האוניברסיטה הפתוחה

אלגוריתמים

מדריך למידה

מנור מנדל ● זאב נוטוב

מדריך למידה לספר **פיתוח אלגוריתמים** פרקים 1-7, ג'. קליינברג, א. טארדוש.
תרגום לעברית בהוצאת האוניברסיטה הפתוחה, 2010.

20417
מהדורה פנימית
לא להפצה ולא למכירה
מק"ט 20417-5111

כותבים: ד"ר מנור מנדל
פרופ' זאב נוטוב

יועץ: דניאל רייכמן, האוניברסיטה הפתוחה

עורכת: אילנה גולן

הדפסה דיגיטלית – אוגוסט 2010

בית ההוצאה לאור של האוניברסיטה הפתוחה, הקריה ע"ש דורותי דה רוטשילד, דרך האוניברסיטה 1, ת"ד 808, רעננה 43107.

The Open University of Israel, The Dorothy de Rothschild Campus, 1 University Road, P.O.Box 808, Raanana 43107. Printed in Israel.

תוכן עניינים

List of Algorithms

הקדמה

מדריך למידה זה ילווה אתכם במהלך הלימוד של הקורס אלגוריתמים. ספר הקורס מכיל שבעה פרקים, והוא תרגום לעברית של הספר Algorithm Design מאת Eva Tardos & Jon Kleinberg , הוצאת Pearson, מהדורה ראשונה, 2006. (במקור 13 פרקים).

המטרות בקורס זה הן:

- תירגול התהליך של פיתוח אלגוריתמים: בהינתן בעיה אלגוריתמית מעשית, הלקוחה מ"מהחיים", עלינו לבצע את הפעולות האלה: 1. פישוט הבעיה והסרת פרטים לא מהותיים מהגדרת הבעיה; 2. ניסוי שיטות אלגוריתמיות שונות; 3. מציאת שיטה "טובה"; 4. כתיבת הוכחה מתמטית של נכונות וניתוח יעילות האלגוריתם המוצע.
- הצגה שיטתית של פרדיגמות אלגוריתמיות בסיסיות, כגון שיטות סריקה בגרפים, שיטות חמדניות, הפרד־ומשול, תכנון דינמי, זרימות וחתכים בגרפים.
- שימוש בשיטות האלה לפיתוח אלגוריתמים לפתרון בעיות אלגוריתמיות קלאסיות, כגון תכנון לוחות־זמנים, מציאת מסלול קצר ביותר בגרפים, או מציאת עץ פורש מינימלי.

קורס זה, כמו הקורסים במבני נתונים, מהווה בסיס חשוב לקורסים שתלמדו בהמשך, אבל השיטות שתלמדו בקורס הזה לפתרון בעיות אלגוריתמיות הנן שיטות יותר מתקדמות מאלה שלמדתם בקורסים קודמים. כמו־כן, מושם בקורס הזה דגש רב יותר על הדרישה להוכחות פורמליות של נכונות אלגוריתמים.

כל פרק בספר מתחיל מתיאור דוגמה שיש לה שימוש מעשי, והספר מתרגם אותה לבעיה אלגוריתמית מופשטת, כך שיהיה אפשר לתקוף אותה בשיטות פורמליות. הספר מציע כמה אפשרויות לפתרון "טבעי" או אינטואיטיבי, ומתקדם תוך כדי פסילה של חלק מהן, עד שהוא מגיע לאלגוריתם שלגביו אין דוגמה שיכולה לפסול אותו. לבסוף, מובא פתרון אלגוריתמי מלא, כולל הוכחת נכונות וניתוח סיבוכיות. אלה הם למעשה השלבים שעובר כל מי שמפתח אלגוריתמים.

לכל פרק בספר יש כ־30 תרגילים, לשניים מהם מובא פתרון מלא (שימו לב, הפתרונות בספר מפורטים יותר מהנדרש בממ"נים). כיוון שהספר מאוד מפורט, ומרבה בדיונים אינטואיטיביים, לא מצאנו לנכון להרחיב את הדיון בכיוונים אלה. במדריך למידה זה תמצאו בין היתר תמצות של האלגוריתמים והוכחת נכונותם, כאשר לעיתים ההוכחות שונות במקצת (בדרך כלל קצרות יותר) מאלה שבספר הלימוד.

