחזורי. למשל, אם $i=10$, $j=14$ וסדרת המצבים בצעדים אלו הייתה q_3,q_2,q_9,q_4,q_3 (כלומר, q_3 היה המצב בצעד ה-10 וה-14, q_2 היה המצב בצעד ה-11 וכן הלאה), אז הראש ימשיך לנוע ימינה כל הזמן, וסדרת המצבים של המכונה תחזור על המעגל $q_3,q_2,q_9,q_4,q_3,q_2,q_9,q_4,\ldots$ עד אינסוף.

תרגיל 3.5

יש בסך הכול $q-2$ מצבים שאינם מצבים סופיים (q_{accept} או q_{reject}). לכן, אם M רצה על w במשך $(q-2)\cdot n\cdot g^n+1$ צעדים, והיא לא הגיעה למצב סופי, הרי שהיא הייתה חייבת לחזור על קונפיגורציה (ולכן להיכנס ללולאה אינסופית), כפי שמשתמע מתוך עקרון שובך היונים, שכן יש רק $(q-2)\cdot n\cdot g^n$ קונפיגורציות שבהן המצב איננו מצב סופי.

תרגיל 3.6

נראה רדוקציה מ- ALL_{CFG} ל- EQ_{CFG} על ידי בניית מכונת טיורינג להכרעת ALL_{CFG} בהינתן מכונת טיורינג R להכרעת EQ_{CFG}:

"בהינתן הקלט $\langle G\rangle$:

1. בנה דקדוק חסר הקשר H כך ש- $L(H)=\Sigma^*$.
2. הרץ את R על $\langle G,H\rangle$. אם R קיבלה, קבל את $\langle G\rangle$. אחרת – דחה אותו."

תרגיל 3.7

יהי M אוטומט סופי דו-ראשי (2DFA) בעל s מצבים. כאשר הוא פועל על קלט x בגודל n, יש לכל היותר $s\cdot(n+2)^2$ קונפיגורציות אפשריות. לכן, M עוצר על הקלט x בתוך $s\cdot(n+2)^2$

צעדים, או נכנס ללולאה אינסופית על קלט זה. אם כן, על מנת להכריע את השפה A_{2DFA} , מספיק לדמות את הפעולה של M על x $s\cdot(n+2)^2$ צעדים. אם M מקבל את x במהלך הצעדים האלו, אז אנו נקבל את הקלט $\langle M, x\rangle$; אחרת – נדחה אותו.

תרגיל 3.8

1. אם הקלט ריק, עצור וקבל אותו (כיוון שהוא מהצורה המתאימה : $a^0b^0c^0$).
2. ודא שהקלט הוא מהצורה $aa*bb*cc*$. זה ניתן לביצוע באמצעות אוטומט סופי רגיל, כיוון שמדובר בביטוי רגולרי.
3. אם הפעולה הקודמת הסתיימה בהצלחה, המשך לשלב הבא בבדיקה. הזז את אחד משני הראשים כך שיצביע על ה-a הראשון, ואת האחר – כך שיצביע על ה-b הראשון. כעת התחל בלולאה שבה תקרא את שני התווים שהראשים מצביעים עליהם ותוודא שהראשון מביניהם הוא a והשני הוא b. אם זה המצב, הזז את שני הראשים ימינה לקראת שלב ההשוואה הבא. אחרת – עבור לצעד הבא.
4. אם התו הראשון הוא a, דחה את הקלט (יש יותר a-ים מ-b-ים). אם התו השני הוא b, דחה את הקלט (יש יותר b-ים מ-a-ים). אחרת, כיוון שכבר וידאנו שהקלט הוא מהצורה $aa*bb*cc*$, אנו מסיקים שמספר ה-a-ים שווה למספר ה-b-ים, ושהראש האחד מצביע כעת על ה-b הראשון, ואילו הראש האחר מצביע על ה-c הראשון.
5. באופן דומה לשלב 3, השווה את מספר ה-b-ים ואת מספר ה-c-ים. עצור וקבל את הקלט רק אם תגיע למצב שבו הראש האחד מצביע על c והראש האחר מצביע על רווח.

תרגיל 3.9

פתרון הבעיה מופיע בספר הלימוד.

תרגיל 3.10

פתרון הבעיה מופיע בספר הלימוד.

תרגיל 3.11

$w'=\perp w$ כאשר $\perp$ הוא תו סימון מיוחד. M' היא מכונה הזהה ל- M בתוספת הכלל $\delta(q,\perp)=(q,\perp,R)$ לכל $q\in Q$.

תרגיל 3.12

התאמה אפשרית היא : $\left[\frac{aa}{a}\right]\left[\frac{aa}{a}\right]\left[\frac{b}{a}\right]\left[\frac{b}{a}\right]\left[\frac{aba}{b}\right]\left[\frac{ab}{abab}\right]\left[\frac{ab}{abab}\right]$

תרגיל 3.13

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 3.14

נניח שהקלט לבעיית ה-PCP האונארית הוא: $\left\{\left[\frac{1^{a_1}}{1^{b_1}}\right],\dots,\left[\frac{1^{a_k}}{1^{b_k}}\right]\right\}$; כלומר, יש בקלט k אבנים, כאשר באבן ה-i יש a_i 1-ים למעלה ו-b_i 1-ים למטה. אלגוריתם ההכרעה יהיה כדלקמן:

1. אם קיים i כך ש-$a_i = b_i$, קבל את הקלט.
2. אם קיימים i ו-j כך ש-$a_i > b_i$ ו-$a_j < b_j$, קבל את הקלט. אחרת – דחה אותו.

במקרה 1, ההתאמה מורכבת מאבן אחת שבה מופיע אותו מספר 1-ים למעלה ולמטה. במקרה 2, אם בכל האבנים מופיעים למעלה יותר 1-ים מאשר למטה (או להפך), אז מובן שאין כל התאמה. אחרת, קיימת התאמה שבה מופיעה האבן ה-i $(b_j - a_j)$ פעמים ואילו האבן ה-j מופיעה $(a_i - b_i)$ פעמים. קל לראות שבהתאמה כזו, מספר ה-1-ים למעלה,

$$(b_j - a_j)\cdot a_i + (a_i - b_i)\cdot a_j = b_j a_i - b_i a_j$$

שווה למספר ה-1-ים למטה,

$$(b_j - a_j)\cdot b_i + (a_i - b_i)\cdot b_j = a_i b_j - a_j b_i.$$

תרגיל 3.15

1. אם הקלט איננו מהצורה $1^m\#1^n$, עבור למצב q_{reject}.
2. אם הקלט הוא מהצורה $1^m\#$ או $\#1^n$, מחק את תוכן כל הסרט ועבור למצב q_{accept}.
3. החלף את ה-1 הראשון ברצף השני ב-1_f.
4. חפש את התו האחרון לפני הרווח.
5. אם תו זה הוא 1, החלף אותו ב-1_l (זה המצב אם $n \geq 2$). אחרת ($n=1$) מחק את התווים $\#1_f$ ועבור למצב q_{accept}.
6. החזר את הראש הקורא לתחילת הסרט וסמן את ה-1 הראשון ב-$\bar{1}$.
7. עבור בלולאה על יתר התווים בתחילת הסרט עד לתו המפריד $\#$. אם נותר שם 1 שאיננו מסומן, סמן אותו $(\bar{1})$ ובצע את פרוצדורת ההעתקה המתוארת למטה. (כל פרוצדורה כזו מוסיפה n 1-ים בסוף הסרט).

8. אם כל התווים עד ל- # מסומנים, הרי שמימין לסימן ה- # יש לנו בשלב זה mn 1-ים, כאשר הראשון וה- n י ביניהם מסומנים כ- 1_f ו- 1_l בהתאמה. עלינו לבצע כעת הזזה שמאלה של כל תוכן הסרט החל מהתו הראשון שמימין ל- #. אפשר לבצע הזזה מייגעת, או לחלופין, למחוק 1-ים בסוף הסרט כנגד $m+1$ התווים שבהתחלה. בסוף יש לזכור לעבור על תוכן הסרט ולשנות בו את כל התווים שאינם 1 ($1_l, 1_f$ או גם $\bar{1}$ או # אם ביצענו מחיקה של תווים מיותרים מהסוף במקום הזזה) כך שיהיו 1.
9. עבור למצב q_{accept}.

פרוצדורת ההעתקה:

1. הבא את הראש הקורא לתו 1_f.
2. סמן אותו ב- $\overline{1_f}$.
3. הוסף 1 לסוף הסרט.
4. החזר את הראש הקורא לתו הימני ביותר המסומן.
5. אם תו זה הוא $\overline{1_l}$, אז הסתיימה פעולת ההעתקה. הסר את הסימון מעל n התווים שסומנו לעיל וסיים.
6. אחרת, זוז ימינה תו אחד. סמן תו זה (כלומר, 1 יסומן כ-$\bar{1}$ ו- 1_l יסומן כ- $\overline{1_l}$) והוסף 1 לסוף הסרט. חזור לשלב 4.

תרגיל 3.16

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 3.17

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 3.18

נראה רדוקציית מיפוי מ-PCP ל-BPCP (בעיית ה-PCP מעל $\Sigma = \{0,1\}$). בהינתן קלט לבעיית ה-PCP נתרגם אותו לקלט לבעיית ה-BPCP כך שכל תו a_i על אבן דומינו מקורית יתורגם לרצף 10^i (דהיינו, התו 1 שאחריו מופיעים i אפסים) באוסף האבנים המהוות קלט לבעיית ה-BPCP.
קל לראות שאם יש התאמה באבנים המקוריות, אז יש התאמה גם באבנים החדשות, ולהפך. לכן זוהי רדוקציית מיפוי (שקל מאוד ליישמה במכונת טיורינג), ולכן, על פי מסקנה 5.23, BPCP איננה כריעה.
אגב – מדוע בחרנו לתרגם את התווים לבינארית באופן הבלתי יעיל המתואר לעיל? מדוע לא נוכל לתרגם את התו a_i לייצוג הבינארי של המספר i?

תרגיל 3.19

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 3.20

1. נניח, בניגוד למה שהוכחנו במשפט 4.11, שקיימת מכונה המכריעה את A_{TM}. כלומר, נניח שקיימת מכונה H המקבלת קלטים מהצורה $\langle M,w\rangle$ אם המכונה M מקבלת את הקלט w, ואחרת – דוחה קלטים כאלו.
2. תהי F מכונת טיורינג המקבלת כקלט מספר טבעי x ומחשבת את $f(x), f(f(x)), f(f(f(x))), \cdots$. המכונה F עוצרת ומקבלת את x אם היא מגיעה לערך 1, אחרת היא נמצאת בלולאה אינסופית.
3. כעת נתאר מכונה אחרת G: בהינתן הקלט w, המכונה G מתעלמת ממנו ומתחילה להריץ את H על הקלטים $\langle F,1\rangle,\langle F,2\rangle,\langle F,3\rangle,\cdots$ עד שהיא מגיעה לשלב i שבו H דוחה את הקלט $\langle F,i\rangle$. רק אז G עוצרת ומקבלת את הקלט w. אם G אינה מגיעה לשלב כזה, אז היא איננה עוצרת. לפיכך, המכונה G מקבלת את הקלט הסתמי שלה w אם ורק אם קיים מספר טבעי i שלגביו הסדרה $f(i), f(f(i)), f(f(f(i))), \cdots$ איננה מסתיימת בערך 1.
4. לבסוף, נבנה מכונת טיורינג המפעילה את H על הקלט $\langle G,w\rangle$ (w יכולה להיות כל מחרוזת שהיא). אם H תקבל את הקלט, משמע ש-G עוצרת ומקבלת את w. לפיכך, בניגוד לכל העדויות, קיים מספר טבעי שעבורו סדרת האיטרציות המתאימה איננה מסתיימת ב-1. אם, לעומת זאת, H תדחה את הקלט, משמע ש-G איננה עוצרת, ולכן סדרת האיטרציות תסתיים ב-1 לכל ערך התחלתי טבעי.

4. סיבוכיות זמן (פרק 7 בספר הלימוד)

קראו את המבוא לפרק 7 בספר הלימוד.

בחלק הראשון של הקורס דנו בשאלה "מה ניתן לחשב?". בחלק הזה של הקורס נתרכז בבעיות הניתנות לחישוב ונשאל את השאלה "כמה זה עולה לנו?". עלות החישוב, הקרויה **סיבוכיות**, נמדדת בזמן ובמקום (זיכרון) – שני המשאבים החשובים ביותר. בפרק זה אנו דנים בסיבוכיות הזמן. בפרק הבא נדון בסיבוכיות המקום.

4.1 מדידת סיבוכיות

קראו את סעיף 7.1 בספר הלימוד עד לפני התת-סעיף "סימוני O גדול ו-o קטן".

זמן הריצה של אלגוריתם הוא מספר הצעדים שמבצע האלגוריתם עד לעצירה. מספר הצעדים תלוי באורכו של הקלט. למשל, אלגוריתם המחפש איבר מסוים בטבלה לא ממוינת על ידי סריקת הטבלה, ירוץ בממוצע $n/2$ צעדים, כאשר n הוא גודל הטבלה (בהנחה שהאיבר שאותו מחפשים אכן נמצא בטבלה) או, במקרה הגרוע ביותר, n צעדים. בדרך כלל נתעניין בזמן הריצה הגרוע ביותר.

בהגדרה 7.1 אנו מדברים על **סיבוכיות הזמן** (או **זמן הריצה**) של מכונת טיורינג, במקום על סיבוכיות הזמן של אלגוריתם. בדברנו על מכונות טיורינג, במקום על המושג השקול (על פי התזה של צ'רץ'-טיורינג) אך הפחות פורמלי **אלגוריתם**, אנו יכולים להגדיר במדויק גם את אורך הקלט וגם את זמן הריצה: אורך הקלט הוא מספר התווים שתופס הקלט על הסרט, ואילו זמן הריצה הוא מספר הצעדים שעושה המכונה עד לעצירה. במובן מסוים, ההבדל בין אלגוריתם ובין מכונת טיורינג המממשת אותו דומה להבדל בין תכנית מחשב הכתובה בשפה עילית לבין הקוד המתאים לתכנית זו בשפת מכונה. לא נוח לחשוב במושגים של שפת מכונה, אך יותר קל להגדיר מושגים כמו אורך הקלט או זמן הריצה באמצעות פריטתם ל"פרוטות" של ביטים או של פעולות מכונה אלמנטריות.

קראו את התת-סעיף "סימוני O גדול ו-o קטן".

אנו נתקלים כאן בסימוני סדר הגודל המוכרים O ו-o. סימנים אלו כבר בוודאי מוכרים לכם היטב מקורסים קודמים במדעי המחשב ובמתמטיקה.

תרגיל 4.1

פתרו את התרגילים 7.1 ו-7.2 בספר הלימוד.

קראו את התת-סעיף "ניתוח אלגוריתמים".

בתת-סעיף זה מתוארות שלוש מכונות טיורינג שונות המכריעות את השפה $A = \{0^k 1^k \mid k \geq 0\}$.

- המכונה M_1:

מכונה זו בודקת תחילה שהקלט הוא מהצורה $0^* 1^*$ (זמן הריצה, כולל החזרת הראש לתחילת הסרט, הוא $O(n)$), ואחר כך נכנסת ללולאה שבה בכל שלב נמחק 0 אחד ו-1 אחד. כל שלב בלולאה אורך $O(n)$ צעדים, ומספר השלבים בלולאה אף הוא $O(n)$ (שכן מספר השלבים חסום על ידי הנמוך מבין מספר ה-0-ים ומספר ה-1-ים). אם נותרים בסוף הלולאה תווים שלא נמחקו, המכונה דוחה את הקלט, אחרת – היא מקבלת אותו. לפיכך, זמן הריצה הוא $O(n^2)$. בסימונים של הגדרה 7.7, אנו אומרים אפוא ש- $A \in \mathrm{TIME}(n^2)$.

- המכונה M_2:

מכונה זו מוחקת את ה-0-ים וה-1-ים בצורה יעילה יותר. גם כאן, החלק העיקרי של האלגוריתם הוא לולאה. כל עוד נותרו 0-ים ו-1-ים על הסרט, בודקים שמספרם הכולל הוא זוגי (אחרת דוחים את הקלט) ואז מוחקים לסירוגין כל 0 אי-זוגי, וכך גם לגבי ה-1-ים.

למשל, אם התחלנו עם הקלט $0^{12}1^{12}$, אז בהמשך תוכן הסרט ישתנה ל- $0^6 1^6$, אחר כך $0^3 1^3$, 01 ולבסוף כלום; לפיכך – קלט זה יזוהה כשייך לשפה (שימו לב: יתכן שהמכונה מוחקת את ה-0-ים וה-1-ים באמצעות כתיבה של תו מיוחד עליהם; לפיכך, יתכן שתווי ה-0 וה-1 הנותרים על הסרט יהיו מופרדים על ידי מופעים של תו מיוחד זה). אם, לעומת זאת, התחלנו עם $0^{13}1^9$, אז בהמשך ישתנה תוכן הסרט ל- $0^6 1^4$ ואחר כך ל- $0^3 1^2$. בשלב זה הקלט יידחה כיוון שהמספר הכולל של תווי 0 ו-1 שנותרו הוא אי-זוגי.

עלות כל שלב בלולאה היא $O(n)$, ומספר הפעמים שהלולאה מתבצעת חסום על ידי $1 + \log n$.
לפיכך, סיבוכיות הזמן של אלגוריתם זה היא $O(n \log n)$. אם כן, זה מוכיח ש- $A \in \mathrm{TIME}(n \log n)$.

הערה: אם $f(n) = O(g(n))$, אז $\mathrm{TIME}(f(n)) \subseteq \mathrm{TIME}(g(n))$. לכן, כיוון ש- $n \log n = O(n^2)$ אז $\mathrm{TIME}(n \log n) \subseteq \mathrm{TIME}(n^2)$.

על פי בעיה 7.20 בספר הלימוד, אין אפשרות להכריע את השפה A בזמן קצר יותר, $f(n) = o(n \log n)$, מכיוון שבזמן כה קצר אפשר להכריע על ידי מכונת טיורינג בעלת סרט יחיד רק שפות רגולריות, ואנו יודעים ש- A איננה רגולרית.

- המכונה M_3:

מכונה זו, בניגוד לשתי קודמותיה, מצוידת בשני סרטים. בדיון על חישוביות ראינו שריבוי סרטים איננו משנה דבר לגבי **מה** שמכונת טיורינג יכולה לחשב (משפט 3.13). אבל ריבוי סרטים יכול

להשפיע על **המהירות** שבה המכונה מסוגלת לחשב. אכן, המכונה M_3 יכולה לבצע את המשימה שלפניה בזמן ליניארי $O(n)$, מכיוון שהיא אינה צריכה לטייל על הקלט הלוך ושוב כדי להשוות את מספר ה-0-ים למספר ה-1-ים. מכונה זו מעתיקה את קטע ה-0-ים לסרט השני, ואז היא "מטיילת" על קטע ה-0-ים המשוכפל על הסרט השני, ובמקביל היא מטיילת גם על קטע ה-1-ים שעל הסרט הראשון ומשווה את אורך שני הקטעים. זמן ההשוואה הזה הוא ליניארי. אלגוריתם ההכרעה הזה מאופיין אפוא בסיבוכיות הזמן המינימלית האפשרית לשפה (שכן נחוצים $O(n)$ צעדים רק בשביל לקרוא את הקלט).

אם כן, סיבוכיות הזמן של הכרעת שפות תלויה בגרסת מכונת טיורינג שנבחר. תלות זו עלולה להצטייר כדבר בעייתי בבואנו לסווג שפות על פי סיבוכיות הזמן שנדרשת להכרעתן. אך, כפי שנראה בהמשך, ההבדלים בין הגרסאות עדיין מאפשרים לנו לסווג שפות למחלקות שונות לפי סיבוכיות הזמן הנחוצה להכרעתן, ושיטת הסיווג בה נדון לא תהיה רגישה להבדלים בין גרסאות אלה.

קראו את התת-סעיף "יחסי סיבוכיות בין מודלים".

בתת-סעיף זה אנו בוחנים שלושה מודלים חישוביים: מכונת טיורינג בעלת סרט אחד, מכונת טיורינג מרובת-סרטים, ומכונת טיורינג אי-דטרמיניסטית.

במשפט 7.8 אנו רואים שעל אף שמכונת טיורינג מרובת-סרטים מסוגלת לחשב מהר יותר (ועל כן להפחית את סיבוכיות הזמן של שפות), שיעור החיסכון אינו יותר מאשר שורש ריבועי. במילים אחרות, אם המכונה מרובת-הסרטים מצליחה להכריע שפה בזמן $t(n)$ (במקרה הגרוע ביותר כאשר הקלט הוא באורך n), אז קיימת מכונת טיורינג בעלת סרט אחד שתעשה את העבודה בזמן $O(t^2(n))$. הבדל כזה יכול בהחלט להיות משמעותי במימוש אלגוריתמים, אך מבחינה תאורטית, כאשר אנו מעוניינים רק למתוח את הקו בין השפות שניתן להכריע בזמן סביר, ובין אלה שלא, זהו הבדל לא משמעותי.

משפט 7.11, לעומת זאת, משווה בין מכונת טיורינג רגילה למכונת טיורינג אי-דטרמיניסטית. המכונה האי-דטרמיניסטית, שהיא מודל חישובי תאורטי בלבד ללא כל מקבילה בעולם האמיתי, **מסתמנת** כמכשיר חישובי חזק הרבה יותר. אם מכונת טיורינג אי-דטרמיניסטית מכריעה שפה בזמן $t(n)$ (במקרה הגרוע ביותר כאשר הקלט הוא באורך n), אז קיימת מכונת טיורינג (רגילה) בעלת סרט אחד שתעשה את העבודה בזמן $2^{O(t(n))}$. נציין שעד עתה לא נמצאה דרך יעילה יותר לממש מכונת טיורינג אי-דטרמיניסטית על ידי מכונת טיורינג דטרמיניסטית. כלומר, ישנן בעיות שהמכונה האי-דטרמיניסטית פותרת בזמן $t(n)$, ואילו לכל מכונת טיורינג דטרמיניסטית נדרש (כך נראה, אך אין זה ודאי) זמן ריצה $2^{O(t(n))}$.

שאלה: שימו לב שסייגנו את דברינו ואמרנו שמכונות טיורינג אי-דטרמיניסטיות רק **מסתמנות** כמודל חישובי חזק יותר. מדוע? האם משפט 7.11 אינו מראה בוודאות שמכונות טיורינג אי-דטרמיניסטיות חזקות בהרבה ממכונות טיורינג רגילות?

תשובה: משפט 7.11 אינו מראה זאת. הוא מראה רק שבהינתן מכונת טיורינג אי-דטרמיניסטית המכריעה שפה בזמן $t(n)$, קיימת מכונת טיורינג רגילה המכריעה את אותה שפה בזמן $2^{O(t(n))}$ במקרה הגרוע ביותר. אך המשפט איננו שולל את האפשרות שבעתיד תשופר תוצאה זו ותימצא מכונת טיורינג רגילה המכריעה אותה שפה בזמן קצר יותר (נאמר, $O(t^k(n))$ עבור חזקה קבועה k). גם אם k יהיה גדול מאוד, ההבדל בין $t(n)$ לבין $O(t^k(n))$ איננו כה דרמטי כמו ההבדל בין $t(n)$ ו- $2^{O(t(n))}$, והוא לא "יורגש" בשיטת הסיווג של סיבוכיויות הזמן שנגדיר בהמשך.

נדון כעת בהוכחת משפט 7.8. מומלץ לחזור ולקרוא את משפט 3.13 ואת הוכחתו בפרק הראשון של מדריך זה. משפט 7.8 חוזר לבנייה המתוארת במשפט 3.13 ומעריך את זמן הריצה של מכונת טיורינג רגילה המדמה את פעולתה של מכונת טיורינג המצוידת ב- k סרטים (k הוא מספר קבוע ולכן איננו משפיע על סדרי הגודל בחישובי הסיבוכיות להלן). הערכת זמן הריצה היא פשוטה למדי:

- אתחול הסרט לפורמט המייצג את k הסרטים מצריך $O(n)$ צעדים.
- כל אחד מ- $t(n)$ הצעדים שעושה המכונה מרובת-הסרטים מתורגם למספר צעדים של המכונה הרגילה המתאימה. מספר הצעדים האלה הוא $O(t(n))$, כיוון שמספר התווים המצויים על כל סרט במכונה מרובת-הסרטים הוא לכל היותר $t(n)$ (זה נובע מכך שבכל צעד מספר התווים על כל סרט של המכונה יכול לגדול באחד לכל היותר).
- לכן, בסך הכול, אנו מקבלים זמן ריצה של $O(n)+t(n)\times O(t(n))=O(t^2(n))$ (זכרו שהנחנו ש- $t(n)\geq n$).

הוכחת משפט 7.11 העוסק במכונות טיורינג אי-דטרמיניסטיות מחזירה אותנו אף היא אחורה, הפעם למשפט 3.16 שבו תיארנו מכונה דטרמיניסטית בעלת שלושה סרטים המדמה מכונה אי-דטרמיניסטית. גם כאן אנו מנתחים את זמן הריצה של המכונה הדטרמיניסטית. נסמן ב- b את מספר ההסתעפויות המקסימלי בעץ ההסתעפויות המתאר את פעולת המכונה האי-דטרמיניסטית. עומקו של העץ, בהינתן קלט באורך n, מוגדר כזמן הריצה הגרוע ביותר, כלומר $t(n)$. המכונה הדטרמיניסטית עלולה, במקרה הגרוע ביותר, לבקר בכל אחד מקדקודי העץ.

לפיכך, כיוון שמספר הקדקודים בעץ זה הוא $O(b^{t(n)})$, והזמן הנחוץ לרדת מהשורש של העץ לקדקוד נתון הוא $O(t(n))$ (בסימולציה המתוארת במשפט 3.16, בכל פעם שאנחנו מבקרים בקדקוד אנחנו סורקים את כל המסלול משורש העץ עד לאותו הקדקוד), מתקבל החסם הבא על זמן הריצה:

$$O(t(n)b^{t(n)})=b^{O(t(n))}=2^{O(t(n))}$$

(פה הסתמכנו על כך ש- b הוא קבוע). כעת, עלינו לתרגם את המכונה הדטרמיניסטית בעלת שלושת הסרטים למכונה רגילה בעלת סרט אחד. המחיר שנשלם על כך הוא לכל היותר ריבוע זמן הריצה – פעולה שמשאירה את סיבוכיות הזמן $2^{O(t(n))}$ על כנה.

4.2 המחלקה P

קראו בסעיף 7.2 את התת-סעיף "זמן פולינומיאלי".

בתת-סעיף זה מוסבר מדוע אנו מבדילים בין זמן ריצה פולינומיאלי ובין זמן ריצה מעריכי (אקספוננציאלי) ומדוע אנו בוחרים לא להבדיל בין זמני ריצה פולינומיאליים כאשר אנו מנסים לאפיין בעיה כ"קלה" או כ"קשה".

זמני הריצה הפולינומיאליים שבהם אנו נתקלים מאופיינים בדרך כלל בחזקות נמוכות. לרוב נפגוש זמני ריצה כמו $O(n^2 \log n)$ או $O(n^4)$ ולא $O(n^{100})$. אלו הם זמני ריצה מעשיים עבור ערכים סבירים של n (כלומר, ערכים של אורכי קלט שבהם נתקלים בדרך כלל בבעיות מעשיות). לעומת זאת, אלגוריתמים בעלי זמן ריצה מעריכי אינם מעשיים כלל ועיקר גם עבור ערכים צנועים של n.

איננו טורחים להבדיל בין פולינומים שונים, משום שאנו רוצים לדון בתכונות הבסיסיות ביותר של המושג **חישוב**, ואיננו רוצים להגביל את עצמנו לגרסה מסוימת של מכונת טיורינג. ראינו, למשל, שכדי להכריע שייכות לשפה $A = \{0^k 1^k \mid k \geq 0\}$ במכונה עם סרט אחד נדרש זמן ריצה שהוא לפחות $O(n \log n)$, בעוד שבמכונה עם שני סרטים אפשר לבצע את המשימה בזמן שהוא $O(n)$. כדי להבין, באופן עקרוני, את הקושי הכרוך בהכרעת שפה זו, עלינו להרים את הדיון אל מעל לרמת המודל החישובי. מצד שני, מכיוון שההבדל בין כל המודלים החישוביים הסבירים הוא פולינומי (כפי שמודגם במשפט 7.8), אז המוסכמה היא לא להבדיל בין פולינומים, אלא להבדיל בין פולינומים לבין פונקציות שגדלות מהר יותר מכל פולינום, כמו הפונקציות המעריכיות. הנחה זו הופכת את החיים לקלים יותר: אם בעיה מסוימת היא קלה לפתרון במודל חישובי אחד (כלומר, היא ניתנת לפתרון בזמן פולינומי במודל זה), היא תהיה קלה לפתרון בכל מודל סביר אחר.

כפי שמוסבר בספר, איננו בוחרים לעצום את העיניים ולהתעלם מההבדל בין, למשל, $O(n^3)$ ו- $O(n^2 \log n)$. זהו אכן הבדל חשוב בעת מימוש מעשי, אך בשלב זה של הדיון אנו בוחרים להתרכז במיון "גס" של הבעיות למחלקות קושי שונות.

קראו בסעיף 7.2 את התת-סעיף "דוגמאות לבעיות ב-P" עד לאחר הוכחת משפט 7.14.

על מנת להראות שבעיה מסוימת ניתנת לפתרון בזמן פולינומיאלי, מתארים אלגוריתם לפתרונה שמספר השלבים בו פולינומיאלי באורך הקלט, וכל שלב ניתן למימוש בזמן פולינומיאלי במודל חישובי סביר, כמו מכונת טיורינג **דטרמיניסטית** (בעלת סרט אחד או מרובת-סרטים, אין זה משנה). בדרך כלל, אנו מנסחים את האלגוריתם כך שכל שלב מורכב מפעולה פשוטה כגון קריאה מהזיכרון, כתיבה לזיכרון, השוואה בין שני ערכים, פעולה אריתמטית (כגון חיבור, חיסור, כפל, חילוק) וכדומה. מכיוון שכל אחת מהפעולות האלה ניתנת למימוש בזמן פולינומיאלי במכונת טיורינג, מספיק להתרכז בהערכת מספר השלבים. אם מספר השלבים חסום על ידי פולינום, הרי שלפנינו אלגוריתם פולינומיאלי.

נקודה נוספת שיש לשים לב אליה היא האופן שבו מיוצג הקלט. הדוגמה הראשונה עוסקת בגרפים. גרפים ניתן לייצג על ידי רשימת הקדקודים ורשימת הקשתות, או לחלופין על ידי רישום מטריצת הסמיכויות המתארת את הגרף. למשל:

$$\langle 1,2,3,4;(1,2),(2,3),(3,4),(4,1)\rangle$$

מתאר מעגל מכוון של ארבעה קדקודים. היינו יכולים לתאר אותו גרף על ידי מטריצת הסמיכויות המתאימה:

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

(1 בעמודה השנייה שבשורה הראשונה מציין כי יש קשת מכוונת בין קדקוד 1 לקדקוד 2 וכך הלאה). במעגל לא מכוון, כל קשת תופיע פעמיים בשיטת הקידוד הראשונה (למשל, קשת המחברת בין שני קדקודים u,v תתואר בשיטת קידוד זו על ידי צמד הקשתות המכוונות (u,v) ו- (v,u)), ובתיאור המטריציאלי נקבל מטריצה סימטרית.

מכיוון שמספר הקשתות בגרף חסום על ידי m^2, כאשר m הוא מספר הקדקודים, אז אורך הקלט המתאר גרף הוא פולינומיאלי ב-m. לפיכך, זמן הריצה של אלגוריתם נתון הוא פולינומיאלי באורך הקלט אם ורק אם הוא פולינומיאלי ב-m.

בבעיה $PATH$ אנו מקבלים כקלט תיאור של גרף מכוון ושני קדקודים בו, s ו-t. עלינו להכריע אם יש מסלול מכוון המקשר בין s ל-t. האלגוריתם פשוט מאוד: הוא מורכב מלולאה שבתחילתה מסומן רק s, ובמהלכה אנו מסמנים עוד ועוד קדקודים שאליהם ניתן להגיע מ-s (זהו למעשה אלגוריתם BFS). אנו עוצרים כאשר הגענו לשלב שבו לא סומן שום קדקוד חדש, ובודקים אז אם הקדקוד t סומן; אם כן, אנו מקבלים את הקלט, אחרת – אנו דוחים אותו.

מכיוון שמספר השלבים בלולאה חסום על ידי m, וכל שלב סורק את כל קשתות הגרף (שמספרן לכל היותר m^2), לפנינו אלגוריתם פולינומיאלי.

המשיכו לקרוא את הדוגמה הבאה עד לאחר הוכחת משפט 7.15.

הדוגמה הקלאסית המתוארת כאן היא הבעיה $RELPRIME$: בהינתן שני מספרים טבעיים, x ו-y, יש לקבוע אם הם זרים, דהיינו אם המחלק המשותף הגדול ביותר שלהם, המסומן $\gcd(x,y)$, שווה ל-1. האלגוריתם לפתרון הבעיה הוא האלגוריתם של אויקלידס, אשר מחשב את **המחלק המשותף המקסימלי** של שני מספרים נתונים. מפעילים את האלגוריתם הזה על שני הערכים הנתונים, x ו-y. אם $\gcd(x,y)=1$, אז שני המספרים זרים, ולפיכך $\langle x,y\rangle \in RELPRIME$. אחרת, $\langle x,y\rangle \notin RELPRIME$.

ננתח כעת את זמן הריצה של האלגוריתם כפונקציה של אורך הקלט.
שימו לב: כאשר נתונים מספרים, אנחנו מניחים כי הם מיוצגים בהצגה בינארית. לכן, גודל הקלט איננו גודלם של המספרים, כי אם מספר הביטים בייצוג הבינארי שלהם. לכן גודל הקלט במקרה זה הינו $\lceil \log_2 y \rceil + \lceil \log_2 x \rceil$ לכל היותר. למעשה, אנו יכולים להניח כל בסיס שאינו אונארי, ולאו דווקא בסיס בינארי. בחירת הבסיס (בתנאי שאינו אונארי) לא ממש משנה, כי הרווח יהיה רק בקבועים. אכן, אילו היינו בוחרים לייצג את המספרים לפי בסיס $b \geq 3$, אז מספר הסיביות הנדרש לייצוג מספר x הוא $\lfloor \log_b x \rfloor + 1$. מכיוון שמתקיים $\frac{\log_2 x}{\log_b x} = \log_2 b$, אז מספר הסיביות הנדרש לייצג מספר x לפי בסיס b קטן פי $\log_2 b$ מהייצוג הבינארי של אותו המספר. למשל, עבור $b=4$ הייצוג מתקצר פי 2, עבור $b=8$ פי 3, וכך הלאה. רווח זה הוא זניח מבחינה תאורטית (מדובר בקבוע, כי b הוא קבוע).

בלולאה העיקרית באלגוריתם מבוצעות שלוש פעולות בסיסיות: חישוב שארית מודולרית (שלב 2), החלפת ערכים (שלב 3), והשוואת ערך לאפס (שלב 1). כל אחת מפעולות אלו ניתנת למימוש בזמן פולינומיאלי, ולכן אנו מתרכזים רק בהערכת מספר השלבים בלולאה.
ההבחנה הבסיסית בכיוון זה היא שהפעולה $x \leftarrow x \bmod y$ מפחיתה את ערכו של x פי 2 לפחות.
מדוע? ראשית, בכל פעם שאנו מבצעים את הפעולה $x \leftarrow x \bmod y$, אז, לפני ביצוע הפעולה, $x \geq y$. כעת נפריד את הדיון לשני מקרים אפשריים: המקרה שבו לפני ביצוע הפעולה $\frac{x}{2} \geq y$ והמקרה האחר שבו $\frac{x}{2} < y$. במקרה הראשון, ערכו של x לאחר ביצוע הפעולה ירד אל מתחת לערכו של y, ולפיכך – אל מתחת למחצית הערך המקורי של x. במקרה השני, מכיוון ש-y גדול ממחצית x, אז $x-y<y$ ולפיכך $x \bmod y = x-y < x-\frac{x}{2} = \frac{x}{2}$.

לאור האמור לעיל, מספר השלבים בלולאה חסום על ידי $2\log_2 x$ וגם על ידי $2\log_2 y$ (הכופל 2 נובע מכך שאנו מחליפים כל פעם בין x ו- y, ולכן ערכו של x מופחת רק בשלבים האי-זוגיים, וזה של y מופחת רק בשלבים הזוגיים). שימו לב שאם ערכו של אחד משני המשתנים מגיע ל-1, אז בשלב הבא נעצור (כיוון שהשארית מודולו 1 של כל מספר טבעי היא 0). לבסוף, כיוון שאורך הקלט הוא $O(\log_2 x + \log_2 y)$, אנו מקבלים שמספר השלבים בלולאה הוא ליניארי באורך הקלט.

תרגיל 4.2

פתרו את תרגיל 7.3 בספר הלימוד.

המשיכו לקרוא את הדוגמה הבאה עד לסוף התת-סעיף.

בסעיף 2.1 הוכחנו שכל שפה חסרת-הקשר היא כריעה (משפט 4.9). כאן אנו מוכיחים שהיא ניתנת להכרעה בזמן פולינומיאלי. בסעיף 2.1 הראינו את כריעותן של שפות חסרות-הקשר על ידי שימוש בדקדוק בצורה הנורמלית של חומסקי היוצר אותן. בתרגיל 2.6 הוכחנו שאם G הוא דקדוק חסר הקשר בצורה הנורמלית של חומסקי ו- $w \in L(G)$ היא מילה באורך n, אז בכל גזירה של w יש בדיוק $2n-1$ צעדים. מכונת טיורינג שבנינו אז להכרעת שפה חסרת-הקשר פעלה כך: בהינתן מילה w באורך n, היא בחנה את כל הגזירות האפשריות באורך $2n-1$ ובדקה אם אחת מהן יוצרת את w. אך מה שהיה טוב בדיון על כריעות, איננו מספיק בדיוננו הנוכחי העוסק בסיבוכיות. מכיוון שמספר הגזירות האפשריות עשוי להיות מעריכי ב- $2n-1$ (שהרי בכל אחד מ- $2n-1$ שלבי הגזירה ייתכן שאנו יכולים לבחור יותר מאשר כלל המרה אחד מבין כללי ההמרה הקיימים בדקדוק), עלינו לשנס מותניים ולאמץ את המוח כדי לנסות למצוא אלגוריתם יעיל יותר. האם קיים אלגוריתם יעיל יותר? בפרט, האם קיים אלגוריתם כזה שזמן ריצתו פולינומיאלי?

אלגוריתם כזה אכן קיים והוא משתמש בטכניקה הקרויה **תכנות דינמי**. הרעיון הוא לבנות טבלה שבה נשמור אינפורמציה על ה"ניתנות לגזירה" של תת-מחרוזות בקלט – כלומר, את רשימת המשתנים בדקדוק שמהם ניתן לגזור את התת-מחרוזת המתאימה.

נניח שהקלט הוא $w = w_1 \cdots w_n$. בטבלה זו, $table(1:n,1:n)$, אנו משתמשים רק באלכסון ובחצי העליון, כלומר רק ב- $table(i,j)$ כאשר $i \le j$. בכניסה $table(i,j)$ נשמור את רשימת כל המשתנים שמהם ניתן לגזור את התת-מחרוזת $w_i \cdots w_j$. אנו מתחילים את מילוי הטבלה מהאלכסון הראשי (המתייחס לכל n התת-מחרוזות באורך 1):

$$\begin{pmatrix} \{\cdots\} & \square & \square & \square & \square \\ & \{\cdots\} & \square & \square & \square \\ & & \{\cdots\} & \square & \square \\ & & & \{\cdots\} & \square \\ & & & & \{\cdots\} \end{pmatrix}$$

אנו עוברים משם לאלכסון שמעליו, דהיינו לכניסות $table(i, i+1)$ המתייחסות לכל $n-1$ התת-מחרוזות באורך 2.

$$\begin{pmatrix} \{\cdots\} & \{\cdots\} & \square & \square & \square \\ & \{\cdots\} & \{\cdots\} & \square & \square \\ & & \{\cdots\} & \{\cdots\} & \square \\ & & & \{\cdots\} & \{\cdots\} \\ & & & & \{\cdots\} \end{pmatrix}$$

אנו ממשיכים כך הלאה ומטפסים בטבלה בכיוון צפון-מזרח, עד שבשלב האחרון אנו מגיעים ל"אלכסון" הקצר ביותר המכיל רק את $table(1, n)$:

$$\begin{pmatrix} \{\cdots\} & \{\cdots\} & \square & \square & \square \\ & \{\cdots\} & \{\cdots\} & \square & \square \\ & & \{\cdots\} & \{\cdots\} & \square \\ & & & \{\cdots\} & \{\cdots\} \\ & & & & \{\cdots\} \end{pmatrix} \rightarrow \begin{pmatrix} \{\cdots\} & \{\cdots\} & \{\cdots\} & \square & \square \\ & \{\cdots\} & \{\cdots\} & \{\cdots\} & \square \\ & & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & & & \{\cdots\} & \{\cdots\} \\ & & & & \{\cdots\} \end{pmatrix} \rightarrow$$

$$\rightarrow \begin{pmatrix} \{\cdots\} & \{\cdots\} & \{\cdots\} & \{\cdots\} & \square \\ & \{\cdots\} & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & & & \{\cdots\} & \{\cdots\} \\ & & & & \{\cdots\} \end{pmatrix} \rightarrow \begin{pmatrix} \{\cdots\} & \{\cdots\} & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & \{\cdots\} & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & & \{\cdots\} & \{\cdots\} & \{\cdots\} \\ & & & \{\cdots\} & \{\cdots\} \\ & & & & \{\cdots\} \end{pmatrix}$$

כניסה זו תכיל את רשימת כל המשתנים שמהם ניתן לגזור את המחרוזת השלמה, $w = w_1 \cdots w_n$.

אם משתנה ההתחלה S יופיע ברשימה זו, w נוצרת על ידי הדקדוק הזה, אחרת – לא.

הרעיון המרכזי בתכנות דינמי הוא להשתמש באינפורמציה שנשמרה בעבר, עבור תת-בעיות קטנות יותר, כדי לפתור את הבעיה הנוכחית, מבלי לעשות מחדש עבודה שכבר נעשתה. למשל, בבואנו למלא את הכניסות בטבלה עבור תת-מחרוזות באורך l, אנו בוחרים תת-מחרוזת מסוימת כזו, $w_i \cdots w_j$ (שלבים 7 ו-8) ואז אנו שוברים אותה לשני חלקים לא ריקים, $w_i \cdots w_k$ ו-$w_{k+1} \cdots w_j$ (שלב 9). לכל כלל גזירה בדקדוק שצורתו $A \rightarrow BC$ אנו בודקים אם $B \in table(i, k)$

(כלומר, אם B יוצר את $w_i \cdots w_k$) וגם $C \in table(k+1, j)$. אם אכן כך, אנו מוסיפים את A ל-$table(i, j)$, שהרי ניתן לגזור את $w_i \cdots w_j$ מהמשתנה A.

ניתוח זמן הריצה של האלגוריתם הוא פשוט למדי והוא מראה שאלגוריתם זה רץ בזמן $O(n^3)$.

תרגיל 4.3

פתרו את תרגיל 7.4 בספר הלימוד.

תרגיל 4.4

פתרו את תרגיל 7.8 בספר הלימוד.

תרגיל 4.5

פתרו את בעיה 7.40 בספר הלימוד.

תרגיל 4.6

פתרו את תרגיל 7.6 בספר הלימוד.

תרגיל 4.7

פתרו את בעיה 7.42 בספר הלימוד.

4.3 המחלקה NP

קראו בסעיף 7.3 עד לפני התת-סעיף "דוגמאות של בעיות ב-NP".

בסעיף זה מוצגת מחלקה חשובה של שפות. זוהי מחלקת השפות שקל **לאמת** אותן (אך, ככל הנראה, קשה **להכריע** חלק מהן). הגדרת המחלקה מתרכזת בחלק הראשון של המשפט הקודם, דהיינו בקלות האימות, כיוון שהחלק השני (קושי ההכרעה) לא הוכח, נכון להיום.

למשל, השפה $HAMPATH$ מורכבת מכל התיאורים של גרף מכוון G בצירוף שני קדקודים בגרף זה, s ו-t, כך שיש מסלול המילטוני המקשר בין שני קדקודים אלו (כלומר, מסלול העובר דרך כל קדקודי הגרף בדיוק פעם אחת, מתחיל ב-s ומסתיים ב-t). נכון להיום, לא ידוע אלגוריתם פולינומיאלי להכרעת שפה זו (אך ידוע אלגוריתם מעריכי). מצד שני, אם מישהו יציג לנו מסלול כנדרש, אנו נוכל בקלות (כלומר, בזמן פולינומיאלי) לוודא שמסלול זה עומד בכל הדרישות, ואז להשתכנע שהקלט שייך לשפה. מסלול כזה משמש **הוכחה** (proof) או **אישור** (certificate) לכך שהגרף הנתון שייך לשפה $HAMPATH$. אם כן, זו שפה שאיננו מכירים דרך להכריע אותה בזמן סביר (כלומר, לזהות קלטים ממנה ללא כל עזרה מבחוץ) אך קל לנו לאמת אותה בזמן סביר:

בהינתן עזרה מבחוץ, דהיינו הוכחה או אישור, קל לנו לבדוק אישור זה ואז להשתכנע שהקלט הנתון שייך לשפה.

הגדרה 7.18 עוסקת ב**מאמת** (verifier). מאמת יודע להכריע שייכות לשפה בהינתן ההוכחה המתאימה בצורת מחרוזת c. למאמת V זמן ריצה פולינומיאלי אם עבור קלט w זמן הריצה של V על w ועל המחרוזת c פולינומיאלי בגודלו של w. שימו לב שאורך ההוכחה c חייב להיות פולינומיאלי באורך הקלט (כיוון שעל המאמת לקרוא את כל ההוכחה, וזמן הריצה של המאמת הוא פולינומיאלי באורך הקלט).

נדגיש: מאמת הוא אלגוריתם שתמיד עוצר (על הקלט $\langle w,c \rangle$), ומקבל את הקלט או דוחה אותו. המאמת מקבל אם c מוכיח ש-w שייכת לשפה. המאמת דוחה אם c לא מוכיח ש-w שייכת לשפה. במקרה השני איננו יודעים האם w שייכת לשפה או לא. כל שאנו יודעים הוא ש-c לא מוכיח ש-w שייכת לשפה.

למשל, המאמת לשפה $HAMPATH$ מקבל אם האישור c שניתן לו הוא מסלול המילטוני מ-s ל-t בגרף G. אחרת, המאמת דוחה. אם המאמת מקבל, אנו יודעים בוודאות ש-$\langle G,s,t \rangle \in HAMPATH$. אם המאמת דוחה, אנו יודעים ש-c איננו מסלול המילטוני מ-s ל-t ב-G, אך איננו יודעים האם מסלול כזה קיים או לא. איננו יודעים האם $\langle G,s,t \rangle \in HAMPATH$ או לא.

השפה A מוגדרת כשפת כל המילים w שיש להן אימות c. כמובן, בהינתן מאמת עבור A, אזי לקלט שאיננו בשפה A אין שום מחרוזת c (הוכחה) הגורמת למאמת לקבל את הקלט.

דוגמה 1:

נתבונן בשפה $HAMPATH$. לשפה זו יש מאמת שזמן ריצתו פולינומיאלי. מאמת זה יקבל את הקלט $\langle \langle G,s,t \rangle ,c \rangle$ רק אם c היא מחרוזת המכילה מסלול המילטוני ב-G בין s ל-t. שימו לב כי ניתן לבדוק בזמן פולינומיאלי האם מסלול נתון בין שני קדקודים הינו מסלול המילטוני. מאמת זה לא יקבל קלט $\langle \langle G,s,t \rangle ,c \rangle$ אם אין ב-G מסלול המילטוני בין s ל-t, מכיוון שלקלט כזה לא קיימת שום מחרוזת c מתאימה.

דוגמה 2:

השפה $COMPOSITES$ מורכבת מכל הקידודים הבינאריים של מספרים טבעיים שאינם ראשוניים. גם לשפה זו יש מאמת שזמן ריצתו פולינומיאלי. מאמת זה יקבל את הקלט $\langle x,d \rangle$ רק אם d הוא מחלק לא טריוויאלי של x (דהיינו, $1 < d < x$ המחלק את x). גם כאן אפשר לוודא בזמן פולינומיאלי בגודלו של x האם מספר נתון הוא מחלק לא טריוויאלי של x. כמובן, אם x

איננו פריק (כלומר, אם x הוא מספר ראשוני) אז לא קיימת מחרוזת פולינומיאלית בגודלו של x שמייצגת מחלק לא טריוויאלי של x.

שימו לב למילים "נכון להיום" שבהן סייגנו את דברינו לעיל: איננו יודעים להוכיח, נכון להיום, כי קיים אלגוריתם שזמן ריצתו פולינומיאלי הפותר את בעיית *HAMPATH*. אך אין זה אומר שאלגוריתם כזה איננו קיים. למשל, עד שנת 2002 ידענו שהשפה *COMPOSITES* שייכת למחלקה NP, כפי שכבר תיארנו לעיל, אך לא היה ברור אם היא שייכת ל-P. בשנת 2002 הצליחו שלושה חוקרים (M. Agrawal, N. Kayal, N. Saxena) לבנות אלגוריתם פולינומיאלי שיכול להכריע שפה זו, ללא כל עזרה מבחוץ בצורת אישור. לפיכך, השפה *COMPOSITES* שייכת למחלקה P. לכן, באופן תאורטי יתכן שדבר דומה יקרה יום אחד גם לשפה *HAMPATH*, כלומר, שגם לה יימצא יום אחד אלגוריתם הכרעה פולינומיאלי. יחד עם זאת, נציין שהסיכוי שדבר כזה יקרה נראה קלוש, מכיוון שהשפה *HAMPATH* "חשודה" כשפה קשה להכרעה (דהיינו, חשודה ככזו שאיננה ניתנת להכרעה בזמן פולינומיאלי), כפי שיוסבר בהמשך.

משפט 7.20 מבאר את משמעות הסימון NP לציון Non-deterministic Polynomial (Time). שפה נמצאת ב-P אם היא ניתנת להכרעה בזמן פולינומיאלי על ידי מכונת טיורינג דטרמיניסטית. לעומת זאת, שפה נמצאת ב-NP אם היא ניתנת להכרעה בזמן פולינומיאלי על ידי מכונת טיורינג אי-דטרמיניסטית. כלומר, אם קיימת מכונת טיורינג אי-דטרמיניסטית שבהינתן קלט מהשפה, אז קיים תרחיש פעולה של המכונה שבסופו היא תקבל את הקלט, בעוד שעבור קלט שאיננו בשפה לא קיים תרחיש כזה; בנוסף, זמן הריצה המקסימלי של המכונה הוא פולינומיאלי באורך הקלט (או, במילים אחרות, כל מסלול חישוב הוא באורך פולינומיאלי בגודל הקלט).

הוכחת משפט 7.20 פשוטה למדי: אם שפה היא ב-NP, כלומר ניתנת לאימות בזמן פולינומיאלי, אז נבנה מכונת טיורינג אי-דטרמיניסטית ש"תנחש" את ההוכחה (המהלכים האי-דטרמיניסטיים של המכונה יתייחסו לאוסף כל הערכים האפשריים שיכולה לקבל ההוכחה, שאורכה חייב להיות פולינומיאלי באורך הקלט). מצד שני, אם שפה ניתנת להכרעה על ידי מכונת טיורינג אי-דטרמיניסטית, אזי ההוכחה שהמאמת שלנו יקבל עבור קלט נתון היא תיאור מסלול במכונה המסתיים בקבלת הקלט.

תרגיל 4.8

פתרו את בעיה 7.43 בספר הלימוד.

קראו את התת-סעיף "דוגמאות של בעיות ב- NP".

כאן מוצגות עוד שתי שפות (או בעיות) קלאסיות השייכות ל-NP: *CLIQUE* ו-*SUBSET-SUM*. כפי שניתן לראות מההוכחות של משפטים 7.24 ו-7.25, ההוכחה ששפה מסוימת שייכת ל-NP היא

לרוב פשוטה מאוד. כיוון שרוב השפות הללו מורכבות מקלטים שעבורם **קיים** משהו (גרף שקיים בו מסלול המילטון, מספר שקיים לו מחלק לא טריוויאלי, קבוצת מספרים שקיימת לה תת-קבוצה שסכומה שווה לערך נתון, וכיוצא באלה), אם נציג את אותו "משהו" (מסלול המילטון, מחלק או תת-קבוצה בדוגמאות לעיל), אז קל לבדוק ש"משהו" זה מספק את תנאי הכניסה לשפה. הקושי בהכרעת שפות אלו ללא הוכחה מתאימה הוא שאיננו מכירים דרך למצוא הוכחה כזו, או להוכיח את קיומה, חוץ מאשר בעזרת **חיפוש ממצה** (exhaustive search) או דרכים אחרות שזמן ריצתן גדול מזמן פולינומיאלי.

שפה A שייכת למחלקה coNP אם משלימתה, $\overline{A}$, שייכת ל-NP. איננו יודעים אם $\mathrm{NP} = \mathrm{coNP}$ או לא. הסברה המקובלת היא ש- $\mathrm{NP} \neq \mathrm{coNP}$.

<u>שאלה:</u>

תהי L שפה ב-NP ותהי M מכונה אי-דטרמיניסטית המכריעה את L בזמן פולינומיאלי. תהי N המכונה המתקבלת מ-M על ידי החלפה של המצבים q_{accept} ו- q_{reject} (כלומר, כאשר M מקבלת אז N דוחה ולהפך). האם N מכריעה את השפה המשלימה של L?

<u>תשובה:</u>

לא! עבור $x \in L$ קיים ב-M לפחות מסלול אחד המסתיים במצב המקבל, אך יתכנו מסלולים רבים ב-M המסתיימים במצב הדוחה. מכאן שגם ב-N יתכנו מסלולים מקבלים עבור הקלט x. לכן יתכן ש-x יהיה שייך גם לשפה המוכרעת על ידי N. לפיכך, השפה המוכרעת על ידי המכונה N איננה בהכרח השפה המשלימה של L.

קראו את התת-סעיף "שאלת הקשר בין P ל-NP".

בתת-סעיף זה מוצגת אחת השאלות החשובות ביותר במתמטיקה ובתאוריה של מדעי המחשב. ברור לנו ש- $\mathrm{P} \subseteq \mathrm{NP}$, שהרי כל שפה שניתנת להכרעה בזמן פולינומיאלי ניתנת גם לאימות בזמן פולינומיאלי (ההכרעה היא אימות שאיננו זקוק להוכחה). אך עד היום טרם נמצאה ולו שפה אחת המצויה ב-NP שהוכח שהיא לא ב-P, כלומר, שהיא מעבר לגבולות החישוב הפולינומיאלי. לפיכך, למרבה ההפתעה, עדיין קיימת האפשרות ש- $\mathrm{P} = \mathrm{NP}$ ופשוט טרם פיתחנו את הטכניקות להכרעת בעיות קשות-למראה כמו $CLIQUE$ או $HAMPATH$ בזמן פולינומיאלי. אפשר לשער ששאלה זו תישאר ללא מענה עוד זמן רב.

תרגיל 4.9

פתרו את תרגיל 7.12 בספר הלימוד.

4.4 NP-שלמות

קראו את המבוא לסעיף 7.4 עד לפני התת-סעיף "ניתנות לרדוקציה בזמן פולינומיאלי".

כאן אנו פוגשים את המושג החשוב הקרוי **NP-שלמות**. שפה נקראת NP-שלמה אם היא שייכת ל-NP ואם היא קשה לפחות כמו כל יתר הבעיות ב-NP. במובן שאם שפה זו תימצא ביום מן הימים כשייכת ל-P, אז מיד נוכל להסיק שכל השפות ב-NP הן ב-P, כלומר ש- $\mathrm{P} = \mathrm{NP}$.

שימו לב: במבט ראשון כלל לא ברור ששפה כזו קיימת! משפט קוק-לוין (משפט 7.37), שבו נדון בהמשך פרק זה, קובע ששפה כזו אכן קיימת, ואף מצביע על שפה כזו, והיא **בעיית הספיקות** (satisfiability או בקיצור SAT) של נוסחאות בוליאניות: בהינתן נוסחה בוליאנית מעל n משתנים, יש לקבוע האם קיימת לה הצבת ערכים מתאימה (אחת מבין 2^n ההצבות האפשריות) המספקת אותה, כלומר, גורמת לה לקבל את הערך 1.

קראו את התת-סעיף "ניתנות לרדוקציה בזמן פולינומיאלי".

המכשיר המתמטי החשוב הקרוי **רדוקציה** יוצר קשר בין בעיה אחת לבעיה אחרת. בסעיף 5.3 בספר הלימוד הגדרנו רדוקציית-מיפוי משפה A לשפה B, כפונקציה **חשיבה** $f : \Sigma^* \to \Sigma^*$ הממירה מילים ב- A במילים ב- B ומילים ב- $\overline{A}$ במילים ב- $\overline{B}$. בדיוננו הנוכחי, שבו אנו עוסקים בסיבוכיות, אנו מתעניינים ברדוקציות מיפוי בעלות סיבוכיות זמן פולינומיאלית. אם קיימת רדוקציה כזו משפה A לשפה B, אנו אומרים ש- A **ניתנת לרדוקציה בזמן פולינומיאלי** ל- B ומסמנים $A \le_{\mathrm{P}} B$.

משפט 7.31 הוא משפט מרכזי וחשוב: אם $A \le_{\mathrm{P}} B$ ו- $B \in \mathrm{P}$ אז גם $A \in \mathrm{P}$. בניסוח פחות פורמלי המשפט אומר שאם הבעיה A איננה קשה יותר מהבעיה B ($A \le_{\mathrm{P}} B$), ו-B בעיה קלה ($B \in \mathrm{P}$), אז גם A בעיה קלה ($A \in \mathrm{P}$). שימו לב שמשפט 7.31 לא היה נכון אילו הרדוקציה מ-A ל-B לא הייתה פולינומיאלית! על כן, חשוב מאוד לזכור לא רק להוכיח ש- $f : \Sigma^* \to \Sigma^*$ מהווה רדוקציית-מיפוי משפה A לשפה B, אלא גם להראות שיש לה סיבוכיות זמן פולינומיאלית.

שימו לב כי יחס הניתנות לרדוקציה בזמן פולינומיאלי, $\le_{\mathrm{P}}$, הוא יחס טרנזיטיבי; כלומר, אם $A \le_{\mathrm{P}} B$ וגם $B \le_{\mathrm{P}} C$ אז $A \le_{\mathrm{P}} C$.

בהינתן שתי שפות A, B, כך שכל אחת מהן ניתנת לרדוקציה פולינומיאלית לשפה השנייה, נאמר ששתי השפות **שקולות פולינומיאלית**, ונסמן זאת כך: $A \equiv_{\mathrm{P}} B$.

<u>שאלה</u>: הראו שהיחס $\equiv_{\mathrm{P}}$ הוא יחס שקילות.

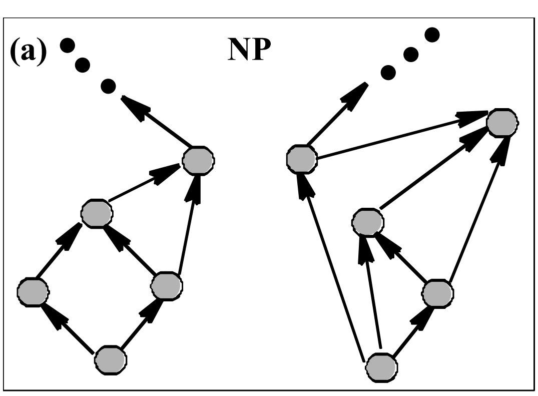
יחס השקילות הפולינומיאלית מחלק אפוא את NP למחלקות שקילות. אחת ממחלקות השקילות היא המחלקה P ללא השפה הריקה וללא שפת כל המילים Σ^*. מחד, כל שפה ששקולה פולינומיאלית לשפה ב-P חייבת אף היא להימצא ב-P, ומאידך – כל שתי שפות ב-P שאינן השפה הריקה ואינן שפת כל המילים מעל האלפבית הנתון, שקולות פולינומיאלית זו לזו.

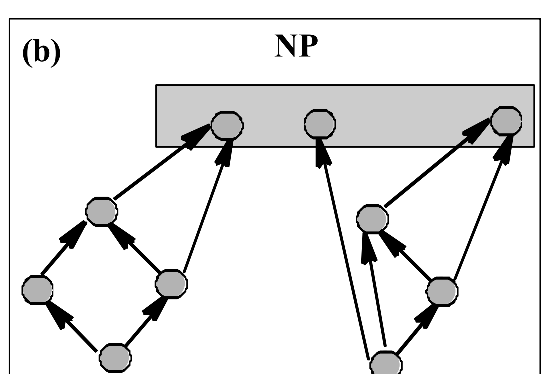
כדי לראות זאת, די אם נראה כי $A \leq_{\mathrm{P}} B$ לכל שתי שפות $A, B \in P$ כך ש- B אינה השפה הריקה ואינה Σ^*, כלומר $B, \overline{B} \neq \varnothing$. למעשה, זה אפילו נכון לכל שפה B (גם אם אינה ב-P) המקיימת $B, \overline{B} \neq \varnothing$. הנה הרדוקציה: בהינתן קלט x לבעיית השייכות ל- A, אנו צריכים לחשב בזמן פולינומיאלי קלט $f(x)$ לבעיית השייכות ל- B, כך שיתקיים: $x \in A$ אם ורק אם $f(x) \in B$. יהיו $y \in B$, $\overline{y} \in \overline{B}$ כלשהם (שימו לב כי $y, \overline{y}$ קיימים, כי $B, \overline{B} \neq \varnothing$). ובכן, פשוט נגדיר $f(x) = y$ אם $x \in A$, ו- $f(x) = \overline{y}$ אם $x \notin A$. כיוון ש- $A \in P$, ניתן לחשב את $f(x)$ בזמן פולינומיאלי, ובבירור $x \in A$ אם ורק אם $f(x) \in B$.

היחס $\leq_{\mathrm{P}}$, בהיותו טרנזיטיבי, מגדיר **היררכיה** (או **סדר**) בין השפות ב-NP. יתכן כי זהו **סדר חלקי** בלבד, במובן זה שייתכן שיש זוגות של שפות שביניהן אין יחס (כלומר, יתכנו שתי שפות A, B שעבורן לא מתקיים $A \leq_{\mathrm{P}} B$ וגם לא מתקיים $B \leq_{\mathrm{P}} A$). טבעי היה לומר כי שפה $A \in \mathrm{NP}$ היא "קשה ביותר ב-NP", אם לכל שפה אחרת $B \in \mathrm{NP}$ המקיימת $B \leq_{\mathrm{P}} A$ מתקיים $B \equiv_{\mathrm{P}} A$.

לפני משפט קוק-לוין (משפט 7.37) היו שני מצבים אפשריים (שבדיעבד התגלו כלא נכונים):

- במחלקה NP אין שפה קשה ביותר (כמו, שלמשל, בקבוצת המספרים הטבעיים אין "מספר גדול ביותר"). איור 1 (a) מדגים מצב זה: כל קדקוד באיור מתאר שפה במחלקה NP וקשת מכוונת משפה A לשפה B מציינת ש- $A \leq_{\mathrm{P}} B$.
- במחלקה NP יש מספר שפות קשות ביותר שאינן שקולות פולינומיאלית זו לזו, כמודגם באיור 1 (b). באיור זה, השפות הקשות ביותר מצוינות על ידי הקדקודים שמהם לא יוצאת אף קשת מכוונת.





איור 1: יחסים אפשריים בין בעיות ב-NP לפני משפט קוק-לוין.

בהמשך, מוגדרת השפה $3SAT$ כשפת כל הנוסחאות הבוליאניות הספיקות ב**צורת CNF** (Conjunctive Normal Form) כך שבכל **פסוקית** (clause) יש בדיוק שלושה **ליטרלים** (literals); צורה זו מסומנת כ**צורת 3CNF**. זוהי שפה חלקית לשפה SAT כיוון שהנוסחאות בה נדרשות להיות בעלות מבנה מסוים (בנוסף להיותן ספיקות).

במשפט 7.32 מודגמת רדוקציה בזמן פולינומיאלי מ-$3SAT$ ל-$CLIQUE$. הוכחת המשפט פשוטה למדי: בהינתן נוסחה ϕ בצורת 3CNF עם k פסוקיות, אנו בונים גרף שבו הקדקודים מחולקים ל- k קבוצות, כך שאין שום קשת בין שני קדקודים הנמצאים באותה קבוצה. כל אחת מקבוצות הקדקודים מתאימה לאחת מהפסוקיות ב- ϕ ומכילה שלושה קדקודים – אחד לכל ליטרל בפסוקית המתאימה. שני קדקודים מקבוצות שונות יחוברו על ידי קשת אלא אם כן הם מתאימים למשתנה ושלילתו.

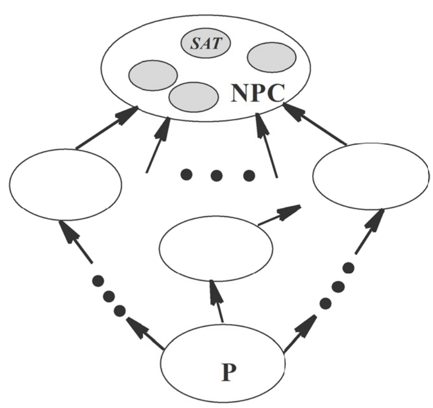
אם יש הצבה למשתני הנוסחה שעבורה ϕ מקבל ערך 1, אז בכל פסוקית יש לפחות ליטרל אחד שערכו 1. נבחר ליטרל אחד כזה מכל פסוקית, ונסתכל על קבוצת k הקדקודים המתאימים בגרף. קבוצה זו מהווה קליקה, כיוון שכל שניים מהקדקודים שבה חייבים להיות מקושרים בקשת על פי האופן שבו הוגדר הגרף. מצד שני, אם קיימת בגרף קליקה בגודל k, אז נחשוב על הצבה שבה כל ליטרל המתאים לקדקוד בקליקה זו מקבל את הערך 1. למשל, אם $k = 4$ וקדקודי הקליקה מתאימים לליטרלים $x_1, \overline{x_2}, x_3, \overline{x_2}$, אז ההצבה תהיה $x_1 = x_3 = 1$ ו-$x_2 = 0$ כאשר משתנים שאינם מיוצגים בקליקה יכולים לקבל כל ערך שהוא. אין בעיה בהגדרה זו, כיוון שלא יתכן שהקליקה תאלץ אותנו לשים ערכים סותרים לאותו משתנה, כי אין בגרף קשתות המחברות בין משתנה לשלילתו. הצבה כזו תיתן ערך 1 לנוסחה ϕ כיוון שהיא נותנת את הערך 1 לפחות לליטרל אחד בכל פסוקית של ϕ. לבסוף, קל לוודא כי הרדוקציה המתוארת לעיל אכן מתבצעת בזמן פולינומיאלי במספר הפסוקיות והמשתנים של הנוסחה המקורית ϕ.

קראו את התת-סעיף *"הגדרה של NP-שלמות"*.

בתת-סעיף זה מוגדרת מחלקת הבעיות ה-NP-שלמות. זוהי למעשה התת-מחלקה של NP המאגדת בתוכה את הבעיות הקשות ביותר ב-NP. משפט קוק-לוין מוכיח את העובדה המפתיעה כי יש בעיה קשה ביותר ב-NP, כלומר, בעיה כזו שכל בעיה ב-NP ניתנת לרדוקציה פולינומיאלית אליה. הבעיה עליה הצביעו קוק ולוין היא בעיית ספיקות הנוסחאות, SAT, שהוגדרה לפני משפט 7.27. לפיכך, הבעיה SAT היא NP-שלמה. זו הייתה ההוכחה הראשונה מאז ומעולם של NP-שלמות. בהינתן NP-שלמות של בעיה אחת (SAT במקרה זה), ניתן להוכיח NP-שלמות של בעיות אחרות במאמץ קטן בהרבה: על פי משפט 7.36, כל מה שעלינו לעשות הוא להראות רדוקציה פולינומיאלית מבעיה שכבר ידועה כ-NP-שלמה לבעיה אחרת שידוע עליה שהיא ב-NP. רדוקציה

כזו תראה שגם הבעיה האחרת היא NP-שלמה, משום שהיחס $\leq_P$ הוא יחס טרנזיטיבי. כיום אנו מכירים בעיות רבות שהן NP-שלמות.

מצב זה מתואר באיור 2, שבו מניחים כי $P \neq NP$ (נציין כי זו הדעה הרווחת כיום). שימו לב שבשכבה העליונה נמצאות כל הבעיות הקשות ביותר -- אלו הן הבעיות ה-NP-שלמות. למעשה, בין כל שתי בעיות כאלה יש קשת דו-כיוונית (כל אחת ניתנת לרדוקציה פולינומיאלית לאחרת). בשכבה התחתונה מתוארות הבעיות הקלות ביותר ב-NP, כלומר מחלקת הבעיות P. שימו לב כי יתכנו בעיות שאינן ב-P וגם אינן NP-שלמות. למעשה, כבר בשנת 1975, זמן לא רב לאחר שהוצג מושג ה-NP-שלמות, הוכיח לדנר (Ladner) שאם $P \neq NP$ אזי בהכרח ישנן רמות ביניים בין שני קצוות אלו, כלומר, יש בעיות שאינן ב-P וגם אינן NP-שלמות.



איור 2: יחסים בין בעיות ב-NP אחרי משפט קוק-לוין.
(NPC היא מחלקת הבעיות ה-NP-שלמות.)

קראו את התת-סעיף "משפט קוק-לוין" עד לפני מסקנה 7.42.

הוכחת משפט זה ארוכה ועמוסה בפרטים, אך אינה קשה להבנה. היא מזכירה במקצת את הוכחת משפט 5.15 שבו הראינו שבעיית ה-PCP איננה כריעה, על ידי כך שבנינו קלט של אבני דומינו עבור בעיית ה-PCP המחקה את פעולתה של מכונת טיורינג נתונה על מילה נתונה. כאן, בהינתן שפה A הניתנת להכרעה בזמן פולינומיאלי על ידי מכונת טיורינג אי-דטרמיניסטית N, אנו בונים לכל מחרוזת w נוסחה בוליאנית $\phi = f(w)$ המחקה את כל תרחישי הפעולה האפשריים של המכונה N על w. אם $w \in A$, כלומר, אם קיים תרחיש פעולה של המכונה N המקבל את w, אז קיימת הצבה למשתני הנוסחה כך שהנוסחה מקבלת את הערך 1 (כלומר-הנוסחה ספיקה); אחרת, ϕ תהיה בלתי-ספיקה. בנוסף, הבנייה אורכת זמן פולינומיאלי באורכה של המחרוזת w.

שלבי ההוכחה (אחרי "שלב 0" שבו מראים כי $SAT \in NP$) הם כדלקמן:

1. תיאור הרדוקציה מ-w ל-$\phi = f(w)$.

2. הוכחה ש- $w \in A$ אם ורק אם $\phi \in SAT$.
3. בדיקה שזמן החישוב של f, כלומר, התרגום של המחרוזת w ל- $\phi = f(w)$, הוא פולינומיאלי.

שלב 0: $SAT \in NP$ **:**

ראשית עלינו להראות ש-SAT היא בעיה ב-NP. אכן, בהינתן אישור בצורת הצבה למשתני הנוסחה, אפשר לבדוק בזמן פולינומיאלי את הערך שנותנת הצבה זו לנוסחה כולה. לפיכך, אם הנוסחה ספיקה, הרי שניתן לאמת זאת בזמן פולינומיאלי.

שלב 1: בניית הרדוקציה מ- w **ל-** $\phi = f(w)$ **:**

זה השלב הארוך ביותר בהוכחה. נניח ש- N היא מכונת טיורינג אי-דטרמיניסטית שמכריעה את השפה A בזמן פולינומיאלי. נניח שזמן הריצה של N על קלט באורך n הוא לכל היותר $n^k - 3$ עבור קבוע כלשהו k. בהינתן מחרוזת w באורך n, יש מספר תרחישי פעולה של N על w. אנו יכולים לתאר כל אחד מהתרחישים הללו על ידי טבלה (tableau) בגודל $n^k \times n^k$, כאשר השורה ה-i בטבלה, $0 \le i \le n^k - 1$, מתארת את הקונפיגורציה של המכונה אחרי i צעדי ריצה על הקלט w. כזכור, הקונפיגורציה של מכונת טיורינג אחרי i צעדי ריצה היא מחרוזת המכילה את התווים על הסרט בצירוף המצב שבו מצויה המכונה (המצב רשום במחרוזת לפני התו שעליו מצוי הראש הקורא-כותב; עיינו בעמוד 168 בספר הלימוד). השורה הראשונה ($i = 0$) מתארת את הקונפיגורציה ההתחלתית (עם סימני # בהתחלה ובסוף, שאותם אנו מוסיפים מטעמי נוחות, כפי שיובהר בהמשך), השורה הבאה מתארת את הקונפיגורציה אחרי צעד אחד של המכונה באותו תרחיש פעולה שאנו מתארים בטבלה זו, וכך הלאה. אם בתרחיש זה המכונה עוצרת אחרי פחות מ- $n^k - 3$ צעדים, אז כל השורות שיופיעו אחרי השורה עם הקונפיגורציה האחרונה (שקיבלה או דחתה את w) אינן חשובות; הבה נניח שהן כולן זהות לשורה שבה הופיעה הקונפיגורציה האחרונה.

שימו לב שבמהלך $n^k - 3$ צעדי ריצה, האורך המקסימלי של המחרוזת הכתובה על הסרט יכול להיות $n^k - 3$ לכל היותר (שכן מכונת טיורינג יכולה בכל צעד לכתוב רק תו אחד לסרט, ובתחילת הריצה נמצא הראש הקורא-כותב של המכונה על התא הראשון בסרט). יחד עם תיאור מצב המכונה (q_i) ושני סימני ה- # בתחילת הקונפיגורציה ובסופה, אנו מקבלים שמספר העמודות הנחוץ בטבלה כזו הוא n^k לכל היותר.

טבלה נקראת ***מקבלת***, אם באחת משורותיה מופיעה קונפיגורציה מקבלת. על מנת לקבוע אם N מקבלת את w, עלינו לקבוע אם יש ל-w טבלה מקבלת.

לאחר שהצגנו את המושג טבלה, אנו מוכנים לתיאור הרדוקציה עצמה. יהי $C = Q \cup \Gamma \cup \{\#\}$ אוסף הסימנים היכולים להופיע בתאי הטבלה (דהיינו מצב, תו סרט, או התו המיוחד #).

- המשתנים הבוליאניים בנוסחה יהיו $n^k \times n^k \times |C|$ במספר:

$$\left\{x_{i,j,s} : 1 \le i, j \le n^k, s \in C\right\}$$

המשתנה $x_{i,j,s}$ יקבל את הערך 1 אם ורק אם התא ה- (i, j) בטבלה מכיל את התו s. כלומר, אוסף המשתנים $\left\{x_{i,j,s} : 1 \le i, j \le n^k, s \in C\right\}$ מייצג את תוכן הטבלה.

- הנוסחה מורכבת מה-AND של ארבע נוסחאות, שלכל אחת מהן יש תפקיד אחר:

$$\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

- הנוסחה הראשונה, ϕ_{cell}, תקבל את הערך 1 אם ורק אם הצבת הערכים למשתנים מתארת את התוכן של טבלה מסוימת. הדרישות שמכתיבה הנוסחה ϕ_{cell} הן שבכל אחד מ- $n^k \times n^k$ תאי הטבלה יהיה בדיוק תו אחד מ- $C = Q \cup \Gamma \cup \{\#\}$, לא פחות ולא יותר.
- הנוסחה השנייה, ϕ_{start}, תקבל את הערך 1 אם ורק אם השורה הראשונה בטבלה מתארת את הקונפיגורציה ההתחלתית על המחרוזת w, כלומר אם ורק אם השורה הראשונה מכילה את התווים $\#, q_0, w_1, \ldots, w_n, \sqcup, \ldots, \sqcup, \#$ (זכרו שאצלנו התו $\sqcup$ מציין את תווי הרווח המופיעים לאחר הקלט על הסרט).
- הנוסחה ϕ_{accept} תקבל את הערך 1 אם ורק אם יש בטבלה קונפיגורציה מקבלת, כלומר, אם ורק אם אחד מתאי הטבלה מכיל את הסימן q_{accept}.
- הנוסחה ϕ_{move} תקבל את הערך 1 אם ורק אם כל קונפיגורציה בטבלה, מהשורה השנייה ואילך, היא קונפיגורציה עוקבת לזו שמופיעה בשורה מעליה. זו הנוסחה המורכבת ביותר ועליה לקודד את כל הכללים של פונקציית המעברים $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$. הרעיון הוא לסרוק את הטבלה ולוודא את החוקיות של כל "חלון" בגודל 2×3. בספר ניתנת דוגמה של שני כללים של פונקציית מעברים כלשהי ואז שש דוגמאות של חלונות חוקיים (איור 7.39) ושלוש דוגמאות של חלונות לא חוקיים (איור 7.40). כפי שמוסבר בספר, בהערה שבתחתית עמוד 307, איננו טורחים להגדיר באופן מדויק מהו חלון חוקי, מפאת החשש שמרוב עצים לא נראה את היער, ואנו מסתפקים בהבנה שהגדרה כזו היא אפשרית ובהדגמה שלה.

כפי שמוסבר לאחר ההוכחה של טענה 7.41 (שנגיע אליה עוד מעט), ניתן להרכיב רשימה של כל החלונות החוקיים שעשויים להופיע בטבלה, ממש כפי שהדגמנו באופן חלקי באיור 7.39. הבה נקרא לחלון

$cell(i, j-1)$	$cell(i, j)$	$cell(i, j+1)$
$cell(i+1, j-1)$	$cell(i+1, j)$	$cell(i+1, j+1)$

בשם החלון ה- (i,j). אז הנוסחה ϕ_{move} תדרוש שכל אחד מהחלונות בטבלה יהיה חלון חוקי. הדרישה שהחלון ה- (i,j) יהיה חוקי מבוטאת על ידי שימוש באופרטור OR שעובר על כל התכנים האפשריים של חלון חוקי. התיאור של תוכן חוקי, לעומת זאת, מתואר על ידי שימוש באופרטור AND שעובר על כל ששת התאים בחלון ומכתיב את תוכנו של כל תא. לבסוף, עושה הנוסחה ϕ_{move} שימוש באופרטור AND שעובר על כל החלונות בטבלה, $1 < j < n^k, 1 \le i < n^k$, על מנת לדרוש שכולם יהיו חוקיים.

שלב 2: הוכחה ש- $w \in A$ אם ורק אם $\phi \in SAT$:

בטענה 7.41 מראים שאם בשורה הראשונה בטבלה מופיעה הקונפיגורציה ההתחלתית וכל חלון בטבלה הוא חוקי, אז כל שורה בטבלה מכילה קונפיגורציה חוקית שניתן לקבלה מהקונפיגורציה בשורה הקודמת בטבלה על פי פונקציית המעברים δ.

לצורך כך אנו מתרכזים בשתי שורות עוקבות של הטבלה ומשווים בין תוכנן. בפסקה הראשונה בעמוד 309 מוסבר מדוע כל תו בקונפיגורציה העליונה, שאיננו סמוך לתו המצב ושאיננו התו #, חייב להישאר כפי שהוא, ובאותו מקום על הסרט, גם בקונפיגורציה התחתונה. זאת משום שבכל חלון חוקי שבו בשורה הראשונה אין תו מצב, בשורה השנייה חייב להופיע בתא האמצעי אותו תו המופיע בתא שמעליו. כלומר, אם $a,c \in \Gamma \cup \{\#\},\ b \in \Gamma$, אז כל חלון חוקי שבשורה הראשונה שלו מופיעים התווים $\boxed{\begin{matrix} a & b & c \\ ? & ? & ? \end{matrix}}$ חייב להכיל בתא האמצעי בשורה השנייה אותו תו שמופיע בשורה הראשונה, $\boxed{\begin{matrix} a & b & c \\ ? & b & ? \end{matrix}}$.

אחר כך, בפסקה השנייה, מטופלים שני התווים בקונפיגורציה העליונה הסמוכים לתו המצב, מימינו ומשמאלו. כאן אנו בוחרים להתרכז בחלונות שבהם תו המצב מופיע בתא האמצעי למעלה: $\boxed{\begin{matrix} a & q & b \\ ? & ? & ? \end{matrix}}$ (כאן $a,b \in \Gamma \cup \{\#\}$ ואילו $q \in Q$). מכיוון שהחלונות החוקיים מצורה זו מקודדים את כל אופני הפעולה של פונקציית המעברים, מובטח לנו גם כאן שתוכן הקונפיגורציה התחתונה נובע מתוכן הקונפיגורציה העליונה על פי פונקציית המעברים.

כעת, נניח ש- $w \in A$. אז קיים תרחיש פעולה של המכונה N על המילה w המסתיים במצב מקבל. לפיכך, קיימת טבלה (ϕ_{cell}) חוקית (ϕ_{move}) המתחילה ב- w (ϕ_{start}) ומקבלת אותה (ϕ_{accept}). על כן $\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$ ספיקה, הווי אומר $\phi \in SAT$.

מצד שני, אם $\phi \in SAT$, אז כל ארבע התת-נוסחאות ספיקות יחדיו. לכן קיימת טבלה (ϕ_{cell}) חוקית (ϕ_{move}) המתחילה ב- w (ϕ_{start}) ומקבלת אותה (ϕ_{accept}). לפיכך, קיים תרחיש פעולה של המכונה N על המילה w המסתיים במצב מקבל. על כן $w \in A$.

שלב 3: הוכחה שזמן הריצה של הרדוקציה $w \mapsto \phi = f(w)$ הוא פולינומיאלי:

חלק זה מוסבר בפשטות בשלוש הפסקאות האחרונות בהוכחה, בעמודים 309 ו-310.

קראו את מסקנה 7.42 ואת ההוכחה שלה.

במסקנה זו מראים כיצד ניתן לעדכן את הוכחת משפט קוק-לוין כך שהנוסחה הבוליאנית ϕ תהיה בצורת 3CNF, על מנת להוכיח שגם $3SAT$ היא NP-שלמה.

בשלב ראשון מראים שניתן לרשום את ϕ בצורת CNF, כלומר AND של פסוקיות שכל אחת מהן היא OR של ליטרלים:

- ϕ_{cell} היא כבר בצורת CNF.
- ϕ_{start} היא כבר בצורת CNF, אם נסתכל על כל אחד מהמשתנים המופיעים בה כפסוקית בגודל 1.
- ϕ_{accept} היא כבר בצורת CNF ויש בה פסוקית אחת.
- ϕ_{move} היא הנוסחה היחידה המצריכה מעט עבודה, על ידי שימוש בחוק הפילוג

$$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$$

בשלב השני מוסבר איך ניתן למפות נוסחה ϕ בצורת CNF לנוסחה ϕ' בצורת 3CNF, כך ש ϕ ספיקה אם ורק אם ϕ' ספיקה. זה משלים את תיאור הרדוקציה הפולינומיאלית מכל שפה ב-NP לשפת $3SAT$.

4.5 בעיות NP-שלמות נוספות

קראו את המבוא לסעיף 7.5.

בהינתן בעיית הכרעה, כיצד נוכיח כי היא NP-שלמה? ראשית נוכיח כי הבעיה ב-NP על ידי הצגת אישור מתאים הניתן לאימות בזמן פולינומיאלי או על ידי הצגת מכונה לא דטרמיניסטית מכריעה בעלת זמן ריצה פולינומיאלי. שנית, נבצע רדוקציה שזמן ריצתה פולינומיאלי מבעיה כלשהי שאנו יודעים שהיא NP-שלמה לבעיה הנדונה. כך, למשל, בעיית הקליקה, $CLIQUE$, היא בעיה NP-שלמה (מסקנה 7.43). למסקנה זו הגענו באופן הבא:

- $CLIQUE$ היא בעיה ב-NP (משפט 7.24);
- $3SAT \leq_{\text{P}} CLIQUE$ (משפט 7.32);
- $3SAT$ היא NP-שלמה (מסקנה 7.42).

בסעיף זה נפגוש חמש בעיות נוספות שהן NP-שלמות:

בעיית הקבוצה הבלתי תלויה (*INDEPENDENT-SET*)

קלט: גרף לא מכוון $G=(V,E)$ ומספר טבעי k.

שאלה: האם יש ב-G קבוצה בלתי תלויה בגודל k?

(קבוצת קדקודים $S \subseteq V$ בגרף $G=(V,E)$ היא בלתי תלויה אם אין ב-S שני קדקודים שהם שכנים ב-G.)

בעיית הכיסוי בקדקודים (*VERTEX-COVER*)

קלט: גרף לא מכוון $G=(V,E)$ ומספר טבעי k.

שאלה: האם יש ל-G כיסוי בקדקודים בגודל k?

(קבוצת קדקודים $S \subseteq V$ בגרף $G=(V,E)$ היא כיסוי בקדקודים של G אם לכל קשת ב-G יש לפחות קצה אחד ב-S.)

בעיית הכיסוי בקבוצות (*SET-COVER*)

קלט: קבוצה סופית U, משפחה $\{C_1,\ldots,C_m\}$ של m תת-קבוצות של U, ומספר טבעי k.

שאלה: האם ניתן לבחור k תת-קבוצות מתוך המשפחה הנתונה כך שאיחודן הוא U?