כיצד לקרוא את המדריך ואת הספר. אנו ממליצים להתחיל את תהליך הלימוד בקריאת מדריך הלמידה (להלן "המדריך"). בדרך־כלל המדריך יפנה אתכם

לקריאת סעיפים בספר הלימוד (בהמשך נכנה אותו בקיצור "הספר"), לאחר־מכן ידון המדריך בחומר הלימוד המוגש בספר, תוך תמצות החומר, והוספת דוגמאות. במקרים אחדים הוספנו נושאים שאינם נכללים בספר, ובחרנו להרחיב ולפרט בהם יותר. בכל פרק של המדריך תמצאו כמה תרגילים – רובם עם פתרונות מלאים. אנו ממליצים לנסות לפתור תרגילים נוספים ממגוון התרגילים המובאים בספר בסוף כל פרק.

שימו לב, המדריך מכיל הפניות לא רק לחומרים שמופיעים במדריך, אלא גם לחומרים שמופיעים בספר. **אם לא מצוין אחרת – ההפניה היא למדריך. בהפניות לספר יצוין "הספר" במפורש.** לדוגמה: הפניה לאיור 4.1 היא הפניה לאיור 4.1 במדריך, ואילו בהפניה לאיור 4.1 בספר תמיד יירשם "איור 4.1 בספר".

מספרי הפרקים במדריך תואמים למספרי הפרקים בספר, אך מספור הסעיפים אינו תואם בהכרח.

אנו תקווה שתפיקו מקורס זה את המרב.

מנור מנדל וזאב נוטוב
רעננה, 2010

פרק 1

מבוא לקורס: בעיות מייצגות

1.1 זיווג יציב

בסעיף הזה נראה דוגמה לבעיה שבאופן מעט מפתיע אפשר לפתור אותה ביעילות בעזרת אלגוריתם פשוט, אך "חכם". הבעיה היא בעיית הזיווג היציב: בהינתן אוסף של נשים וגברים שלכל אחד ואחת מהם יש סדר מלא של העדפות לבן־זוג מהמין השני, יש למצוא זיווג, כך שלא יווצרו "בגידות".

קראו בספר את סעיף 1.1

להלן נביא תיאור מדויק וקצר של הבעיה. נניח כי $M = \{m_1, \ldots, m_n\}$ ו־$W = \{w_1, \ldots, w_n\}$ הן שתי קבוצות, בנות n עצמים ("גברים" ו"נשים" בהתאמה). אוסף של זוגות סדורים $S \subseteq M \times W$ נקרא זיווג [matching] אם אין אף גבר ואין אף אישה המופיעים יותר מפעם אחת ב־S. זיווג S נקרא זיווג מושלם [perfect matching] אם כל גבר וכל אישה מופיעים בדיוק פעם אחת ב־S.

נניח עתה שלכל גבר m יש רשימת העדפות P_m שבה מסודרות הנשים בסדר מלא, ובאופן דומה, לכל אישה w יש רשימת העדפות P_w שבה מסודרים הגברים בסדר מלא.

אנו נאמר ש־S איננו יציב אם ישנם שני זוגות $(m, w), (m', w') \in S$ כך ש־m מעדיף את w' על־פני w, ו־w' מעדיפה את m על־פני m'. במלים אחרות, בזיווג S יש שני זוגות (m, w) ו־(m', w') שהם זוגות לא־יציבים, כי m ו־w' מעדיפים זה את זו על־פני בני הזוג הנוכחיים שלהם ב־S. הזוג $(m, w') \notin S$ יקרא אי־יציבות ביחס ל־S.

הגדרה 1.1 זיווג יציב הוא זיווג מושלם שאין אי־יציבות ביחס אליו. $\diamondsuit$

החשיבות החברתית של זיווג יציב היא ברורה: זהו סוג של שיווי משקל שבו אין גבר m ואין אישה w' ששניהם רוצים להחליף את בני הזוג הנוכחיים שלהם לטובת יצירת הזוג (m, w'). יש כמובן זיווגים יציבים בהם חלק מהגברים והנשים אינם מזווגים לבן הזוג המועדף עליהם, כפי שאפשר לראות בדוגמה הבאה.