בעיית המסלול ההמילטוני (*HAMPATH*)

קלט: גרף מכוון $G=(V,E)$ ושני קדקודים s ו-t $(s,t \in V)$.

שאלה: האם יש ב-G מסלול המילטוני (מסלול פשוט העובר בכל קדקוד בדיוק פעם אחת) שמתחיל ב-s ומסתיים ב-t?

(הגרסה הלא מכוונת של *HAMPATH* נקראת *UHAMPATH*.)

בעיית הסכום של תת-קבוצה (*SUBSET-SUM*)

קלט: קבוצה $S=\{x_1,\ldots,x_n\}$ של מספרים טבעיים ומספר טבעי t.

שאלה: האם S מכילה תת-קבוצה $\{y_1,\ldots,y_l\} \subseteq S$ שסכום המספרים בה $\sum_{i=1}^{l} y_i$ הוא בדיוק t?

ההוכחה כי שלוש הבעיות הראשונות הן NP-שלמות תתבצע באופן הבא. ראשית, נראה כי כל אחת מהן היא ב-NP. לאחר מכן נוכיח את שרשרת הרדוקציות:

$$CLIQUE \le_P INDEPENDENT-SET \le_P VERTEX-COVER \le_P SET-COVER.$$

כיוון שכבר הוכחנו כי $CLIQUE$ היא NP-שלמה, נקבל כי שלוש הבעיות לעיל גם הן NP-שלמות. אלו הן רדוקציות קלות יחסית, ולכן הן ייעשו בתרגילים.

ההוכחה ששתי הבעיות $HAMPATH$ ו- $SUBSET-SUM$ הן NP-שלמות תתבצע על ידי הדגמת רדוקציה מבעיית $3SAT$ אליהן, כאשר הרעיון הוא לדמות את המשתנים הבוליאניים והפסוקיות בנוסחאות הבוליאניות באמצעות מבנים מתאימים בשפה הנידונה הקרויים **עזרים** (gadgets).

הערה:

אם אתם תוהים מדוע התמקדנו בבעיית $3SAT$ ולא התחלנו את שרשרת הרדוקציות שלנו עם $2SAT$ – הסיבה היא פשוטה מאוד: $2SAT \in \mathrm{P}$. כלומר, ידוע אלגוריתם פולינומיאלי הפותר את $2SAT$. נעיר כי "סף" דומה קיים גם לבעיות רבות אחרות, למשל:

***k*-צביעה של גרף** ($k-COLORING$): בהינתן גרף לא מכוון G ומספר טבעי k, האם ניתן לצבוע את קדקודי G ב- k צבעים, כך שכל שני קדקודים שכנים ב- G יהיו צבועים בצבעים שונים? עבור $k=2$ הבעיה שקולה לבדיקה האם הגרף הוא דו-צדדי, ולכן ניתנת לפתרון בזמן פולינומיאלי. עבור $k=3$, הבעיה היא NP-שלמה (ראו בעיה 7.38 בספר).

אריזה של קבוצות בגודל *k* ($k-SET-PACKING$): בהינתן אוסף $\{C_1,\ldots,C_m\}$ של תת-קבוצות של קבוצה U, כל אחת בגודל k, ומספר טבעי t, האם יש באוסף t תת-קבוצות שהן זרות בזוגות? עבור $k=2$ הבעיה שקולה לבעיית זיווג המקסימום, ולכן ניתנת לפתרון בזמן פולינומיאלי. עבור $k=3$, הבעיה היא NP-שלמה.

תרגיל 4.10

1) הוכיחו כי בעיית הקבוצה הבלתי תלויה היא NP-שלמה.
2) הוכיחו כי בכל גרף $G=(V,E)$, S היא קבוצה בלתי תלויה אם ורק אם קבוצת הקדקודים המשלימה $V \setminus S$ מכסה את קשתות הגרף (כלומר, אחד הקצוות של כל קשת בגרף G נמצא ב- $V \setminus S$).
3) הסיקו מכאן כי בעיית הכיסוי בקדקודים אף היא NP-שלמה.

תרגיל 4.11

הוכיחו כי $SET-COVER$ היא בעיה NP-שלמה.

סעיף רשות: המשיכו לקרוא את התת-סעיף "בעיית הכיסוי בקדקודים".

במשפט 7.44 מוכיחים שבעיית הכיסוי בקדקודים היא NP-שלמה. כפי שכבר ראינו לעיל, ניתן להוכיח זאת בקלות על ידי סדרת הרדוקציות

$$CLIQUE \leq_P INDEPENDENT-SET \leq_P VERTEX-COVER$$

(ראו תרגיל 4.10 לעיל). ההוכחה בספר הלימוד, לעומת זאת, היא הוכחה באמצעות רדוקציה ישירות מבעיית $3SAT$. זהו מאמץ מיותר, ולכן הנכם יכולים לוותר על קריאת ההוכחה, אך כתרגול נוסף של טכניקת ההוכחה באמצעות רדוקציות, הבה נסקור כאן את הוכחת משפט 7.44. נתרכז רק בדיון ברדוקציה:

נניח שנוסחת ה-3CNF היא $\phi = \phi_1 \wedge \cdots \wedge \phi_l$ כאשר $\phi_i = y_{i,1} \vee y_{i,2} \vee y_{i,3}$ הן הפסוקיות, $y_{i,j} \in \{x_1, \overline{x_1}, \cdots, x_m, \overline{x_m}\}$ הם הליטרלים ו-x_i הם המשתנים. נתאר את הצמד $\langle G,k \rangle$ המתאים לנוסחה זו, כאשר $G=(V,E)$ הוא גרף לא מכוון ו- k מספר טבעי.

בגרף יהיו $2m+3l$ קדקודים כדלקמן:

$$V = \{x_i, \overline{x_i} : 1 \leq i \leq m\} \cup \{y_{i,1}, y_{i,2}, y_{i,3} : 1 \leq i \leq l\}$$

כלומר, לכל משתנה x_i יהיו שני קדקודים, האחד מסומן x_i והאחר מסומן $\overline{x_i}$. כמו כן, לכל פסוקית נקצה שלושה קדקודים שכל אחד מהם מסומן על ידי אחד הליטרלים המופיעים בפסוקית.

קשתות הגרף יהיו כדלקמן:

1. קבוצת הקשתות הראשונה: קשת בין כל שני קדקודים המתייחסים לאותו משתנה: $(x_i, \overline{x_i})$, $1 \leq i \leq m$.
2. קבוצת הקשתות השנייה: שלוש קשתות בין כל שלושה קדקודים המייצגים את הליטרלים של אותה פסוקית: $(y_{i,1}, y_{i,2}), (y_{i,2}, y_{i,3}), (y_{i,3}, y_{i,1})$, $1 \leq i \leq l$.
3. קבוצת הקשתות השלישית: קשת בין כל קדקוד של ליטרל בפסוקית לקדקוד של אותו ליטרל בקבוצת הקדקודים של המשתנים (ה-x_i-ים). כלומר, לכל $1 \leq i \leq l$ ו- $1 \leq j \leq 3$, אם $y_{i,j} = x_r$, אז נוסיף את הקשת $(y_{i,j}, x_r)$, אחרת- אם $y_{i,j} = \overline{x_r}$, נוסיף את הקשת $(y_{i,j}, \overline{x_r})$.

לבסוף, אנו מגדירים את המספר הטבעי המתאים והוא $k = m + 2l$.

ברור שזמן הריצה של רדוקציה זו הוא פולינומיאלי באורך של ϕ, ולכן אנו ממשיכים להוכחה שמדובר ברדוקציית מיפוי תקפה מ- $3SAT$ ל-$VERTEX\text{-}COVER$.

כיוון ראשון: $\phi \in 3SAT \Rightarrow \langle G,k \rangle \in VERTEX-COVER$ **:**

נניח של- ϕ יש הצבה מספקת. אז אנו יכולים לבחור בכל פסוקית ליטרל אחד שערכו, תחת הצבה זו, שווה 1 (יתכן שיש כמה ליטרלים שערכם תחת ההצבה שווה 1, אבל תמיד נוכל לבחור אחד); נקרא לליטרל זה הליטרל המסומן. אנו טוענים שבמקרה זה יש ב- G קבוצה של k קדקודים המכסה את כל הקשתות: נשים בקבוצה זו את m קדקודי המשתנים שערכם 1 ואת $2l$ קדקודי הליטרלים שאינם מסומנים. למשל, אם בדוגמה המופיעה באיור 7.45 בספר הלימוד ההצבה היא $x_1=0, x_2=1$, אז הקדקודים שייבחרו הם $\overline{x_1}$ ו- x_2 מהחלק העליון של הגרף, x_1, x_1 מתוך שלישיית קדקודי הליטרלים הראשונה (בחלק התחתון של הגרף משמאל), $\overline{x_2}, \overline{x_2}$ מתוך שלישיית קדקודי הליטרלים השנייה, ו- $\overline{x_1}, x_2$ מהשלישייה האחרונה (בשלישייה זו אנו יכולים לבחור כל שני קדקודים כיוון שכולם בעלי ערך 1 תחת הצבה זו).

קל לראות שאוסף זה של קדקודים מכסה את כל קשתות הגרף. הקשתות מהקבוצה הראשונה מכוסות על ידי m קדקודי המשתנים שערכם 1. הקשתות מהקבוצה השנייה מכוסות על ידי $2l$ קדקודי הליטרלים שאינם מסומנים (שכן בכל משולש שני קדקודים מכסים את כל שלוש הקשתות). לבסוף, הקשתות מהקבוצה השלישית מתחלקות לשתי תת-קבוצות: אלו שקשורות לקדקוד ליטרל מסומן מכוסות על ידי קדקוד המשתנה המתאים (שערכו 1 ולכן הוא שייך לכיסוי) ואילו הקשתות שקשורות לקדקודי הליטרלים שאינם מסומנים מכוסות על ידי קדקודים אלו. (שימו לב: זה המקום ברדוקציה שבו אנו משתמשים בעובדה כי ההצבה שבעזרתה בחרנו את הקדקודים המסומנים הינה הצבה מספקת).

כיוון שני: $\langle G,k \rangle \in VERTEX-COVER \Rightarrow \phi \in 3SAT$ **:**

נניח שיש ב- G קבוצה של $k=m+2l$ קדקודים המכסה את כל הקשתות. על מנת לכסות את כל m הקשתות מקבוצת הקשתות הראשונה אנו חייבים לבחור לכל משתנה x_i לפחות אחד מבין שני הקדקודים המייצגים אותו – x_i או $\overline{x_i}$. על מנת לכסות את כל $3l$ הקשתות מקבוצת הקשתות השנייה, אנו חייבים לבחור לכל פסוקית לפחות שני קדקודי ליטרלים. אילוצים אלו קובעים שכל כיסוי של G בעל k קדקודים חייב לכלול לכל $1 \le i \le m$ בדיוק אחד מבין הקדקודים x_i או $\overline{x_i}$, ולכל $1 \le i \le l$ בדיוק שניים מבין קדקודי הליטרלים $\{y_{i,1}, y_{i,2}, y_{i,3}\}$.

נסתכל על ההצבה המתאימה שתתן ערך 1 לכל קדקודי המשתנים שבכיסוי זה. ניתן לוודא שהצבה זו חייבת לתת ערך 1 לכל ליטרל שאיננו בכיסוי, כיוון שאם לא כן, אז הקשת המחברת בין ליטרל כזה לקדקוד המשתנה המתאים איננה מכוסה. לכן יש בכל אחת מהפסוקיות לפחות ליטרל אחד בעל ערך 1; משמע, הצבה זו מספקת את הנוסחה.

המשיכו לקרוא את התת-סעיף "בעיית המסלול ההמילטוני" עד לסוף הוכחת משפט 7.46.

בהוכחת משפט זה אנו רואים רדוקציית מיפוי בזמן פולינומיאלי מבעיית $3SAT$ לבעיית $HAMPATH$, שבה אנו מקבלים גרף מכוון ושני קדקודים בו, ועלינו להכריע אם קיים מסלול המילטוני המקשר שני קדקודים אלה (כלומר, מסלול מכוון המתחיל בקדקוד s, עובר דרך כל קדקודי הגרף ללא חזרות ומסתיים בקדקוד t).

תיאור הרדוקציה בספר מאיר עיניים ועל כן נסקור אותו כאן בקצרה. בהינתן נוסחה בוליאנית ϕ בצורת 3CNF בעלת l משתנים ו- k פסוקיות, אנו בונים גרף G עם קדקודים וקשתות כמתואר, באופן חלקי, באיור 7.49:

- לכל משתנה אנו מקדישים מעוין של קדקודים המקושרים בקשתות בצורת זיג-זג, כמתואר באיור 7.47 עבור משתנה אחד.
- לכל פסוקית אנו מקדישים קדקוד אחד בלבד "העומד בצד".
- מספר הקדקודים הניצבים בשורה האופקית בכל אחד מן המעוינים הוא $3k+1$ (וזאת מבלי למנות את שני הקדקודים המהווים את קדקודי המעוין). כפי שניתן לראות באיור 7.50, קדקודים אלו הם (משמאל לימין):
  - קדקוד מפריד;
  - 2 קדקודים לפסוקית c_1;
  - קדקוד מפריד;
  - 2 קדקודים לפסוקית c_2;
  - קדקוד מפריד;
  - ...;
  - קדקוד מפריד;
  - 2 קדקודים לפסוקית c_k;
  - קדקוד מפריד.
- אם המשתנה x_i מופיע בפסוקית c_j בצורה חיובית, אנו מוסיפים קשתות מהמבנה המייצג אותו לקדקוד המייצג את אותה פסוקית כמתואר באיור 7.51. אם המשתנה x_i מופיע בפסוקית c_j בצורה שלילית, אנו מוסיפים קשתות מהמבנה המייצג אותו לקדקוד המייצג את אותה פסוקית כמתואר באיור 7.52.
- לבסוף, קדקודי המקור והיעד, s ו- t, הם הקדקודים בראש ובתחתית מגדל המעוינים, כמתואר באיור 7.49.

זוהי בנייה בעלת סיבוכיות פולינומיאלית. נותר רק להראות ש- ϕ ספיקה אם ורק אם יש בגרף מסלול המילטוני מ-s ל- t.

נניח ש- ϕ ספיקה. אז אנו מתארים תחילה את עמוד השדרה של המסלול - המסלול ש"מזגזג" דרך כל קדקודי הגרף, למעט קדקודי הפסוקיות שעומדים בצד. צורת הזיגזוג שנבחרת מוכתבת על ידי ההצבה: **זיג-זג** ימינה דרך קדקודי המשתנים שערכם תחת ההצבה המספקת הוא 1 **וזיג-זג** שמאלה דרך קדקודי המשתנים שערכם 0, כמתואר באיור 7.53. כעת עלינו להוסיף **מעקפים** (detours) על מנת לכלול במסלול זה את קדקודי הפסוקיות. מעקפים אלו ניתנים למימוש באמצעות הקשתות המסתעפות אל קדקודים אלו כפי שתואר באיורים 7.51 ו-7.52. אנו נבחר בכל פסוקית את אחד הליטרלים בה שערכו 1 (ויש לפחות אחד כזה) ואז נכלול את אותה פסוקית במסלול על ידי הוספת שתי קשתות המעקפים מקדקודי המשתנה המתאים. האופן שבו קבענו את כיווני הקשתות מבטיח שהמעקפים הללו משתלבים בכיוון הזרימה לאורך עמוד השדרה של המסלול שתואר לעיל.

נניח כעת שקיים בגרף מסלול המילטוני. כפי שמוסבר בפסקה האחרונה של ההוכחה, מסלול כזה חייב להיות **נורמלי** במובן שהמצב המתואר באיור 7.54 הוא בלתי אפשרי. כלומר, אם אנו "קופצים" הצידה לקדקוד של פסוקית מתוך מעוין המייצג משתנה אחד, לא יתכן שנחזור אל קדקוד הכלול במעוין המייצג משתנה אחר, משום שאז לא נוכל לכסות את אחד הקדקודים בגרף (a_2 בדוגמה שבאיור 7.54). לפיכך, אנו יכולים "לקרוא" מתוך המסלול את ההצבה המתאימה למשתנים: אם המסלול עושה זיג-זג ימינה דרך קדקודי המשתנה x_i אז $x_i = 1$; אחרת, $x_i = 0$. כמו כן, המעקפים הכוללים את הפסוקית c_j מראים לנו איזה ליטרל באותה פסוקית מקבל את הערך 1 תחת ההצבה הזו.

סיימו את קריאת התת-סעיף "בעיית המסלול ההמילטוני".

במשפט 7.55 אנו רואים שגם הגרסה הלא מכוונת של בעיית המסלול ההמילטוני, *UHAMPATH*, היא NP-שלמה. ההוכחה מסתמכת על רדוקציה פשוטה מבעיית *HAMPATH*.

קראו את התת-סעיף "בעיית הסכום של תת-קבוצה".

גם כאן, ההוכחה שהבעיה הנדונה, *SUBSET-SUM*, היא NP-שלמה היא באמצעות רדוקציה מ-*3SAT*.

תהי ϕ נוסחה מעל l משתנים, $x_1,\dots,x_l$, ובעלת k פסוקיות, $c_1,\dots,c_k$. הרדוקציה המתוארת ממפה נוסחה זו לקלט עבור הבעיה *SUBSET-SUM*. קלט זה הוא הזוג הסדור $\langle S,t \rangle$, כאשר S היא קבוצה של $2l+2k$ מספרים טבעיים ו- t הוא מספר היעד – אותו ערך שעבורו יש למצוא קבוצה חלקית של S שסכומה שווה לו.

מתארים את קבוצת המספרים S ואת מספר היעד t באמצעות טבלה, כפי שמודגם באיור 7.57. כל שורה בטבלה זו מתארת את אחד המספרים. כל המספרים בקבוצה S הם מספרים עשרוניים שספרותיהם הן 1 או 0 בלבד. המספר t הוא מספר עשרוני שספרותיו הן 1 ו-3 בלבד. מספר

הספרות במספרים אלו הוא לכל היותר $l+k$. לכל מספר $s \in S$ נסמן ב- s_L את l הספרות השמאליות במספר וב- s_R את k הספרות הימניות בו. כלומר, $s = 10^k \cdot s_L + s_R$.

המספרים ב- S הם כדלקמן:

1. למשתנה x_i מתאימים שני מספרים ב- S, המסומנים y_i ו- z_i, $1 \le i \le l$. החלק השמאלי שלהם הוא $y_{i,L} = z_{i,L} = 10^{l-i}$ (למשל, אם $l=6$ אז $y_{1,L} = z_{1,L} = 100000$ ואילו $y_{6,L} = z_{6,L} = 1$). החלק הימני ב- y_i (z_i) הוא מספר עשרוני בן k ספרות, כאשר הספרה ה- j משמאל היא 1 אם x_i מופיע בצורה חיובית (שלילית) בפסוקית c_j, ואפס - אחרת.
2. לפסוקית c_j מתאימים שני מספרים שווים ב- S, המסומנים g_j ו- h_j, $1 \le j \le k$ (זכרו שהקבוצה S היא למעשה קבוצה מוכללת – multiset, שיכולה להכיל ערכים החוזרים על עצמם). החלק השמאלי במספרים אלו שווה אפס, ואילו החלק הימני נתון על ידי $g_{j,R} = h_{j,R} = 10^{k-j}$.

לבסוף, המספר t נתון על ידי הייצוג העשרוני $t = \overbrace{1\cdots1}^{l \text{ times}}\overbrace{3\cdots3}^{k \text{ times}}$.

הרעיון במספרים אלו הוא שניתן לחבר אותם בכל עמודה בנפרד מבלי שנקבל ערך נגרר (carry) מעמודה אחת לשנייה, משום שבכל עמודה יש לכל היותר 5 אחדים (שהרי בכל פסוקית יש לכל היותר שלושה ליטרלים שערכם 1).

אם יש לנוסחה הצבה מספקת, נבחר את l המספרים מקרב $\{y_i, z_i : 1 \le i \le l\}$ המתאימים לליטרלים שיש להם ערך 1 תחת הצבה זו; נוסיף להם מספר מתאים של מספרים מקרב $\{g_j, h_j : 1 \le j \le k\}$ כדי שהסכום בחלק הימני של המספר יתן את הספרה 3 בכל עמודה כנדרש. לפיכך, אם $\phi \in 3SAT$ אז $\langle S, t \rangle \in SUBSET-SUM$.

מצד שני, אם יש תת-קבוצה של S שסכומה שווה ל- t, הרי שמתוך עיון בחלק השמאלי של המספרים ברור שבתת-קבוצה זו יש בדיוק מספר אחד מבין $\{y_i, z_i\}$ לכל $1 \le i \le l$; הבחירה הזו בין y_i ו- z_i לכל $1 \le i \le l$ מכתיבה הצבה למשתנים $x_1, \ldots, x_l$. מתוך עיון בחלק הימני של המספרים ברור שבעמודה המתאימה לכל פסוקית חייב להיות לפחות ליטרל אחד שערכו 1, על מנת שהסכום באותה עמודה יגיע לערך 3, כפי שמופיע באותה עמודה במספר היעד t. לכן קיבלנו תיאור של הצבה המספקת את ϕ.

לבסוף, עלות הרדוקציה היא $O\left(n^2\right)$, כאשר n הוא אורך התיאור של הקלט ϕ, ולכן מדובר ברדוקציה פולינומיאלית.

סיכום:

הבה נחזור ונסכם את השלבים בהוכחה כי שפה מסוימת B היא NP-שלמה:

1. מוכיחים ש- B היא ב-NP.
2. מוצאים שפה מתאימה A שידוע עליה שהיא NP-שלמה ומראים ש- $A \le_P B$ על ידי הצגת רדוקציה פולינומיאלית מ- A ל- B. הוכחת הרדוקציה היא כדלקמן:

א. מגדירים פונקציה f הממפה מופעים המתאימים לשפה A למופעים המתאימים לשפה B (כלומר, אם השפה A מורכבת, למשל, מנוסחאות בוליאניות בעלות תכונה מסוימת והשפה B מורכבת מגרפים בעלי תכונה אחרת, אז הפונקציה f ממפה נוסחאות בוליאניות לגרפים).

ב. מראים ש- $x \in A$ אם ורק אם $f(x) \in B$.

ג. מוכיחים שזמן הריצה של f הוא פולינומיאלי באורך הקלט.

תרגיל 4.12

פתרו את בעיה 7.45 בספר הלימוד. עליכם להראות שאם P=NP אז קיימת רדוקציית מיפוי בזמן פולינומיאלי מכל $B \in \mathrm{NP}$ לכל $A \in \mathrm{NP} - \{\varnothing, \Sigma^*\}$.

תרגיל 4.13

פתרו את בעיה 7.48 בספר הלימוד. (מסלול פשוט הוא מסלול שאין בו לולאות, כלומר מסלול שכל הקדקודים לאורכו שונים זה מזה.)

תרגיל 4.14

פתרו את בעיה 7.49 בספר הלימוד.

תרגיל 4.15

פתרו את בעיה 7.50 בספר הלימוד.

תרגיל 4.16

פתרו את בעיה 7.29 בספר הלימוד.

תרגיל 4.17

פתרו את בעיה 7.36 בספר הלימוד.

שימו לב: המחלקות P ו- NP כוללות **שפות**, או **בעיות הכרעה**. שפה היא תת-קבוצה של Σ^*, ובעיית ההכרעה המתאימה לה היא הבעיה שבה אנו מקבלים מחרוזת $w \in \Sigma^*$ ועלינו להכריע אם המחרוזת שייכת לשפה.

לעומת זאת, במקרה שלפנינו עלינו לפתור **בעיית חישוב**. בהינתן ייצוג בינארי של מספר, אין זה מספיק להוציא כפלט ביט אחד בלבד המציין אם הקלט הוא ראשוני או לא; עלינו לתת את הפירוק המלא שלו לגורמים! עליכם להראות שאם P=NP, אז קיים אלגוריתם פולינומיאלי

דטרמיניסטי המוצא את הפירוק המלא לגורמים של כל מספר נתון (אלגוריתם כזה כמובן חזק יותר מאלגוריתם ההכרעה שרק בודק ראשוניות).

<u>הדרכה:</u> היעזרו בשפה הזאת - $F = \{\langle a,b,c \rangle : \exists p \in [b,c] \text{ such that } p \mid a\}$.

פתרון התרגילים

תרגיל 4.1

התשובות לשאלות בתרגיל 7.1: כן, לא, לא, כן, כן, כן.

התשובות לשאלות בתרגיל 7.2: לא, כן, כן, כן, לא, לא.

תרגיל 4.2

חלק א:

10505=1274*8+313

1274=313*4+22

313=22*14+5

22=5*4+2

5=2*2+1

2=1*2+0

לכן $\gcd(10505,1274) = 1$, ולפיכך $\langle 10505,1274\rangle \in RELPRIME$.

חלק ב:

8029=7289*1+740

7289=740*9+629

740=629*1+111

629=111*5+74

111=74*1+37

74=37*2+0

לכן $\gcd(8029,7289) = 37$, ולפיכך $\langle 8029,7289\rangle \notin RELPRIME$.

תרגיל 4.3

הטבלה היא:

$$\begin{pmatrix} \{T\} & \{T,R\} & \{S\} & \{S,R,T\} \\ & \{R\} & \{S\} & \{S\} \\ & & \{T\} & \{T,R\} \\ & & & \{R\} \end{pmatrix}$$

למשל, נניח שכבר מילאנו את האלכסון הראשון (המתייחס לתת-מחרוזות באורך 1) ואת האלכסון השני (תת-מחרוזות באורך 2) וכעת ברצוננו למלא את הכניסה הראשונה $table(1,3)$ באלכסון השלישי. יש שני אופני שבירה של התת-מחרוזת $w_1w_2w_3 = \text{bab}$: האחד הוא ab || b והשני הוא b || ba.

אופן השבירה הראשון: b נוצר רק מהמשתנה T ו-ab נוצר רק מהמשתנה S. כיוון שאין שום כלל המרה הממיר משתנה במשתנים TS, דרך שבירה זו אינה מניבה דבר.
אופן השבירה השני: ba נוצר רק מהמשתנים T, R ו-b נוצר רק מהמשתנה T. כיוון שיש רק כלל המרה אחד הממיר משתנה ב- TT או ב- RT, וזהו הכלל $S \to RT$, אז $table(1,3) = \{S\}$.

תרגיל 4.4

שלב 1 באלגוריתם מאתר בתיאור הגרף את הקדקוד הראשון ברשימת הקדקודים ומסמן אותו. סיבוכיות פעולה זו היא $O(n)$. שלב 2 באלגוריתם הוא לולאה. מספר הפעמים שבהן נחזור על פעולות לולאה זו הוא לכל היותר n מכיוון שבכל פעם, למעט האחרונה, אנחנו מסמנים לפחות קדקוד אחד מבין n הקדקודים. גוף הלולאה הוא בעל סיבוכיות $O(n^3)$ כיוון שעלינו לעבור על n קדקודים, ולכל אחד מהם לבדוק $O(n^2)$ קשתות. לכן, הסיבוכיות הכוללת של הלולאה היא $O(n^4)$. לבסוף, סיבוכיות שלב 4 היא $O(n)$. לפיכך, הסיבוכיות הכוללת של האלגוריתם להכרעת השפה $CONNECTED$ היא $O(n^4)$, ולכן שפה זו היא ב-P.

תרגיל 4.5

האלגוריתם הנאיבי לחישוב $a^b \bmod p$ יכפיל את a בעצמו $b-1$ פעמים. אך זמן הריצה של אלגוריתם כזה הוא מעריכי באורך הייצוג הבינארי של b (אורך הייצוג הבינארי של b הוא $\lfloor \log b \rfloor + 1$). במקום זה, נציג אלגוריתם פולינומיאלי לחישוב חזקה מודולרית, בהתבסס על הרעיון של העלאה חוזרת בריבוע. לצורך כך, נניח שהייצוג הבינארי של b הוא $b = b_k b_{k-1} \cdots b_0$ (כלומר, b_k היא הסיבית המשמעותית ביותר, b_0 היא הסיבית הכי פחות משמעותית).
החישוב יתבצע כך:

1. $r = 1$
2. For $i = k, \ldots, 0$ do:
   a. If $b_i = 0$ then $r = r^2 \bmod p$
   b. If $b_i = 1$ then $r = ar^2 \bmod p$

בסוף החישוב, נבדוק אם $c = r \bmod p$. אם כן, נקבל את הקלט $\langle a, b, c, p \rangle$. אחרת, נדחה אותו.
האלגוריתם הוא פולינומיאלי, כיוון שהלולאה מתבצעת $O(\log b)$ פעמים, וכל אחת משתי הפעולות בלולאה ניתנת ליישום בזמן פולינומיאלי. עתה נסביר מדוע האלגוריתם נכון. כל מעריך המחושב במהלך השגרה הוא כפול מהמעריך הקודם או גדול ב-1 מפעמיים המעריך הקודם.

הייצוג הבינארי של b נקרא משמאל לימין, וערכה של הסיבית b_i קובע אלו פעולות יש לבצע. בכל שלב בלולאה אנו משתמשים בזהות:

$$a^{2d} \bmod p = \left(a^d\right)^2 \bmod p$$

אם $b_i = 0$, או בזהות:

$$a^{2d+1} \bmod p = a \cdot \left(a^d\right)^2 \bmod p$$

אם $b_i = 1$.

למעשה, השגרה מחשבת בכל פעם את $a^d \bmod p$ כאשר d גדל מ- 0 עד b באמצעות העלאות בריבוע והכפלות ב- a.

תרגיל 4.6

איחוד: $A, B \in \mathrm{P} \Rightarrow A \cup B \in \mathrm{P}$

בהינתן קלט, נריץ עליו את שני האלגוריתמים הפולינומיאליים המכריעים את A ו-B. אם הוא שייך לפחות לאחת מהן, נקבלו, אחרת- נדחה אותו. זמן הריצה של אלגוריתם זה הוא סכום זמני הריצה של כל אחד מהאלגוריתמים הפולינומיאליים להכרעת כל אחת משתי השפות. מכיוון שסכום פולינומים אף הוא פולינום, הרי שאיחוד שפות ששייכות ל-P אף הוא ב-P.

משלים: $A \in \mathrm{P} \Rightarrow \overline{A} \in \mathrm{P}$

נשנה את האלגוריתם להכרעת A על ידי החלפה בין המצבים של קבלה ודחייה. לשינוי זה אין השפעה על זמן הריצה הפולינומיאלי של האלגוריתם המקורי להכרעת A, ולכן קיבלנו כאן אלגוריתם פולינומיאלי המכריע את השפה המשלימה ל-A.

שרשור: $A, B \in \mathrm{P} \Rightarrow AB \in \mathrm{P}$

בהינתן קלט, נשבור אותו לשתי תת-מחרוזות בכל אחד מ-$n+1$ האופנים האפשריים. נריץ על התת-מחרוזת הראשונה את האלגוריתם הפולינומיאלי המכריע את A ועל השנייה את האלגוריתם הפולינומיאלי המכריע את B. נקבל את הקלט אם ורק אם באחת מן הבדיקות האלה שתי התת-מחרוזות נמצאו שייכות לשפות A ו-B בהתאמה. זמן הריצה של בדיקת כל אחד מאופני השבירה הוא סכום זמני הריצה של כל אחד מהאלגוריתמים הפולינומיאליים להכרעת כל אחת משתי השפות. מכיוון שסכום פולינומים אף הוא פולינום, והוא נותר פולינום גם לאחר שמכפילים אותו ב-$n+1$, הרי ששרשור שפות ששייכות ל-P אף הוא ב-P.

תרגיל 4.7

נניח ש-$A \in \mathrm{P}$ ותהי M מכונה דטרמיניסטית המכריעה את A בזמן פולינומיאלי. נתאר כעת אלגוריתם S המכריע את A^* בזמן פולינומיאלי. בהינתן מילת קלט $w = w_1 \cdots w_n$, האלגוריתם יבנה טבלה $T\left(1:n, 1:n\right)$ כך שעבור $1 \le i \le j \le n$ תציין הכניסה $T\left(i, j\right)$ אם התת-מחרוזת

$w_i \cdots w_j$ שייכת ל-A^* או לא. להלן תיאור האלגוריתם, הדומה מאוד לאלגוריתם שבנינו בהוכחת משפט 7.16, כיוון ששניהם משתמשים בתכנות דינמי:

1. If $w = \varepsilon$, accept.
2. Initialize $T\left(i, j\right) = 0$ for $1 \le i \le j \le n$.
3. For $i = 1$ to n:
4. Set $T(i,i) = 1$ if $w_i \in A$. [Test substrings of length 1]
5. For $l = 2$ to n: [l is the length of the substring]
6. For $i = 1$ to $n - l + 1$: [i is the substring start position]
7. Let $j = i + l - 1$. [j is the substring end position]
8. If $w_i \cdots w_j \in A$, set $T\left(i, j\right) = 1$.
9. For $k = i$ to $j - 1$: [k is the split position]
10. If $T\left(i,k\right) = 1$ and $T\left(k+1, j\right) = 1$, set $T\left(i, j\right) = 1$.
11. Accept if $T\left(1, n\right) = 1$; otherwise reject.

יש באלגוריתם זה $O(n^3)$ שלבים, שכל אחד מהם אורך זמן פולינומיאלי. לפיכך, זמן הריצה של האלגוריתם הוא פולינומיאלי.

תרגיל 4.8

פתרון בעיה זו מופיע בספר הלימוד.

תרגיל 4.9

בהינתן קלט מהצורה $\langle G, H\rangle$ המתאר שני גרפים, נבצע את הפעולות הבאות במכונת טיורינג אי-דטרמיניסטית להכרעת השפה ISO:

1. יהי m מספר הקדקודים בגרף G. אם ב-H יש מספר שונה של קדקודים, דחה את הקלט.
2. בחר באופן אי-דטרמיניסטי תמורה π של m איברים.
3. לכל זוג קדקודים x, y ב-G בדוק שהם מחוברים בקשת ב-G אם ורק אם $\pi(x), \pi(y)$ מחוברים בקשת ב-H. אם בדיקה זו העלתה אפילו חוסר התאמה אחד, דחה את הקלט. אחרת, אם נמצאה התאמה מלאה, קבל את הקלט.

אם $\langle G, H\rangle \in ISO$ אז קיימת תמורה π של m איברים שתביא להצלחת הבדיקה בשלב 3, ולכן קיים תרחיש פעולה שבו האלגוריתם לעיל יקבל קלט זה. אם, לעומת זאת, $\langle G, H\rangle \notin ISO$, אזי אין שום תמורה π שעבורה תסתיים הריצה בקבלת הקלט, כלומר, הוא תמיד יידחה. מכיוון שזמן הריצה של האלגוריתם לעיל הוא תמיד פולינומיאלי ב-m, אז $ISO \in \mathrm{NP}$.

הערה: ISO היא אחת הבעיות הבודדות ב-NP שעבורן לא ידוע שהן ב-P וגם לא ידוע שהן NP-שלמות.

תרגיל 4.10

חלק (1):

<u>בשלב הראשון</u> נוכיח כי בעיית הקבוצה הבלתי תלויה נמצאת ב-NP. הוכחה זו דומה מאוד להוכחה כי בעיית הקליקה נמצאת ב-NP. בהינתן קלט מהשפה, $\langle G,k\rangle \in INDEPENDENT-SET$ (כלומר גרף G ומספר טבעי k כך שהגרף G מכיל קבוצה בלתי תלויה של k קדקודים), אישור מתאים הוא קבוצה בלתי תלויה של k קדקודים. סיבוכיות הזמן של הבדיקה שהגרף G איננו מכיל שום קשת המקשרת בין שניים מתוך k הקדקודים הללו היא פולינומיאלית באורך של הקלט $\langle G,k\rangle$. לפיכך, $INDEPENDENT-SET \in \mathrm{NP}$.

<u>בשלב השני</u> נתאר רדוקציה שזמן ריצתה פולינומיאלי מבעיית הקליקה לבעיית הקבוצה הבלתי תלויה. כל מה שדרוש הוא ההבחנה שקבוצה בלתי תלויה בגרף G היא קליקה בגרף המשלים $\bar{G}$, שהוא גרף ה"נגטיב" של G הבנוי על אותם קדקודים ומכיל את כל הקשתות החסרות ב-G. בהינתן גרף $G=(V,E)$ ומספר k כקלט לבעיית הקליקה, נעבור על כל הזוגות של קדקודים ב-G ונבנה את הגרף המשלים $\bar{G}$ (כלומר, אם (u,v) היא קשת ב-G, אז היא לא תהיה קשת ב-$\bar{G}$, ולהפך – אם (u,v) איננה קשת ב-G, אז היא תהיה קשת ב-$\bar{G}$). קל לראות ש-$\langle G,k\rangle \in CLIQUE$ אם ורק אם $\langle \bar{G},k\rangle \in INDEPENDENT-SET$. זמן הריצה של הרדוקציה הינו $O\left(n^2\right)$, כאשר n הוא מספר הקדקודים בגרף, ולפיכך מדובר ברדוקציה שזמן ריצתה פולינומיאלי. בכך השלמנו את הוכחת נכונות הרדוקציה.

משני השלבים לעיל נקבל כי בעיית הקבוצה הבלתי תלויה היא NP-שלמה.

חלק (2):

תהי S קבוצה בלתי תלויה. נניח בשלילה שקבוצת הקדקודים המשלימה $V-S$ איננה מכסה את כל הקשתות בגרף. אז קיימת קשת (u,v) ששני קצותיה, u ו-v, אינם ב-$V-S$. מכאן ש-u וגם v נמצאים ב-S. אבל אז S איננה בלתי תלויה. סתירה!

מצד שני, נניח ש-$V-S$ מכסה את כל קשתות הגרף. אם S איננה בלתי תלויה, אז היא מכילה קשת שאיננה מכוסה על ידי הקדקודים ב-$V-S$. אבל אז נקבל סתירה להנחתנו ש-$V-S$ מכסה את כל קשתות הגרף.

חלק (3):

נראה שבעיית הקבוצה הבלתי תלויה ניתנת לרדוקציה פולינומיאלית לבעיית הכיסוי בקדקודים.

בהינתן קלט לבעיית הקבוצה הבלתי תלויה, $\langle G,k\rangle$, כאשר G הוא גרף בעל n קדקודים, הקלט

שתיצור הרדוקציה לבעיית הכיסוי בקדקודים הוא $\langle G,n-k\rangle$. לפי הסעיף הקודם, יש ב-G קבוצה בלתי תלויה בגודל k אם ורק אם יש ב-G כיסוי בקדקודים בגודל $n-k$. על כן, הרדוקציה נכונה. לבסוף, קל לוודא כי זמן הריצה של הרדוקציה הינו פולינומיאלי בגודל הקלט, כדרוש.

צריך גם להראות שבעיית הכיסוי בקדקודים שייכת ל-NP. ההוכחה דומה מאוד להוכחות של בעיית הקליקה ובעיית הקבוצה הבלתי תלויה.

תרגיל 4.11

ראשית נראה ש-*SET-COVER* שייכת ל-NP. אישור במקרה זה יהיה רשימה של k מתוך m התת-קבוצות הנתונות. אלגוריתם הבדיקה יעבור על כל אחד מאיברי הקבוצה U ויחפש אותו בין איברי k התת-קבוצות המופיעות באישור. אם כל איברי U יימצאו, אז נקבל את הקלט; אחרת – נדחה אותו. קל לראות שזמן הריצה של אלגוריתם בדיקה זה הוא פולינומיאלי.

נוכיח כעת כי *SET-COVER* היא NP-שלמה על ידי תיאור רדוקציה מ-*VERTEX-COVER* לבעיה הנדונה. בהינתן קלט $\langle G,k\rangle$ לבעיית *VERTEX-COVER*, נמפה אותו לקלט מתאים לבעיית *SET- COVER* כדלקמן:

- U תהיה קבוצת כל הקשתות ב-G.
- משפחת התת-הקבוצות תכיל n תת-קבוצות כאשר n הוא מספר הקדקודים של G: לכל קדקוד u בגרף G, המשפחה תכיל תת-קבוצה מתאימה $C(u)$ המורכבת מכל הקשתות הנוגעות בקדקוד u.
- המספר k בקלט הנדון של *SET-COVER* יהיה זהה למספר k במופע של *VERTEX-COVER*.

הרדוקציה דורשת מעבר על כל קדקודי G ובחינת השכנים של כל קדקוד. ניתן לבצע זאת בזמן לינארי במספר הקדקודים והקשתות של G.

כעת נוכיח את נכונות הרדוקציה: אם ל-G יש כיסוי קדקודים המכיל k קדקודים, אזי איחוד התת-קבוצות המתאימות לקדקודים אלו מכסה את כל קשתות הגרף. לכן הקלט שתואר לעיל שייך לשפה *SET-COVER*. מצד שני, נניח שאוסף התת-קבוצות בקלט ל-*SET-COVER* מכיל k תת-קבוצות שאיחודן הוא כל U. אז אוסף הקדקודים ב-G המתאים לתת-קבוצות אלו מהווה כיסוי בקדקודים ב-G. אכן, נניח בשלילה שלא כך הדבר. אז קיימת קשת e שאף אחד מהקדקודים בכיסוי הנבחר לא נוגע בה. מכאן, לפי הגדרת $C(u)$, הקשת e איננה שייכת לאף אחת מהתת-קבוצות $C(u)$ שבפתרון הנדון לבעיה *SET-COVER*. זו סתירה, שכן איחוד הקבוצות בכיסוי שווה לקבוצת כל הקשתות שב-G. סתירה זו משלימה את נכונות הרדוקציה.

תרגיל 4.12

תהי A שפה אי- טריוויאלית ב-P. אז A גם ב-NP. לכן על מנת להראות ש- A NP-שלמה, מספיק להראות רדוקציה פולינומיאלית מכל $B \in \mathrm{NP} = \mathrm{P}$ אליה. לצורך כך נבחר שתי מילים: אחת מתוך השפה, $x_{\text{in}} \in A$, ואחת מחוצה לה, $x_{\text{out}} \notin A$ (דבר זה אפשרי כיוון שהנחנו ש- A שפה אי-טריוויאלית, כלומר, היא איננה ריקה מצד אחד ואיננה הכל מצד שני). כעת, הרדוקציה מ- B ל- A תהיה כך: בהינתן מילה w, נפעיל עליה את האלגוריתם הפולינומיאלי להכרעת B. אם הוא יכריע ש- $w \in B$, אז הפלט של הרדוקציה יהיה $f(w) = x_{\text{in}}$, אחרת – $f(w) = x_{\text{out}}$.

תרגיל 4.13

חלק א:

האלגוריתם הפולינומיאלי להכרעת $SPATH$ הוא שכלול של האלגוריתם להכרעת $PATH$ שתואר בהוכחת משפט 7.14. שם, בכל שלב בלולאה סימנו עוד קדקודים שגילינו שניתן להגיע אליהם מקדקוד המקור. הסימון היה בינארי: "לא מסומן" פירושו היה "טרם גילינו מסלול המגיע לקדקוד זה מקדקוד המקור", בעוד "מסומן" משמעו היה "ניתן להגיע לקדקוד זה מקדקוד המקור". זה היה מספיק להכרעת $PATH$ – שם היה חשוב רק **קיומו** של מסלול בין שני קדקודים נתונים, אך לא **אורכו**. מכיוון שבבעיית $SPATH$ יש לנו עניין גם באורכו של המסלול (או ליתר דיוק, באורכו של המסלול הקצר ביותר בין שני הקדקודים ובהשוואתו לערך המספרי הנתון k), נשכלל את האלגוריתם הקודם על ידי שימוש בסימונים מספריים. תחילה, נסמן את קדקוד המקור ב-0 ואת כל השאר ב--1. בכל שלב i בלולאה, אם נגלה שקדקוד המסומן ב--1 מקושר לקדקוד שכבר סומן, אז נסמן אותו בסימון i, שיציין שאורך המסלול הקצר ביותר שנמצא אל קדקוד זה אורכו i. כך נמשיך עד לשלב שבו לא נסמן עוד שום קדקוד. כעת נבדוק את הסימון של קדקוד היעד b. אם הסימון שלו שונה מ--1 וקטן או שווה ל- k, נקבל את הקלט. אחרת - נדחה אותו.

חלק ב:

$SPATH$ שואלת לגבי אורכו של המסלול הקצר ביותר בין שני קדקודים, ואילו $LPATH$ שואלת על אורכו של המסלול הארוך ביותר בין שני קדקודים. בעיה זו כנראה קשה יותר.

קל לראות ש- $LPATH \in \mathrm{NP}$, כיוון שאפשר לתת כאישור מסלול פשוט בין שני הקדקודים שאורכו לפחות k. לצורך הוכחת NP-שלמות, נראה ש- $UHAMPATH \leq_{\mathrm{P}} LPATH$. הרדוקציה תהיה

$$\langle G, a, b \rangle \rightarrow \langle G, a, b, k-1 \rangle$$

כאשר $\langle G, a, b \rangle$ הוא קלט עבור בעיית $UHAMPATH$ (כלומר, G הוא גרף בלתי מכוון ו- a, b הם שני קדקודים בו), k הוא מספר הקדקודים ב- G ו-$\langle G, a, b, k-1 \rangle$ הוא קלט עבור בעיית $LPATH$.

זמן הריצה של הרדוקציה הינו פולינומיאלי.

כיוון ראשון: אם $\langle G,a,b\rangle \in UHAMPATH$, אז קיים מסלול המילטון מ- a ל- b. מסלול זה אורכו $k-1$ כיוון שהוא עובר דרך כל הקדקודים ב- G. לכן $\langle G,a,b,k-1\rangle \in LPATH$.

כיוון שני: אם $\langle G,a,b,k-1\rangle \in LPATH$, אז קיים מסלול פשוט באורך לפחות $k-1$ מ- a ל- b. מסלול פשוט לא יכול לחזור לאותו קדקוד פעמיים. לכן, מכיוון שיש ב- G בדיוק k קדקודים, אז אורך מסלול זה חייב להיות בדיוק $k-1$. לפיכך, זהו מסלול המילטון מ- a ל- b, הווי אומר $\langle G,a,b\rangle \in UHAMPATH$.

תרגיל 4.14

$DOUBLE-SAT \in \mathrm{NP}$ כיוון שמכונת טיורינג אי- דטרמיניסטית יכולה לנחש שתי הצבות עבור משתני הנוסחה, לבדוק ששתיהן מספקות אותה ושהן שונות זו מזו. כל זאת ניתן לביצוע בזמן פולינומיאלי. להוכחת השלמות, נראה רדוקציה מ-*SAT* ל-*DOUBLE-SAT*:

בהינתן קלט $\langle \phi \rangle$ לבעיית *SAT*, נתרגם אותו לקלט $\langle \phi' \rangle$ עבור בעיית *DOUBLE-SAT* כאשר $\phi' = \phi \wedge (x \vee \overline{x})$ ו-x הוא משתנה חדש. (עלות החישוב הזה היא כמובן פולינומיאלית.)

אם $\phi \in SAT$, אז $\phi' \in DOUBLE-SAT$ כיוון שמתוך ההצבה המספקת את ϕ ניתן ליצור שתי הצבות שונות המספקות את ϕ' – אחת שבה $x=0$ ואחרת שבה $x=1$. מצד שני, אם $\phi' \in DOUBLE-SAT$, אז בפרט אנו מסיקים ש- ϕ (המהווה חלק מ-ϕ') ספיקה, כלומר $\phi \in SAT$.

תרגיל 4.15

פתרון הבעיה מופיע בספר הלימוד.

תרגיל 4.16

$SET-SPLITTING \in \mathrm{NP}$ כיוון שמכונת טיורינג אי-דטרמיניסטית יכולה לנחש צביעה של איברי הקבוצה S ולבדוק, בזמן פולינומיאלי, אם צביעה זו מספקת את הדרישות הרצויות.

נראה כעת רדוקציה פולינומיאלית מבעיית *3SAT* ל-*SET-SPLITTING*.

בהינתן נוסחת-3CNF ϕ במשתנים $x_1,\ldots,x_m$, בעלת l פסוקיות $c_1,\ldots,c_l$, נגדיר קלט לבעיית *SET-SPLITTING*, $\phi \to f(\phi) = \langle S,C\rangle$, באופן הבא:

- $S=\{x_1,\overline{x_1},\ldots,x_m,\overline{x_m},y\}$ - כלומר, הקבוצה S כוללת איבר אחד לכל משתנה בנוסחה, איבר אחד לכל שלילה של משתנה, ואיבר אחד נוסף y.
- לכל פסוקית c_i בנוסחה ϕ נגדיר תת-קבוצה של S כתת-קבוצה C_i המכילה את כל הליטרלים המופיעים בפסוקית וגם את האיבר y. כמו כן נגדיר לכל משתנה x_i

את התת-קבוצה $D_i = \{x_i, \overline{x_i}\}$. אוסף התת-קבוצות המתאים יהיה $C = \{C_1, \ldots, C_l, D_1, \ldots, D_m\}$.

למשל, אם הנוסחה היא $\phi = (x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee x_4)$ אז

$$S = \{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3, \overline{x_3}, x_4, \overline{x_4}, y\}$$

והאוסף C מורכב מהקבוצות האלה:

$$C_1 = \{x_1, \overline{x_3}, x_4, y\}\ ,\ C_2 = \{\overline{x_2}, x_3, x_4, y\}\ ,\ D_i = \{x_i, \overline{x_i}\}\ ,\ 1 \le i \le 4$$

זמן הריצה של הרדוקציה הינו $O(m+l)$, שכן אנו עוברים על כל פסוקית c_i פעם אחת כדי לייצר את הקבוצה C_i, והאורך של כל פסוקית הוא 3. בנוסף, אנו סורקים את כל המשתנים בנוסחה פעם אחת.

נניח שיש ל- ϕ הצבה מספקת. נצבע את כל האיברים המתאימים לליטרלים שערכם 1 באדום, ואת כל אלו שמתאימים לליטרלים שערכם 0 בכחול; כמו כן נצבע את y בכחול. בכל תת-קבוצה D_i יש איבר אחד אדום ואחד כחול. גם בתת-קבוצות C_i יש ייצוג לשני הצבעים כיוון שכל תת-קבוצה כזו מכילה לפחות איבר אחד הצבוע באדום (כי כל פסוקית מכילה לפחות ליטרל אחד שערכו 1) וגם את האיבר y שצבעו כחול.

אם כן $\phi \in SAT \Rightarrow f(\phi) = \langle S, C\rangle \in SET - SPLITTING$.

נניח כעת שקיימת צביעה של איברי S כך שכל תת-קבוצה ב- C מכילה את שני הצבעים. אז נבחר לתת ערך 0 לכל הליטרלים שצבעם זהה לצבע של y, ואת הערך 1 לכל האחרים. מתקבלת הצבה חוקית (כיוון שהתת-קבוצות D_i מבטיחות שכל משתנה יקבל ערך הפוך מזה של שלילתו) ובכל פסוקית קיים לפחות ליטרל אחד שערכו 1.

לפיכך $f(\phi) = \langle S, C\rangle \in SET - SPLITTING \Rightarrow \phi \in SAT$.

תרגיל 4.17

השפה $F = \{\langle a,b,c\rangle : \exists p \in [b,c] \text{ such that } p \mid a\}$ היא ב-NP, כיוון שבהינתן מספר טבעי p, מכונת טיורינג דטרמיניסטית יכולה לאמת בזמן פולינומיאלי ש- p שייך ל- $[b,c]$ ושהוא מחלק את a. תחת ההנחה ש-P=NP, אנו מסיקים ששפה זו ניתנת להכרעה גם ללא קבלת אישור כזה. נסמן את האלגוריתם המכריע את השפה F באות F.

להלן אלגוריתם D המוצא בזמן פולינומיאלי את המחלק הקטן ביותר של מספר נתון a:

1. קרא לאלגוריתם F עם הקלט $\langle a, 2, a-1\rangle$. אם F יקבע שאין ל- a מחלק בקטע זה, עצור והכרז ש- a ראשוני.

2. התחל חיפוש בינארי על הקטע $[2, a-1]$ אחר מחלק של a. חפש תחילה בקטע של המספרים הקטנים יותר. רק אם בקטע זה לא נמצא מחלק, חפש בקטע של המספרים הגדולים יותר.
3. עצור כאשר תגיע לקטע חיפוש המכיל מספר אחד בלבד ו-F מצא שקטע זה מכיל מחלק של a. הוצא מחלק זה כפלט.

דוגמת ריצה עם המספר $a = 24$:

1. הקטע $[2, 23]$ מכיל מחלק.
2. הקטע $[2, 12]$ מכיל מחלק.
3. הקטע $[2, 6]$ מכיל מחלק.
4. הקטע $[2, 3]$ מכיל מחלק.
5. הקטע $[2, 2]$ מכיל מחלק. עצור והוצא כפלט "המספר 2 הוא המחלק הקטן ביותר של 24".

זמן הריצה של אלגוריתם זה הוא פולינומיאלי, כיוון שמספר הקריאות לאלגוריתם הפולינומיאלי להכרעת F הוא $O(\log_2 a)$.

מכיוון שהאלגוריתם מוצא את המחלק הקטן ביותר של a, המחלק d שהוא מוצא הוא מספר ראשוני. אם $d \neq a$, נפעיל את האלגוריתם D על a/d למציאת גורמים ראשוניים נוספים של a. נמשיך כך עד שהקלט ל-D יהיה מספר ראשוני. כעת נמצאים בידנו כל הגורמים הראשוניים של a.

מספר הקריאות ל-D (שהוא בעצמו בעל זמן ריצה פולינומיאלי) הוא k, כאשר k הוא מספר הגורמים הראשוניים של הקלט a (למשל, אם $a = 200 = 2^3 \cdot 5^2$, אז $k = 5$). כל מחלק ראשוני גדול או שווה ל-2. לכן $k = O(\log_2 a)$. לפיכך, גם האלגוריתם למציאת הפירוק המלא לגורמים רץ בזמן פולינומיאלי.

5. סיבוכיות מקום (פרק 8 בספר הלימוד)

קראו את המבוא לפרק 8 בספר הלימוד

בפרק זה נעסוק בסיבוכיות מקום (זיכרון) של בעיות. זמן ומקום הם שני המשאבים החשובים ביותר בפתרון בעיות חישוביות מעשיות, ולאחר שבפרק הקודם ניתחנו את סיבוכיות הזמן של בעיות, ננתח כעת את סיבוכיות המקום של בעיות כאלה ונסווג אותן על פי סיבוכיות זו.

על פי הגדרה 8.1, סיבוכיות המקום של מכונת טיורינג היא הפונקציה $f:\mathbb{N}\rightarrow\mathbb{N}$, כאשר $f(n)$ הוא המספר המקסימלי של תאי סרט שהמכונה משתמשת בהם עבור קלטים באורך n. המקסימום מחושב על פני כל הקלטים האפשריים באורך n וגם (במכונות אי-דטרמיניסטיות) על פני כל מסלולי החישוב האפשריים. השוו הגדרה זו להגדרות 7.1 ו-7.9 של סיבוכיות הזמן של מכונות טיורינג דטרמיניסטיות ואי-דטרמיניסטיות.

בהגדרה 8.2 מוגדרות המחלקות $\mathrm{SPACE}(f(n))$ ו- $\mathrm{NSPACE}(f(n))$ באופן אנלוגי להגדרות 7.7 ו-7.21 שהתייחסו לסיבוכיות זמן.

בדוגמה 8.3 מודגמת מכונת טיורינג פשוטה המכריעה את שפת הנוסחאות הספיקות, SAT, בסיבוכיות מקום ליניארית בגודל הקלט. הרעיון המרכזי במכונה הוא שימוש חוזר בסרט הזיכרון של המכונה. אנו עוברים על כל ההשמות האפשריות ובודקים לגבי כל אחת מהן אם מדובר בהשמה מספקת. לאחר שבדקנו השמה מסוימת, אנו מוחקים אותה ורושמים השמה חדשה על המקום שבו רשמנו את ההשמה הקודמת. העובדה כי ניתן לעשות שימוש חוזר במקום (בניגוד לזמן!) גורמת לנו לשער כי $\mathrm{TIME}(f(n))$ מוכלת ממש ב- $\mathrm{SPACE}(f(n))$. יחד עם זאת, נכון להיום כל שידוע לנו הוא ש- $\mathrm{TIME}(f(n))\subseteq\mathrm{SPACE}(f(n))$. ייתכן ששתי המחלקות שוות, אך הסברה המקובלת היא ש- $\mathrm{TIME}(f(n))\subsetneq\mathrm{SPACE}(f(n))$.

<u>שאלה:</u>

האם יתכן שסיבוכיות המקום של בעיה מסוימת תהיה גדולה יותר מסיבוכיות הזמן שלה?

בדוגמה 8.4 מוצג אלגוריתם לא דטרמיניסטי שמכריע את השפה $\overline{ALL_{\mathrm{NFA}}}$ במקום ליניארי. $\overline{ALL_{\mathrm{NFA}}}$ היא שפת התיאורים של אוטומטים סופיים לא דטרמיניסטיים שיש לפחות מילה אחת שהם לא מקבלים. האלגוריתם המופיע בדוגמה מוכיח שהשפה $\overline{ALL_{\mathrm{NFA}}}$ שייכת ל- $\mathrm{NSPACE}(n)$.

לא ידוע על השפה $\overline{ALL_{\text{NFA}}}$ שהיא שייכת ל- NP או ל- coNP. האלגוריתם שמופיע בדוגמה 8.4 לא מוכיח שהיא שייכת ל- NP, משום שזמן הריצה שלו איננו פולינומיאלי בגודל הקלט. צעד 3 באלגוריתם עשוי להתבצע 2^q פעמים, כאשר q הוא מספר המצבים של האוטומט. זהו מספר אקספוננציאלי בגודל הקלט. אנו רואים שוב שבעזרת שימוש חוזר במקום, אפשר להשתמש במקום ליניארי כדי להריץ אלגוריתמים בעלי זמן ריצה אקספוננציאלי.

האלגוריתם N שומר את קבוצת המצבים שבהם יכול האוטומט הלא דטרמיניסטי להימצא באותו רגע של החישוב. את זה אפשר לממש בעזרת מקום שגודלו ליניארי במספר מצבי האוטומט. בנוסף שומר האלגוריתם מונה שערכו ההתחלתי הוא 2^q, והוא קטן ב-1 לפני כל פעם שמתבצע שלב 3 של האלגוריתם. כאשר המונה מתאפס, אנו יודעים שהלולאה של שלבים 2 ו-3 התבצעה 2^q פעמים. את המונה הזה אפשר לשמור בייצוג בינרי. לכן המקום הדרוש למונה הוא $O(q)$. זהו מקום ליניארי במספר המצבים של האוטומט.

5.1 משפט Savitch

קראו את משפט 8.5 אך דלגו על הוכחתו

במשפט זה (שאיננו מתעכבים על הוכחתו בקורס זה) מוכיחים כי ניתן לדמות כל מכונת טיורינג אי-דטרמיניסטית, הפועלת בסיבוכיות מקום $f(n)$, באמצעות מכונת טיורינג דטרמיניסטית שסיבוכיות המקום שלה היא $f^2(n)$, לכל היותר. אם כן, מבחינת סיבוכיות מקום, כל מה שניתן לחישוב בסיבוכיות מקום פולינומיאלית על מכונה אי-דטרמיניסטית, ניתן לחישוב בסיבוכיות מקום פולינומיאלית גם על מכונה דטרמיניסטית. כזכור, המצב מבחינת סיבוכיות זמן שונה: בהינתן מכונה אי-דטרמיניסטית שרצה בזמן $f(n)$, לא ידועה לנו מכונת טיורינג דטרמיניסטית המדמה את פעילותה של מכונה זו בזמן שאיננו אקספוננציאלי ב- $f(n)$. לפיכך, מבחינת סיבוכיות זמן מבדילים בין המחלקות P ו-NP, ואילו מבחינת סיבוכיות מקום אין הבדל בין המחלקות PSPACE ו- NPSPACE.

5.2 המחלקה PSPACE

קראו את סעיף 8.2 בספר הלימוד

לאחר שהגדרנו את המחלקה PSPACE (הגדרה 8.6), נבחן את הקשר בין מחלקות הסיבוכיות השונות שפגשנו עד כה:

1. $P \subseteq PSPACE$ מכיוון שמכונה שרצה בזמן $t(n) \ge n$ אינה יכולה להשתמש ביותר מ- $t(n)$ תאים על הסרט.
2. באופן דומה $NP \subseteq NPSPACE$, ועל פי משפט Savitch NPSPACE=PSPACE.
3. $PSPACE \subseteq EXPTIME$ מהסיבה הבאה: נניח ש-M היא מכונה עם q מצבים המשתמשת בלא יותר מ- $f(n)$ תאים על הסרט על קלט באורך n. מספר הקונפיגורציות השונות של מכונה כזו הוא לכל היותר $q \cdot f(n) \cdot |\Gamma|^{f(n)}$ (מספר המצבים הוא q, הראש יכול להיות ב- $f(n)$ מקומות שונים לאורך הסרט ומספר התכנים האפשריים של הסרט הוא $|\Gamma|^{f(n)}$, כאשר Γ הוא אלפבית הסרט). מכיוון שמכונה אשר עוצרת תמיד איננה יכולה לחזור פעמיים על אותה קונפיגורציה (חזרה כזו אפשרית רק במכונות שבהן תיתכן לולאה אינסופית), אז זמן הריצה של מכונה כזו חסום על ידי $q \cdot f(n) \cdot |\Gamma|^{f(n)}$. מכיוון ש- q ו- $|\Gamma|$ הם קבועים ו- $f(n)$ הוא פולינום, אז זמן הריצה של מכונה כזו חסום על ידי $2^{d \cdot f(n)}$ עבור קבוע d כלשהו, ולפיכך גם חסום על ידי 2^{n^k} עבור k כלשהו.

כל ההכלות לעיל עשויות להיות שוויון, אך לפחות אחת מהן חייבת להיות הכלה ממש, מכיוון שבפרק 9 נוכיח ש- $P \ne EXPTIME$.

תרגיל 5.1

פתרו את תרגיל 8.4 בספר הלימוד.

5.3 PSPACE-שלמות

קראו את המבוא לסעיף 8.3 בספר הלימוד עד לפני התת-סעיף "בעיית ה-TQBF"

אנו מגדירים כאן את המושג PSPACE-שלמות: שפה היא **PSPACE-שלמה** אם היא שפה במחלקה PSPACE, וכל שפה אחרת במחלקה PSPACE ניתנת לרדוקציית זמן פולינומיאלית אליה. תת-מחלקה זו מציינת אפוא את אוסף הבעיות הקשות ביותר במחלקה PSPACE.

באופן כללי, בעיה A נחשבת לקשה ביותר במחלקה שלה אם כל בעיה אחרת באותה מחלקה ניתנת לרדוקציה אליה "**בקלות**" יחסית לקושי של המחלקה. במקרה של PSPACE מדובר על רדוקציה בזמן פולינומיאלי. פולינומיאליות הרדוקציה היא קריטית, כיוון שברצוננו לדעת שפתרון בזמן פולינומיאלי לבעיה A יגרור פתרון בזמן פולינומיאלי לכל הבעיות במחלקה. זו הסיבה שאנו דורשים גם בהגדרה זו **רדוקציית זמן** פולינומיאלית ולא **רדוקציית מקום** פולינומיאלית (כפי שאולי ניתן היה לצפות). שלמות ביחס לרדוקציית מקום פולינומיאלית הייתה

אומרת שפתרון במקום פולינומיאלי לבעיה A יגרור פתרון במקום פולינומיאלי לכל הבעיות במחלקה. אבל זה ממילא קיים, גם ללא הרדוקציה.

<u>שאלה:</u>

שפה נקראת **NP-קשה** (NP-hard), אם כל שפה ב-NP ניתנת לרדוקציה בזמן פולינומיאלי אליה.
שפה NP-קשה שגם שייכת בעצמה ל-NP היא שפה NP-שלמה.
באופן דומה, שפה נקראת **PSPACE-קשה** (PSPACE-hard) , אם כל שפה ב-PSPACE ניתנת לרדוקציה בזמן פולינומיאלי אליה. שפה PSPACE-קשה שגם שייכת בעצמה ל-PSPACE היא שפה PSPACE-שלמה.
מדוע כל שפה שהיא PSPACE-קשה היא גם NP-קשה?

<u>תשובה:</u>

אם A שפה PSPACE-קשה, אז כל שפה $B \in \mathrm{PSPACE}$ מקיימת $B \leq_{\mathrm{P}} A$.
אך $\mathrm{NP} \subseteq \mathrm{PSPACE}$. לכן A היא שפה NP-קשה.

המשיכו לקרוא בסעיף 8.3 בספר הלימוד את התת-סעיף "בעיית ה-TQBF"

ניתן לדלג על הוכחת משפט 8.9

משפט (sentence) הוא **נוסחה בוליאנית עם כמתים** (quantified Boolean formula) שבה כל המשתנים **קשורים** (bound) על ידי **כמתים** (quantifiers) - **הכמת הכולל** $\forall$ (the universal quantifier) או **הכמת היישי** $\exists$ (the existential quantifier). כל משפט כזה הוא אמיתי או שקרי בהתאם לתחום (universe) שממנו נלקחים ערכי המשתנים.

תרגיל 5.2

להלן חמישה משפטים. קבעו את ערך האמת של כל אחד מהם בכל אחת מחמש קבוצות המספרים האלה: $\mathbb{N}$ (קבוצת המספרים הטבעיים), $\mathbb{Z}$ (קבוצת המספרים השלמים), $\mathbb{Q}$ (קבוצת המספרים הרציונליים), $\mathbb{R}$ (קבוצת המספרים הממשיים), ו- $\mathbb{C}$ (קבוצת המספרים המרוכבים).

1. $\forall x \exists y \left[y > x \right]$
2. $\forall x \exists y \left[y < x \right]$
3. $\forall x \forall y \exists z \left[(x = y) \vee (x < z < y) \vee (y < z < x) \right]$
4. $\forall x \exists y \left[(x < 0) \vee (y^2 = x) \right]$
5. $\forall x \exists y \left[y^2 = x \right]$

שפת ה-TQBF מורכבת מכל המשפטים הבוליאניים האמיתיים. זוהי שפה PSPACE-שלמה כפי שמוכח במשפט 8.9. כלומר, היא ב-PSPACE, וכל שפה אחרת ב-PSPACE ניתנת לרדוקציה אליה **בזמן** פולינומיאלי.

הוכחת משפט 8.9

בשלב הראשון בהוכחה מראים ש- $TQBF \in \text{PSPACE}$. אלגוריתם להכרעת השפה בודק את אמיתות המשפט באופן רקורסיבי על ידי "קילוף" שכבות הכמתים אחד אחד: אם הכמת החיצוני הוא $\exists x_i \psi$, אז האלגוריתם מקבל את הקלט אם $\psi[x_i = 0]$ או $\psi[x_i = 1]$ נכון (דהיינו, אם אחד משני המשפטים המתקבלים מהצבת $x_i = 0$ ב- ψ או $x_i = 1$ ב- ψ נכון). אם הכמת החיצוני הוא $\forall x_i \psi$, אז האלגוריתם מקבל את הקלט אם $\psi[x_i = 0]$ וגם $\psi[x_i = 1]$ נכונים. אלגוריתם זה סורק למעשה את כל 2^m ההצבות האפשריות למשתנים $x_1, \ldots, x_m$. לפיכך, זמן הריצה של אלגוריתם זה הוא לפחות 2^m; אך סיבוכיות המקום של האלגוריתם היא ליניארית מכיוון שהבדיקה של כל אחד מ- 2^m ערכי ההצבות עושה שימוש באותו שטח עבודה. (לשם פשטות, התעלמנו מעומק הרקורסיה ומהזיכרון הנלווה לרקורסיה).

בשלב השני מראים רדוקציה מכל שפה $A \in \text{PSPACE}$, הניתנת להכרעה על ידי מכונת טיורינג M במקום n^k (עבור קבוע כלשהו k) ל- $TQBF$. המיפוי הוא ממחרוזת w למשפט ϕ כך ש- ϕ נכון אם ורק אם M מקבלת את w.

הרדוקציה תתבצע כדלקמן: כבר ראינו בהוכחת משפט קוק-לווין כיצד אפשר לייצג קונפיגורציה של מכונת טיורינג על ידי אוסף של משתנים בוליאניים. נסמן ב- c_1 וב- c_2 שני אוספים של משתנים כאלו המייצגים קונפיגורציות של המכונה M, ויהי t מספר טבעי המהווה חזקה של 2 (זו הנחה המפשטת את הבנייה). נבנה נוסחה $\phi_{c_1,c_2,t}$ שתהיה נכונה אם ורק אם שני אוספי המשתנים c_1 ו- c_2 מייצגים קונפיגורציות חוקיות של המכונה M, ו- M יכולה לעבור מהקונפיגורציה המיוצגת על ידי c_1 לזו המיוצגת על ידי c_2 ב- t צעדים לכל היותר. שימו לב שיש לנו משתנה לכל תא בסרט (יש כאלו n^k לכל היותר), לכל מצב של המכונה ולכל סמל באלפבית. לכן מספר המשתנים הוא $O\left(n^k\right)$.

תהי c_{start} קונפיגורציית ההתחלה של המכונה M על הקלט $w = w_1 \cdots w_n$. תהי c_{accept} קונפיגורציית הקבלה של המכונה M. (ללא הגבלת הכלליות, אנו יכולים להניח שאם M מקבלת קלט כלשהו, אז היא מוחקת את תוכן הסרט, מחזירה את הראש לתחילת הסרט ונכנסת למצב q_{accept}, כך שיש רק קונפיגורציית קבלה אחת). יהי $h = 2^{d \cdot f(n)}$ זמן הריצה המקסימלי האפשרי עבור המכונה M, כאשר $f(n) = n^k$ היא סיבוכיות המקום שלה (הראינו זאת בסעיף 8.2 לצורך

ההוכחה ש- $\text{PSPACE} \subseteq \text{EXPTIME}$). אז המשפט $\phi_{c_{\text{start}},c_{\text{accept}},h}$ יהיה נכון אם ורק אם M מקבלת את w.

נבנה אפוא את המשפט $\phi_{c_1,c_2,t}$. אם $t=1$, אז המשפט יקודד את אחת משתי האפשרויות הבאות: או ששתי הקונפיגורציות שוות, או שניתן לעבור מהאחת לאחרת בצעד אחד. את האפשרות השנייה, המורכבת יותר, ניתן לקודד כפי שעשינו בהוכחת משפט קוק-לווין, על ידי התמקדות בחלונות של שלושה תאים סמוכים בשתי הקונפיגורציות.

אם $t=2^r$ עבור $r \ge 1$, אז אנו ממשיכים בבנייה רקורסיבית של המשפט. הרעיון הראשון לבנייה רקורסיבית הוא:

$$\phi_{c_1,c_2,t} = \exists m_1 \left[\phi_{c_1,m_1,\frac{t}{2}} \wedge \phi_{m_1,c_2,\frac{t}{2}} \right]$$

כלומר, ניתן לעבור מהקונפיגורציה המיוצגת על ידי c_1 לזו המיוצגת על ידי c_2 ב- t צעדים לכל היותר, אם קיימת קונפיגורציית ביניים, המיוצגת על ידי אוסף המשתנים המסומן כאן m_1, כך שניתן לעבור מ- c_1 אליה ב- $\frac{t}{2}$ צעדים לכל היותר, וממנה ל- c_2 גם כן ב- $\frac{t}{2}$ צעדים לכל היותר. אך הרעיון הזה נכשל מכיוון שהוא מייצר משפט גדול מדי. כל שלב בבנייה הרקורסיבית הזו מכפיל, בערך, את גודל המשפט. לכן נקבל בסופו של דבר משפט שגודלו בערך t. כיוון שאנו מבקשים לבנות כך את המשפט $\phi_{c_{\text{start}},c_{\text{accept}},h}$, כאשר $h=2^{d \cdot f(n)}$, נקבל משפט שגודלו מעריכי ב- n. אך זה אומר שהרדוקציה $w = w_1 \cdots w_n \rightarrow \phi_{c_{\text{start}},c_{\text{accept}},h}$ היא בעלת סיבוכיות **זמן** מעריכית, כיוון שהכתיבה של משפט באורך מעריכי לבדה (אפילו מבלי להביא בחשבון את זמן החישוב של המשפט) אורכת מספר מעריכי של צעדים.

הרעיון השני לבנייה רקורסיבית הוא:

$$\phi_{c_1,c_2,t} = \exists m_1 \forall (c_3,c_4) \in \{(c_1,m_1),(m_1,c_2)\} \left[\phi_{c_3,c_4,\frac{t}{2}} \right]$$

כפי שמוסבר בספר, ניתן לקודד את המשפט הזה כך:

$$\phi_{c_1,c_2,t} = \exists m_1 \forall c_3 \forall c_4 \left[\left((c_3=c_1 \wedge c_4=m_1) \vee (c_3=m_1 \wedge c_4=c_2) \right) \rightarrow \phi_{c_3,c_4,\frac{t}{2}} \right]$$

לבסוף, אופרטור הגרירה $p \rightarrow q$ מתורגם ל- $\neg p \vee q$ כך שהמשפט לעיל ניתן לרישום באמצעות שלושת האופרטורים הבסיסיים בלבד - $\neg \wedge \vee$.
מדוע בנייה זו עובדת? מכיוון שבכל שלב ברקורסיה המשפט גדל על ידי הוספת שלושה אוספי משתנים חדשים - m_1, c_3, c_4 - שכל אחד מהם באורך $O(n^k)$, ומתווסף לו החלק הראשון $(c_3=c_1 \wedge c_4=m_1) \vee (c_3=m_1 \wedge c_4=c_2)$ שאורכו אף הוא $O(n^k)$. לכן, מכיוון שמספר

השלבים ברקורסיה הוא $O\left(n^k\right)$, אנו מקבלים שאורך המשפט הסופי הוא $O\left(n^{2k}\right)$. כאן תמה הוכחת המשפט.

תרגיל 5.3

פתרו את בעיה 8.27 בספר הלימוד.

5.4 המחלקות L ו-NL

קראו את סעיף 8.4 בספר הלימוד

בסעיף זה אנו פוגשים לראשונה סיבוכיות תת-ליניארית. כיוון שבדרך כלל חישובים חייבים לקרוא את **כל** הקלט, הרי שסיבוכיות הזמן וגם סיבוכיות המקום של חישובים היא בדרך כלל ליניארית לפחות, $\Omega(n)$. אך ישנם חישובים שבהם מקום העבודה הנדרש, מעבר למקום הנחוץ לשמירת הקלט ולקריאה ממנו, קטן באופן משמעותי מ-n. שפות שלצורך הכרעתן נחוץ שטח עבודה של $O(\log n)$ במכונת טיורינג דטרמיניסטית שייכות למחלקה L. באופן דומה, שפות שלצורך הכרעתן נחוץ שטח עבודה של $O(\log n)$ במכונת טיורינג אי-דטרמיניסטית שייכות למחלקה NL. על מנת לאפיין שפות כאלו, אנו מציגים מודל חישובי חדש: מכונת טיורינג בעלת שני סרטים – סרט לקריאה בלבד, שעליו מאוחסן הקלט (שאורכו n), וסרט נוסף לעבודה, שהשטח הנחוץ עליו לצורך הכרעת שפות ב-L או ב-NL הוא $O(\log n)$.

בדוגמה 8.18 אנו רואים ש-$A=\left\{0^k1^k : k\geq 0\right\}\in \mathrm{L}$ על ידי שימוש במונה בינארי לספירת מספר ה-0-ים, ובמונה בינארי נוסף לספירת ה-1-ים, והשוואה ביניהם.

בדוגמה 8.19 אנו רואים ש-$PATH\in \mathrm{NL}$. נניח שהקלט הוא $\langle G,s,t\rangle$, כאשר G הוא גרף מכוון בעל m קדקודים ו-s ו-t הם שניים מהקדקודים האלו. האלגוריתם האי-דטרמיניסטי המכריע האם $\langle G,s,t\rangle\in PATH$ פועל כדלקמן: הוא בוחר מסלול שרירותי ב-G באורך $\geq m$ המתחיל ב-s, והוא מקבל את הקלט אם ורק אם המסלול מגיע בשלב כלשהו לקדקוד הרצוי t (שימו לב שאם קיים מסלול בין s ל-t, אז קיים מסלול כזה שאורכו הוא m לכל היותר). הרעיון בבנייה זו הוא שכל מה שצריך לזכור על סרט העבודה של המכונה הוא את מספרו של הקדקוד הנוכחי שאליו הגיע המסלול, את אורכו הנוכחי של המסלול ואת מספר הקדקודים m; שלושת המספרים ייוצגו באופן בינארי על מנת לשמור על סיבוכיות מקום לוגריתמית. בכל שלב בלולאה תסקור המכונה את סרט הקלט ותבחר באופן אי-דטרמיניסטי את אחד הקדקודים המחוברים בקשת לקדקוד הנוכחי שאליו הגיע המסלול; המכונה תכתוב את מספרו הסידורי של הקדקוד החדש במקום המספר הסידורי של הקדקוד האחרון, ואחר כך תגדיל את אורכו הנוכחי של המסלול

באחד. האלגוריתם יעצור באחד משני המקרים האלה: או שבשלב מסוים יגיע המסלול לקדקוד t (ואז נקבל את הקלט) או שאורך המסלול יגיע לערך m (ואז נדחה את הקלט).

בסעיף 8.2 הראינו שאם למכונת טיורינג יש סיבוכיות מקום $f(n)$, אז סיבוכיות הזמן שלה היא $2^{O(f(n))}$. טענה זו איננה נכונה בהכרח במודל המכונה עם שני סרטים – סרט לאחסון הקלט וקריאה בלבד וסרט נוסף לעבודה. אך בהמשך מוסבר שהטענה עדיין נכונה עבור סיבוכיות מקום שאינה נמוכה מדי, $f(n) \geq \log n$.

תרגיל 5.4

תנו דוגמה של מכונת טיורינג בעלת סיבוכיות מקום $f(n) = o(n)$ כך שסיבוכיות הזמן שלה גדולה מ- $2^{O(f(n))}$.

בסוף הסעיף נטענת טענה נוספת המצדיקה את הבחירה בסיבוכיות מקום לוגריתמית כמחלקת סיבוכיות מעניינת בפני עצמה. משפט Savitch, שאותו תיארנו בסעיף 8.1 ללא הוכחה, טוען שהמחיר שעלינו לשלם בסיבוכיות מקום עבור המעבר מהמודל האי-דטרמיניסטי למודל הדטרמיניסטי הוא ריבוע הסיבוכיות, בהנחה ש- $f(n) \geq n$ (כלומר, סיבוכיות מקום אי-דטרמיניסטית של $f(n) \geq n$ ניתנת לתרגום לסיבוכיות מקום דטרמיניסטית של $O\left(f^2(n)\right)$). משפט Savitch נותר נכון גם עבור סיבוכיות מקום נמוכה יותר בתנאי ש- $f(n) \geq \log n$.

תרגיל 5.5

פתרו את תרגיל 8.5 בספר הלימוד.

תרגיל 5.6

פתרו את תרגיל 8.7 בספר הלימוד.

תרגיל 5.7

פתרו את בעיה 8.33 בספר הלימוד.

5.5 NL-שלמות

קראו את סעיף 8.5 בספר הלימוד

היחס בין המחלקות L ו-NL איננו ברור ממש כמו היחס בין המחלקות P ו-NP. ברור שהמחלקה הדטרמיניסטית מוכלת בזו האי-דטרמיניסטית, אך טרם נמצאה בעיה השייכת ל-NL אך לא ל-L. לפיכך, יתכן ש-L=NL. שאלה זו נותרת פתוחה בשלב זה.

שפה נקראת **NL-שלמה** אם היא שייכת ל-NL והיא ברמת הקושי הגבוהה ביותר במחלקה זו, במובן שכל שפה אחרת בשפה ניתנת לרדוקציה קלה אליה. כאשר הגדרנו NP-שלמוּת או PSPACE-שלמוּת, הגדרנו רדוקציה קלה ככזו שסיבוכיות הזמן שלה היא פולינומיאלית. אך הגדרה כזו איננה מתאימה למחלקה NL כיוון שכל השפות ב-NL ניתנות להכרעה בזמן פולינומיאלי (את זה נוכיח בהמשך).

תרגיל 5.8

הניחו ש- $\mathrm{P} \subseteq \mathrm{NL}$. הסיקו כי אם $A, B \in \mathrm{NL}$ ו- $B \notin \{\varnothing, \Sigma^*\}$ אז $A \le_{\mathrm{P}} B$.

לפיכך, רדוקציות מיפוי בזמן פולינומיאלי הן חזקות מדי בהקשר זה - אין באפשרותן לבדל חלק מהשפות ב-NL כקשות יותר מהאחרות. לפיכך, אנו פונים לרדוקציית מיפוי "עדינה" יותר: **רדוקציית מקום לוגריתמי** (הגדרה 8.21). הבה נחזור על פרטי הגדרה זו:

- **מתמר מקום לוגריתמי** (log space transducer) הוא מכונת טיורינג בעלת שלושה סרטים: סרט קלט לקריאה בלבד, סרט פלט לכתיבה בלבד, וסרט עבודה לכתיבה וקריאה, שיכול להכיל $O(\log n)$ תווים (n הוא אורך הקלט).
- כל מתמר מקום לוגריתמי M מגדיר פונקציה $f : \Sigma^* \to \Sigma^*$ באופן הבא: $f(w)$ היא המחרוזת הרשומה על סרט הפלט לאחר ש- M סיים את פעולתו על הקלט w. פונקציה f כזו נקראת **פונקציה חשיבה במקום לוגריתמי** (log space computable function).
- שפה A **ניתנת לרדוקציית מקום לוגריתמי** (log space reducible) לשפה B, דבר המסומן על ידי $A \le_{\mathrm{L}} B$, אם קיים מתמר מקום לוגריתמי המיישם רדוקציית מיפוי מ- A ל- B.

שפה נקראת **NL-שלמה** (הגדרה 8.22), אם היא ב-NL וכל שפה אחרת ב-NL ניתנת לרדוקציית מקום לוגריתמי אליה.

תרגיל 5.9

נניח ש- $A \leq_{\mathrm{L}} B$ באמצעות פונקציה חשיבה במקום לוגריתמי f. הראו שאורכו של $f(w)$ הוא לכל היותר פולינומיאלי ב-$|w|$.

אין זה מפתיע שאם $A \leq_{\mathrm{L}} B$ ו-$B \in \mathrm{L}$ אז גם $A \in \mathrm{L}$ (משפט 8.23). אך ההוכחה איננה מיידית לחלוטין משום שיש לנקוט בה משנה זהירות. הרכבה של המכונה להכרעת B על המתמר מ-A ל-B, כמו שעשינו בהוכחת משפט 7.31 בהקשר של רדוקציות בזמן פולינומיאלי, אומנם מניבה מכונה המכריעה את A, שנסמנה M_A, אך סיבוכיות המקום שלה עלולה להיות גדולה מסיבוכיות לוגריתמית. התיאור של M_A הוא כדלקמן:

1. בהינתן קלט $w \in \Sigma^*$, M_A תחשב עבורו את $f(w)$ באמצעות מתמר המקום הלוגריתמי מ-A ל-B.
2. M_A תפעיל את המכונה להכרעת B, שנקרא לה M_B, על $f(w)$. אם M_B תכריע ש-$f(w) \in B$, אזי M_A תעצור ותקבל את w כקלט מהשפה A; אחרת – היא תדחה אותו.

השלב הראשון באלגוריתם לעיל צורך שטח עבודה לוגריתמי ב-$|w|$, כנדרש. גם השלב השני איננו בעייתי לכשעצמו: יהי k קבוע כך ש-$|f(w)| \leq |w|^k$ (ראו תרגיל 5.9 לעיל). אז סיבוכיות המקום של M_B היא לוגריתמית באורך הקלט שלה, כלומר:

$$O\left(\log|f(w)|\right) \leq O\left(\log|w|^k\right) = O\left(k\log|w|\right) = O\left(\log|w|\right)$$

הבעיה היא אורכו של הפלט של השלב הראשון שהוא הקלט של השלב השני, דהיינו - אורך המחרוזת $f(w)$. השלב הראשון חייב להעביר את $f(w)$ אל השלב השני, ולצורך כך הוא כותב את המחרוזת $f(w)$ על סרט העבודה. אך על פי האמור בתרגיל 5.9 לעיל, אורכו של $f(w)$ יכול להיות פולינומיאלי ב-$|w|$, ולפיכך הוא עלול להיות ארוך יותר מ-$O\left(\log|w|\right)$.

התיקון לאלגוריתם לעיל הוא פשוט ומבוסס על האבחנה שמכונת טיורינג היא "צרת-הסתכלות" - בכל פעם היא קוראת תו אחד מסרט הקלט; משמע, היא אינה מסוגלת לראות את כל תוכן סרט הקלט בבת אחת. לפיכך, המכונה M_A "המשופצת" לא תריץ את השלב הראשון פעם אחת ותחשב מראש את כל $f(w)$, מחרוזת שעלולה להיות ארוכה מדי. במקום זה היא תזכור בכל שלב היכן אמור לעמוד הראש הקורא של המכונה M_B על גבי הקלט שלה, $f(w)$. אם המכונה M_B אמורה לקרוא את התו ה-i-י מהמחרוזת $f(w)$, המכונה M_A תפעיל את המיפוי של השלב הראשון עד שהוא יחזיר את התו ה-i-י מהמחרוזת $f(w)$, ואז היא תעביר תו זה לעותק של המכונה M_B שהיא מריצה. (שימו לב: בכל פעם שאנו מפעילים את המיפוי הנדון, אנו שבים ומשתמשים באותו מקום בסרט העבודה.) אפשר לעשות זאת משום שפעולת המכונה M_A איננה

מושפעת מתוכן סרט הפלט שלה שהוא סרט לכתיבה בלבד, ולכן אפשר בכל פעם לחשב רק סמל אחד מ $f(w)$. זהו חישוב לא יעיל כלל מבחינת זמן ריצה, אך הוא שומר על סיבוכיות מקום נמוכה, ובפרט - לוגריתמית באורך הקלט w.