דוגמה: יהיו $M = \{m, m'\}$, $W = \{w, w'\}$ ונניח ש־$P_m = (w, w')$ (בסדר עדיפות יורד), $P_{m'} = (w', w)$, $P_w = (m', m)$ ו־$P_{w'} = (m, m')$. זיווג יציב אחד הוא

$$\{(m, w),\ (m', w')\}.$$

במקרה הזה, שני הגברים קיבלו את העדפתם הראשונה והם לא ירצו להחליף בנות זוג, לכן זהו זיווג יציב. הנשים, לעומת זאת, מזווגות כל אחת להעדפה השנייה ברשימה שלה. זיווג יציב אחר הוא

$$\{(m, w'),\ (m', w)\}.$$

בזיווג הזה, הנשים זווגו לפי ההעדפה הראשונה שלהן ולכן הן לא ירצו להחליף בני זוג. לעומת זאת, הגברים יוצאים לא מרוצים. ♣

תרגיל 1.1 נתון מופע חלקי של בעיית הזיווג היציב:

$$\begin{aligned} M &= \{m_1, m_2, m_3\} & W &= \{w_1, w_2, w_3\} \\ P_{m_1} &= (w_1, w_2, w_3) & P_{w_1} &= (m_3, m_2, m_1) \\ P_{m_2} &= (w_1, w_3, w_2) & P_{w_2} &= (m_3, m_1, m_2) \\ P_{m_3} &= (w_2, w_1, w_3) & P_{w_3} &=? \end{aligned}$$

רשימת ההעדפות של w_3 אינה מוגדרת עדיין. עבור כל אחת מ־$3! = 6$ רשימות העדיפות האפשריות של w_3 מצאו זיווג יציב עבור המופע המתקבל.

פתרון בעמוד 13

אלגוריתם G-S. האלגוריתם של גייל ושאפלי לחישוב זיווג יציב מתואר בספר. לשם הנוחות העתקנו אותו לכאן, והוא רשום כאלגוריתם 1.1.

Algorithm 1.1 Gale–Shapley

Initially all $m \in M$ and $w \in W$ are free
while there is a man m who is free and hasn't proposed to every woman **do**
  Choose such a man m
  Let w be the highest-ranked woman in m's preference list to whom m has not yet proposed
  if w is free **then**
    (m, w) become engaged
  else $\{w$ is currently engaged to $m'\}$
    if w prefers m' to m **then**
      m remains free
    else $\{w$ prefers m to $m'\}$
      (m, w) become engaged
      m' becomes free
return the set S of engaged pairs

נחזור עתה על ניתוח האלגוריתם. בשלב זה נוכיח רק את נכונות האלגוריתם. סיבוכיות האלגוריתם תנותח בפרק 2, יחד עם הגדרת מבני הנתונים הנדרשים.

ניתוח נכונות קל לראות (טענה 1.3 בספר) שהאלגוריתם מסתיים לאחר n^2 איטרציות של לולאת While לכל היותר, כי לכל זוג (m, w) במהלך האלגוריתם, m מציע אירוסין ל־w לכל היותר פעם אחת; בכל איטרציה של הלולאה מוצעים אירוסין, ויש רק n^2 זוגות מהצורה (m, w).

תרגיל 1.2 תנו דוגמה לרשימת העדפות של n גברים ו־n נשים שבה, מספר האיטרציות שיבצע אלגוריתם G-S (בכל ריצה אפשרית על קלט זה) יהיה:

1. $\Omega(n^2)$.
2. $O(n)$.

פתרון בעמוד 13

קל לראות שבמהלך ריצת האלגוריתם, אוסף הזוגות המאורסים מהווה זיווג, כיוון שזוג יכול להיווסף לזיווג רק אם שני בני הזוג היו חופשיים, או שהאשה הייתה מאורסת, ובמקרה זה הארוס הקודם שלה הפך לחופשי.

בשלב הבא מראים שהאלגוריתם מסתיים עם זיווג מושלם (טענה 1.5 בספר). נניח בשלילה שקיימים $m \in M$, $w \in W$ שאינם מזווגים בזיווג אותו מחזיר אלגוריתם G-S. נשים לב כי:

1. אישה המתארסת בשלב כלשהו של האלגוריתם, לא הופכת לחופשייה עד סוף האלגוריתם.
2. כיוון שבסיום ביצוע האלגוריתם m עודנו חופשי, משתמע שהוא בהכרח הציע אירוסין לכל הנשים.

בפרט, נובע בהכרח ש־m הציע ל־w, ולכן לאחר הצעתו, w הייתה מאורסת (אם היא לא הייתה מאורסת לפני־כן, היא מחויבת לקבל את ההצעה של m), מכאן ש־w סיימה את האלגוריתם מאורסת, בסתירה להנחת השלילה.