מסקנה מידית ממשפט זה (מסקנה 8.24) היא שאם שפה NL-שלמה תתגלה אי פעם כשייכת ל-L, אז כל המחלקה NL "תקרוס" אל L, כלומר, נוכל להסיק ש-L=NL.

תרגיל 5.10

הראו שאם $B \leq_{\mathrm{L}} A$ ו- $C \leq_{\mathrm{L}} B$, אז $C \leq_{\mathrm{L}} A$.

במשפט 8.25 מוכיחים ש- $PATH$ היא NL-שלמה. מכיוון שכבר ראינו בדוגמה 8.19 שבעיה זו שייכת ל-NL, כל שנותר להראות הוא רדוקציית מקום לוגריתמי מכל שפה ב-NL אליה.

נניח ש-A ניתנת להכרעה על ידי מכונת טיורינג אי-דטרמיניסטית M במקום לוגריתמי. נבנה רדוקציית מיפוי $w \to \langle G,s,t \rangle$ כדלקמן (w הוא הקלט עליו M רצה): G הוא גרף מכוון שקדקודיו הם אוסף כל הקונפיגורציות האפשריות של M על w, וקשת מקשרת את הקדקוד c_1 לקדקוד c_2 אם המכונה M יכולה לעבור מהקונפיגורציה המתאימה ל-c_1 לזו המיוצגת על ידי c_2 בצעד אחד. הקדקוד s יציין את הקונפיגורציה ההתחלתית של M על w, והקדקוד t יציין את הקונפיגורציה המקבלת של M (ללא הגבלת הכלליות, אנו יכולים לשנות את M כך שתהיה לה קונפיגורציה מקבלת אחת בלבד, בדומה להוכחת משפט 8.9).

קל לראות שזו אכן רדוקציית מיפוי מ-A ל-$PATH$: $w \in A$ אם ורק אם קיים ב-G מסלול מ-s ל-t, כלומר אם ורק אם $\langle G,s,t \rangle \in PATH$. עלינו להמשיך ולהראות שהרדוקציה ניתנת למימוש במקום לוגריתמי.

למתמר שלנו יש שלושה סרטים: סרט קלט עליו רשום w; סרט עבודה; וסרט פלט שעליו יירשם התיאור הסופי $\langle G,s,t \rangle$. המתמר יכתוב לסרט הפלט את התיאור של כל קונפיגורציה אפשרית של M על w. תיאור זה מורכב מ:

- תוכן סרט העבודה – $O(\log n)$;
- מקום הראש של M על סרט הקלט – $\log n$;
- מקום הראש של M על סרט העבודה – $O(\log \log n)$;
- ומצב המכונה M – $O(1)$.

בתום שלב זה יכיל סרט הפלט את תיאור כל הקונפיגורציות האפשריות של M על w, כלומר את כל הקדקודים בגרף G. (המכונה למעשה בודקת כל מחרוזת אפשרית אשר יכולה להיות

קונפיגורציה, תוך שימוש חוזר במקום. נקודה עקרונית כאן היא העובדה שהאורך של כל קונפיגורציה הינו $O(\log n)$. בשלב הבא יסקור המתמר את כל הזוגות הסדורים של קונפיגורציות, (c_1, c_2), ויבדוק אם ניתן לעבור מ- c_1 ל- c_2 באמצעות צעד אחד של M. אם כן, הוא יכתוב את תיאור הקשת הזו בהמשך סרט הפלט. כל הפעולות הללו ניתנות למימוש באמצעות שימוש ב- $O(\log n)$ תאים על סרט העבודה. לסיום יכתוב המתמר את הקודקוד $s = c_{start}$ ואת הקדקוד $t = c_{accept}$.

<u>הערה:</u> שימו לב כי הבעיה $PATH$ הוגדרה על גרף מכוון.

מסקנה מידית ממשפט זה (מסקנה 8.26) היא ש- $\mathrm{NL} \subseteq \mathrm{P}$. אנו רואים כאן את התועלת הרבה בקיומן של בעיות שלמות עבור מחלקות סיבוכיות. כדי להוכיח כי NL כולה מוכלת ב-P, די לנו להוכיח זאת עבור בעיה אחת שהיא NL-שלמה, $PATH$ במקרה זה. ההיררכיה של מחלקות הסיבוכיות שפגשנו עד כה היא אפוא:

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$$

תרגיל 5.11

פתרו את בעיה 8.23 בספר הלימוד.

5.6 NL שווה ל-coNL

קראו את סעיף 8.6 בספר הלימוד. דלגו על הוכחת משפט 8.27

בסעיף זה מוכחת תוצאה מפתיעה: $\mathrm{NL} = \mathrm{coNL}$ (משפט 8.27).

נשים לב שהשאלה המקבילה האם $\mathrm{NP} = \mathrm{coNP}$ היא שאלה פתוחה, ומרבית החוקרים סוברים שאלה מחלקות שונות.

בסוף הסעיף מסוכם מה שידוע על היחס בין המחלקות L, NL, coNL, P, NP ו- PSPACE.

פתרון התרגילים

תרגיל 5.1

PSPACE סגורה תחת איחוד : נניח ש- $L_1, L_2 \in \mathrm{PSPACE}$ ו- M_1, M_2 הן מכונות המכריעות את השפות הללו בסיבוכיות מקום פולינומיאלית, $f_1(n)$ ו- $f_2(n)$ בהתאמה. נבנה מכונה שבהינתן קלט, היא מריצה עליו תחילה את M_1 ואחר כך את M_2 (תוך שימוש חוזר במקום) ומקבלת את הקלט אם ורק אם לפחות אחת משתי מכונות אלו קיבלה אותו. סיבוכיות המקום של מכונה כזו היא $\max(f_1(n), f_2(n))$, כלומר, גם פולינומיאלית. לפיכך $L_1 \cup L_2 \in \mathrm{PSPACE}$.

PSPACE סגורה תחת פעולת המשלים, כיוון שאם M היא מכונה המכריעה שפה בסיבוכיות מקום פולינומיאלית, נחליף בה את המצבים q_{accept} ו- q_{reject} ונקבל מכונה המכריעה את השפה המשלימה באותה סיבוכיות.

לבסוף, אם $L \in \mathrm{PSPACE}$ ו- M היא מכונה המכריעה שפה זו בסיבוכיות מקום פולינומיאלית, נתאר מכונה אי-דטרמיניסטית המכריעה את L^* בסיבוכיות מקום פולינומיאלית (מה שיוכיח ש- $L^* \in \mathrm{NPSPACE} = \mathrm{PSPACE}$). המכונה האי-דטרמיניסטית תחלק את הקלט למספר תת-מחרוזות באופן אי-דטרמיניסטי, תפעיל את המכונה M על כל אחת מהתת-מחרוזות (תוך שימוש חוזר במקום), ותקבל את הקלט אם ורק אם כל התת-מחרוזות זוהו כמילים בשפה L. מכיוון שסיבוכיות המקום של כל הרצה של M היא פולינומיאלית באורך התת-מחרוזת (ולפיכך פולינומיאלית באורך הקלט כולו), אז סיבוכיות המקום הכוללת אף היא פולינומיאלית.

תרגיל 5.2

ב- $\mathbb{N}$ רק משפט 1 נכון.

ב- $\mathbb{Z}$ רק משפטים 1 ו-2 נכונים.

ב- $\mathbb{Q}$ רק משפטים 1, 2 ו-3 נכונים.

ב- $\mathbb{R}$ רק משפטים 1, 2, 3 ו-4 נכונים.

ב- $\mathbb{C}$ אין משמעות לארבעת המשפטים הראשונים, כיוון ש- $\mathbb{C}$ הוא שדה שאיננו סדור (לא מוגדר בו יחס סדר), אך משפט 5 נכון.

תרגיל 5.3

נניח שכל שפה NP-קשה גם הייתה PSPACE-קשה. אז SAT הייתה שפה PSPACE-קשה. לכן כל שפה ב-PSPACE הייתה ניתנת לרדוקציה ל- SAT בזמן פולינומיאלי. אבל אז, מכיוון שאת SAT ניתן להכריע על ידי מכונת טיורינג אי-דטרמיניסטית בזמן פולינומיאלי, היינו יכולים להכריע כל שפה ב-PSPACE על ידי מכונת טיורינג אי-דטרמיניסטית בזמן פולינומיאלי. מכאן היינו מסיקים ש- $\mathrm{PSPACE} \subseteq \mathrm{NP}$ ובפרט ש- $\mathrm{NP} = \mathrm{PSPACE}$.

תרגיל 5.4

נתאר מכונה המקבלת מחרוזות בינאריות $w = w_1 \cdots w_n \in \{0,1\}^*$ ומחשבת את ביט הזוגיות המתאים $p = w_1 \oplus \cdots \oplus w_n$. המכונה תחשב את ביט הזוגיות אל תוך הביט הראשון של סרט העבודה, שערכו ההתחלתי יהיה 0. בכל שלב בלולאה תקרא המכונה את הביט הבא מסרט הקלט ותעשה לו XOR עם הביט הראשון (והיחיד) על סרט העבודה. סיבוכיות המקום של מכונה זו היא $f(n) = 1$ אך סיבוכיות הזמן היא $O(n)$.

תרגיל 5.5

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 5.6

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 5.7

תהי $w \in \{(,)\}^*$ מחרוזת של סוגריים באורך n. נגדיר את המונים c_i עבור $0 \le i \le n$ באופן הבא:

$$c_0 = 0 \quad , \quad c_i = \begin{cases} c_{i-1} + 1 & \text{if} \quad w_i = (\\ c_{i-1} - 1 & \text{if} \quad w_i =) \end{cases} \quad i > 0$$

קל לראות ש-w מייצגת סוגריים המקוננים באופן חוקי אם ורק אם $c_i \ge 0$ לכל $0 \le i \le n$ ו-$c_n = 0$ (הוכיחו זאת באינדוקציה על אורך המחרוזת n; ניתן כמובן להתרכז רק בערכי n זוגיים).

מכונת טיורינג יכולה להכריע שפה זו בסיבוכיות מקום לוגריתמית. סרט העבודה יאותחל לערך אפס והוא ישמור בכל שלב בלולאה את הערך המתאים של c_i, מקודד בינארית. אם בשלב כלשהו בריצה תקרא המכונה מסרט הקלט סוגר ימני, כאשר ערך המונה הנוכחי הוא אפס, היא תעצור ותדחה את הקלט. אם מצב זה לא יקרה אף פעם, היא תקבל את הקלט אם ורק אם ערך המונה בתום הריצה הוא אפס.

תרגיל 5.8

מכיוון ש-$B \notin \{\varnothing, \Sigma^*\}$, אנו יכולים לבחור שתי מילים: אחת בשפה $u \in B$, והשנייה מחוצה לה $v \notin B$. כעת, בהינתן קלט $w \in \Sigma^*$, נפעיל עליו את האלגוריתם הפולינומיאלי להכרעת השפה A. אם נגלה ש-$w \in A$, אז הערך שתתאים לו הרדוקציה יהיה $f(w) = u \in B$, אחרת $f(w) = v \notin B$.

תרגיל 5.9

יהי M מתמר מקום לוגריתמי מהשפה A לשפה B. נניח שב-M יש q מצבים, ונסמן ב-$|\Gamma|$ את גודל אלפבית הסרט שלו. קונפיגורציה של M מורכבת מהפרטים האלה:

- תוכן סרט העבודה – $O(\log n)$ תווים מתוך אלפבית בגודל $|\Gamma|$;
- מקום הראש של M על סרט הקלט – n ערכים אפשריים;
- מקום הראש של M על סרט העבודה – $O(\log n)$ ערכים אפשריים;
- ומצב המכונה M – q מצבים אפשריים.

לכן, מספר הקונפיגורציות השונות של M הוא $|\Gamma|^{O(\log n)} \cdot n \cdot O\left(\log n\right) \cdot q$. מספר זה חסום על ידי $2^{k \cdot \log n} = n^k$ עבור קבוע כלשהו k. לכן זמן הריצה של M חסום אף הוא על ידי n^k, שכן מכונת טיורינג שעוצרת תמיד איננה יכולה להיות פעמיים באותה קונפיגורציה. לכן אורכו של $f(w)$ הוא לכל היותר n^k.

תרגיל 5.10

נניח ש-M_f הוא מתמר המחשב במקום לוגריתמי פונקציה f, כך ש-$w \in A$ אם ורק אם $f(w) \in B$. כמו כן נניח ש-M_g הוא מתמר המחשב במקום לוגריתמי פונקציה g כך ש-$w \in B$ אם ורק אם $g(w) \in C$. נגדיר $h = g \circ f$. ברור ש-$w \in A$ אם ורק אם $h(w) \in C$. על כן די להראות ש-h חשיבה במקום לוגריתמי, כלומר שהיא ניתנת לחישוב על ידי מתמר מקום לוגריתמי. לצורך כך נבנה מכונה M_h המחשבת את h על ידי הפעלת M_f על הקלט, ואז הפעלת M_g על הפלט של M_f. הבעיה במכונה זו היא נקודת התפר בין שני השלבים שלה: M_f מחשבת את $f(w)$ ואמורה לרשום מחרוזת זו על סרט העבודה כדי שתשמש כקלט ל-M_g. אך אורך $f(w)$ עלול לחרוג ממגבלות המקום הלוגריתמי. על כן, נשנה את פעולת המכונה M_h באמצעות אותו תכסיס שבו נקטנו בהוכחת משפט 8.23: המכונה M_h תפעיל כל פעם את M_f על מנת לחשב רק את התו הנוכחי מתוך $f(w)$ הנחוץ באותו שלב ל-M_g. כך, על ידי המרת סיבוכיות מקום בסיבוכיות זמן, אנו מקבלים מכונה שסיבוכיות המקום שלה היא לוגריתמית.

תרגיל 5.11

פתרון בעיה זו מופיע בספר הלימוד.

6. משפטי היררכיה (סעיף 9.1 בספר הלימוד)

קראו בספר הלימוד מתחילת פרק 9 עד לפני משפט 9.3.

בעיות כריעוֹת נקראות בעיות **קשות לפתרון** (intractable) אם פתרונן מחייב עלויות זמן או מקום שאינן מעשיות. בעיות רבות, כגון SAT, הן בעיות שאיננו יודעים איך לפתור אותן בצורה יעילה, והן מסתמנות כבעיות קשות לפתרון, אך אין לנו כל הוכחה לכך. בפרק זה נוכיח את קיומן של בעיות כריעוֹת שהן קשות לפתרון. נוכיח כאן משפטי היררכיה שאומרים, במילים פשוטות, שככל שעומדים לרשותנו יותר משאבי מקום או זמן, כוחנו החישובי עולה. כלומר, מכונות טיורינג שרשאיות לרוץ בזמן n^3 למשל (או להשתמש ב- n^3 תאים על הסרט), יוכלו להכריע יותר שפות ממכונות המוגבלות לזמן ריצה (או מקום) של n^2. מזה ינבע שיש שפות שאפשר להכריע אותן בזמן 2^n, למשל, אך אי אפשר להכריע אותן בזמן פולינומיאלי n^k לכל k טבעי.

נתחיל במשפט ההיררכיה של מקום. לצורך כך מגדירים (הגדרה 9.1) פונקציה $f:\mathbb{N}\to\mathbb{N}$ **ניתנת לבנייה במגבלת מקום עצמית** (space constructible), אם $f(n)=\Omega(\log n)$ וניתן לחשב את המיפוי מהמחרוזת 1^n למחרוזת המייצגת בינארית את $f(n)$ בסיבוכיות מקום של $O(f(n))$.

על מנת לכלול בהגדרה זו גם פונקציות תת-ליניאריות, $f(n)=o(n)$ (כגון $f(n)=\lfloor\sqrt{n}\rfloor$ או $f(n)=\lfloor\log n\rfloor$), אנו מניחים שבמקרה של פונקציה תת-ליניארית הקלט 1^n נמצא על סרט קלט לקריאה בלבד, וסיבוכיות המקום $O(f(n))$ מתייחסת למספר התאים שהמכונה משתמשת בהם על סרט העבודה.

קראו את משפט היררכיית המקום (משפט 9.3) והוכחתו

משפט היררכיית המקום אומר שאם $f:\mathbb{N}\to\mathbb{N}$ היא פונקציה הניתנת לבנייה במגבלת מקום עצמית, אז יש שפה $A\in \mathrm{SPACE}(f(n))$ כך ש- $A\notin \mathrm{SPACE}(g(n))$ לכל $g(n)=o(f(n))$.

כלומר, אם $f:\mathbb{N}\to\mathbb{N}$ היא פונקציה הניתנת לבנייה במגבלת מקום עצמית ו- $g(n)=o(f(n))$, אז שתי מחלקות הסיבוכיות המתאימות יוצרות היררכיה אמיתית במובן ש- $\mathrm{SPACE}(g(n))\subsetneq \mathrm{SPACE}(f(n))$.

במשפט מצביעים על שפה A הניתנת להכרעה בסיבוכיות מקום $O(f(n))$ אך איננה ניתנת להכרעה בסיבוכיות מקום $o(f(n))$. התיאור של השפה A איננו תיאור ישיר. במקום לומר מהו מבנה המילים בשפה או מה היא מתארת, כפי שנהגנו לעשות, אנו מתארים את השפה A תיאור

עקיף על ידי תיאור פעולתה של מכונה המכריעה אותה. נסמן מכונה זו ב- D. המכונה D תגביל את עצמה לשימוש ב- $O\left(f(n)\right)$ מקום, על מנת להבטיח ש- $A \in \mathrm{SPACE}(f(n))$. כמו כן, על מנת לוודא ש- $A \notin \mathrm{SPACE}(g(n))$ לכל $g(n) = o\left(f(n)\right)$, נדאג שהשפה ש- D מכריעה תהיה שונה מכל אחת מהשפות המוכרעות על ידי מכונות טיורינג שסיבוכיות המקום שלהן היא $o\left(f(n)\right)$.

הבנייה משתמשת בעקרונות של שיטת האלכסון שבה השתמשנו בהוכחת משפט 4.11. בהינתן ל- D קלט w, המכונה D בודקת אם w מהווה תיאור של מכונת טיורינג כלשהי. רוב המחרוזות, כמובן, אינן מהוות תיאורים של מכונות טיורינג; בהינתן קלטים כאלו, D תדחה אותם. לעומת זאת, אם $w = \langle M \rangle$ עבור מכונת טיורינג כלשהי M, אז D תתחיל להריץ את M על הקלט $w = \langle M \rangle$ (זהו בדיוק ה"אלכסון": ההפעלה של מכונה על עצמה כקלט). אבל D תריץ את M בזהירות: היא תגביל את הריצה לשימוש ב- $f(n)$ תאי סרט בלבד (n הוא אורך הקלט, $n = |w|$). אם M תעצור מבלי לחרוג ממגבלת מקום זו, אז D תקבל את הקלט $w = \langle M \rangle$ אם M דחתה אותו, ולהפך - היא תדחה אותו אם M קיבלה אותו. מצד שני אם D תגלה שהמכונה M מנסה לחרוג מעבר למגבלת המקום של $f(n)$, היא תעצור את הסימולציה ותדחה את הקלט.

אם כן, תיארנו בקווים כלליים מכונה D המכריעה שפה A. המכונה, מעצם תיאור אופן הפעולה שלה, איננה משתמשת ביותר מ- $f(n)$ תאי סרט. לכן, השפה A שהיא מכריעה מקיימת $A \in \mathrm{SPACE}(f(n))$, כנדרש, מחד. מאידך, מכונת טיורינג כלשהי, M, הצורכת $o\left(f(n)\right)$ תאי סרט, תהיה מכונה שתעצור במסגרת מגבלת המקום ש- D מציבה לה, ואז, לפי אופן הפעולה של D, $L(M) \neq A$ (אם $w = \langle M \rangle \in L(M)$, אז $w \notin A$ ולהפך). לכן, A שונה מכל השפות הכריעות בסיבוכיות מקום $o\left(f(n)\right)$.

התיאור שניתן לוקה בחסר. נניח ש- $f(n) = n^2$ וש- M היא מכונה שסיבוכיות המקום שלה היא $g(n) = 100n$. ברור ש- $g(n) = o\left(f(n)\right)$. מכאן נובע שעבור ערכים מספיק גדולים של n, $g(n) < f(n)$ (בדוגמה זו, מתקיים האי-שיוויון לכל $n > 100$). אך דא עקא: עבור n-ים קטנים יתכן ש- $g(n) > f(n)$. אז כאשר D תריץ את M לעיל על גבי קלט שאורכו, נאמר, $n = 50$, היא תגביל את הריצה ל- 50^2 תאי סרט. מכיוון ש- M תנסה להשתמש ביותר תאים במקרה זה, המכונה D תעצור את הריצה ותדחה את הקלט, משום שהיא "תחשוב" שהמכונה M "בסדר" (דהיינו, שסיבוכיות המקום שלה גדולה מ- $o\left(f(n)\right)$). D תמשיך כעת בפעולתה מבלי שווידאה כי קיימת מחרוזת בשפה המוכרעת על ידי M שאיננה ב-A (או מחרוזת ב-A שאיננה בשפה המוכרעת על ידי M). אם כן, יתכן, כי $L(M) = A$, ואז המסקנה היא שהבנייה שלנו לא הוכיחה את הדרוש.

התיקון לבעיה שתוארה לעיל הוא פשוט: המכונה D שתוארה לעיל פעלה באופן "מושכל" רק על קלטים המהווים תיאור של מכונת טיורינג כלשהי, $w = \langle M \rangle$, ודחתה מיד כל קלט שאיננו כזה.

נתקן את המכונה D כך שהיא תטפל באופן דומה בכל קלט שצורתו $w = \langle M \rangle 1\overbrace{0\cdots 0}^{k}$ עבור k טבעי כלשהו. כלומר, כל קלט שאיננו מהצורה הזו יידחה, אך בהינתן קלט מהצורה הזו היא תריץ את המכונה המתאימה M על הקלט ותגביל את השימוש בתאי סרט ל- $f(n)$ לכל היותר, כאשר $n = |w|$. אם M היא מכונה בעלת סיבוכיות מקום $g(n) = o\left(f(n)\right)$, אז קיים k כזה שעבורו יתקיים $g(n) < f(n)$ (בעצם, קיימים אינסוף ערכי k כאלה). לכן, השפות A ו- $L(M)$ נבדלות זו מזו במילים שצורתן $w = \langle M \rangle 1\overbrace{0\cdots 0}^{k}$ עבור ערכי k כאלה. המטרה הושגה.

תיקון טכני אחרון: D צריכה להיזהר מהרצת מכונות M שנכנסות ללולאה אינסופית, מכיוון ש- D צריכה להיות מכונה מכריעה, דהיינו, מכונה שעוצרת תמיד. הסכנה קיימת רק לגבי מכונות שמסתפקות ב- $f(n)$ תאי סרט (אם המכונה M תרצה, בשלב מסוים, לחרוג ממגבלת המקום הזו, אז פעולתה תופסק מיד, ולכן אין זה משנה אם מדובר במכונה הנכנסת ללולאה אינסופית). אך מכונה שמסתפקת ב- $f(n)$ תאי סרט חייבת לעצור תוך $q \cdot f(n) \cdot |\Gamma|^{f(n)}$ צעדים, אחרת היא תיכנס ללולאה אינסופית, מכיוון שזהו מספר הקונפיגורציות האפשריות שלה (q הוא מספר המצבים של המכונה M ו- $|\Gamma|$ הוא גודל אלפבית הסרט). לפיכך, D תנהל מונה צעדים ותגביל את ההרצה של M לא רק במקום, אלא גם בזמן. אם M תחרוג ממגבלת הזמן לעיל, הרי שזו איננה מכונה מכריעה ולכן אין ל- D שום סיבה "לחשוש" ממנה; במקרה זה נעצור ונדחה את הקלט.

כעת, תיאור המכונה D המופיע בהוכחת המשפט וההסברים המופיעים לאחר ההוכחה אמורים להיות ברורים. שימו לב שבשלב 4 יש לתקן את מגבלת הזמן מ- $2^{f(n)}$ ל- $q \cdot f(n) \cdot |\Gamma|^{f(n)}$; המכונה D יכולה לחשב את הערך הזה כיוון ש- $f(n)$ ניתנת לבנייה במגבלת מקום עצמית (כלומר, החישוב שלה איננו מצריך יותר מ- $O\left(f(n)\right)$ תאי סרט), הערכים q ו- $|\Gamma|$ ניתנים לחילוץ מתוך $\langle M \rangle$ והביטוי $q \cdot f(n) \cdot |\Gamma|^{f(n)}$ אף הוא חשיב במגבלת המקום שלנו.

המשיכו לקרוא עד לפני הגדרה 9.8

בקטע זה מוצגות ארבע מסקנות פשוטות ממשפט היררכיית המקום. כולן עוסקות בהפרדה בין מחלקות סיבוכיות המוכלות זו בזו. המסקנה האחרונה, מסקנה 9.7, מוכיחה את קיומן של בעיות קשות לפתרון (intractable). כלומר, יש בעיות הניתנות לפתרון במקום מעריכי שאין כל אפשרות לפתור אותן במקום פולינומיאלי.

<u>שאלה:</u>

הראו ש- $n^{\log n} = o\left(2^n\right)$.

<u>תשובה:</u>

מכיוון ש- $n^{\log n} = \left(2^{\log n}\right)^{\log n} = 2^{(\log n)^2}$ וקל לראות ש- $(\log n)^2 = o(n)$, אז אכן $n^{\log n} = o\left(2^n\right)$.

המשיכו לקרוא עד מסקנה 9.13

כעת נעסוק בהירככיית זמן. תחילה מגדירים (הגדרה 9.8) **פונקציות הניתנות לבנייה במגבלת זמן עצמית** (time constructible). **משפט היררכיית הזמן** אומר שאם $t : \mathbb{N} \to \mathbb{N}$ היא פונקציה הניתנת לבנייה במגבלת זמן עצמית, אז יש שפה $A \in \mathrm{TIME}(t(n))$ כך ש- $A \notin \mathrm{TIME}(s(n))$ לכל $s(n) = o\left(t(n) / \log t(n)\right)$. כלומר, אם $t : \mathbb{N} \to \mathbb{N}$ היא פונקציה הניתנת לבנייה במגבלת זמן עצמית ו- $s(n) = o\left(t(n) / \log t(n)\right)$, אז שתי מחלקות הסיבוכיות המתאימות יוצרות הירכיה אמיתית במובן ש- $\mathrm{TIME}(s(n)) \subsetneq \mathrm{TIME}(t(n))$.

ההוכחה דומה להוכחה של משפט 9.3. גם כאן מוגדרת השפה A על ידי תיאור המכונה D המכריעה אותה. המכונה D תמיד עוצרת בזמן $O\left(t(n)\right)$, כך ש- $A = L(D) \in \mathrm{TIME}(t(n))$. מצד שני, D תדאג לפעול באופן שונה מכל מכונה M שרצה בזמן שהוא $o\left(t(n) / \log t(n)\right)$. זה נעשה על ידי בחינת קלטים שצורתם $w = \langle M \rangle 10^*$ והרצת M עליהם. אם M עוצרת בתוך $t(n) / \log t(n)$ צעדי זמן, אז D תוציא את הפלט ההפוך לזה ש- M נותנת. אחרת, אם M אינה עוצרת בתוך $t(n) / \log t(n)$ צעדי זמן או אם הקלט איננו מהצורה $w = \langle M \rangle 10^*$, אז D דוחה את הקלט.

ההבדל בין משפט זה למשפט המקביל, שעסק בהירככיית מקום, הוא שעל D למנות את צעדי **הזמן** של M ולא את צריכת **המקום** שלה. במשפט 9.3, אם M הייתה זקוקה ל- $g(n)$ תאי סרט, אז D הייתה זקוקה ל- $d \cdot g(n)$ תאי סרט, עבור קבוע כלשהו d. סיבת הניפוח (האפשרי) בפקטור הכפלי d נעוצה בכך שייתכן שהאלפבית של המכונה M רחב יותר מזה של המכונה D: למשל, אם D היא מכונה בינארית ואילו M היא מכונה עשרונית, אז D תיאלץ להקדיש $d = 4$ תאים על הסרט שלה על מנת לקודד תא אחד על הסרט של M. ברגע שחושב פקטור הניפוח d, קל לשלוט על צריכת המקום של M ללא תשלום מחיר נוסף: המכונה D תסמן מראש, באמצעות תו מיוחד, עד היכן בסרט מותר לה להגיע, ואז אם במהלך הסימולציה היא תפגוש בתו זה – היא תדע שבוצעה חריגת מקום. גם חישוב ערכו של מונה הצעדים והטיפול במונה זה בזמן הריצה של המכונה D ניתנים לביצוע במגבלת המקום $O\left(f(n)\right)$.

במקרה שלנו, D אמורה להחזיק מונה צעדים לספירת צעדי המכונה M. תחילה מחשבים את מספר הצעדים המותר $\lceil t(n)/\log t(n) \rceil$. לאחר מכן מחסרים 1 מן המונה על כל סימולציה של צעד אחד של M. כדי לצמצם את מספר הצעדים הדרוש לניהול המונה, מחזיקים את המונה סמוך לראש הקורא של D, כפי שמוסבר בהמשך. זה דורש הזזה של המונה לאחר ביצוע הסימולציה של כל צעד של M. הגודל של המונה הוא $O\left(\log\left(t(n) / \log t(n)\right)\right)$ (המונה מוחזק בייצוג בינרי). לכן מספר הצעדים הדרוש להזזתו הוא $O\left(\log\left(t(n) / \log t(n)\right)\right) = O\left(\log t(n)\right)$. זו הסיבה לכך שאנו צריכים "להרחיק" את סיבוכיות הזמן של M מזו של D בגורם $1/\log t(n)$.

האלגוריתם של D המתואר בתחילת ההוכחה של משפט היררכיית הזמן דומה לאלגוריתם של המכונה שתוארה בהוכחה של משפט היררכיית המקום – ההבדל היחידי הוא קריטריון העצירה: שם עצרנו אם חרגנו ממגבלת המקום או אם עברנו מספר צעדים שהעיד על כך שהמכונה שאנו מדמים נכנסה ללולאה אינסופית; כאן אנו מסתפקים בבדיקת מספר הצעדים של המכונה הנבדקת.

הדבר הראשון באימות האלגוריתם של D הוא בדיקת מספר הצעדים שלו. עלינו לוודא שהמכונה המתוארת D עוצרת בזמן $O\left(t(n)\right)$:

1. שלב 2 - חישוב $t(n)$ אורך $O\left(t(n)\right)$ צעדים, בשל היותה של $t(n)$ פונקציה הניתנת לבנייה במגבלת זמן עצמית. אחר כך עלינו לחשב את הערך $\lceil t(n)/\log t(n) \rceil$. לא קשה לוודא שבהינתן מספר בינארי k, ניתן לחשב את $\lceil k/\log k \rceil$ ב-$O(k)$ צעדים. לכן גם חישוב זה איננו מביא לחריגה ממגבלת הזמן שברצוננו לעמוד בה.
2. שלב 3 - הבדיקה שהקלט הוא מהצורה $w = \langle M \rangle 10^*$ ניתנת לביצוע במספר צעדים שהוא $O(n) = O\left(t(n)\right)$.

השלבים שהתעכבנו עליהם עד כה הם השלבים הקלים. אך עיקר העבודה של D הוא בשלב 4 – חיקוי הפעולה של M על w, ובשלב 2 – הפחתת ערך מונה הצעדים ב-1. הבעיה היא שלרשות D עומד סרט עבודה אחד המכיל גם את תיאור המכונה M, גם את הקונפיגורציה הנוכחית שלה (מצבה, מקום הראש ותוכן הסרט שלה) וגם את מונה הצעדים שמנהלת D. אם לא נהיה זהירים בבניית המכונה D, אנו עלולים לבזבז הרבה זמן ב**דילוגים** לאורך הסרט כדי לאסוף את פיסות האינפורמציה הנחוצות לחישוב הצעד הנוכחי של M וכדי לעדכן ערכים על הסרט בעקבות ביצוע צעד זה. דילוגים אלו עלולים להביא לכך שכל צעד של המכונה M יתורגם ליותר מאשר $O\left(\log t(n)\right)$ צעדים של המכונה D. מכיוון שהסימולציה של M נמשכת $\lceil t(n)/\log t(n) \rceil$ צעדים, זה יביא את סיבוכיות הזמן של D אל מעבר למגבלת ה-$O\left(t(n)\right)$ שהצבנו לעצמנו.

אילו עמדו לרשות המכונה D כמה סרטים, היינו יכולים לפצל את האינפורמציה בין סרטים אלו, ואז הראש הקורא על כל סרט היה צריך לנוע מספר קטן של צעדים בכל שלב. מכיוון ש-D מצוידת בסרט אחד בלבד, אנו יוצרים בו באופן מלאכותי שלוש **רצועות** (tracks) על מנת לדמות שלושה סרטי עבודה. יצירת רצועות שונות בסרט אחד יכולה להתבצע בשני אופנים: על ידי יצירת סרט מסורג, כך ששלושת הסרטים האלה

a	b	c	d	e	$\cdots$

1	2	3	4	5	$\cdots$

*	#	&	+	$-$	$\cdots$

ייוצגו על סרט אחד באופן הבא:

a	1	*	b	2	#	c	3	&	d	4	+	e	5	$-$	$\cdots$

או על ידי יצירת סרט שכבות, כך ששלושת הסרטים מהדוגמה לעיל ייוצגו על סרט אחד באופן הבא:

a	b	c	d	e	
1	2	3	4	5	$\cdots$
*	#	&	+	$-$	

בסרט השכבות לעיל הגדלנו את אלפבית הסרט כך שכל אות מייצגת שלשה של אותיות על הסרטים המקוריים.

במימוש שלנו של D יהיו, כאמור, שלוש רצועות:

1. הרצועה הראשונה (העליונה) תשמור את התוכן הנוכחי של הסרט של M ותו מיוחד שיסמן את מיקומו הנוכחי של הראש של M על הסרט.
2. הרצועה השנייה תשמור את המצב הנוכחי של M ואת תיאור פונקציית המעברים שלה.
3. הרצועה השלישית תשמור את מונה הצעדים של M שאותחל בשלב 2 לחסם העליון ומופחת ב-1 על כל צעד של M.

פונקציית המעברים של D תדאג לעדכן בכל צעד את כל שלוש הרצועות על הסרט. זכרו שהקלטים המעניינים אותנו הם קלטים מהצורה $w = \langle M \rangle 1 \overbrace{0 \cdots 0}^{k}$, כאשר k הוא מספר טבעי כלשהו, שיכול להיות גדול מאוד. יחסית ל-k שיכול לגדול כרצוננו, אפשר להתייחס לגודל של $\langle M \rangle$ כאל קבוע. לפיכך, אורך הרצועה הראשונה בתחילת ריצתה של המכונה הוא $O\left(n = |\langle M \rangle| + 1 + k\right)$. הרצועה השנייה קצרה ואורכה קבוע – $O\left(|\langle M \rangle|\right)$. אורך הרצועה השלישית הוא $O\left(\log \lceil t(n) / \log t(n) \rceil\right) = O\left(\log t(n)\right)$. הרעיון הוא לדאוג להזיז את תוכן הרצועה השנייה והרצועה השלישית, כך שהוא תמיד יופיע סמוך למקום הראש הקורא ברצועה הראשונה. לפיכך, על כל צעד של M המכונה D:

1. תקרא את תו הקלט מהרצועה הראשונה, המיוצג על ידי d תווי קלט באלפבית של D.

   עלות: $O(1)$

2. תקרא את מצב המכונה M מהרצועה השנייה.

   עלות: $O(1)$

3. תסרוק את תיאור פונקציית המעברים של M, כפי שמופיע על הרצועה השנייה, ותשלוף את כלל המעבר המתאים לתו ולמצב שנקראו.

   עלות: $O(1)$

4. תעדכן את התו על הרצועה הראשונה בהתאם לכלל המעבר שנקרא בשלב 3.

   עלות: $O(1)$

5. תעדכן את המצב של M על הרצועה השנייה בהתאם לכלל המעבר שנקרא בשלב 3.

   עלות: $O(1)$

6. תפחית 1 מהמונה ברצועה השלישית.

   עלות: $O\left(\log t(n)\right)$

7. תעדכן את מקום הראש של M על גבי הרצועה הראשונה בהתאם לכלל המעבר שנקרא בשלב 3.

   עלות: $O(1)$

8. תזיז את תוכן הרצועה השנייה בהתאם לתנועת הראש בשלב 7.

   עלות: $O(1)$

9. תזיז את תוכן הרצועה השלישית בהתאם לתנועת הראש בשלב 7.

   עלות: $O\left(\log t(n)\right)$

אם כך, כל אחד מ-$O\left(t(n)/\log t(n)\right)$ הצעדים של המכונה M מתורגם ל-$O\left(\log t(n)\right)$ צעדים של המכונה D. מכיוון ש-D עוצרת לאחר $\left\lceil t(n)/\log t(n)\right\rceil$ צעדים של המכונה M, היא עוצרת לאחר $O\left(t(n)\right)$ צעדים.

אם כן $A = L(D)$ ניתנת להכרעה בזמן $O\left(t(n)\right)$, כנדרש. נותר להראות שהיא אינה כריעה בזמן $g(n) = o\left(t(n)/\log t(n)\right)$. נניח, בשלילה, שהיא כריעה על ידי מכונה כלשהי M בזמן $g(n)$, ונניח שהזמן הדרוש לביצוע הסימולציה של M על-ידי D הוא $dg(n)$ עבור קבוע d כלשהו. קיים n_0 מספיק גדול שעבורו $dg(n) < t(n) / \log t(n)$ לכל $n \geq n_0$. לכן, בהינתן קלטים מהצורה $w = \langle M \rangle 1\overbrace{0\cdots0}^{n}$ כאשר $n \geq n_0$, מובטח לנו שהסימולציה ש-D עושה למכונה M

תגיע לכדי עצירה. לכן, אם M מקבלת את w, אז D תדחה אותה, ולהפך. לפיכך, $L(M) \neq A = L(D)$, בסתירה להנחתנו.

בזאת הסתיימה ההוכחה של משפט הירכיית הזמן.

תרגיל 6.1

פתרו את תרגיל 9.1 בספר הלימוד.

תרגיל 6.2

פתרו את תרגיל 9.2 בספר הלימוד.

תרגיל 6.3

פתרו את תרגיל 9.3 בספר הלימוד.

תרגיל 6.4

פתרו את בעיה 9.20 בספר הלימוד.

תרגיל 6.5

פתרו את בעיה 9.22 בספר הלימוד.

תרגיל 6.6

עיינו בהגדרה של פונקציית הריפוד (pad) בבעיה 9.21. פתרו את בעיה 9.23 בספר הלימוד.

פתרון התרגילים

תרגיל 6.1

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 6.2

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 6.3

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 6.4

הכשל ב"הוכחה" מופיע בשורה השלישית. כל שפה ב-NP אכן ניתנת לרדוקציית זמן פולינומיאלית ל- SAT, אך מעלת הפולינום המתאר את סיבוכיות הרדוקציה יכולה להיות מעלה כלשהי. לכן, אם $SAT \in \mathrm{TIME}\left(n^k\right)$, ואילו הרדוקציה $f : A \to SAT$ ניתנת לחישוב ב- $O\left(n^m\right)$, אז כל שאנו יכולים להסיק הוא ש- $A \in \mathrm{TIME}\left(n^{mk}\right)$. לפיכך, איננו יכולים להסיק, כמו שנעשה ב"הוכחה", ש- $\mathrm{NP} \subseteq \mathrm{TIME}\left(n^k\right)$.

תרגיל 6.5

השפה $pad(A,2^n)$ מוגדרת באופן הבא:

$$pad(A,2^n) = \left\{ s\#^{\left(2^{|s|}-|s|\right)} : s \in A \right\}$$

כלומר, זו השפה המתקבלת מהמילים בשפה A על ידי הוספת תווי $\#$ לסוף כל מילה, כך שאם אורך המילה המקורית היה n, אורך המילה ה"מרופדת" יהיה 2^n.

תהי $A \in \mathrm{NEXPTIME}$. אנו טוענים שבמקרה זה $pad(A,2^n)$ שייכת ל-NP: תהי M מכונת טיורינג אי-דטרמיניסטית המכריעה את A בזמן מעריכי. תהי N מכונת טיורינג הפועלת כדלקמן: אם אין תווי $\#$ בסוף מילת הקלט, N דוחה את מילת הקלט. אם יש תווי $\#$ בסוף מילת הקלט, כלומר מילת הקלט היא מהצורה $s\#^k$ כאשר s לא מכילה תווי $\#$, N בודקת האם $k = 2^{|s|} - |s|$. אם לא, N דוחה את מילת הקלט. אם כן, N "מקלפת" את תווי ה-$\#$, ואז מריצה את M על s.

המכונה האי-דטרמיניסטית N מכריעה את $pad(A,2^n)$ בזמן **פולינומיאלי** בגודל הקלט, שכן הבדיקות שהיא מבצעת ופעולת "קילוף" תווי ה-# ניתנות לביצוע בזמן פולינומיאלי באורך הקלט, ולאחר מכן – זמן הריצה של M על המילה שנותרה הוא מעריכי ב-n, כלומר פולינומיאלי ב-2^n, וזה בדיוק אורך מילת הקלט של N. על כן $pad(A,2^n) \in \mathrm{NP}$.

כעת נניח ש- NP=P. מהדיון לעיל נובע כי לכל שפה $A \in \mathrm{NEXPTIME}$, $pad(A,2^n)$ שייכת למחלקה NP, ולפיכך גם למחלקה P. אבל כעת אנו יכולים לבנות אלגוריתם דטרמיניסטי שזמן ריצתו מעריכי המכריע את A: בהינתן מחרוזת s, נחשב את $r = pad(s,2^{|s|})$; שימו לב ש-$|r| = 2^{|s|}$. כעת, מכיוון ש-$pad(A,2^n) \in \mathrm{P}$, אנו יכולים להכריע בזמן פולינומיאלי ב-$|r| = 2^{|s|}$ אם $r \in pad(A,2^n)$ או לא. אם נמצא ש-$r \in pad(A,2^n)$, אז נכריע ש-$s \in A$, אחרת $s \notin A$. זמן ריצה פולינומיאלי ב-$|r| = 2^{|s|}$ הוא מעריכי באורך הקלט המקורי, $|s|$. אם כן, A ניתנת להכרעה על ידי אלגוריתם דטרמיניסטי שזמן ריצתו מעריכי. מכיוון ש-A הייתה שפה כלשהי ב-NEXPTIME, אנו מסיקים ש- NEXPTIME=EXPTIME.

לסיכום, אם NP=P אז NEXPTIME=EXPTIME. כלומר, אם $\mathrm{NEXPTIME} \neq \mathrm{EXPTIME}$, אז $\mathrm{P} \neq \mathrm{NP}$.

תרגיל 6.6

פתרון הבעיה מופיע בספר הלימוד.

7. נושאים מתקדמים בתורת הסיבוכיות (פרק 10 בספר הלימוד)

7.1 אלגוריתמי קירוב

קראו את סעיף 10.1 בספר הלימוד

בבעיות אופטימיזציה עלינו להביא למינימום (או למקסימום) פונקציית מטרה f בכפוף לאוסף אילוצים. בחשבון דיפרנציאלי ואינטגרלי פגשנו בעיות כאלו עבור פונקציות של משתנים רציפים. למשל, בהינתן פונקציה $f(x,y)$ של שני משתנים ממשיים, $x,y \in \mathbb{R}$, מהם ערכי הקיצון (מינימום ומקסימום) שלה המתקבלים בתחום כלשהו (נאמר, במעגל היחידה שבו המשתנים x,y מוגבלים על-ידי האילוץ $x^2+y^2 \le 1$). בתאוריה של מדעי המחשב, לעומת זאת, כאשר מדברים על בעיות אופטימיזציה, הפונקציות הנדונות הן פונקציות של משתנים המקבלים ערכים על האוסף $2^{|A|}$ של התת-קבוצות של קבוצה נתונה A.

בסעיף זה מנותחות שתי בעיות מסוג זה.

כיסוי מינימום של קשתות על ידי קדקודים (*MIN-VERTEX-COVER*)

- <u>קלט:</u> גרף לא מכוון $G=(V,E)$.
- <u>פלט:</u> כיסוי כל הקשתות של G על ידי קבוצת קדקודים ב-G שגודלה מינימלי. במילים אחרות, עלינו למצוא $U \subseteq V$, כך שכל קשת ב- E נוגעת באיזשהו קדקוד ב- U, ו-$|U|$ מינימלי.

חתך מקסימום (*MAX-CUT*)

- <u>קלט:</u> גרף לא מכוון $G=(V,E)$.
- <u>פלט:</u> חתך ב- G בעל גודל מקסימלי. (חתך הוא חלוקה של צומתי הגרף לשתי קבוצות S ו- T ($S\cup T=V$, $S\cap T=\varnothing$). גודל החתך הוא מספר הקשתות המחברות צומת ב- S עם צומת ב- T).

נוכל להציג את *MIN-VERTEX-COVER* כבעיית אופטימיזציה באופן הבא: f היא פונקציה על התת-קבוצות של V, המתאימה לכל תת-קבוצה $X \subseteq V$ את גודלה $f(X)=|X|$. לכל מופע נתון $G=(V,E)$ של הבעיה, נסמן ב- $VC(G)$ את אוסף כל התת-קבוצות של V שהן כיסויים בקדקודים (זהו התחום שבו אנו מחפשים את הפתרון). אנו רוצים לפתור את הבעיה

$$\min\left\{ f(X) : X \in VC(G) \right\}$$

כלומר, למצוא $X \in VC(G)$ שעבורו $f(X)$ מינימלי.

האלגוריתם A המתואר לפתרון בעיית *MIN-VERTEX-COVER* מוציא כפלט כיסוי של הגרף שאיננו בהכרח אופטימלי, אך גודלו הוא לכל היותר פעמיים גודל כיסוי אופטימלי. האלגוריתם פשוט מאוד. אנו בונים את הכיסוי X בהדרגה. מתחילים עם $X = \varnothing$, וכל עוד יש ב- G קשתות, בוחרים קשת $e = (u,v)$, מוסיפים את הקצוות שלה לקבוצה X, כלומר $X \leftarrow X \cup \{u,v\}$, ומוחקים מהגרף את u ו- v ואת כל הקשתות הסמוכות אליהם. כלומר, האלגוריתם הוא:

אתחול: $X \leftarrow \varnothing$, $H \leftarrow \varnothing$

כל עוד יש ב- G קשת $e = (u,v)$, בצע:

$X \leftarrow X \cup \{u,v\}$ (הוסף את u,v ל- X)

$H \leftarrow H \cup \{e\}$ (הוסף את e ל- H)

$G \leftarrow G \setminus \{u,v\}$ (הסר מהגרף את u,v ואת כל הקשתות הסמוכות אליהם)

נעיר כי הקבוצה H והעדכון שלה במהלך האלגוריתם נועדו לצורכי ניתוח יחס הקירוב בלבד, ולמעשה ניתן להשמיט אותה מהאלגוריתם. קל לראות שני דברים אם זוכרים שאין ב- H קשתות הסמוכות זו לזו:

- $|X| = 2|H|$.
- לכל פתרון אפשרי Y (כלומר, קבוצת קדקודים המכסה את כל קשתות הגרף), מתקיים $|Y| \geq |H|$ (כי Y חייב להכיל לפחות קצה אחד של כל קשת ב- H, ואין ב- H שתי קשתות עם צומת משותף).

מכאן נובע כי $|X| \leq 2|Y|$ לכל פתרון אפשרי Y (ובפרט, אם Y הוא פתרון אופטימלי).

אנו אומרים שאלגוריתם זה הוא בעל יחס קירוב 2 (המושג יחס קירוב יוגדר בהמשך בצורה מדויקת), מכיוון שהוא מחשב פתרון שגרוע יותר מהפתרון האופטימלי בגורם כפלי לכל היותר 2. נעיר שלא ידוע אלגוריתם פולינומיאלי בעל יחס קירוב טוב יותר (כלומר, קטן יותר) מ- 2 ל-*MIN-VERTEX-COVER*; יתרה מכך, ישנן עדויות שלפיהן כנראה אין לבעיה אלגוריתם פולינומיאלי שיש לו יחס קירוב $2-\varepsilon$, עבור איזשהו $\varepsilon > 0$ (אלא אם כן $\mathrm{P} = \mathrm{NP}$).

בהמשך מתואר אלגוריתם B המחשב פתרון מקורב לבעיית *MAX-CUT*. ניתוח יחס הקירוב של האלגוריתם הזה הוא גם כן פשוט מאוד: האלגוריתם מוצא חתך בגודל $|E|/2$ לפחות, בעוד שגודלו של כל חתך הוא לכל היותר $|E|$. נעיר כאן כי לבעיית *MAX-CUT* ידוע אלגוריתם מסובך הרבה יותר עם יחס קירוב $1/0.87856$, (של החוקרים M. Goemans, D. Williamson משנת

1995). זהו יחס הקירוב הטוב ביותר שידוע לבעיה זו, ולאחרונה יש עדויות כי ייתכן שלא ניתן להשיג יחס קירוב טוב יותר (אלא אם כן $\mathrm{P} = \mathrm{NP}$).

נמשיך ונתאר עוד שתי בעיות אופטימיזציה חשובות:

בעיית הסוכן הנוסע (Traveling Salesman Problem או בקיצור *TSP*)

- קלט: גרף לא מכוון מלא $G = (V, E)$, עם מחירים אי-שליליים על הקשתות.
- פלט: מעגל המילטוני ב-G בעל עלות מינימלית. (מעגל המילטוני הוא מעגל העובר דרך כל הקדקודים בגרף, פעם אחת בדיוק בכל קדקוד.)

עץ שטיינר מינימלי (Minimal Steiner Tree)

- קלט: גרף לא מכוון $G = (V, E)$, עם מחירים אי-שליליים על הקשתות ותת-קבוצה של הקדקודים, $R \subseteq V$.
- פלט: עץ בעל מחיר מינימלי מבין כל העצים המוכלים ב-G ושקבוצת הקדקודים שלהם מכילה את R.

הבעיה האחרונה היא הכללה של בעיית העץ הפורש המינימלי (המתקבלת כאשר $R = V$), והיא בעלת מוטיבציה דומה. יש לבנות רשת שתקשר בין טרמינלים (הקדקודים ב-R), כאשר ניתן לקשר בין טרמינלים גם דרך קדקודי מעבר (הקדקודים ב-$V \setminus R$). המטרה היא לבנות רשת זולה ככל האפשר המקיימת את הדרישה.

באופן כללי, לבעיית אופטימיזציה Π יש אוסף (בדרך כלל אינסופי של) קלטים $\Im_\Pi$. בהינתן קלט $I \in \Im_\Pi$, יש לכל פתרון אפשרי σ של הבעיה ערך מסוים $f(\sigma) \in \mathbb{R}$ (המציין את עלות הפתרון, או את הרווח שפתרון זה יכול להפיק). בהינתן קלט I, אנו מחפשים פתרון σ כך ש-$f(\sigma)$ יהיה אופטימלי (כלומר מינימלי או מקסימלי, בהתאם לדרישה). בבעיית הסוכן הנוסע, הקלטים הם כל הגרפים עם מחירים על הקשתות (יש כמובן אינסוף גרפים כאלה). פתרון אפשרי σ הוא סידור של הקדקודים במעגל, שכן סידור כזה קובע מיד פתרון לפי סדר הקדקודים במעגל. לחלופין, אפשר גם להגדיר פתרון אפשרי כ"קבוצת קשתות היוצרות מעגל המילטוני". ערך הפתרון $f(\sigma)$ הוא סכום מחירי הקשתות המשתתפות במעגל (הקשתות שמחברות את הקדקודים העוקבים במעגל).

כפי שכבר ראינו, רבות מבעיות היסוד שייכות לקטגוריה של בעיות NP-קשות. מציאת אלגוריתם אשר רץ בזמן סביר (כלומר, בזמן פולינומיאלי בגודל הקלט) עבור אחת מבעיות אלה, תגרור קיום אלגוריתם פולינומיאלי לכל בעיה ב-NP. מתחילת שנות ה- 70 של המאה הקודמת, עסקו חוקרים רבים בעולם בעיצוב אלגוריתמים יעילים לבעיות אופטימיזציה שונות, וביניהן בעיות NP-קשות. לרבות מהבעיות נמצאו אלגוריתמים פולינומיאליים, אך לא נמצא אלגוריתם פולינומיאלי לאף

בעיה NP-קשה. לפיכך, נראה כי בעיות NP-קשות אינן ניתנות לפתרון מעשי עבור קלטים גדולים. ישנן שתי גישות עיקריות לטיפול בבעיות כאלה:

1. מציאת פתרון מדויק, בסיבוכיות סבירה, **בחלק מהמקרים**. כאן מדובר באלגוריתמים שזמן הריצה שלהם אינו בהכרח פולינומיאלי, אבל לעיתים הם מוכיחים את עצמם בפתרון בעיות מעשיות מסוימות. מבחינה תאורטית, זמן הריצה של אלגוריתם כזה אינו פולינומיאלי: יש מקרים שבהם האלגוריתם מייצר פתרון אופטימלי בזמן סביר, אבל איננו יודעים מראש אם האלגוריתם ירוץ בזמן סביר ויחזיר פתרון משביע רצון, או שהוא ירוץ זמן רב בלי להחזיר פתרון. טיבו של אלגוריתם כזה נמדד בניסויים, אך אינו נתמך במדד תאורטי.
2. מציאת **פתרון מקורב בזמן פולינומיאלי**. המדד המקובל לטיב הקירוב שמניב אלגוריתם הוא המנה בין ערך הפתרון שהאלגוריתם מניב לבין הערך האופטימלי, עבור הקלט הגרוע ביותר (דהיינו, הקלט שעבורו מתקבלת המנה הגרועה ביותר). אלגוריתם כזה נקרא **אלגוריתם קירוב**. הניסיון מראה כי בהרבה מקרים אלגוריתמים כאלה מייצרים פתרון אופטימלי, או פתרון קרוב מאוד לאופטימלי.

הדיון בסעיף זה מתרכז בגישה השנייה. יהי A אלגוריתם פולינומיאלי לבעיה Π. עבור קלט I, נסמן ב-$\mathrm{OPT}(I)$ את הערך של הפתרון האופטימלי לבעיה, וב-$\mathrm{A}(I)$ את ערך הפתרון שמייצר האלגוריתם. אנו אומרים כי לאלגוריתם A יש **יחס קירוב** ρ, או שהוא ρ**-מקרב**, אם לכל קלט I שעבורו יש לבעיה פתרון מתקיים:

$\frac{\mathrm{A}(I)}{\mathrm{OPT}(I)} \le \rho$ אם Π היא בעיית מינימיזציה (כאשר $\rho \ge 1$);

$\frac{\mathrm{OPT}(I)}{\mathrm{A}(I)} \le \rho$ אם Π היא בעיית מקסימיזציה (כאשר $\rho \ge 1$).

(בספר נעשה שימוש במונח k-optimal. אנו מעדיפים להשתמש במונח *מקרב* במקום *אופטימלי*).

צעד מרכזי בתכנון אלגוריתם קירוב לבעיית אופטימיזציה Π הוא מציאת חסמים "טובים" עבור ערך האופטימום $\mathrm{OPT}(I)$ ועבור הערך המושג על ידי אלגוריתם הקירוב $\mathrm{A}(I)$. עלינו להוכיח את הטענה שערך האופטימום $\mathrm{OPT}(I)$ "לא יכול להיות טוב מדי" מחד, ומאידך שהערך המושג על ידי אלגוריתם הקירוב עבור הבעיה הנידונה "לא יכול להיות גרוע מדי", על מנת להסיק עד כמה הפתרון המתקבל על ידי אלגוריתם הקירוב "גרוע יותר" מפתרון אופטימלי. בבעיית מינימיזציה, למשל, עלינו להראות ש-$\mathrm{OPT}(I) \ge m$ וש-$\mathrm{A}(I) \le M$ עבור m ו-M כלשהם התלויים בנתוני הבעיה. במצב כזה,

$$\frac{\mathrm{A}(I)}{\mathrm{OPT}(I)} \le \rho := \frac{M}{m}.$$

בעיית הסוכן הנוסע

עבור הבעיה הכללית של הסוכן הנוסע, שהוצגה לעיל, לא קיים אלגוריתם פולינומיאלי המבטיח יחס קירוב כלשהו, אלא אם כן P=NP. על מנת להוכיח טענה זו, נראה שאילו היה קיים אלגוריתם קירוב כזה, אפילו בעל יחס קירוב גדול מאוד, היינו יכולים להכריע בזמן פולינומיאלי את בעיית *UHAMCIRCUIT* - הבעיה שבה יש להכריע האם גרף לא מכוון נתון מכיל מעגל המילטוני. בעיה זו ידועה כבעיה NP-שלמה.

בהינתן קלט $G'=(V,E')$ לבעיית המעגל ההמילטוני, נגדיר קלט $G=(V,E)$ לבעיית ה-TSP על-ידי כך שנשלים את G' לגרף מלא בעל מחירי קשתות כדלקמן:

$$c(e)=\begin{cases}1 & e\in E'\\ rn & e\in E\setminus E'\end{cases}$$

כאשר $n=|V|$ ו-$r>1$ קבוע כלשהו.

לצורך הדיון, נבחין בין שני מקרים:

- אם יש ב-G' מעגל המילטוני, אזי ערך הפתרון האופטימלי ל-TSP ב-G הוא בדיוק n. זאת משום שכל פתרון לבעיית ה-TSP ב-G מכיל בדיוק n קשתות, ומשקל כל אחת מהן הוא לפחות 1 (ומכאן שכל פתרון לבעיית ה-TSP ב-G הוא בעל ערך n לכל הפחות); מצד שני, כל מעגל המילטוני ב-G' הוא גם מעגל המילטוני ב-G שמשקלו בדיוק n. על כן, מעגל זה הוא פתרון אופטימלי לבעיית ה-TSP ב-G וערכו הוא בדיוק n.
- אם אין ב-G' מעגל המילטוני, אז כל פתרון לבעיית ה-TSP ב-G חייב לכלול לפחות קשת אחת $e\in E\setminus E'$ שמחירה rn. לכן, ערך הפתרון האופטימלי לבעיית ה-TSP ב-G הוא לפחות $rn+n-1$.

כעת נניח שקיים אלגוריתם קירוב בעל יחס קירוב $\rho>1$ כלשהו. נפעיל אלגוריתם זה על הגרף $G=(V,E)$ שהוגדר לעיל, כאשר $r>\rho$. אז אם יש ב-G' מעגל המילטוני, הרי שערך הפתרון האופטימלי ל-TSP ב-G הוא בדיוק n, ולכן ערך הפתרון שיחזיר אלגוריתם הקירוב יהיה ρn לכל היותר. אם, לעומת זאת, אין ב-G' מעגל המילטוני, אז ערך הפתרון שיחזיר אלגוריתם הקירוב יהיה לכל הפחות $rn+n-1$ וערך זה גדול ממש מ-ρn משום שבחרנו $r>\rho$.

לפיכך, אלגוריתם קירוב פולינומיאלי כזה יכול לקבל כקלט גרף $G'=(V,E')$ ולהכריע אם יש בו מעגל המילטוני: אם ערך הפתרון המקורב לבעיית ה-TSP ב-G קטן או שווה ל-ρn, הרי שקיים ב-G' מעגל המילטוני, אחרת – לא קיים מעגל כזה.

מסקנה: לא קיים ל-TSP אלגוריתם קירוב בעל יחס קירוב ρ כלשהו, אלא אם כן P=NP.

לאור האמור לעיל, נצמצם את הדיון בבעיית הסוכן הנוסע למקרה המיוחד הקרוי **בעיית הסוכן הנוסע המטרית**, שבו מחירי הקשתות מקיימים את אי-שוויון המשולש. כלומר, לכל זוג קדקודים $u, v \in V$ קיים מחיר $c(u,v) \geq 0$ המציין את מחיר הקשת המחברת אותם (כל שני קדקודים מחוברים בקשת) ולכל $u, v, w \in V$ מתקיים $c(u,w) \leq c(u,v) + c(v,w)$. (במקרה זה, מהווה הפונקציה $c(\cdot,\cdot)$ **מטריקה** על קבוצת הקדקודים V.)

הנחה זו מתקיימת בהרבה יישומים מעשיים. היישום הטבעי ביותר לבעיית הסוכן הנוסע, המהווה גם את המקור לשם הבעיה, הוא כדלקמן: נתונה רשימת ערים (המיוצגות על ידי קדקודי הגרף) והמרחקים (או מחירי המעבר) ביניהם. סוכן נוסע מעוניין לבקר בכל הערים, בכל עיר בדיוק פעם אחת, ולחזור לעיר שממנה יצא, כך שהמרחק הכולל (או העלות הכוללת של הנסיעה) יהיה מינימלי.

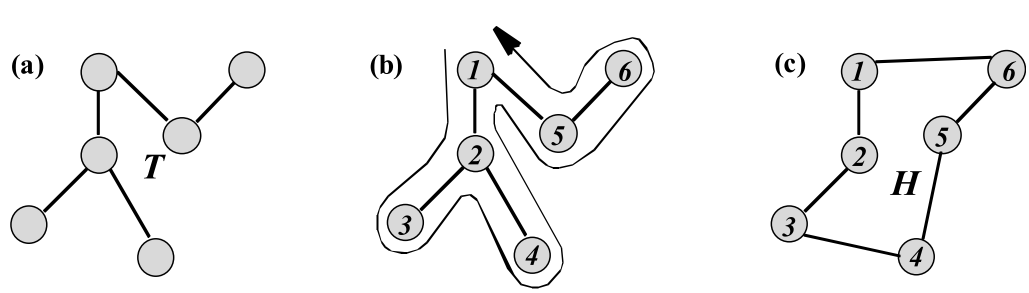
לבעיה המטרית של הסוכן הנוסע יש אלגוריתם פשוט מאוד בעל יחס קירוב 2 שנתאר אותו כאן. האלגוריתם מסתמך על הטענה הפשוטה הבאה:

טענה:

יהי T עץ פורש כלשהו בגרף לא מכוון ומלא $G=(V,E)$, עם מחירים אי-שליליים על הקשתות המקיימים את אי-שוויון המשולש. אז קיים אלגוריתם פולינומיאלי המוצא מעגל המילטוני H ב-G כך שמתקיים: $c(H) \leq 2c(T)$, כאשר $c(H)$ מציין את סכום מחירי הקשתות ב-H ואילו $c(T)$ מציין את סכום מחירי הקשתות ב-T.

הוכחה:

נקבע קדקוד כלשהו של T כשורש העץ, ונבצע בעץ סריקה תחילית (ראו איור 3, (a) ו-(b)). כזכור, סריקה תחילית מבקרת באופן רקורסיבי בכל קדקוד בעץ, ומוסיפה לרשימה כל קדקוד כאשר מבקרים בו, עוד לפני שנערך ביקור בצאצאיו. אנו טוענים כי הסדר שקובעת הסריקה התחילית מגדיר מעגל המילטוני H כנדרש, באופן המתואר להלן:



איור 3: (a) עץ פורש. (b) סריקה תחילית בעץ. (c) המעגל ההמילטוני המתאים לסריקה.

הסריקה התחילית עוברת על כל קשת בדיוק פעמיים ומגדירה סדר על הקדקודים. תחילה נבנה מעגל שאינו פשוט המכיל כל קשת של T בדיוק פעמיים. בדוגמה שבאיור 3(b), המעגל יהיה כדלקמן:

$$(1,2)\,(2,3)\,(3,2)\,(2,4)\,(4,2)\,(2,1)\,(1,5)\,(5,6)\,(6,5)\,(5,1)$$

מחירו של מעגל זה הוא $2c(T)$. לאחר מכן, לכל $1 \le i \le n-1$, כאשר n הוא מספר הקדקודים, נחליף את המסלול המקשר את הקדקודים i ו-$i+1$ במעגל זה, בקשת $(i,i+1)$, ואילו את המסלול המקשר את הקדקודים n ו-1 נחליף בקשת $(n,1)$. למשל, המסלול $(3,2)\,(2,4)$ באיור 3 (b) מוחלף בקשת $(3,4)$ באיור 3 (c), והמסלול $(4,2)\,(2,1)\,(1,5)$ מוחלף בקשת $(4,5)$. בגלל אי-שוויון המשולש, פעולה זו של החלפת מסלולים בקשתות אינה מעלה את המחיר. לפיכך, עלות המעגל ההמילטוני המתקבל היא לכל היותר $2c(T)$.

מכאן נובע באופן מיידי אלגוריתם שיחס הקירוב שלו 2 לבעיית הסוכן הנוסע (המטרית). תחילה, מוצאים עץ פורש מינימלי T ב- G. לאחר מכן, מוצאים מעגל המילטוני H ב- G כך שמתקיים $c(H) \le 2c(T)$, כמתואר בטענה לעיל. יהי H^* מעגל המילטוני אופטימלי. כיוון שכל מעגל המילטוני, ובפרט H^*, מכיל עץ פורש (שהוא מסלול), וכיוון ש-T הוא עץ פורש מינימלי, אזי $c(T) \le c(H^*)$. לפיכך, $c(H) \le 2c(T) \le 2c(H^*)$, מה שמוכיח שיחס הקירוב של אלגוריתם זה הוא 2.

הערות:

1. קיימת גרסה "מוחלשת" של בעיית הסוכן הנוסע שבה מחלישים את הדרישה "עוברים בכל קדקוד **בדיוק** פעם אחת" לדרישה "עוברים בכל קדקוד **לפחות** פעם אחת". בבעיה זו, מחפשים מעגל בעל אורך מינימלי העובר בכל הקדקודים, אבל המעגל איננו חייב להיות פשוט. גרסה מוחלשת זו ניתנת לרדוקציה לגרסה המטרית של הבעיה על ידי תהליך הקרוי **השלמה מטרית**: לכל $u,v \in V$ נגדיר $c'(u,v)$ כמחיר הזול ביותר של מסלול בין u ל- v. לא קשה לראות כי המחירים החדשים c' מקיימים את אי-שוויון המשולש, וכי כל פתרון עם המחירים c' ניתן להמרה בזמן פולינומיאלי לפתרון בעל אותו מחיר עבור הגרסה "המוחלשת".
2. לבעיית הסוכן הנוסע המטרית ידוע אלגוריתם מסובך יותר שיחס הקירוב שלו 1.5 (אלגוריתם שגילה Christofides בשנת 1976). זהו יחס הקירוב הטוב ביותר שידוע כיום לבעיה זו.
3. לבעיית עץ שטיינר המינימלי קיים אלגוריתם שיחס הקירוב שלו 2 הדומה לאלגוריתם בעל אותו יחס קירוב לבעיית הסוכן הנוסע שתואר לעיל. אלגוריתם הקירוב הטוב ביותר הידוע כיום לבעיית עץ שטיינר המינימלי הוא בעל יחס קירוב $1+\frac{1}{2}\ln 3 \approx 1.55$. אלגוריתם זה (שהתגלה בשנת 2000 על-ידי החוקרים G. Robins, A. Zelikovsky), הוא מורכב למדי.

במסגרת הדיון הנוכחי באלגוריתמי קירוב נסתפק בתיאור אלגוריתמים פשוטים בלבד. הקורס **אלגוריתמי קירוב** עוסק באלגוריתמים מורכבים יותר.

בעיית אריזה בקופסאות זהות

כעת נתאר דוגמה נוספת של אלגוריתם קירוב פשוט יחסית. זהו אלגוריתם קירוב חמדני לבעיה הבאה:

אריזה בקופסאות זהות (0/1-Bin Packing)

- <u>קלט:</u> קבוצה של n מספרים $A=\{a_1,\dots,a_n\}$, כאשר $0\le a_i\le 1$ לכל $1\le i\le n$.
- <u>פלט:</u> חלוקה של A ל- k תת-קבוצות זרות, $A=\bigcup_{j=1}^{k}A_j$, כך שלכל $1\le j\le k$ מתקיים $\sum\{a:a\in A_j\}\le 1$, ו- k מינימלי.

השם "אריזה בקופסאות זהות" נובע מהפירוש האפשרי הבא של הבעיה. נתון אוסף פריטים ועלינו לארוז את הפריטים בקופסאות. לכל הקופסאות אותו נפח, וקיים מלאי בלתי מוגבל של קופסאות. כל פריט תופס איזשהו חלק מנפח הקופסה, והמספר $0<a_i\le 1$ מייצג את חלק הקופסה שתופס הפריט ה-i. ניתן לארוז קבוצת פריטים B בקופסה נתונה רק אם נפחם המצטבר אינו עולה על נפח הקופסה. כלומר, אם $\sum\{a_i : a_i\in B\}\le 1$. הדרישה להביא את k למינימום פירושה שאנו מנסים לארוז את הפריטים במספר קטן ככל האפשר של קופסאות.
להלן אלגוריתם קירוב פשוט לבעיה שיחס הקירוב שלו 2:

אלגוריתם חמדני לבעיית האריזה בקופסאות

1. *פתח קופסה ריקה חדשה*
2. *עבור* $i=1,\dots,n$ *בצע:*
   a. *אם בקופסה האחרונה שנפתחה יש מקום לאיבר* a_i*, הכנס אותו אליה.*
   b. *אחרת, סגור קופסה זו, פתח קופסה חדשה והכנס לתוכה את* a_i.

נדגים את האלגוריתם על הקבוצה הבאה: $A=\{\frac{1}{10},\frac{1}{10},\frac{9}{10},\frac{9}{10}\}$.
בשני השלבים הראשונים נקבל: $A_1=\{\frac{1}{10},\frac{1}{10}\}$. כלומר, שני האיברים הראשונים ייארזו באותה קופסה A_1, שתיסגר מיד לאחר מכן (בניצולת של 20% בלבד מנפחה). לאחר מכן נצטרך לפתוח שתי קבוצות נוספות: $A_2=\{\frac{9}{10}\}$ ו- $A_3=\{\frac{9}{10}\}$. עם זאת, ברור כי הפתרון האופטימלי מורכב משתי קבוצות בלבד: $\{\frac{9}{10},\frac{1}{10}\}$, $\{\frac{9}{10},\frac{1}{10}\}$.

שימו לב שאילו הקלט היה מסודר באופן אחר, $A=\{\frac{1}{10},\frac{9}{10},\frac{1}{10},\frac{9}{10}\}$, דהיינו, אילו הפריטים היו מוצגים לאלגוריתם בסדר המתואר לעיל (כלומר "קטן, גדול, קטן, גדול" במקום "קטן, קטן, גדול, גדול"), אז אלגוריתם הקירוב שלנו היה מניב אריזה אופטימלית.

בתרגיל 7.1 שלהלן נראה שיחס הקירוב של האלגוריתם הזה הוא 2. נציין שישנם אלגוריתמי קירוב טובים יותר לבעיה, אך שוב, עקב מורכבותם, לא נתאר אותם כאן.

תרגיל 7.1

א. יהי $t=\sum_{a\in A} a$. הראו ש- $\mathrm{OPT}(I)\geq\lceil t\rceil$.

ב. הסיקו מכך כי יחס הקירוב של האלגוריתם החמדני שלנו הוא 2 לכל היותר.

ג. הראו כי לא קיים קבוע $\rho<2$ כך שלאלגוריתם הזה יש יחס קירוב ρ . כלומר, יחס הקירוב 2 הוא הדוק עבור האלגוריתם.

7.2 אלגוריתמים הסתברותיים

קראו את המבוא לסעיף 10.2 בספר הלימוד ואת התת-סעיף "המחלקה BPP"

עד עתה היה המודל החישובי שלנו מכונת טיורינג **דטרמיניסטית**. זהו מודל מוגבל מאוד כפי שכבר העלה דיוננו עד עתה: ישנן בעיות טבעיות רבות שאינן ניתנות לחישוב או שלא ידוע עבורן אלגוריתם פתרון יעיל על מכונות כאלה. ישנם מודלים אחרים לחישוב השונים ממודל מכונת טיורינג דטרמיניסטית. אחד מהם כבר פגשנו - מכונת טיורינג אי-דטרמיניסטית. זהו מודל חישובי המסתמן כיעיל יותר, אך הוא אינו מעשי.

כעת, נדון במודל אחר – מכונת טיורינג עם גישה לביטים אקראיים: כלומר, מכונה שבמהלך ריצתה יכולה לייצר ביטים אקראיים ולעשות בהם שימוש. נשאלת השאלה, האם יכולת זו עשויה לסייע בהשגת פתרונות יעילים יותר לבעיות חישוביות. קיימות דוגמאות שבהן שימוש באקראיות בחישוב מאפשר לקבל אלגוריתמים יעילים יותר מאלו הידועים לנו על מכונות דטרמיניסטיות.
נדגיש כי השאלה האם קיימת אקראיות בטבע היא שאלה פתוחה ומורכבת החורגת מתחום הדיון הנוכחי. למחשבים כיום יש **מחוללי מספרים אקראיים** (Random number generators) היוצרים מספר "אקראי"[1]. השם הזה מטעה מעט מכיוון שמחוללים אלה מייצרים למעשה מספרים הנראים כמספרים אקראיים, על אף שאינם כאלה. (מספרים כאלה נקראים מספרים **פסאודו-אקראיים** והגדרתם המתמטית המדויקת מורכבת ולכן לא נציגה כאן.)

[1] סביר שבמהלך ניסיונכם בתכנות השתמשתם במחוללים אלו לפחות פעם אחת, על ידי קריאה לפקודות כמו random.

בהגדרה 10.3 מוגדרת **מכונת טיורינג הסתברותית**. מכונה זו היא מכונת טיורינג אי-דטרמיניסטית, שבה לכל מצב q ותו סרט a מתקיים אחד משני המקרים הבאים:

1) $|\delta(q,a)|=1$; כלומר התנהגות המכונה במקרה זה היא דטרמיניסטית – למכונה יש אפשרות יחידה להמשך החישוב.

2) $|\delta(q,a)|=2$; במקרה זה יש למכונה שתי אפשרויות להמשיך את החישוב והיא נדרשת לבצע בחירה אי-דטרמיניסטית בין שתיהן, על ידי **הטלת מטבע**. במקרה זה בוחרת המכונה את אחת משתי האפשרויות בהסתברות שווה של $\frac{1}{2}$ לכל אפשרות.

מההגדרה ברור כי ההסתברות שייבחר מסלול חישוב נתון המכיל k בחירות אי-דטרמיניסטיות היא 2^{-k}: בכל בחירה אי-דטרמיניסטית כזו ההסתברות שנבחר את הבחירה המותירה אותנו במסלול החישוב הנתון היא $\frac{1}{2}$. מכיוון שהבחירות מתבצעות באופן בלתי תלוי, אזי ההסתברות שייבחר מסלול החישוב הנתון היא מכפלת ההסתברויות לבחור בכל קדקוד בחירה את האפשרות הנכונה, כלומר 2^{-k}.

כעת אנחנו יכולים להגדיר את ההסתברות כי מכונת טיורינג הסתברותית תקבל קלט w: הסתברות זו שווה לסכום ההסתברויות שנבחר מסלול מקבל. כלומר, זוהי ההסתברות שהמכונה שלנו תעצור בקונפיגורציה מקבלת בריצתה על w.

כמו בפרקים הקודמים, גם כאן אנו עוסקים בבעיות הכרעה. כלומר, עלינו להכריע אם מחרוזת נתונה w שייכת לשפה L או לא. אנו אומרים כי מכונת טיורינג הסתברותית M מכריעה שפה L עם הסתברות שגיאה הקטנה מ- ε עבור $0 \le \varepsilon < \frac{1}{2}$, אם לכל מחרוזת x מתקיים:

1. אם $x \in L$, ההסתברות כי M עוצרת בקונפיגורציה מקבלת בריצתה על x היא לפחות $1-\varepsilon$.
2. אם $x \notin L$, ההסתברות כי M עוצרת בקונפיגורציה דוחה בריצתה על x היא לפחות $1-\varepsilon$.

במילים אחרות, M מחזירה בהסתברות גבוהה, $1-\varepsilon$, את התשובה הנכונה **לכל** מחרוזת אפשרית.

בניגוד למכונה דטרמיניסטית, שריצתה על קלט מסוים קבועה וכל הרצה חוזרת שלה על אותו קלט תשחזר באופן מדויק את ריצתה הקודמת על אותו קלט, מכונה הסתברותית עשויה לתת תוצאות שונות בריצות שונות שלה על אותו קלט ממש. למשל, אם מכונה M מכריעה שפה L עם הסתברות לשגיאה $\varepsilon = 0.1$, אז כל קלט מהשפה יתקבל לפחות ב- 90% מהריצות של המכונה M עליו, וכל קלט שאיננו מהשפה יידחה לפחות ב- 90% מהריצות של המכונה M עליו.

שימו לב: שגיאה נפוצה היא לחשוב ש- M תיתן תשובה נכונה עבור 90% מהקלטים. אין זה נכון, משום שההסתברות מחושבת על הבחירות האקראיות שהמכונה מבצעת, ולא על בחירת הקלט.

חשוב להדגיש כי אקראיות אינה יכולה לסייע בפתרון בעיות הכרעה שאינן כריעות, למשל בפתרון בעיית העצירה. זאת מכיוון שכל מכונת טיורינג M, שזמן ריצתה סופי, משתמשת במספר סופי של ביטים אקראיים, נאמר t. אנו יכולים לדמות את ריצתה של M על קלט נתון עם כל 2^t הבחירות האפשריות של ערכי t הביטים האקראיים ולקבוע באופן דטרמיניסטי מה ההסתברות שהקלט הנדון מתקבל או נדחה על ידי M. לכן, מכונת טיורינג הסתברותית אינה יכולה להכריע בעיות שאינן כריעות. השימוש העיקרי של אקראיות הוא בהשגת זמן ריצה *מהיר* יותר בפתרונן של בעיות הכרעה כריעות.

זמן הריצה של מכונה הסתברותית M על קלט x מוגדר כאורך המסלול הארוך ביותר של M בריצתה על x. המחלקה BPP מוגדרת (הגדרה 10.4) כמשפחת השפות שעבורן קיימת מכונת טיורינג הסתברותית המכריעה אותן בזמן ריצה פולינומיאלי עם הסתברות לשגיאה של $\frac{1}{3}$ לכל היותר.

תרגיל 7.2

פתרו את תרגיל 10.7 בספר הלימוד.

אתם עשויים לתהות מדוע נבחר הקבוע $\frac{1}{3}$ ולא קבוע אחר. בפרט, נראה כי הסתברות לשגיאה בשיעור $\frac{1}{3}$ היא גבוהה למדי. למה 10.5 עונה על תהיות אלו. בלמה זו מראים כי כל הסתברות לשגיאה שהיא בין 0 ל- $\frac{1}{2}$ פירושה כי אנו יכולים להשיג שגיאה שהיא קטנה באופן מעריכי בגודל הקלט. במילים אחרות: אם קיימת מכונת טיורינג הסתברותית M_1, שזמן ריצתה פולינומיאלי, המכריעה שפה נתונה עם הסתברות לשגיאה ε לכל היותר, כאשר $0 \le \varepsilon < \frac{1}{2}$, אזי קיימת מכונת טיורינג הסתברותית M_2 המכריעה את אותה שפה, שזמן ריצתה פולינומיאלי ושההסתברות שלה לשגיאה היא $2^{-\text{poly}(n)}$ (כאשר $\text{poly}(n)$ הוא פולינום ב- n ו- n הוא אורך הקלט).

הרעיון מאחורי הוכחת למה 10.5 הוא כדלקמן: לכל קלט y, M_2 תריץ את M_1 על y "מספר גדול" של ריצות בלתי תלויות. כלומר, בכל ריצה של M_1 על y, היא תבצע הטלות מטבע בלתי תלויות בהטלות המטבע שביצעה בריצותיה הקודמות. כעת, תחליט M_2 אם לקבל או לדחות את הקלט y על סמך התשובה שהתקבלה יותר פעמים בריצות אלו. כלומר, M_2 תקבל את הקלט y

אם רוב (יותר ממחצית) ההרצות של M_1 הסתיימו בקבלת הקלט, והיא תדחה את הקלט – אחרת.

ולהוכחה עצמה: בוחרים שלם k , שערכו יוגדר מאוחר יותר, ומריצים את M_1 באופן בלתי תלוי $2k$ פעמים על הקלט w. מחליטים אם לקבל או לדחות את w בהתאם לתוצאת רוב הריצות כפי שהוסבר קודם (ההתנהגות במצב של "תיקו" איננה משנה כיוון שהסבירות לכך נמוכה מאוד). תהי S סדרה כלשהי באורך $2k$ של "כן" ו"לא" המתארת את תוצאות $2k$ ההרצות של M_1 על הקלט w. נניח כי c הוא מספר התשובות הנכונות ב-S ו-w הוא מספר התשובות השגויות ב-S (שימו לב, אין קשר בין ה-w שהוגדר כעת לבין הקלט המסומן בספר אף הוא ב-w). ברור כי $c+w=2k$. אנו אומרים כי S *רעה* אם $c \le w$. במקרה זה M_2 תחזיר תשובה שגויה. כעת אנו שואלים מה ההסתברות ש-M_2 תחזיר תשובה שגויה? (או לחלופין, מה ההסתברות לקבל סדרה רעה?)

נסמן את ההסתברות ש-M_1 תטעה בריצתה על הקלט w ב-ε_w. על פי הנחתנו, $\varepsilon_w \le \varepsilon$. לכן, מכיוון ש-$0 \le \varepsilon < \frac{1}{2}$, אז גם $0 \le \varepsilon_w < \frac{1}{2}$. ללא הגבלת הכלליות נניח ש-$\varepsilon_w = \varepsilon$ ונמשיך בחישובנו עם הסימון ε במקום ε_w. אם כן, ההסתברות לקבלת הסדרה S שבה c תשובות נכונות ו-w תשובות שגויות היא $p_S = \varepsilon^w(1-\varepsilon)^c$. אם הסדרה רעה, אז $w=k+t$ ו-$c=k-t$ עבור t חיובי כלשהו (למשל, אם $k=100$, כלומר, ערכנו 200 ריצות, ו-$w=110$ ואילו $c=90$ אז $t=10$). לכן:

$$p_S = \varepsilon^w(1-\varepsilon)^c = \varepsilon^k(1-\varepsilon)^k \cdot \left(\frac{\varepsilon}{1-\varepsilon}\right)^t < \varepsilon^k(1-\varepsilon)^k$$

כאשר האי-שוויון האחרון נובע מכך ש-$\varepsilon < \frac{1}{2}$ ולכן $\frac{\varepsilon}{1-\varepsilon} < 1$.

מספר הסדרות הרעות הוא לכל היותר מספר הסדרות באורך $2k$, כלומר 2^{2k}. מכאן נובע שההסתברות לקבל סדרה רעה חסומה על ידי

$$2^{2k} \cdot \varepsilon^k(1-\varepsilon)^k = (4\varepsilon(1-\varepsilon))^k$$

מכיוון ש-$\varepsilon \cdot (1-\varepsilon) < \frac{1}{4}$ לכל $0 \le \varepsilon < \frac{1}{2}$, אז $4\varepsilon(1-\varepsilon) < 1$. לכן $(4\varepsilon(1-\varepsilon))^k$ היא הסתברות קטנה מעריכית ב-k. כעת, אם אנו רוצים לקבל חסם של 2^{-t}, עבור $t = \text{poly}(n)$, על הסתברות השגיאה, נבחר $k = \frac{t}{\alpha}$ עבור $\alpha = -\log_2(4\varepsilon(1-\varepsilon))$.

כזכור, הרצנו את M_1 $2k$ פעמים על הקלט w. $k = \frac{t}{\alpha}$, $t = poly(n)$ ו- α הוא גודל קבוע. לכן $2k$ הוא פולינום ב- n (n הוא האורך של w). כלומר, הרצנו את M_1 מספר פולינומיאלי של פעמים. לכן זמן הריצה של התהליך כולו פולינומיאלי.

תרגיל 7.3

מדוע דרשנו כי ההסתברות לשגיאה תהיה קטנה ממש מ- $\frac{1}{2}$?

קראו את התת-סעיף "ראשוניות" בספר הלימוד **ניתן לדלג על הוכחת למה 10.8**

בעיית ההכרעה הנדונה בתת-סעיף הנוכחי היא **בעיית הראשוניות**, $PRIMES$. בבעיה זו אנו מקבלים ייצוג בינארי של מספר טבעי a ומטרתנו היא להכריע האם מדובר במספר ראשוני. האלגוריתם הנאיבי הבודק את כל המחלקים האפשריים עד $\sqrt{a}$ הוא מעריכי באורך הקלט, שכן זמן הריצה שלו אינו קטן מ- $O(\sqrt{a})$ ואילו גודל הקלט הוא $O(\log a)$ (כזכור, $\sqrt{a} = 2^{(\log a)/2}$).