נותר לטעון שהזיווג המושלם המתקבל אינו מכיל אי־יציבות (טענה 1.6 בספר). לצורך זה אנו מבחינים תחילה שלכל אישה w, סדרת בני הזוג להם היא מאורסת במהלך האלגוריתם היא סדרה עולה ברשימת ההעדפות שלה (של w). עתה נניח בשלילה שיש אי־יציבות בזיווג המושלם S שהאלגוריתם מחזיר, כלומר יש שני זוגות $(m, w), (m', w') \in S$ כך ש־m מעדיף את w' על־פני w, ו־w' מעדיפה את m על־פני m'. כיוון ש־m סיים מאורס ל־w הוא בהכרח הציע ל־w אירוסין, וכיוון ש־m הציע אירוסין ל־w רק לאחר שהציע לכל הנשים בעדיפות גבוהה יותר ברשימות ההעדפות שלו, הוא בהכרח הציע אירוסין לפני זה ל־w'. לאחר הצעת האירוסין מ־m, w' בהכרח מאורסת למישהו שנמצא ברמת עדיפות של m או גבוהה יותר. ומרגע זה היא "לא יורדת ברמה" כלומר היא כבר לא תהיה מאורסת למישהו ברמת עדיפות נמוכה יותר. אך על־פי ההנחה היא סיימה את האלגוריתם מאורסת ל־m' שנמצא נמוך יותר מ־m ברשימת ההעדפות שלה, וזו סתירה. בזאת הסתיימה הוכחת הנכונות של אלגוריתם G-S. □

נציין כי אלגוריתם 1.1, אינו מגדיר תוואי ריצה יחיד – כל גבר חופשי יכול להציע אירוסין. למרות זאת, בכל תוואי ריצה, הפלט תמיד יהיה אותו דבר: כל גבר m מזווג לאישה שנמצאת בעדיפות הגבוהה ביותר ברשימת ההעדפות שלו, מבין כל הנשים שמזווגות ל־m באיזשהו זיווג יציב (טענה 1.7 בספר). באופן סימטרי, כל אישה w מזווגת לגבר שנמצא בעדיפות הנמוכה ביותר שלה מבין הגברים שמזווגים ל־w באיזשהו זיווג יציב (טענה 1.8 בספר).

תרגיל 1.3 (זיווג פוליגיני) נניח כעת ש־ $n = |M| \leq n' = |W|$ וכי לכל גבר $m \in M$ נתון

מספר $t_m \in \mathbb{N}$ כך שמתקיים:

$$\sum_{m \in M} t_m = n'.$$

אנו מכלילים את מושג הזיווג S ל"זיווג פוליגיני" שבו כל גבר $m \in M$ צריך להיות מזווג ל־t_m נשים. וכל אישה צריכה להיות מזווגת לגבר אחד. בנוסף לכך, הזיווג S צריך להיות יציב, כלומר, לא קיימים $(m, w), (m', w') \in S$ כך ש־$m \neq m'$ m מעדיף את w' על־פני w, ו־w' מעדיפה את m על־פני m'.

הוכיחו שבתנאים האלה קיים תמיד זיווג פוליגיני יציב, והציעו אלגוריתם יעיל לחישוב זיווג כזה.

פתרון בעמוד 14

1.2 בעיות מייצגות

קראו בספר מתחילת סעיף 1.2 עד תת־הסעיף "קבוצה בלתי־תלויה" (לא כולל)

בסעיף 1.2 בספר מוצגות חמש בעיות אלגוריתמיות; אתם מתבקשים לקרוא רק את שלוש הראשונות, והן:

1. **תזמון מקטעים:** בהינתן קבוצה של מקטעים, יש למצוא את תת־הקבוצה הגדולה ביותר של מקטעים זרים בזוגות.
2. **תזמון מקטעים ממושקלים:** בהינתן קבוצה $\mathcal{I}$ של מקטעים ממושקלים, יש למצוא תת־קבוצה $\mathcal{J} \subseteq \mathcal{I}$ של מקטעים זרים בזוגות, המביאה למקסימום את סכום משקולות המקטעים ב־$\mathcal{J}$.
3. **זיווג מקסימלי:** בהינתן אוסף של גברים ונשים ואוסף E של זוגות מעורבים, יש למצוא זיווג $M \subseteq E$ גדול ככל האפשר.