במשך זמן רב לא היה ידוע אם קיים אלגוריתם פולינומיאלי המכריע את ראשוניותו של מספר נתון. בשנות השבעים והשמונים של המאה הקודמת הוכיחו מילר ורבין כי בעיית ההכרעה הזו מצויה ב-BPP. כלומר, קיים אלגוריתם הסתברותי שזמן ריצתו פולינומיאלי המכריע את הראשוניות של מספר נתון, והסתברות השגיאה שלו קטנה מ- $\frac{1}{3}$. בשנת 2002 הוכיחו החוקרים Agrawal, Kayal & Saxena כי קיים אלגוריתם פולינומיאלי דטרמיניסטי הפותר את בעיית הראשוניות (כלומר, ש- $PRIMES \in \mathrm{P}$). למרות זאת, יש לאלגוריתם ההסתברותי חשיבות רבה והוא עדיין האלגוריתם שבו משתמשים בפועל. (ביישומים רבים של אבטחת מידע יש צורך להכריע בדבר הראשוניות של מספרים גדולים מאוד.) האלגוריתם ההסתברותי עדיף על האלגוריתם הדטרמיניסטי, משום שהוא פשוט יותר להבנה ולניתוח ומשום שהוא יעיל יותר. אנו רואים, אם כן, שאקראיות עשויה לסייע במתן פתרונות יעילים יותר לבעיות חישוב. המחיר שעלינו לשלם על יעילות זו הוא ההסתברות לשגיאה. ואולם, כפי שראינו, ניתן להוריד את ההסתברות לשגיאה עד כדי מספרים זניחים ביותר (2^{-1000} לדוגמה) בלי לשנות את סדר הגודל של זמן הריצה של האלגוריתם.

הערה:

בספר הלימוד מוגדרות הקבוצות הבאות: $\mathrm{Z}_p = \{0,1,\ldots,p-1\}$ ו- $\mathrm{Z}_p^+ = \{1,\ldots,p-1\}$. נציין שהסימון האחרון איננו הסימון המקובל – בספרים אחרים מסומנת קבוצה זו כך:

$Z_p^* = \{1, \ldots, p-1\}$. אך על מנת לשמור על עקביות, נאמץ במדריך זה את הסימונים של ספר הלימוד.

כיצד נבדוק ראשוניות? רעיון אחד למבחן ראשוניות מסתמך על **המשפט הקטן של פרמה** (משפט 10.6): אם p ראשוני, אז $a^{p-1} \equiv 1 \pmod{p}$ לכל $a \in Z_p^+ = \{1, 2, \ldots, p-1\}$. מצד שני, אם p איננו ראשוני, אז תמיד קיים $a \in Z_p^+$ שעבורו $a^{p-1} \not\equiv 1 \pmod{p}$.

תרגיל 7.4

הוכיחו את המשפט הקטן של פרמה.

הדרכה: הסתכלו בקבוצה $A = \{ax \mid x \in Z_p^+\}$, כאשר $a \in Z_p^+ = \{1, 2, \ldots, p-1\}$ והכפל מבוצע מודולו p. הראו שקבוצה זו שווה לקבוצה Z_p^+, ואז השוו את מכפלת כל האיברים בכל אחת משתי הקבוצות האלו.

תרגיל 7.5

הראו שאם p מספר פריק ו- a מחלק לא טריוויאלי של p (כלומר, $1 < a < p$), אז $a^{p-1} \not\equiv 1 \pmod{p}$.

אם p מספר פריק, אז $a \in Z_p^+$ המקיים $a^{p-1} \not\equiv 1 \pmod{p}$ נקרא **עד** (witness) לפריקותו של p, שכן, בשל המשפט הקטן של פרמה, מספר כזה מהווה הוכחה לכך ש- p איננו ראשוני. לעומת זאת, כל $a \in Z_p^+$ שעבורו $a^{p-1} \equiv 1 \pmod{p}$ נקרא **לא-עד** (nonwitness) או **שקרן** (liar), שכן הוא "מרמז" על כך ש- p עלול להיות ראשוני.

האלגוריתם *PSEUDOPRIME* מתבסס על העובדה שאם p מספר פריק, אז יש עדים לפריקותו, ואילו אם p מספר ראשוני, אז לא קיימים עדים כאלו. השגרה בוחרת k מספרים אקראיים ב- Z_p^+ (כל מספר ב- Z_p^+ נבחר בהסתברות שווה של $\frac{1}{p-1}$). כל אחד מהמספרים שנבחרו מועלה בחזקת $p-1$ ומושווה ל-1 מודולו p. אם אחד ממספרים אלו מקיים $a^{p-1} \not\equiv 1 \pmod{p}$, אז הוא עד לפריקותו של p. במצב זה אנו דוחים את הקלט (כלומר, אנו קובעים שהמספר p איננו ראשוני). אחרת אנו מקבלים את הקלט. פעולות החזקה והמודולו ניתנות לביצוע בזמן פולינומיאלי ב- $\log p$ (כמו שראינו בפתרון תרגיל 4.5 במדריך זה), ולכן מדובר באלגוריתם שזמן ריצתו פולינומיאלי.

כעת, הבה נבחן את ההסתברות שאלגוריתם זה יחזיר תשובה שגויה. אם מצאנו עד לפריקותו של p, אז p איננו ראשוני, כפי שנובע מהמשפט הקטן של פרמה. מכאן שעבור קלטים p הנמצאים

בשפה (כלומר, עבור מספרים ראשוניים), ההסתברות ש-*PSEUDOPRIME* יחזיר תשובה שגויה היא 0. הבעיה היא שקיימים קלטים שאינם בשפה (כלומר, מספרים פריקים) שעבורם נקבל תשובה שגויה **בהסתברות גבוהה**. מצב זה קורה כאשר קיימים "הרבה" לא-עדים מודולו p ואז האלגוריתם *PSEUDOPRIME* מגריל בשלב הראשון k לא-עדים ללא שום עד שיצביע על האמת. למרבה הצער, קיימים אינסוף מספרים p שעבורם קיימים הרבה לא-עדים. מספרים אלו קרויים **מספרי Carmichael** והם בעלי התכונה הבאה: לכל מספר $a \in Z_p^+$ הזר ל-p מתקיים $a^{p-1} \equiv 1 \pmod p$. אפשר להראות (אם כי לא נעשה זאת כאן) כי עבור מספרי קרמייקל יש הרבה מספרים $a \in Z_p^+$ הזרים ל-p, ולכן האלגוריתם *PSEUDOPRIME* טועה בהסתברות גדולה מדי. מכיוון שאנו מעוניינים באלגוריתם שההסתברות שלו לשגיאה קטנה עבור **כל** קלט, האלגוריתם *PSEUDOPRIME* איננו עונה על דרישתנו.

אם כן, השימוש במשפט הקטן של פרמה לבדיקת ראשוניות אינו מספק את התוצאות שלהן קיווינו. ואולם מסתבר כי שימוש במשפט הקטן של פרמה בשילוב תכונה נוספת של מספרים ראשוניים נותן לנו את האלגוריתם המיוחל.

מספר שלם $s \in Z_p^+$ הוא **שורש ריבועי** של 1 (מודולו p), אם הוא מקיים $s^2 \equiv 1 \pmod p$. אם p **ראשוני**, אז השורשים הריבועיים **היחידים** של 1 הם 1 ו-1- (טענה זו מוכחת במסגרת הוכחת למה 10.7. -1 הוא $p-1$). שורשים אלה קרויים **שורשים טריוויאלים** של 1. אם, לעומת זאת, p אינו ראשוני, אינו זוגי, ואינו חזקה של ראשוני, אז יש ל-1 לפחות שורש ריבועי אחד לא טריוויאלי; כלומר, קיים $s \not\equiv \pm 1 \pmod p$ המקיים $s^2 \equiv 1 \pmod p$ (טענה זו תוכח בהמשך).

מצב זה מודגם בספר עבור $p = 21$. מודולו $p = 21$ יש ל-1 שני שורשים ריבועיים לא טריוויאלים: 8 ו-13 (שימו לב ש-$13 \equiv -8 \pmod{21}$). אכן,

$$8^2 = 64 = 21 \times 3 + 1 \equiv 1 \pmod{21}$$

ואילו

$$.13^2 = 169 = 8 \times 21 + 1 \equiv 1 \pmod{21}$$

נניח כעת ש-p מספר פריק אי-זוגי שעבר בהצלחה את *PSEUDOPRIME* (כלומר, p "הצליח להתחזות" כמספר ראשוני). שימו לב, שאנו רשאים לצמצם את דיוננו למספרים אי-זוגיים, כיוון שמספרים זוגיים (גדולים מ-2) הם בבירור פריקים ולכן אינם מעניינים אותנו בדיון זה. יהי a אחד המספרים שנבחרו על ידי *PSEUDOPRIME*. מכיוון ש-p עבר בהצלחה את *PSEUDOPRIME*, נובע כי

$$a^{p-1} \equiv 1 \pmod p$$

הרעיון הוא לנסות "לייצר" מ-a^{p-1} שורש ריבועי אי-טריוויאלי של 1. כיצד? מכיוון ש-p אי-זוגי, אז $p-1$ זוגי. לפיכך, אנו בוחנים את המספר $a^{(p-1)/2}$. אם $a^{(p-1)/2} \not\equiv \pm 1 \pmod p$, אזי

$a^{(p-1)/2}$ הוא שורש ריבועי אי-טריוויאלי של 1, ולפיכך p פריק. אם, לעומת זאת, $a^{(p-1)/2} \equiv 1 \pmod p$ ו- $(p-1)/2$ זוגי, אז אפשר לחזור על התהליך ולבדוק אם $a^{(p-1)/4} \not\equiv \pm 1 \pmod p$. אנו עורכים אפוא לולאת בדיקה כאשר בשלב ה- i בלולאה אנו בוחנים את המספר $a^{(p-1)/2^i} \pmod p$ ומשווים אותו ל- $\pm 1 \pmod p$. לולאה זו מסתיימת כאשר אנו מגיעים לשלב i שבו החזקה כבר איננה זוגית ולכן איננו יכולים להמשיך ולחלק אותה, או כאשר הגענו לתוצאה שאיננה שווה ל-1 מודולו p. לכן, אנו עוצרים באחד משלושת המקרים הבאים:

1. לאחר חזרה על התהליך i פעמים, הגענו למצב שבו $a^{(p-1)/2^i} \not\equiv \pm 1 \pmod p$, בעוד ש- $a^{(p-1)/2^{i-1}} \equiv 1 \pmod p$. מצאנו אפוא שורש ריבועי אי-טריוויאלי של 1 מודולו p.
2. לאחר חזרה על התהליך i פעמים, הגענו למצב שבו $a^{(p-1)/2^i} \equiv -1 \pmod p$.
3. לאחר חזרה על התהליך i פעמים, הגענו למצב שבו $a^{(p-1)/2^i} \equiv 1 \pmod p$ ו- $(p-1)/2^i$ איננו זוגי.

במקרה הראשון מצאנו עד לפריקותו של p. במקרים השני והשלישי לא הצלחנו למצוא עד כזה.

האלגוריתם $PRIME$ המתואר בראש עמוד 401 מיישם את הרעיון דלעיל. אנו מתארים כאן את האלגוריתם בצורה שונה במעט מזו המתוארת בספר; תיאור האלגוריתם שלהלן קרוב יותר ברוחו לאופן שבו נתכנת אלגוריתם זה.

האלגוריתם $PRIME(p)$:

1. אם $p = 2$, **קבל** את הקלט; אם p זוגי גדול מ- 2, **דחה** את הקלט; אחרת, המשך.
2. יהיו s,t מספרים כך ש- $p-1 = s \cdot t$, המספר s אי-זוגי ו- $t = 2^h$ עבור h טבעי כלשהו (שימו לב: h הוא מספר האפסים התחתונים בייצוג הבינארי של $p-1$ ואילו s הוא המספר הנותר מ- $p-1$ לאחר קיצוץ האפסים הללו. למשל, אם הייצוג הבינארי של $p-1$ הוא 1101000_2, אז $h = 3$, כלומר $t = 8$, ואילו $s = 1101_2 = 13$).
3. עבור $i = 1,\dots,k$ בצע:
   a. קרא לפונקציה $TEST(p)$.
   b. אם הפונקציה $TEST$ דחתה את הקלט, **דחה** את הקלט ועצור; אחרת, המשך בלולאת הבדיקה.
4. אם בכל k הפעמים לעיל קיבלה הפונקציה $TEST$ את הקלט p, **קבל** את הקלט ועצור.

הפונקציה $TEST(p)$:

1. בחר באופן אקראי $a \in Z_p^+$.
2. אם $a^{p-1} \not\equiv 1 \pmod p$, **דחה** את הקלט. אחרת, המשך.
3. אתחל $b = \dfrac{p-1}{2} = s \cdot 2^{h-1}$.

4. אם $a^b \not\equiv \pm1 \pmod p$, **דחה** את הקלט; אם $a^b \equiv -1 \pmod p$ **קבל** את הקלט; אחרת, כלומר אם $a^b \equiv 1 \pmod p$, המשך לשלב הבא.
5. אם b זוגי, חשב $b = \frac{b}{2}$ וחזור לשלב 4. אחרת, **קבל** את הקלט.

האלגוריתם $PRIME$ מריץ על הקלט שלו k בדיקות בלתי תלויות. כל בדיקה כזו, הממומשת על ידי הפונקציה $TEST$, בלתי תלויה בקודמותיה מכיוון שהיא מגרילה מספר אקראי $a \in Z_p^+$ ומשתמשת בו לביצוע הבדיקה.

אם הפונקציה $TEST$ דוחה את הקלט, כלומר – מצביעה עליו כפריק, אז תשובה זו נכונה בוודאות ולכן אין טעם להמשיך בבדיקות נוספות. אכן, $TEST$ דוחה את הקלט רק אם היא מצאה עד לפריקותו בשלב 2 או בשלב 4 בתיאור לעיל של הפונקציה $TEST$. (שימו לב: שלבים אלו מתאימים לשלבים 4 ו-7 בהתאמה בתיאור האלגוריתם בספר.) בלמה 10.7 אנו מוכיחים שאם p ראשוני, אז אין לו עדים בשלב 2 (מה שקרוי בספר "a stage 4 witness"), על פי המשפט הקטן של פרמה, וגם אין לו עדים בשלב 4 (מה שקרוי בספר "a stage 7 witness"), מכיוון שהשורשים היחידים של 1 מודולו p הם ±1. (נזכיר ש--1 הוא $p-1$).

לסיכום, אם הפונקציה $TEST$ דוחה את הקלט, כלומר – מצביעה עליו כפריק, אז תשובה זו נכונה בוודאות; או, במילים אחרות, פונקציית בדיקה זו לעולם איננה טועה בהינתן לה קלט מהשפה, דהיינו מספר ראשוני.

לעומת זאת, אם p פריק, פונקציית הבדיקה $TEST$ עלולה לטעות. בלמה 10.8 מוכיחים שהסתברות שלה לטעות במקרה זה היא לכל היותר $\frac{1}{2}$. זו הסיבה שהאלגוריתם $PRIME$ מריץ אותה k פעמים. כך, אם הקלט p הוא פריק, ההסתברות שהאלגוריתם $PRIME$ יקבל אותו (כלומר, יטעה) שווה להסתברות שפונקציית הבדיקה $TEST$ תיכשל בכל k הפעמים. הסתברות זו, על פי למה 10.8, אינה עולה על 2^{-k}. לכן, אנו יכולים לבחור את k על פי מידת האמינות שאנו מעוניינים להשיג: ככל ש-k גדול יותר, הסבירות לטעות קטנה ועל כן האמינות של תשובת האלגוריתם בדבר ראשוניותו של הקלט גדלה.

הוכחת למה 10.8:

יהי p מספר פריק אי-זוגי. עד (witness) הוא מספר $w \in Z_p^+$ שאם $TEST$ מגרילה אותו בשלב הראשון שלה אז היא תדחה את הקלט p. לא-עד (nonwitness) הוא כל מספר $w \in Z_p^+$ שמביא להכרעה שגויה של $TEST$, דהיינו, לקבלת הקלט p כמספר ראשוני. אם כן, על מנת להראות

שהסתברות הטעות של $TEST$ במקרה זה איננה עולה על $\frac{1}{2}$, מספיק להראות שמספר העדים גדול לפחות כמספר הלא-עדים.

יש שני סוגים של לא-עדים: אלו המביאים לקבלת הקלט בשלב 5 של הפונקציה $TEST$ (שאליהם נתייחס מעתה כאל לא-עדים מהסוג הראשון), ואלו המביאים לקבלת הקלט בשלב 4 של הפונקציה $TEST$ (לא-עדים מהסוג השני). במילים אחרות, אם $w \in Z_p^+$ הוא לא-עד מהסוג הראשון, אז כל הערכים המחושבים בשלב 4 בלולאת הבדיקה בפונקציה $TEST$ יוצאים 1; אם $w \in Z_p^+$ הוא לא-עד מהסוג השני, אז סדרת הערכים המחושבים בשלב 4 בלולאת הבדיקה מסתיימת ב--1. למשל, $w = 1$ הוא לא-עד מהסוג הראשון ואילו $w = p - 1$ הוא לא-עד מהסוג השני, כפי שמוסבר בהוכחת הלמה.

מכל קבוצת הלא-עדים מהסוג השני (שעל פי האמור לעיל איננה קבוצה ריקה), נבחר לא-עד כזה שעבורו מגיעה סדרת הערכים המחושבים בשלב 4 בלולאת הבדיקה ל--1 מוקדם ככל האפשר. נסמן לא-עד זה ב-h ונניח ש-$h^{s\cdot 2^j} \equiv -1 \pmod p$ הוא הערך שבו עצרנו בלולאת הבדיקה בשלב 4.

כעת, הדיון מתפצל לשני מקרים. המקרה שבו ניתן לפרק את p למכפלת שני מספרים קטנים ממנו הזרים זה לזה, $p = q \cdot r$, והמקרה שבו דבר זה איננו אפשרי, כלומר $p = q^e$ כאשר q ראשוני ו-$e > 1$.

מקרה א: $p = q \cdot r$ כאשר $q, r > 1$ ו-$\gcd(q, r) = 1$.

ראשית, הבה ניזכר במשפט החשוב הקרוי **משפט השאריות הסיני:**

נניח כי p ו-q הם שלמים הזרים זה לזה. אז לכל $a \in Z_p = \{0, 1, \ldots, p-1\}$ ו-$b \in Z_q = \{0, 1, \ldots, q-1\}$ קיים $r \in Z_{pq} = \{0, 1, \ldots, pq-1\}$ יחיד כך ש:

$$r \equiv a \pmod p \quad \text{וגם} \quad r \equiv b \pmod q$$

אם כן, לפי משפט השאריות הסיני, קיים מספר $t \in Z_p$ המקיים את שתי המשוואות:

$$t \equiv h \pmod q$$

$$t \equiv 1 \pmod r$$

מכאן נובע ש-

$$t^{s\cdot 2^j} \equiv -1 \pmod q$$

ו-

$$t^{s\cdot 2^j} \equiv 1 \pmod r$$

תרגיל 7.6
הוכיחו את שני השוויונים האחרונים.

מהשוויון מודולו q לעיל אנו מסיקים ש- $t^{s\cdot 2^j} \not\equiv 1 \pmod p$. מהשוויון מודולו r לעיל אנו מסיקים ש- $t^{s\cdot 2^j} \not\equiv -1 \pmod p$. מצד שני, $t^{s\cdot 2^{j+1}} \equiv 1 \pmod p$, כפי שנובע משני השוויונים הללו גם יחד. לכן t הוא עד לפריקותו של p.

בחרנו אפוא לא-עד מהסוג השני, h, ומצאנו בעזרתו עד t. נסמן את קבוצת הלא-עדים (משני הסוגים) ב-NW ואת קבוצת העדים ב-W. נגדיר העתקה $f: NW \to W$ על ידי $f(d) = dt$. בספר מראים כי זו אכן העתקה מ-NW ל-W ושהעתקה זו היא חד-חד-ערכית. מכאן נובע ש- $|NW| \le |W|$. לפיכך, זה מסיים את ההוכחה במקרה זה.

מקרה ב: $p = q^e$ כאשר q ראשוני ו-$e > 1$.

נסמן $t = 1 + q^{e-1}$. אנו טוענים ש- t הוא עד בשלב 2 של $TEST$ ("a stage 4 witness"). כלומר, $t^{p-1} \not\equiv 1 \pmod p$. דבר זה מוכח באופן הבא: נניח בשלילה ש- $t^{p-1} \equiv 1 \pmod p$. אז $t^p \equiv t \pmod p$. אבל, כפי שמוכח בספר, $t^p \equiv 1 \pmod p$; מצד שני $t \not\equiv 1 \pmod p$ (משום ש- $t - 1 = q^{e-1}$ וערך זה איננו אפס מודולו $p = q^e$). סתירה זו מוכיחה את מה שרצינו ולכן t הוא עד.
כמו במקרה הקודם, נגדיר העתקה $f: NW \to W$ על ידי $f(d) = dt$. כפי שמראים בספר, זו אכן העתקה מ-NW ל-W והיא אף חד-חד-ערכית. לכן $|NW| \le |W|$.
בכך הושלמה הוכחת למה 10.8.

משפט 10.9 מסכם את מה שנובע מדיוננו עד כה: $PRIMES \in \mathrm{BPP}$. למעשה, הוכחנו טענה חזקה יותר: הראינו אלגוריתם הסתברותי להכרעת השפה $PRIMES$ שיש לו טעות מסוג אחד בלבד: האלגוריתם עלול לטעון שמספר פריק הוא ראשוני, ואולם הוא לעולם לא יטען שמספר ראשוני הוא פריק. מחלקת השפות שיש להן אלגוריתם הסתברותי בעל זמן ריצה פולינומיאלי המקבל כל קלט בשפה בהסתברות $\frac{1}{2}$ לפחות, ודוחה כל קלט שאינו בשפה בהסתברות 1, נקראת RP. בפרט שפת המספרים הפריקים, $COMPOSITES$, שייכת ל-RP.

המחלקה coRP מורכבת מכל השפות שעבורן קיים אלגוריתם הסתברותי בעל זמן ריצה פולינומיאלי המקבל כל קלט בשפה בהסתברות 1, ודוחה כל קלט שאינו בשפה בהסתברות $\frac{1}{2}$ לפחות.

מהדיון לעיל נובע ששפת המספרים הראשוניים, $PRIMES$, שייכת ל-coRP.

תרגיל 7.7

הוכיחו ש:

א. $RP \subseteq BPP$

ב. $coRP \subseteq BPP$

ג. $RP \subseteq NP$

פתרון התרגילים

תרגיל 7.1

א. $\mathrm{OPT}(I)$ הוא מספר הקופסאות באריזה האופטימלית. זהו מספר שלם המייצג את הנפח הכולל של הקופסאות באריזה, כיוון שהנפח של כל קופסה הוא 1. מאחר שהקופסאות מכילות את כל הפריטים, הרי שנפחן הכולל חייב להיות לפחות כסכום נפחי כל הפריטים, דהיינו t. המספר השלם המינימלי הגדול מ- t הוא $\lceil t \rceil$.

ב. בהינתן "קופסה" B, נסמן ב- a $S(B) = \sum_{a \in B}$ את נפח כל הפריטים שארוזים בה. סכום הנפחים של הפריטים בכל שתי קופסאות עוקבות גדול מ-1. כלומר, לכל שתי קופסאות עוקבות A_i ו-A_{i+1}, $S(A_i) + S(A_{i+1}) > 1$. אחרת, היינו מכניסים את כל הפריטים של A_{i+1} ל- A_i.

כעת נראה שהאלגוריתם משתמש בלא יותר מ-$2\lceil t \rceil$ קופסאות. נניח בשלילה שהוא משתמש ביותר קופסאות. נחלק את הקופסאות לזוגות עוקבים. מכיוון שסכום הנפחים של כלל הפריטים הוא t ומספר הזוגות העוקבים גדול מ- t, חייב להיות זוג קופסאות עוקבות שסכום הנפחים בשתי הקופסאות קטן מ-1, בסתירה למה שהראינו. הואיל ובסעיף א הוכחנו כי $\mathrm{OPT(I)} \geq \lceil t \rceil$, יחס הקירוב של האלגוריתם שלנו הוא 2 לכל היותר.

אבל יתכן שעל ידי הוכחה זהירה ומדויקת יותר, נצליח להראות יחס קירוב טוב יותר עבור אלגוריתם זה. מסתבר שלא. בסעיף ג להלן אנו מראים שיחס הקירוב של אלגוריתם זה הוא 2, על ידי בניית קלט שעבורו מתקבל יחס קירוב של "כמעט" 2.

ג. נתבונן בסדרת הקלט $\frac{1}{2}, \varepsilon, \frac{1}{2}, \varepsilon, \ldots$. כלומר, הפריטים המוצגים לאלגוריתם הם בגודל $\frac{1}{2}$ ו- ε לסירוגין ($a_{2i-1} = \frac{1}{2}$ ו- $a_{2i} = \varepsilon$) כאשר $\varepsilon > 0$ הוא מספר קטן מאוד. אנו רואים ש- a_1, a_2 ייכנסו לאותה הקופסה, A_1, אך בשביל a_3 נצטרך לפתוח קופסה חדשה, A_2, שאליה ייכנס גם a_4. באופן כללי, אם מספר הפריטים הוא $n = 2k$, האלגוריתם ישתמש ב- $k = n/2$ קופסאות, שכל אחת מהן מכילה פריטים בנפח כולל של $\varepsilon + \frac{1}{2}$.

כמובן, היה יעיל יותר לרכז את כל הפריטים בגודל ε באותה הקופסה, אך זו בדיוק הבעיה באלגוריתם הפשטני שלנו: הוא אינו יכול לחזור ולתקן פעולות שהוא ביצע בעבר. ברגע שהוא ארז פריט בגודל ε עם פריט בגודל $\frac{1}{2}$, אין הוא יכול לחזור בו ולפרק את הקופסה הזו.

כעת נבדוק בכמה קופסאות ישתמש הפתרון האופטימלי? אם $\varepsilon \leq 1/k$, אז האלגוריתם האופטימלי ישתמש בקופסה אחת עבור כל הפריטים בגודל ε, ובעוד $\lceil k/2 \rceil$ קופסאות בשביל

הפריטים בגודל $\frac{1}{2}$. לכן $\text{OPT(I)}=\lceil k/2 \rceil+1$. לפיכך, יחס הקירוב המתקבל עבור קלטים מהצורה הזו הוא:

$$\frac{A(I)}{OPT(I)} = \frac{k}{\lceil k/2 \rceil + 1}$$

בפרט, כאשר k שואף לאינסוף, היחס לעיל שואף ל-2. לכן, המספר ρ המינימלי שעבורו מתקיים $\frac{\text{A}(I)}{\text{OPT}(I)} \le \rho$ לכל הקלטים האפשריים הוא $\rho = 2$.

תרגיל 7.2

פתרון התרגיל מופיע בספר הלימוד.

תרגיל 7.3

ראשית, אנו רואים שהוכחת למה 10.5 נכשלת אם $\varepsilon \ge \frac{1}{2}$. אם $\varepsilon = \frac{1}{2}$, אז $4\varepsilon(1-\varepsilon) = 1$ ולכן חסם השגיאה יוצא 1. אם $\varepsilon > \frac{1}{2}$, אז ההוכחה נכשלת עוד קודם מכיוון שאז $\frac{\varepsilon}{1-\varepsilon} > 1$.

כעת נמשיך וניתן הסבר ישיר מדוע במקרה ש-$\varepsilon = \frac{1}{2}$ אין אפשרות להגיע להסתברות שגיאה קטנה כרצוננו. נסתכל על מחלקת השפות L שעבורן קיימת מכונת טיורינג הסתברותית M כך שאם $w \in L$, אז M מקבלת את w בהסתברות $\frac{1}{2}$ ואם $w \notin L$, אז M דוחה את w בהסתברות $\frac{1}{2}$. אולם, לכל שפה קיימת מכונת טיורינג הסתברותית כזו: המכונה פשוט מטילה מטבע הוגן ומחליטה לקבל או לדחות את הקלט בהתאם לתוצאות הטלת המטבע; מכונה כזו **תמיד** מחזירה תשובה נכונה בהסתברות $\frac{1}{2}$.

תרגיל 7.4

נגדיר לכל $a \in Z_p^+ = \{1,2,\ldots,p-1\}$ פונקציה $f: Z_p^+ \to Z_p^+$ על ידי הנוסחה

$$f(x) = ax \bmod p.$$

אנו טוענים כי f חח"ע. אם לא, אז קיימים $m,n \in Z_p^+$ שונים (נניח למשל ש-$m > n$) שעבורם $am \equiv an \pmod p$. לפיכך, p מחלק את ההפרש בין שני מספרים אלו, כלומר את $a \cdot (m-n)$. אך אז, כיוון ש-p ראשוני, עליו לחלק לפחות את אחד משני הכופלים הללו. כיוון שגם a וגם $m-n$ הם מספרים הגדולים מאפס אך קטנים מ-p, דבר זה לא ייתכן. סתירה זו מוכיחה ש-$m = n$ ולכן f חח"ע.

נסתכל כעת בקבוצה $A = \{ax \mid x \in Z_p^+\}$. לאור האמור לעיל, היא מכילה $p-1$ איברים שונים ולכן היא זהה לקבוצה Z_p^+. לפיכך, מכפלת כל האיברים ב-A מודולו p שווה למכפלת כל האיברים ב-Z_p^+ מודולו p. מכיוון שהכפל מודולו p הוא חילופי,

$$a^{p-1}(p-1)! \equiv (p-1)! \pmod p$$

לכן $(a^{p-1}-1)\cdot(p-1)! \equiv 0 \pmod p$, כלומר $(a^{p-1}-1)\cdot(p-1)!$ מתחלק ב-p. אך מכיוון ש-p ראשוני, אין הוא מחלק את $(p-1)!$. מסקנה: p מחלק את $a^{p-1}-1$. מכאן ש-$a^{p-1}-1 \equiv 0 \pmod p$, כפי שנדרשנו להראות.

תרגיל 7.5

נראה שאם a מחלק של p וש-$a^{p-1} \equiv 1 \pmod p$, אז בהכרח $a = 1$. מתוך כך ש-$a^{p-1} \equiv 1 \pmod p$ נובע ש-$p \mid a^{p-1}-1$. a מחלק את p, ולכן $a \mid a^{p-1}-1$. מכיוון ש-$a \mid a^{p-1}$, מקבלים ש-a מחלק את 1. זה ייתכן רק אם $a = 1$.

תרגיל 7.6

השוויון השני (מודולו r) מיידי. לגבי השוויון הראשון, נתון לנו ש-$t \equiv h \pmod q$. לכן, $t = mq + h$ עבור שלם כלשהו m. לכן

$$t^{s\cdot 2^j} = nq + h^{s\cdot 2^j}$$

עבור שלם כלשהו n. לכן

$$t^{s\cdot 2^j} \equiv h^{s\cdot 2^j} \pmod q$$

אבל $h^{s\cdot 2^j} \equiv -1 \pmod p$, כלומר $h^{s\cdot 2^j}+1$ הוא כפולה שלמה של p. מכיוון ש-$p = q\cdot r$, אז $h^{s\cdot 2^j}+1$ הוא גם כפולה שלמה של q. לפיכך, $t^{s\cdot 2^j} \equiv h^{s\cdot 2^j} \equiv -1 \pmod q$.

תרגיל 7.7

א. נניח ש-$L \in \mathrm{RP}$ ותהי M מכונת טיורינג הסתברותית, שזמן ריצתה פולינומיאלי והיא בעלת התכונה שאם $x \in L$ אזי M מקבלת את x בהסתברות $\frac{1}{2}$ לפחות ואם $x \notin L$ אזי M דוחה את x בהסתברות 1.

נבנה מכונת טיורינג חדשה M', אשר בהינתן קלט x היא תריץ עליו באופן בלתי תלוי את M פעמיים. M' תדחה את הקלט רק אם M דחתה אותו פעמיים, ואחרת – היא תקבל את הקלט. אם $x \notin L$, אז M' תדחה אותו בהסתברות 1; כלומר, כאן היא אינה טועה לעולם. אם, לעומת זאת, $x \in L$, M' תטעה, כלומר תדחה את x, רק אם M דחתה אותו פעמיים. דבר זה יקרה

בהסתברות $\frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2}$ לכל היותר. בכל מקרה, ההסתברות לתשובה שגויה של M' קטנה מ- $\frac{1}{3}$.

מכיוון שזמן הריצה של M' הוא בבירור פולינומיאלי, אז $L \in \text{BPP}$.

ב. אותה הבנייה שתיארנו בסעיף הקודם מראה גם שאם $L \in \text{coRP}$, אז $L \in \text{BPP}$.

ג. נניח ש- $L \in \text{RP}$ ותהי M מכונת טיורינג הסתברותית שזמן ריצתה פולינומיאלי בעלת התכונה שאם $x \in L$ אזי M מקבלת את x בהסתברות $\frac{1}{2}$ לפחות, ואם $x \notin L$ אזי M דוחה את x בהסתברות 1. לפיכך, מסלול מקבל של M בריצתה על קלט x מהווה מסמך אישור להיותו של x בשפה; אורך מסמך האישור הוא לכל היותר פולינומיאלי באורך הקלט. מסלול כזה יהיה קיים תמיד עבור $x \in L$, שכן הסתברות הקבלה של קלט כזה היא לפחות $\frac{1}{2}$.

8. נספח – נושאים מתקדמים בתורת החישוביות (פרק 6 בספר הלימוד)

8.1 משפט הרקורסיה

דלגו על המבוא לסעיף 6.1 בספר הלימוד וקראו את התת-סעיף "התייחסות עצמית"

בתת-סעיף זה מתוארת היכולת של מכונות טיורינג לייצר עותקים של עצמן. התוצאה היסודית הראשונה היא קיומה של מכונת טיורינג שמדפיסה כפלט את התיאור שלה (תוך התעלמות מהקלט). התיאור של מכונה כזו עושה שימוש בלמה 6.1 שבה נאמר שקיימת פונקציה חשיבה $q:\Sigma^*\to\Sigma^*$, כך שלכל $w\in\Sigma^*$, $q(w)$ הוא התיאור של מכונת טיורינג P_w המדפיסה את w ועוצרת.

תרגיל 8.1

א. בהינתן מחרוזת $w=w_1\cdots w_n$, רשמו תיאור מפורט של מכונת טיורינג P_w שהפלט שלה, לכל קלט נתון, הוא w. ניתן לתאר מכונה יעילה יותר מזו המתוארת באופן סכמתי בהוכחת למה 6.1.

ב. תארו מכונת טיורינג המקבלת כקלט $w=w_1\cdots w_n$ ומוציאה כפלט את התיאור המפורט של P_w, שאותה תיארתם בסעיף הקודם. הנכם רשאים להשתמש במכונה בעלת כמה סרטים.

ג. אילו תווים נוספים, מלבד התווים ב-Σ, נחוצים לתיאור של P_w?

המכונה SELF:

כעת נתאר את המכונה SELF, שכל פעולתה היא הדפסת התיאור שלה עצמה. הבנייה של SELF מתעתעת במקצת וייתכן שחלק מכם התקשה בהבנת התיאור שלה.

המכונה SELF מורכבת משני מרכיבים, A ו-B. במילים אחרות, האלגוריתם שהמכונה SELF מממשת ניתן להפרדה לשני שלבים. תחילה פועל המרכיב המסומן A, ואחריו פועל המרכיב המסומן B. כאשר A יסיים לעבוד, יהיה על הסרט תיאור של המרכיב B. כלומר, יהיה תיאור של מכונת טיורינג שכולל בתוכו רק את המצבים ואת הפעולות הרלוונטיים ל-B. אחר כך, לאחר ש-B יסיים לעבוד, יתווסף לתיאור זה גם התיאור של A, למען השלמת התיאור של SELF כולה.

המרכיב A פועל כדלקמן: נניח שהיה לנו תיאור של המרכיב B, כלומר $\langle B\rangle$. אז A אמור להיות מכונת טיורינג שמדפיסה את $\langle B\rangle$. אבל זה ניתן למימוש ברוח תרגיל 8.1 לעיל. לכן, כל מה שנחוץ לנו על מנת לממש את A הוא לקבל תיאור של B. מיד תשתכנעו שזה דבר קל.

המרכיב B פועל לאחר המרכיב A. כלומר, המרכיב B מתחיל לרוץ כאשר הפלט של A כבר מופיע על הסרט והוא אמור לייצר תיאור של מרכיב A. אך גם זה ניתן למימוש ברוח תרגיל 8.1. המרכיב B יקרא את תוכן הסרט ועל ידי שימוש במכונה M מחלק ב׳ של תרגיל 8.1 ייצר תיאור של מכונת טיורינג שזה הפלט שלה. לאחר מכן ישולב תיאור זה עם התיאור שמופיע על הסרט ושילוב זה יהיה הפלט הסופי. אם כן, המרכיב B פועל באופן הבא:

1. קרא את הקלט מהסרט (זה התיאור של B עצמו כפי שנכתב על ידי A).
2. חשב את התיאור של מכונת טיורינג המדפיסה פלט זה (זה יהיה התיאור של A).
3. שלב את שני התיאורים כך שבשילוב יופעל קודם A ואחר כך B.
4. הדפס את התיאור של מכונת טיורינג המשולבת ועצור.

קראו שוב בעיון את התיאור של המרכיב B. שימו לב: זהו תיאור כללי ושלם שאיננו תלוי בשום מידע נוסף. לפיכך, אנו יכולים לרשום תיאור מפורט של מכונת טיורינג שמבצעת את הפעולות של המרכיב B, כפי שהן מתוארות לעיל. לכן, המרכיב A יכול לחשב ולהדפיס את $\langle B\rangle$ ללא שום קושי. לבסוף, המכונה SELF מבצעת תחילה את A ואחר כך את B, כפי שמתואר בראש עמוד 248 בספר הלימוד.

המשיכו לקרוא את משפט הרקורסיה עד לפני התת-סעיף "יישומים"

משפט הרקורסיה מאפשר לתכניות מחשב להתייחס לעצמן. באופן פורמלי, המשפט מראה שמכונות טיורינג יכולות לחשב את התיאור שלהן עצמן ואז להשתמש בתיאור זה כאחד מהארגומנטים המעורבים בחישוב מסוים. כלומר, אם המכונה SELF שהוצגה לעיל מסוגלת לחשב את התיאור שלה עצמה ואז להדפיס אותו, משפט הרקורסיה מראה שניתן גם לבצע חישוב התלוי בתיאור זה.

משפט הרקורסיה: תהי T מכונת טיורינג המחשבת פונקציה $t:\Sigma^*\times\Sigma^*\to\Sigma^*$. אז קיימת מכונת טיורינג R המחשבת פונקציה $r:\Sigma^*\to\Sigma^*$ כאשר $r(w)=t(\langle R\rangle,w)$ לכל $w\in\Sigma^*$.

כלומר, קיימת מכונת טיורינג R המחשבת אותה פונקציה של שני משתנים כמו המכונה T, אלא שהמכונה T ציפתה לקבל מהקלט את שני הארגומנטים של הפונקציה, ואילו המכונה החדשה R מצפה לקבל מהקלט רק את הארגומנט השני, והארגומנט הראשון הוא התיאור שלה עצמה.

נניח ש-T מצפה לקבל מהקלט שתי מחרוזות המקודדות באופן הבא: $v\#w_{\sqcup}$; כלומר, $v \in \Sigma^*$ הוא הארגומנט הראשון של הפונקציה $t(\cdot,\cdot)$, $w \in \Sigma^*$ הוא הארגומנט השני, התו $\#$ משמש כמפריד בין שני הארגומנטים ואילו תו הרווח מציין את סוף הקלט.

המכונה R שנתאר כעת תקבל קלט מהצורה $w_{\sqcup}$ ותחזיר את $r(w) = t(\langle R\rangle, w)$. אופן הפעולה של R דומה מאוד לאופן הפעולה של SELF מלבד ההבדלים האלה:

- במכונה R יהיו שלושה מרכיבים: המרכיב A והמרכיב B, כמו ב-SELF, והמרכיב השלישי T שיבצע את פעולת החישוב (ב-SELF היו רק שני מרכיבים, כיוון שלא היה מרכיב של חישוב).
- המרכיב A צריך להדפיס גם את המרכיב B וגם את המרכיב T (ב-SELF המרכיב A הדפיס רק את המרכיב B).
- לכל אורך הדרך יש לשמור על הסרט את המחרוזת w כדי שהמרכיב האחרון T יוכל לחשב איתה (במכונה SELF התעלמנו לחלוטין מן הקלט).

אם כן, נתאר את המכונה R:

1. תוכן הסרט בהתחלת הפעולה הוא $w_{\sqcup}$.
2. המרכיב A: הדפס על סרט הקלט את התיאור של המרכיב B, אחריו את התיאור של המרכיב T, ולבסוף את הסיפא $\#w_{\sqcup}$. התיאור של המרכיב B ניתן לחישוב, באותו אופן שבו הוא חושב ב-SELF (מכיוון שהתיאור של פעולת B, שמפורט להלן, הוא תיאור שלם שאיננו זקוק לשום מידע נוסף ועל כן אנו יכולים לרשום תיאור מפורט של מכונה כזו). התיאור של המרכיב T ידוע, כיוון ש-T היא מכונה נתונה.
3. המרכיב B: בחן את תוכן הסרט עד לתו $\#$ וייצר תיאור של מכונת טיורינג המדפיסה מחרוזת זו, אחר כך היא מדפיסה $\#$ ולבסוף היא מדפיסה את הערך שהיה רשום במקור על הסרט. כלומר, זוהי וריאציה קטנה על המכונה שתוארה בלמה 6.1 ובתרגיל 8.1. אחר כך שלב את תיאור המכונה שקיבלת ($\langle A\rangle$) ביחד עם התיאור שהופיע במקור על הסרט ($\langle BT\rangle$) לתיאור של מכונה אחת שעושה קודם את הפעולות של המרכיב A, ואחריו את פעולות שני המרכיבים B ו-T. את התיאור הסופי הזה רשום על הסרט במקום מה שהיה רשום במקור עד לתו $\#$. כל מה שהיה רשום מעבר לתו $\#$ חייב להישמר (זהו הקלט w שעומד סוף-סוף להיכנס לפעולה).
4. המרכיב T: הפעל את המכונה T על תוכן הסרט הנוכחי. מאחר שהסרט מכיל כעת את המחרוזת $\langle R\rangle\#w_{\sqcup}$, אז הפלט הסופי שנקבל יהיה בדיוק מה שרצינו, כלומר $r(w) = t(\langle R\rangle, w)$.

תרגיל 8.2

פתרו את בעיה 6.22 בספר הלימוד. הדרכה: המכונות M ו-N אמורות להיות שונות, אך השוני יכול להיות קטן מאוד.

המשיכו לקרוא בספר הלימוד את התת-סעיף "יישומים"

המסר העיקרי ממשפט הרקורסיה הוא שאנו יכולים לחשוב על מכונות טיורינג M שבהן מופיעה ההוראה "קבל תיאור עצמי $\langle M\rangle$". בתת-סעיף זה מופיעים שלושה יישומים של המשפט.

במשפט 6.5 אנו רואים הוכחה קצרה ואלגנטית לכך שבעיית הקבלה איננה כריעה. נניח שקיימת מכונה H המכריעה את A_{TM}. המכונה B המתוארת כאן מחשבת את התיאור שלה עצמה ואז מפעילה את H על $\langle B,w\rangle$ (דהיינו, על התיאור שלה עצמה ועל הקלט שהיא קיבלה). אם H מקבלת את $\langle B,w\rangle$ (כלומר, אם H קובעת ש-B מקבלת את w), אז B דווקא תדחה את w. ולהפך, אם H דוחה את $\langle B,w\rangle$ (כלומר, אם H קובעת ש-B דוחה את w או לא עוצרת עליה), אז B דווקא תקבל את w. בקיצור, B כזו מדגימה שהמכונה H חייבת להיכשל בניבוי ההתנהגות של חלק מהמכונות.