את הבעיה הראשונה אפשר לפתור בשיטות "חמדניות" – נלמד אותן בפרק 4. הבעיה השנייה היא הכללה של הראשונה והיא מסובכת יותר לפתרון. עבורה נפתח (בפרק 6) שיטה הנקראת "תכנון דינמי". את הבעיה השלישית אפשר לפתור על־ידי רדוקציה לבעיה "זרימות ברשתות" שבה נדון בפרק 7.

לפני־כן, בפרק 2, נחזור על המושג סיבוכיות אסימפטוטית, ועל מבני נתונים פשוטים הנדרשים באלגוריתמים שיוצגו במהלך הקורס. שני הנושאים האלה נלמדו בקורסים שעסקו במבני נתונים, ולכן נדון בהם בקצרה.

בפרק 3 נלמד את המושג גרף, שתפקידו לייצג קשרים בין אובייקטים, והוא מאפשר, למשל, להגדיר באלגנטיות את שלוש הבעיות שלעיל. בנוסף נלמד כמה שיטות בסיסיות לסריקת גרפים.

פרק 5 יעסוק בשיטה אלגוריתמית הנקראת "הפרד־ומשול" שנמצאת למשל בבסיס האלגוריתם מיון־מיזוג והאלגוריתם מיון־מהיר שנלמדו בקורס "מבני נתונים ומבוא לאלגוריתמים".

לבסוף נציין שבהמשך הסעיף בספר נדונות עוד שתי בעיות. לבעיות אלה לא ידועים אלגוריתמים יעילים הפותרים אותן, והם לא ידונו בקורס הנוכחי. אתם תלמדו עליהם בקורס "חישוביות ומבוא לסיבוכיות" או בקורס לתואר שני "אלגוריתמי קירוב".

1.3 פתרונות לתרגילים

פתרון תרגיל 1.1 מעמוד 10

השאלה לא הגדירה מופע יחיד של בעיית הזיווג היציב, אלא 6 מופעים שונים בהתאם לערכה של P_{w_3}. במקום לעבור על כל אחד מהמופעים הללו, אנו נסרוק את מרחב הפתרונות האפשריים. כל זיווג יציב הוא בפרט זיווג מושלם. ישנם $3! = 6$ זיווגים מושלמים. נעבור על הזיווגים הללו ונבדוק אלו מהם יכולים להוות זיווג יציב לאחד מ־6 בעיות הזיווג היציב שבפנינו. אנו נראה שכולם, למעט אחד, אינם יציבים ללא תלות ב־P_{w_3}. רק הזיווג M_5 (המוגדר למטה), הוא יציב, עבור כל ערך של P_{w_3}.

1. הזיווג $\{(m_1, w_1), (m_2, w_2), (m_3, w_3)\} = M_1$ אינו זיווג יציב כי הזוג (m_3, w_2) הוא אי־יציבות ביחס ל־M_1.
2. הזיווג $\{(m_1, w_1), (m_2, w_3), (m_3, w_2)\} = M_2$ אינו זיווג יציב כי הזוג (m_2, w_1) הוא אי־יציבות ביחס ל־M_2.
3. הזיווג $\{(m_1, w_2), (m_2, w_1), (m_3, w_3)\} = M_3$ אינו זיווג יציב כי הזוג (m_3, w_2) הוא אי־יציבות ביחס ל־M_3.
4. הזיווג $\{(m_1, w_2), (m_2, w_3), (m_3, w_1)\} = M_4$ אינו זיווג יציב כי הזוג (m_3, w_2) הוא אי־יציבות ביחס ל־M_4.
5. הזיווג $M_5 = \{(m_1, w_3), (m_2, w_1), (m_3, w_2)\}$ הוא זיווג יציב: m_2 ו־m_3 מזווגים לעדיפות הראשונה שלהם ולכן לא יהוו חלק מאי־יציבות. נותרו רק (m_1, w_1) ו־(m_1, w_2) כאי־יציבויות פוטנציאליות. הזוג (m_1, w_1) אינו אי־יציבות כי w_1 מעדיפה את בן־הזוג הנוכחי שלה, m_2, על־פני m_1. גם הזוג (m_1, w_2) אינו אי־יציבות כי w_1 מעדיפה את הבן זוג הנוכחי שלה, m_3, על־פני m_1.
6. הזיווג $\{(m_1, w_3), (m_2, w_2), (m_3, w_1)\} = M_6$ אינו זיווג יציב כי הזוג (m_1, w_2) הוא אי־יציבות ביחס ל־M_6.

♣

פתרון תרגיל 1.2 מעמוד 11

1. נגדיר לכל הגברים אותה רשימת העדפות, לדוגמה,

$$P_{m_i} = (w_1, w_2, \ldots, w_n) \quad i \in \{1, \ldots, n\}\ .$$

אלגוריתם G-S מחשב זיווג יציב כלשהו

$$\{(m_1, w_{\pi(1)}),\ (m_2, w_{\pi(2)}),\ \ldots, (m_n, w_{\pi(n)})\}\ ,$$

כאשר π היא תמורה [permutation] כלשהי של $\{1, \ldots, n\}$. על־פי הדרך שבה אלגוריתם G-S עובד, m_i צריך להציע אירוסין ל־$\pi(i)$ נשים, ולכן מספר הצעות האירוסים (= מספר האיטרציות) הוא בסך־הכל:

$$\sum_{i=1}^{n} \pi(i) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

2. נגדיר לגברים רשימות העדפות כך שלכל גבר תהיה עדיפות ראשונה שונה, לדוגמה:

$$
\begin{aligned}
P_{m_1} &= (w_1, w_2, \ldots, w_n),\\
P_{m_2} &= (w_2, w_1, w_3, w_4, \ldots, w_n),\\
P_{m_3} &= (w_3, w_1, w_2, w_4, w_5, \ldots, w_5),\\
&\vdots\\
P_{m_i} &= (w_i, w_1, w_2, \ldots, w_{i-1}, w_{i+1}, w_{i+2}, \ldots, w_n),\\
&\vdots\\
P_{m_n} &= (w_n, w_1, w_2, \ldots, w_{n-1}),
\end{aligned}
$$

בריצת האלגוריתם G-S על הקלט הזה, כל גבר m_i יציע תחילה אירוסין ל־w_i, וריצת האלגוריתם תסתיים לאחר n הצעות אירוסין. ♣

פתרון תרגיל 1.3 מעמוד 11

בעיית הזיווג הפוליגני היציב בברור מכלילה את בעיית הזיווג היציב: עבור מופע של בעיית הזיווג היציב, נבנה מופע של בעיית הזיווג הפוליגני כאשר $t_m = 1$ לכל $m \in M$.

כפי שנראה עתה, ניתן בקלות יחסית גם לעבור בכיוון ההפוך: בהינתן מופע $\mathcal{I} = (M, W, (t_m)_{m\in M}, (P_m)_{m\in M}, (P_w)_{w\in W})$ של בעיית הזיווג הפוליגני היציב, נבנה (בעזרת אלגוריתם פשוט בזמן לינארי) מופע $\mathcal{I}' = (M', W', (P'_{m'})_{m'\in M'}, (P'_{w'})_{w'\in W'})$ של בעיית הזיווג היציב אשר מפתרונו – הזיווג היציב S' – ניתן יהיה בקלות לגזור זיווג פוליגני יציב S ל־$\mathcal{I}$.

בניית המופע $\mathcal{I}'$. באופן לא־פורמלי, $\mathcal{I}'$ יתקבל מ־$\mathcal{I}$ ע"י החלפת כל גבר m ב־t_m "עותקים" שלו. פורמלית לכל $m \in M$ נגדיר t_m "עותקים" $m_1, \ldots, m_{t_m}$. עתה נגדיר $M' = \{m_i : m \in M,\, i \in \{1, \ldots, t_m\}\}$, $P'_{m_i} = P_m$, $W' = W$, ו־P'_w מתקבל מ־P_w ע"י שיכפול כל גבר m ל־t_m עותקים ב־P'_w כך שהסדר בין גברים שונים נשמר. כלומר לכל שני גברים שונים $m, m' \in M$, $m \neq m'$, אם w מעדיפה את m על־פני m' ב־P_w אז w מעדיפה את m_i על־פני m'_j ב־P'_w לכל $i \in \{1, \ldots, t_m\}$ ולכל $j \in \{1, \ldots, t_{m'}\}$.

לאחר פתרון מופע בעיית הזיווג היציב $\mathcal{I}'$ מתקבל זיווג יציב S'. ממנו נייצר זיווג פוליגני כדלקמן:

$$S = \{(m, w) |\ \exists i \in \{1, \ldots, t_m\}\ (m_i, w) \in \mathcal{M}'\}.$$

כלומר, כל גבר m מזדווג לכל הנשים שזווגו ל"עותקים" שלו.