<u>שאלה</u>: המכונה B המתוארת בהוכחת משפט 6.5 מקבילה למכונה R בהוכחת משפט 6.3. תארו את פעולת המכונה T המתאימה במקרה זה.
<u>תשובה</u>: המכונה T מקבלת שני ארגומנטים, M ו-w, ומחזירה את ההפך ממה שמחזירה H על הקלט $\langle M,w\rangle$. המכונה B מבצעת אותה פעולה כמו המכונה T, אלא שבמקום הארגומנט הראשון היא משתמשת בתיאור שלה עצמה, $\langle B\rangle$.

משפט 6.7 אומר ששפת המכונות בעלות אורך מינימלי איננה ניתנת לזיהוי-טיורינג (מכונת טיורינג נקראת מינימלית, אם לא קיימת מכונת טיורינג שקולה בעלת תיאור קצר יותר).

תרגיל 8.3

הראו שכל תת-קבוצה אינסופית של MIN_{TM} (למשל, אוסף מכונות טיורינג המינימליות שיש להן מספר זוגי של מצבים) – גם היא איננה ניתנת לזיהוי-טיורינג.

לבסוף, משפט 6.8 אומר שלכל טרנספורמציה חשיבה של מכונות טיורינג (כלומר, פונקציה המקבלת תיאור של מכונת טיורינג ומשנה אותו לתיאור חוקי של מכונת טיורינג אחרת) קיימת נקודת שבת. כלומר, קיימת מכונת טיורינג שהטרנספורמציה משנה אותה למכונה השקולה לה (כלומר, מכונה F כך ש-$\langle F\rangle$ ו-$t(\langle F\rangle)$ הם תיאורים של שתי מכונות שקולות).

תרגיל 8.4

פתרו את בעיה 6.23 בספר הלימוד.

תרגיל 8.5

פתרו את בעיה 6.25 בספר הלימוד.

8.2 כריעוּת של תורות לוגיות

מומלץ לקרוא את התת-סעיף "פונקציות ויחסים" בסעיף 0.2 בפרק המבוא של הספר (עמודים 7 עד 10), כדי להיזכר במושגים הבסיסיים בנוגע ליחסים.

קראו את המבוא לסעיף 6.2 בספר הלימוד עד לפני התת-סעיף "תורה כריעה"

הבעיות שבהן אנו עוסקים בסעיף זה הן לגבי אמיתות של טענות מתמטיות. כלומר, בהינתן טענה מתמטית, יש להכריע אם היא נכונה או לא. למעשה, השאיפה הגלומה בשאלה זו היא לייצר "מתמטיקאי אוטומטי": מכונה שיכולה להחליף את עבודת המתמטיקאי האנושי ולהוכיח או להפריך טענות מתמטיות. אנו נראה שכריעוּת בעיה זו תלויה באופי הטענות המתמטיות שלגביהן נשאלת השאלה. ניתן לומר שאם הטענות המתמטיות האמורות הן פשוטות מאוד, אז קיימת מכונה שתוכל להחליף את המתמטיקאי האנושי. אך אם הטענות הן מעט פחות פשוטות, לא קיימת מכונה כזו.

אנו מתרכזים כאן בשתי משפחות של טענות מתמטיות. המשפחה הראשונה מורכבת מטענות מתמטיות שבהן מופיעים משתנים וקבועים שהם מספרים טבעיים, והפעולה היחידה המופיעה בטענות אלו היא פעולת החיבור. למשל, הטענות

$$\forall x \in \mathbb{N} \quad \exists y \in \mathbb{N} \quad [y = x + 1]$$

ו-

$$\forall x \in \mathbb{N} \quad \exists y \in \mathbb{N} \quad [x = y + y]$$

הן טענות כאלו. הראשונה קובעת שלכל מספר טבעי יש עוקב (ולכן היא נכונה). השנייה קובעת שכל מספר טבעי הוא זוגי (ולכן איננה נכונה).

המשפחה השנייה מורכבת מטענות דומות שמערבות גם את פעולת הכפל. למשל, בדוגמה מספר 1 בספר, בעמוד 252 מוצגת טענה כזו. הטענה בדוגמה זו קובעת שלכל מספר טבעי q קיים ראשוני גדול ממנו; במילים אחרות, שיש אינסוף ראשוניים (טענה זו נכונה, כמובן). גם דוגמה מספר 3 היא כזו. הטענה בדוגמה זו קובעת שיש אינסוף ראשוניים-תאומים, כלומר זוגות מספרים מהצורה $p, p+2$ ששניהם ראשוניים (כמו 17 ו-19 או 29 ו-31). טענה זו טרם הוכחה וטרם הופרכה. דוגמה מספר 2 איננה ניתנת לתיאור באמצעות כפל וחיבור בלבד מכיוון שמופיעה שם פעולת החזקה כאשר המעריך הוא *משתנה*. אילו המעריך היה מספר קבוע, אז ניתן היה לרשום

את החזקה באמצעות מכפלות; אך כאשר בחזקה מופיע משתנה, אין אפשרות לתרגם את החזקה למכפלות.

שתי התוצאות המרכזיות בסעיף זה הן:

- בעיית הנכונות של טענות מהמשפחה הראשונה היא כריעה.
- בעיית הנכונות של טענות מהמשפחה השנייה איננה כריעה.

בהמשך, מגדירים באופן פורמלי את השפה המתאימה. האלפבית כולל את:

1. האופרטורים הבולאניים $\neg \vee \wedge$;
2. הכמתים $\exists \; \forall$;
3. סוגר ימני וסוגר שמאלי;
4. משתנים (כאשר, כפי שמוסבר בספר, התו x מספיק לציון כל מספר של משתנים; למשל, אם יש צורך בשלושה משתנים, נסמנם ב- x, xx, xxx);
5. סימני יחסים R_i שיקבלו מאוחר יותר משמעות קונקרטית על ידי הצבת יחס.

נוסחה אטומית היא מחרוזת מהצורה $R_i(x_1,\ldots,x_j)$ (כאן לא נשתמש בסימון המייגע x, xx, xxx כדי לפשט את ההצגה). טיפוס היחס (arity) הוא מספר המשתנים שבו. למשל, $R(x, y, z)$ הוא יחס מטיפוס 3. הצבה אפשרית עבור יחס כזה היא $x + y = z$. הצבה אפשרית אחרת היא $x \times y > z$.

נוסחה היא כל ביטוי שניתן לקבל מנוסחאות אטומיות על ידי שימוש באופרטורים בולאניים ובכמתים.

<u>שאלה:</u> בנו בשלבים את הנוסחה שבדוגמה 1, בעמוד 252 בספר, מן הנוסחאות האטומיות שלה.
<u>תשובה:</u> יש בנוסחה זו שלושה יחסים:

- הראשון מטיפוס 1: $R_1(x) : x > 1$;
- השני מטיפוס 2: $R_2(x, y) : x > y$;
- השלישי מטיפוס 3: $R_3(x, y, z) : xy \neq z$.

הנוסחה ניתנת כעת לרישום כ: $\forall q[\exists p[\forall x[\forall y[F]]]]$, כאשר מה שמופיע בתוך הסוגריים הפנימיים הוא הביטוי: $F = R_2(p, q) \wedge \left(\neg\left(R_1(x) \wedge R_1(y)\right) \vee R_3(x, y, p)\right)$.

משתנה ייקרא **חופשי** אם הוא אינו קשור על ידי כמת. למשל, בנוסחה $\forall x[\exists y[x \leq y + z]]$, המשתנים x ו- y קשורים, אך המשתנה z הוא חופשי. נוסחה שאין בה משתנים חופשיים נקראת **טענה** או **משפט** (למשל, אם נקשור את z בנוסחה האחרונה על ידי הכמת היישי, נקבל את הטענה $\forall x[\exists y[\exists z[x \leq y + z]]]$).

מודל (או **פירוש** או **מבנה**) הוא התחום שבו המשתנים מקבלים ערכים, יחד עם הצבות (או פירושים) לכל סימני היחסים המופיעים בנוסחה. למשל, הסתכלו בנוסחה שבדוגמה הראשונה מבין השלוש בעמוד 252 ; צורתה היא :

$$\forall q[\exists p[\forall x[\forall y[R_2(p,q) \wedge (\neg(R_1(x) \wedge R_2(y)) \vee R_3(x,y,p))]]]]$$

במקרה זה, המודל הוא :

$$(\mathbb{N},\ \ R_1(x) : x > 1,\ \ R_2(x,y) : x > y,\ \ R_3(x,y,z) : xy \neq z)$$

כלומר, המשתנים הם מספרים טבעיים, סימן היחס האונארי מציין שהארגומנט של היחס גדול מ-1, סימן היחס הבינארי מציין שהארגומנט הראשון גדול מהארגומנט השני, וסימן היחס המשולש מציין שמכפלת שני הארגומנטים הראשונים שונה מהארגומנט השלישי. במודל זה, הנוסחה נכונה, כיוון שהיא מתפרשת כטענה האומרת שיש מספרים ראשוניים גדולים כרצוננו.

אוסף כל הנוסחאות שמשתמשות רק בהצבות היחסים המופיעים לעיל נקרא **שפת המודל**. אם ϕ היא משפט בשפת המודל, להבדיל מנוסחה, אז היא אמיתית או שקרית. אוסף כל המשפטים הנכונים במודל נקרא **התורה של המודל**. אם כן, אנו שואלים אילו תורות הן כריעות.

תרגיל 8.6

פתרו את בעיה 6.26 בספר הלימוד.

תרגיל 8.7

פתרו את בעיה 6.27 בספר הלימוד.

קראו בספר הלימוד את התת-סעיף "תורה כריעה"

תרגיל 8.8

א. פתרו את בעיה 1.37 בעמוד 89 בספר הלימוד (קראו תחילה את בעיה 1.36 וחשבו עליה).

ב. יהי Σ_i אלפבית המורכב מ- 2^i האותיות מהצורה $\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \end{bmatrix}$, כאשר $b_1,\ldots,b_i \in \{0,1\}$ (למשל, האלפבית בחלק א של התרגיל היה Σ_3). כל מחרוזת מעל Σ_i ניתנת לפירוש כייצוג של i מספרים הרשומים בשורות לרוחב המחרוזת. יהיו $1 \le k,l,m \le i$ שלושה אינדקסים של שורות. הראו ששפת כל המחרוזות ב- $\Sigma_i{}^*$ שבהן המספר בשורה ה- m שווה לסכום המספרים בשורות ה- k וה- l היא שפה רגולרית. לשם כך מספיק לתאר במילים כלליות ולא באופן מדוקדק אוטומט המזהה שפה זו.

נעיין תחילה ברעיון ההוכחה של משפט 6.12, כפי שהוא מופיע בספר. נדגים את הרעיון על הטענה הבאה:

$$\forall x \exists y \exists z \left[x = y + z \right] \quad x, y, z \in \mathbb{N}$$

כיוון ש-$\mathbb{N}$ כולל גם את 0, אז הטענה נכונה (לעומת זאת, אילו $\mathbb{N}$ היה מתחיל ב-1, אז הטענה לא הייתה נכונה; מדוע?). במקרה זה $l = 3$ והתת-נוסחאות הן:

$$\phi_3 = \psi = [x = y + z]$$
$$\phi_2 = \exists z[x = y + z]$$
$$\phi_1 = \exists y \exists z[x = y + z]$$
$$\phi_0 = \forall x \exists y \exists z[x = y + z]$$

<u>שאלה:</u> מהן השפות שהאוטומטים A_1, A_2, A_3 המתאימים לנוסחאות ϕ_1, ϕ_2, ϕ_3 יזהו בדוגמה לעיל?

<u>תשובה:</u> האוטומט A_3 יקבל את כל השלשות (a_1, a_2, a_3) שיהפכו את ϕ_3 לנכונה. כלומר, את כל השלשות שבהן $a_1 = a_2 + a_3$. האוטומט A_2 יקבל את כל הזוגות (a_1, a_2) שבהם $a_1 \geq a_2$ (זכרו שהמספרים הטבעיים תמיד גדולים או שווים לאפס). האוטומט A_1 יקבל את כל המספרים a_1.

כעת ניגש להוכחה עצמה. נתחיל באוטומט A_l. אוטומט זה מזהה את כל ערכי ההצבות $(a_1, \ldots, a_l)$, המיוצגות כמחרוזות מעל Σ_l, שעבורן $\phi_l = \psi$ נכונה. כיצד ניתן לבנות אוטומט כזה? הנוסחה ψ מורכבת מנוסחאות אטומיות הקשורות על ידי אופרטורים בוליאניים. אנו יכולים לבנות אוטומט לכל אחת מהנוסחאות האטומיות, כפי שהתבקשתם להראות בחלק ב' של תרגיל 8.8. אחר כך נרכיב את האוטומט A_l לפי הכללים המתארים בניית אוטומט המתאים לחיתוך, איחוד או משלים של שפות רגולריות.

כעת נעבור לתיאור הבנייה של האוטומט A_i בהסתמך על האוטומט A_{i+1}. צריך לטפל בשני מקרים:

- המקרה שבו $\phi_i = \exists x_{i+1} \phi_{i+1}$.
- המקרה שבו $\phi_i = \forall x_{i+1} \phi_{i+1}$.

מכיוון שאין בעיה לתרגם אוטומט נתון המזהה שפה מסוימת לאוטומט המזהה את השפה המשלימה, אז המקרה השני ניתן לרדוקציה למקרה הראשון דרך הזהות

$$\forall x_{i+1} \phi_{i+1} = \neg \exists x_{i+1} \neg \phi_{i+1}.$$

על כן, מספיק להתרכז במקרה הראשון. העיקרון הוא פשוט: האוטומט שאנו צריכים לבנות, A_i, אמור לזהות את כל המחרוזות מעל Σ_i המייצגות i מספרים טבעיים $(a_1, \ldots, a_i)$ שמספקים את

הנוסחה $\phi_i = \exists x_{i+1}\phi_{i+1}$. האוטומט A_{i+1}, לעומת זאת, שאותו כבר בנינו, מזהה את כל המחרוזות מעל Σ_{i+1} המייצגות $i+1$ מספרים טבעיים $(a_1,\ldots,a_i,a_{i+1})$ שמספקים את ϕ_{i+1}.

מה נעשה? ניקח את A_{i+1} ובמקום שהוא יצפה לקבל בקלט את a_{i+1}, נהפוך אותו לאוטומט המקבל רק i מספרים טבעיים $(a_1,\ldots,a_i)$, ואז, תוך שימוש באי-דטרמיניזם, האוטומט מנחש ערך עבור a_{i+1} ואז בודק אם $i+1$ המספרים $(a_1,\ldots,a_i,a_{i+1})$ מספקים את ϕ_{i+1}. אם קיים ניחוש מוצלח של a_{i+1} כך ש-$(a_1,\ldots,a_i,a_{i+1})$ מספקים את ϕ_{i+1}, אז ברור ש-$(a_1,\ldots,a_i)$ מספקים את $\phi_i = \exists x_{i+1}\phi_{i+1}$; במקרה זה, האוטומט האי-דטרמיניסטי יקבל את $(a_1,\ldots,a_i)$. אם, לעומת זאת, לא קיים ניחוש כזה, הרי שהאוטומט הזה ידחה את $(a_1,\ldots,a_i)$.

המורכבות נמצאת בפרטים. להלן תיאור בנייה של האוטומט A_i השונה במעט מזה המתואר בספר. האוטומט החדש יכיל העתק של A_{i+1}. הוא יקרא בכל פעם תו קלט ב-Σ_i המייצג עמודת ביטים ב-i המספרים $(a_1,\ldots,a_i)$, מהביט המשמעותי ביותר ומטה (דהיינו, משמאל לימין). בכל פעם שהאוטומט יקרא תו קלט כזה, הוא ינחש את ערכו של הביט המתאים ב-a_{i+1}. כל אחד משני הניחושים האפשריים ייצור תו אפשרי אחר ב-Σ_{i+1}. למשל, אם $i=3$ והתו הנקרא הוא $\begin{bmatrix}1\\0\\1\end{bmatrix}$, אז האוטומט יסתעף ממצבו הנוכחי למצב שאליו A_{i+1} היה מגיע אילו קרא במצב זה את התו $\begin{bmatrix}1\\0\\1\\0\end{bmatrix}$ וגם למצב שאליו A_{i+1} היה מגיע אילו קרא במצב זה את התו $\begin{bmatrix}1\\0\\1\\1\end{bmatrix}$. על ידי יצירת הסתעפויות אי-דטרמיניסטיות שכאלו, אנו מנחשים את ערכו של a_{i+1} ביט אחרי ביט.

העניין המצריך עיון נוסף נובע מן האפשרות שהמספר a_{i+1} ארוך יותר מכל אחד מהמספרים $(a_1,\ldots,a_i)$. למשל, אם $i=3$ והקלט לאוטומט הוא

$$\begin{bmatrix}0\\1\\0\end{bmatrix}\begin{bmatrix}1\\1\\1\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix}$$

(כלומר, $a_1 = 5, a_2 = 14, a_3 = 6$), איננו יכולים להגביל את עצמנו לניחושים שבהם גם ב- a_4 יש רק ארבעה ביטים (כלומר, $a_4 \le 15$). למשל, אם ברצוננו לבדוק גם את הערך $a_4 = 46$ (ואכן, אנו רוצים להיות מסוגלים לבדוק **כל** ערך טבעי שהוא), עלינו להיות מסוגלים ליצור בתהליך האי-דטרמיניזם גם את הקלט הבא:

$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}\begin{bmatrix}0\\0\\0\\0\end{bmatrix}\begin{bmatrix}0\\1\\0\\1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\begin{bmatrix}0\\1\\1\\1\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

אם כן, תהליך הניחוש של a_{i+1} יכלול שני שלבים:

1. בשלב הראשון נייצר את העמודות המתאימות לביטים של a_{i+1} ש"חורגים" מן הטווח שהגדירו $(a_1,\ldots,a_i)$ (בדוגמה שלעיל אלו הן שתי העמודות השמאליות ביותר). בכל שלב כזה נבחר באופן אי-דטרמיניסטי את העמודה $\begin{bmatrix}0\\0\\\vdots\\0\\0\end{bmatrix}$ או את $\begin{bmatrix}0\\0\\\vdots\\0\\1\end{bmatrix}$.
2. בשלב השני נייצר את העמודות שבהן כבר יש ביטים שונים מאפס בלפחות אחד מהמספרים $(a_1,\ldots,a_i)$ (אלו הן ארבע העמודות הימניות בדוגמה לעיל).

כדי לממש שני שלבים כאלו (התיאור שאנו מביאים כאן שונה במעט מהתיאור בספר הלימוד), יהיו באוטומט החדש שני עותקים של A_{i+1}: האחד יממש את שלב היצירה הראשון, והשני יממש את שלב היצירה השני. באוטומט זה יהיה מצב התחלה חדש שיאפשר לנו לקפוץ (באמצעות מעברי- ε) או לעותק של A_{i+1} המממש את שלב היצירה הראשון (כדי לייצר ערכי a_{i+1} שאורכם בביטים גדול יותר מהאורך של כל אחד מ- i המספרים הנתונים) או ישירות לעותק המממש את שלב היצירה השני. כמו כן, יהיו מעברי- ε מהעותק הראשון לשני, שיבואו לידי ביטוי כאשר נרצה לעבור משלב היצירה הראשון לשלב היצירה השני. בעותק הראשון לא יהיו מצבים מקבלים; כל המצבים המקבלים יהיו רק בעותק השני. כך נוכל לנחש כל ערך טבעי שהוא עבור a_{i+1}.

לבסוף, A_0 יקבל כל מחרוזת אם ורק אם $\phi = \phi_0$ נכונה. לכן בבדיקה הסופית אנו בודקים אם A_0 מקבל את המחרוזת הריקה ε. זה מסיים את תיאור הבנייה של אוטומט המכריע את התורה $\mathrm{Th}(\mathbb{N},+)$.

תרגיל 8.9

פתרו את תרגיל 6.5 בספר הלימוד.

תרגיל 8.10

פתרו את בעיה 6.28 בספר הלימוד.

קראו בספר הלימוד את התת-סעיף "תורה לא כריעה" עד סוף הוכחת משפט 6.13

התורה $\mathrm{Th}(\mathbb{N},+,\times)$ מורכבת מכל המשפטים הנכונים שבהם הנוסחאות האטומיות הן:

$$R_1(x,y,z)=\{x+y=z\}\ ,\ R_2(x,y,z)=\{x\times y=z\}$$

תורה זו איננה כריעה.

תרגיל 8.11

המשפט $\forall x\forall y\neg\exists z\left[x>0\wedge y>0\wedge z>0\wedge x^3+y^3=z^3\right]$ הוא משפט נכון ב- $\mathrm{Th}(\mathbb{N},+,\times)$. זהו המקרה הפרטי המתאים לערך $n=3$ של המשפט האחרון של פרמה המופיע בדוגמה 2 בעמוד 252. העובדה שהמשפט נכון הייתה ידועה לפרמה עצמו כבר במאה ה-17. נסחו את המשפט באמצעות היחסים R_1,R_2 לעיל, אופרטורים בוליאניים וכמתים.

הוכחת האי-כריעות היא על ידי רדוקציה מבעיית הקבלה. בהינתן קלט $\langle M,w\rangle$ לבעיית הקבלה A_{TM}, הרדוקציה בונה נוסחה $\phi_{M,w}$ בשפה של $\mathrm{Th}(\mathbb{N},+,\times)$ שיש בה משתנה חופשי אחד, x. לנוסחה זו יש התכונה שקיים x המספק אותה (כלומר המשפט $\exists x\phi_{M,w}$ הוא נכון) אם ורק אם M מקבלת את w. אם קיים מספר טבעי כזה x, הוא מהווה קידוד של היסטוריית חישוב מקבלת של M על w. עיקר העבודה הוא בבנייה של נוסחה כזו שמקודדת באופן אריתמטי טענות חישוביות (זהו התוכן של למה 6.14). מאחר שבעיית הקבלה איננה כריעה, אין בנמצא מכונת טיורינג המסוגלת להכריע משפטים מהצורה $\exists x\phi_{M,w}$. לפיכך, התורה $\mathrm{Th}(\mathbb{N},+,\times)$ איננה כריעה.

המשיכו לקרוא את סעיף 6.2 עד סופו

משפט האי-שלמות של גדל מהווה אחת התוצאות המרכזיות והחשובות במתמטיקה של המאה ה-20. אם נפשט את המשפט, הטענה הנטענת בו היא שקיימים משפטים נכונים בתורת המספרים שאין כל דרך להוכיחם. נעיין למשל במשפט האחרון של פרמה, הגורס שלא קיימים מספרים טבעיים $a,b,c\geq 1$ כך ש- $a^n+b^n=c^n$, עבור $n\geq 3$ שלם. עדויות רבות התומכות בנכונותו הצטברו מאז המאה ה-17 שבה נוסח המשפט. אך למרות מאמצים ניכרים שהושקעו על ידי טובי המוחות, לא נמצאה עבורו הוכחה עד שנת 1994, שבה הוכרע על ידי אנדרו ויילס, שהוכיח את נכונותו. עד אותה שנה, היו שלוש אפשרויות:

1. המשפט איננו נכון, אך טרם נמצאה הדוגמה הנגדית שתפריך אותו.
2. המשפט נכון, וניתן להוכיחו, אך ההוכחה טרם נמצאה.
3. המשפט נכון, אך לא ניתן להוכיחו.

עד לשנה שבה הוכח משפט גדל, רק שתי האפשרויות הראשונות היו קיימות. לאחר התגלית המדהימה של קורט גדל, עלתה על הפרק האפשרות השלישית: שמשפט פרמה, שכל כך קל להשתכנע בנכונותו וכל כך קשה להוכיחו, הוא אולי דוגמה קונקרטית למשפט נכון בתורת המספרים שאיננו **ניתן להוכחה** או שאינו **יכיח** (provable). לבסוף, כאמור, נמצאה ההוכחה, אך משפטים נכונים ובלתי יכיחים עדיין קיימים.

הוכחה פורמלית של משפט היא סדרת טענות, שהראשונה או הראשונות בה הן הנחות המשפט, האחרונה היא המשפט עצמו, וכל טענה בסדרה נובעת מקודמותיה על ידי כללי היסק והיקש מקובלים או שהיא אקסיומה:

- דוגמה לכלל היסק מקובל הוא הכלל הידוע בשם **מודוס פוננס**: אם טענה אחת בסדרה קבעה ש- $P \to Q$ וטענה אחרת בסדרה קבעה ש- P נכונה, אנו רשאים להסיק שגם Q נכונה.
- דוגמה לאקסיומה מקובלת בתורת המספרים: לכל מספר טבעי n קיים מספר עוקב $n+1$.

<u>הנחות:</u>

1. ניתן לוודא כל הוכחה על ידי מכונה.
2. אם משפט הוא יכיח, אז הוא נכון (משום שאנו מקבלים את האקסיומות כאמיתיות, ואת כללי ההיקש כהגיוניים).

על מנת להוכיח את משפט האי-שלמות של גדל, מראים, ראשית, שאוסף כל המשפטים היכיחים ב- $\mathrm{Th}(\mathbb{N},+,\times)$ ניתן לזיהוי-טיורינג (משפט 6.15). ההוכחה היא פשוטה: נבנה מכונת טיורינג שבהינתן לה משפט ϕ לבדיקה, היא עוברת על כל ההוכחות האפשריות, אחת לאחת, ובודקת אם הן מהוות הוכחה של ϕ (פה אנו מסתמכים על הנחה 1 לעיל). אם ϕ יכיח, אז בשלב מסוים נגיע להוכחה שלו ואז נעצור. לעומת זאת, קיימים משפטים שעליהם מכונה זו תרוץ לעד (למשל, משפטים שאינם נכונים ולכן – נטולי הוכחה). שימו לב שאין כל בעיה לסרוק את כל ההוכחות האפשריות, שהרי כל הוכחה היא מחרוזת סופית, ואוסף המחרוזות הסופיות מעל אלפבית נתון ניתן למנייה.

מכאן מסיקים (משפט 6.16) שקיים ב- $\mathrm{Th}(\mathbb{N},+,\times)$ משפט נכון ϕ שאיננו יכיח. עבור משפט נכון אך בלתי יכיח כזה, ϕ, מתקיים שגם הוא וגם שלילתו $\neg\phi$ אינם יכיחים (שימו לב ש- $\neg\phi$ איננו יכיח כיוון שמנכונות ϕ נובע ש- $\neg\phi$ איננו נכון ולכן, עפ"י הנחה 2 לעיל, אף הוא איננו יכיח). נניח בשלילה שלכל משפט ϕ מתקיים שאחד מבין המשפטים ϕ ו- $\neg\phi$ יכיח. לפי הנחה 2 לעיל, רק אחד מהם יכול להיות יכיח, כיוון שרק אחד מהם נכון. לכן, נבנה מכונת טיורינג שמקבלת כקלט משפט ϕ ומחפשת במקביל הוכחה ל- ϕ וגם הוכחה ל- $\neg\phi$. כיוון שבדיוק אחד מהם יכיח,

המכונה תמצא את הוכחתו בזמן סופי כלשהו, ואז היא תוכל להכריע אם ϕ נכון. כלומר, מכונה כזו תכריע את $\mathrm{Th}(\mathbb{N},+,\times)$, בסתירה למשפט 6.13. לכן, חייב להיות קיים משפט ϕ כך שגם הוא וגם שלילתו $\neg\phi$ אינם יכיחים.

לבסוף, במשפט 6.17 מנסחים משפט מפורש שהוא נכון אך בלתי יכיח. הבה נחזור על ההוכחה:
בהוכחה בונים מכונת טיורינג S המבצעת את הפעולות הבאות לכל קלט (אין היא מתחשבת בקלט שהיא קראה):

1. S מקבלת את התיאור העצמי שלה, $\langle S\rangle$, על ידי שימוש במשפט הרקורסיה.
2. S משתמשת ב-$\langle S\rangle$ על מנת להרכיב את המשפט $\psi=\neg\exists c\left[\phi_{S,0}\right]$ (c הוא המשתנה החופשי ב-$\phi_{S,0}$), באמצעות למה 6.14; כלומר, המשפט ψ גורס ש-S איננה מקבלת את הקלט 0 (להזכירכם, S פועלת באותו אופן על כל הקלטים ולכן בחירתנו בקלט 0 כאן שרירותית לחלוטין).
3. S מריצה את האלגוריתם P מהוכחת משפט 6.15 על ψ. להזכירכם, P יכול לעצור ולקבל את ψ כמשפט יכיח, או לרוץ עד אינסוף.
4. אם השלב הקודם הסתיים בעצירת P ובקבלת ψ כמשפט יכיח, S מקבלת את הקלט שלה ועוצרת.

אנו טוענים שלא ייתכן ש-S תגיע לשלב 4 ותעצור. נניח בשלילה ש-S מגיעה לשלב 4 ועוצרת. משמע, P עצר וקיבל את ψ כמשפט יכיח. לכן ψ משפט נכון. לפיכך S איננה מקבלת את 0 (כיוון שזה בדיוק מה שאומר המשפט ψ). סתירה! לכן אפשרות זו נפסלת. מסקנה, S לעולם לא תגיע לשלב 4. כלומר, P לא מצא הוכחה ל-ψ. לפיכך ψ איננו משפט יכיח. אך ψ משפט נכון כיוון ש-S אכן איננה מקבלת את 0. אז מצאנו משפט נכון שאיננו יכיח.

פתרון התרגילים

תרגיל 8.1

חלק א:

נתאר מכונת טיורינג P_w יעילה מעט יותר מזו המתוארת בהוכחת למה 6.1. במקום למחוק את הסרט ורק אז לכתוב את המחרוזת $w = w_1 \cdots w_n$, המכונה המתוארת להלן כותבת ישר את w על הסרט ואחר כך מוחקת את כל תווי הקלט שאולי עוד נותרו מימין לסוף המחרוזת w.

האלפבית של הקלט נתון והוא Σ. אלפבית הסרט הוא $\Gamma = \Sigma \cup \{\sqcup\}$, כאשר $\sqcup$ הוא תו הרווח המציין את סוף הקלט. המצבים ב- P_w יהיו כדלקמן:

1. q_0 הוא מצב ההתחלה
2. $q_1, \ldots, q_n$ הם המצבים בזמן כתיבת המחרוזת $w = w_1 \cdots w_n$ לסרט כאשר המצב q_i מציין שכבר סיימנו לכתוב את התווים $w_1 \cdots w_i$.
3. q_n הוא גם המצב בזמן מחיקת תווי הקלט העודפים (אם יש כאלו) מימין ל- n התווים של $w = w_1 \cdots w_n$.
4. q_{accept} הוא המצב המקבל.
5. q_{reject} הוא המצב הדוחה, שאין כל אפשרות להגיע אליו.

פונקציית המעברים היא כדלקמן:

$$\delta(q_{i-1}, \gamma) = (q_i, w_i, R) \quad \forall \gamma \in \Gamma,\ 1 \le i \le n$$
$$\delta(q_n, \sigma) = (q_n, \sqcup, R) \quad \forall \sigma \in \Sigma$$
$$\delta(q_n, \sqcup) = (q_{\text{accept}}, \sqcup, R)$$

חלק ב:

כעת עלינו לתאר מכונת טיורינג M שבונה תיאור של המכונה לעיל בהינתן המחרוזת $w = w_1 \cdots w_n$. לשם פשטות, נניח שלמכונה יש שלושה סרטים. על הראשון מופיעה המחרוזת w והתוכן שלו לעולם לא משתנה. הסרט השני ישמש לספירת התווים ב- w כדי שנוכל למספר את המצבים q_i. הסרט השלישי יהיה סרט הפלט שעליו יירשם התיאור של P_w.

1. בשלב הראשון M תרשום לסרט הפלט את Σ ואחר כך את $\Gamma = \Sigma \cup \{\sqcup\}$. שני מרכיבים אלו בתיאור של P_w אינם תלויים ב- w.

2. בשלב הבא M תרשום את קבוצת המצבים ב- P_w. מצב ההתחלה יסומן ב- $q|$.
המצב q_i יסומן ב- $q11\cdots1|$ כאשר מספר ה-1-ים בסימון הוא i. לצורך יצירת סימונים אלו, תשתמש M בסרט השני. לבסוף, הסימונים $qa|$ ו- $qr|$ ישמשו לסימון המצב המקבל והמצב הדוחה, בהתאמה.
3. בשלב הבא תחזיר M את הראש הקורא לתחילת הסרט הראשון ותאפס את הסרט השני, כדי להתחיל בתיאור פונקציית המעברים. רישום התיאור של פונקציית המעברים יתנהל באופן דומה. בהתחלה יירשמו הכללים $\delta(q|,\gamma)=(q1|,w_1,R)$ לכל $\gamma\in\Gamma$. אחר כך תניע M את הראש הקורא על הסרט הראשון לתו הבא, תוסיף 1 לסרט השני ותרשום את הכללים $\delta(q1|,\gamma)=(q11|,w_2,R)$ לכל $\gamma\in\Gamma$. וכך הלאה.
לאחר מכן M תרשום את המעברים המתאימים למצב q_n.
4. בשלב האחרון תרשום M מהו המצב ההתחלתי, המצב המקבל והמצב הדוחה.

<u>חלק ג:</u>

התיאור של P_w משתמש בתווים הבאים:

- כל התווים מ- Σ המופיעים ב- w;
- התווים $q,1,a,r,|$ המשמשים לציון המצבים;
- התו $\sqcup$;
- התווים R,L לציון כיוון התנועה של הראש (אין שימוש מפורש בתו L);
- תו פיסוק מוסכם (למשל "#") להפרדה בין החלקים השונים בתיאור המכונה;
- תו פיסוק מוסכם (למשל ",") להפרדה בין המצבים השונים ובין ערכים שונים של פונקציית המעברים;
- תו פיסוק מסוים (למשל ";") להפרדה בין שני הארגומנטים של δ ובין שלושת הערכים שהיא מחזירה.

תרגיל 8.2

הרעיון הוא לבנות את M ו- N זהות כמעט למכונה SELF, בשינויים קלים. גם M וגם N יהיו מורכבות משני מרכיבים, כמו SELF, אך יהיו למרכיבים אלו שתי גרסאות שונות. למרכיב A יהיו הגרסאות A_L ו- A_R ולמרכיב B יהיו הגרסאות B_L ו- B_R. שתי המכונות יהיו $M=A_LB_R$ ו- $N=A_RB_L$. נתאר כעת את פעולת המרכיבים הללו כדי להשתכנע ש- M אכן מדפיסה את $\langle N\rangle$ ואילו N מדפיסה את $\langle M\rangle$.

1. A_L מדפיסה תיאור של B_L, שאת פעולתו נתאר להלן.

2. באופן דומה, A_R מדפיסה תיאור של B_R.
3. B_L ו- B_R זהות למרכיב B שתואר בבנייה של SELF, למעט שני הבדלים: (א) בכל פעם ש- B עוצר, הראש ינוע שמאלה ב- B_L וימינה ב- B_R (מידע זה לא בא לידי ביטוי בתיאור של B מכיוון ששם זה לא היה חשוב; כאן זהו מידע חשוב, כי זה האופן שבו אנו יוצרים הבדל בין שתי המכונות). (ב) המרכיב B קרא את תוכן הסרט ואז חישב תיאור של מכונת טיורינג שהדפיסה תוכן זה. אצלנו, המרכיבים B_L ו- B_R יקראו את תוכן הסרט, שאותו נסמן ב- $\langle C\rangle$ (זהו סימון מוצלח, כיוון שאכן מדובר בתיאור של מכונת טיורינג כלשהי, C), ואז הם יחשבו תיאור של מכונת טיורינג המדפיסה את $S(\langle C\rangle)$, כאשר $S(\langle C\rangle)$ הוא תיאור של מכונת טיורינג הזהה ל- $\langle C\rangle$ למעט ההבדל שבכל פעם שמגיעים למצב מקבל q_{accept}, אז אם ב- $\langle C\rangle$ הראש נע שמאלה, אז ב- $S(\langle C\rangle)$ הוא ינוע ימינה, ולהפך.

נבחן את פעולת $M = A_L B_R$. בהתחלה המרכיב A_L רץ ומשאיר על הסרט את $\langle B_L\rangle$. כעת נכנס לפעולה B_R. מרכיב זה יקרא את תוכן הסרט, שהוא $\langle B_L\rangle$, אבל אז יחשב תיאור של מכונה המדפיסה את $S(\langle B_L\rangle)=\langle B_R\rangle$, כלומר הוא יחשב את $\langle A_R\rangle$. לבסוף, B_R ימזג את שני המרכיבים וישאיר על הסרט את התיאור $\langle A_R B_L\rangle$ שהוא בדיוק $\langle N\rangle$. באופן דומה, N תדפיס את $\langle M\rangle$.

תרגיל 8.3

ההוכחה של משפט 6.7 עובדת גם כאשר מדובר בתת-קבוצה אינסופית של MIN_{TM}, כיוון שבתת-קבוצה כזו, בשל אינסופיותה, תמיד יהיו מכונות עם תיאור ארוך כרצוננו, ובפרט תמיד נמצא מכונה D עם תיאור ארוך יותר מזה של C.

תרגיל 8.4

כל מכונה שלעולם אינה עוצרת (כלומר מכונה שבה פונקציית המעברים δ היא כזו ששום קלט לא יגרום למכונה להגיע ל- q_{accept} או ל- q_{reject}) היא מכונה כזו. למשל, המכונה הבאה שרק מניעה את הראש הקורא ימינה על הסרט היא כזו:

$$\delta(q_0,\gamma)=(q_0,\gamma,R)\quad \forall\gamma\in\Gamma$$

כל מכונה שעבורה קיים אפילו קלט אחד שעליו היא תגיע ל- q_{accept} או ל- q_{reject} חייבת להשתנות בעקבות הטרנספורמציה המתוארת, כיוון שהמכונה לאחר הטרנספורמציה תתנהג באופן שונה מהמכונה המקורית, בהינתן הקלט הזה.

תרגיל 8.5

פתרון הבעיה מופיע בספר הלימוד.

תרגיל 8.6

פתרון הבעיה מופיע בספר הלימוד.

תרגיל 8.7

$(\mathbb{N}, =, <)$.

תרגיל 8.8

<u>חלק א:</u>

זוהי בעיה שקל לפתור באמצעות אוטומט סופי, כיוון שיש לזכור רק שני דברים.

ראשית, האם עד שלב זה אומתו כל עמודות הביטים. אם הייתה עמודה אחת שבה הסכום שונה מכפי שהיה צריך להיות, אין טעם להמשיך הלאה בבדיקה. למשל, אם ננסה לאמת את הקלט

$$\begin{bmatrix}1\\0\\1\end{bmatrix}\begin{bmatrix}0\\1\\1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix}\begin{bmatrix}1\\0\\1\end{bmatrix}$$

המייצג את השאלה האם $1011_2 + 0110_2 = 1101_2$ (כלומר, האם $11 + 6 = 13$) שתי העמודות הימניות יאומתו, אך בעמודה השלישית מימין תתגלה הבעיה ולכן אין טעם להמשיך בבדיקה.

הדבר השני שיש לזכור הוא את הערך הנגרר (carry) מהעמודה הקודמת. למשל, בדוגמה שלעיל, החיבור בעמודה הראשונה מימין (ה-least significant) יוצר ערך נגרר 0, בעוד שהחיבור בעמודה השנייה מימין יוצר ערך נגרר 1 שאותו יש לזכור לצרף לחיבור בעמודה הבאה בתור.

לפיכך, באוטומט שלנו יהיו שלושה מצבים:

1. q_0 – המצב ההתחלתי וכן המצב המציין שעד כה החיבור אומת והערך נגרר הנוכחי הוא 0. זה גם יהיה המצב המקבל.
2. q_1 – המצב המציין שעד כה החיבור אומת והערך הנגרר הנוכחי הוא 1.
3. q_2 - המצב המציין שהתגלתה שגיאה בחיבור ולכן אין טעם להמשיך בבדיקה. זהו מצב שאין אפשרות לצאת ממנו.

פונקציית המעברים תהיה כדלקמן:

1. ממצב q_0: $\begin{bmatrix}1\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix},\begin{bmatrix}0\\0\\0\end{bmatrix}$ מובילים ל- q_0, $\begin{bmatrix}1\\1\\0\end{bmatrix}$ מוביל ל- q_1 ויתר ארבעת הערכים מובילים ל- q_2.

2. ממצב q_1: $\begin{bmatrix}0\\0\\1\end{bmatrix}$ מוביל ל- q_0, $\begin{bmatrix}1\\1\\1\end{bmatrix},\begin{bmatrix}1\\0\\0\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix}$ מובילים ל- q_1, ויתר ארבעת הערכים מובילים ל- q_2.

3. ממצב q_2 כל המעברים מובילים ל- q_2.

<u>חלק ב:</u>

האוטומט כאן זהה כמעט לחלוטין לאוטומט בחלק א. כאשר האוטומט קורא את תו הקלט הבא, הוא מתרכז בערכי הביטים המופיעים בתו זה בשורות k,l,m ונוהג בדיוק כמו האוטומט שתואר לעיל, תוך התעלמות מערכי הביטים בשאר השורות. למשל, נניח ש- $i=5$ ו- $k=5$, $l=1$, $m=3$. אז האוטומט ינהג באופן זהה בכל ארבעת התווים שצורתם $\begin{bmatrix}1\\0,1\\1\\0,1\\0\end{bmatrix}$, ואופן הפעולה שלו יהיה זהה לאופן הפעולה של האוטומט בחלק א בהינתן תו הקלט $\begin{bmatrix}0\\1\\1\end{bmatrix}$.

תרגיל 8.9

פתרון תרגיל זה מופיע בספר הלימוד.

תרגיל 8.10

פתרון בעיה זו מופיע בספר הלימוד.

תרגיל 8.11

$$\forall x \forall y \forall z \forall t_1 \forall s_1 \forall t_2 \forall s_2 \forall t_3 \forall s_3 \begin{bmatrix} x > 0 \wedge y > 0 \wedge z > 0 \wedge \\ R_2(x, x, t_1) \wedge R_2(x, t_1, s_1) \wedge \\ R_2(y, y, t_2) \wedge R_2(y, t_2, s_2) \wedge \\ R_2(z, z, t_3) \wedge R_2(z, t_3, s_3) \rightarrow \\ \neg R_1(s_1, s_2, s_3) \end{bmatrix}$$

השורה השנייה מגדירה את $s_1 = x \cdot t_1 = x \cdot x^2 = x^3$. השורות השלישית והרביעית מגדירות את $s_2 = y^3$ ו- $s_3 = z^3$ באופן דומה. לבסוף, בשורה האחרונה נטען ש- $s_1 + s_2 \neq s_3$.

מהדורה פנימית

לא להפצה ולא למכירה

מק"ט 20585-5059