טענה 1.1 S הינו זיווג פוליגני יציב עבור $\mathcal{I}$.

הוכחה. כיוון ש־S' הינו זיווג מושלם, נובע מיידית ש־S הינו זיווג פוליגני מושלם. נבדוק עתה שהוא גם יציב. נניח בשלילה שקיימים $(m, w), (m', w') \in S$, $m \neq m'$ וגם m מעדיף את w' על פני w, ו־w' מעדיפה את m על פני m' (ב־P_w). מהגדרת S קיימים $i \in \{1, \ldots, t_m\}$ ו־$i' \in \{1, \ldots, t_{m'}\}$, כך ש־$(m_i, w), (m'_{i'}, w') \in S'$.

כיוון שסדרת העדפות של m_i זהה לסדרת ההעדפות של m, m_i מעדיף את w' על פני w. כיוון ש-w' מעדיפה את m על פני m' ב־P_w, על־פי הגדרת P'_w היא מעדיפה את m_i על פני $m'_{i'}$ ב־P'_w. מכאן שמצאנו אי־יציבות ב־S', וזו סתירה. $\square$

♣

פרק 2

יסודות ניתוח אלגוריתמים

פרק זה מהווה בחלקו חזרה על חלקים מהקורס "מבני נתונים ומבוא לאלגוריתמים". במקומות בהם הוא מהווה חזרה אנו נציין זאת ונשאיר לכם להחליט אם אתם זקוקים לרענון ידיעותיכם.

2.1 פתירות חישובית

קראו בספר מתחילת פרק 2 עד סוף סעיף 2.1

סעיף זה מציג את אמת־המידה העיקרית שתשמש אותנו כדי לציין אלגוריתם יעיל (תאורטית) שהיא: זמן הריצה הגרוע ביותר על קלטים באורך n החסום מלמעלה על־ידי פולינום כלשהו ב־n.

תרגיל 2.1 עבור זמני הריצה הבאים, בדקו אם הם זמני ריצה פולינומיאליים (כלומר חסומים על־ידי פולינום כלשהו) או לא.

1. $T(n) = 3n^{3.5}$,
2. $T(n) = 5n^2 + 3n2^{\log^2 n}$,
3. $T(n) = 2^{2^{\sqrt{\log n}}}$,
4. $T(n) = (\log n)^{\log n}$.

פתרון בעמוד 19

בקורס זה אנו מעוניינים לדעת, על־פי־רוב, את סדר הגודל של זמן הריצה באופן מדויק יותר (עד כדי קבועים). לחלק מהבעיות האלגוריתמיות יש כמה פרמטרי "גודל" טבעיים; אנו נציין את סיבוכיות האלגוריתמים לבעיות אלה כפונקציה של הפרמטרים האלה, ולא נסתפק בציון אורך הקלט בלבד.

2.2 קצב גידול אסימפטוטי

חומר זה נלמד במסגרת הקורס "מבני נתונים ומבוא לאלגוריתמים". אם ברצונכם לרענן את זכרונכם:

קראו בספר את סעיף 2.2

2.3 יישום יעיל לאלגוריתם G-S

קראו בספר את סעיף 2.3

בפרק 1 הובא תיאור של אלגוריתם G-S (אלגוריתם 1.1) לחישוב זיווג יציב. ראינו שהאלגוריתם מבצע $O(n^2)$ איטרציות, אך לא ביצענו ניתוח זמן ריצה, מפני שלא תיארנו כיצד בדיוק לממש כל איטרציה. מימוש נאיבי של האלגוריתם עלול להוביל לזמן ריצה בסדר גודל n לכל איטרציה, וזמן ריצה כולל של $\Omega(n^3)$. בסעיף זה אנו רואים ששימוש נבון במבני נתונים פשוטים – מערכים, רשימות מקושרות ומטריצות (שנלמדו בקורס "מבני נתונים ומבוא לאלגוריתמים") – מאפשר לבצע כל איטרציה בזמן קבוע, והזמן הכולל לביצוע האלגוריתם יהיה אם־כן $O(n^2)$.

2.4 סקירת זמני ריצה שכיחים

קראו בספר את סעיף 2.4

2.5 תור קדימויות

סעיף זה חוזר על התיאור של תור קדימויות ומימוש יעיל בעזרת ערמה – מושגים שנלמדו במסגרת הקורס "מבני נתונים ומבוא לאלגוריתמים". אם ברצונכם לרענן את זכרונכם:

קראו בספר את סעיף 2.5

2.6 פתרונות לתרגילים

פתרון תרגיל 2.1 מעמוד 17

1. פולינומיאלי. הפולינום $S(n)$ המוגדר להלן מקיים:
$$S(n)4n^4 = T(n) \cdot \sqrt{n} \geq T(n).$$
הא־ישיוויון האחרון מתקיים כיוון ש־$n \geq 1$.

2. לא פולינומיאלי. יהי $d \geq 1$ ו־$S(n) = a_d n^d + \cdots + a_1 n + a_0$ פולינום כלשהו, ונוכיח שקיים $n_0 \in \mathbb{N}$ כך ש־$T(n) > S(n)$ עבור כל $n \geq n_0$ (לתנאי הזה אנו קוראים באופן לא־פורמלי "$T(n) > S(n)$ עבור n מספיק גדול"). נסמן $A = |a_d| + \cdots + |a_1| + |a_0|$, ו־$R(n) = An^d$. ברור ש־$R(n) \geq S(n)$ לכל $n \geq 1$. עתה נראה ש־$T(n) > R(n) \geq S(n)$ עבור n מספיק גדול ובזאת סיימנו. ואמנם, עבור $n > A2^d$,
$$T(n) > 2^{\log^2 n} = n^{\log n} > n^{\log A} n^d \geq An^d.$$
האי־שיוויון האחרון מתקיים כיוון ש־$n > 2^d \geq 2^1 = 2$, ולכן $n^{\log A} \geq 2^{\log A} = A$.

3. לא פולינומיאלי. כמו בסעיף הקודם, מספיק להראות שלכל $d \in \mathbb{N}$ ו־$A > 0$, $T(n) > An^d$ עבור n מספיק גדול. נוציא $\log$ פעמיים משני האגפים ונניח ש־$n \geq \max\{2^{d+1}, A\}$. מכאן ש־
$$\log\log(An^d) \leq \log\log n^{d+1} = \log(d+1) + \log\log n \leq 2\log\log n.$$
לכן מספיק להראות ש־$\sqrt{\log n} > 2\log\log n$ עבור n מספיק גדול. ע"י הצבה $x = \sqrt{\log n}$ מספיק להראות $x > 2\log(x^2) = 4\log x$ עבור x מספיק גדול. ואמנם, קל לראות שעבור $x > 16$, מתקיים $x > 4\log x$.

4. לא פולינומיאלי. נשים לב ש־$(\log n)^{\log n} = n^{\log\log n}$. ולכן, כמו בסעיף הקודם, מספיק להראות שעבור כל $d \in \mathbb{N}$, קיים $n_0 \in \mathbb{N}$ כך ש־$\log\log n > d$. תנאי זה הוא ברור כיוון שניתן לבחור $n_0 = d + 1$.

♣

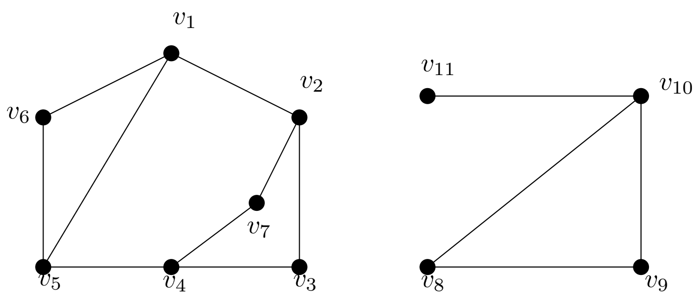
פרק 3

גרפים

גרף הוא אובייקט מתמטי המתאר קבוצה של עצמים ("קודקודים" או "צמתים") ואת היחסים ביניהם ("קשתות"). חלק גדול מהבעיות האלגוריתמיות המופיעות בקורס זה ובקורסי המשך במדעי המחשב, מוגדרות ונפתרות בעזרת גרפים. בפרק זה נלמד מושגים בסיסיים בתורת הגרפים, וגם כמה אלגוריתמי סריקה בגרפים.

3.1 הגדרות ויישומים בסיסיים

קראו בספר מתחילת פרק 3 עד סוף סעיף 3.1



איור 3.1: גרף שיש לו שני רכיבי קשירות

נפתח בהצגת מילון מושגים בסיסיים בגרפים.

גרף [**Graph**] הוא זוג $G = (V, E)$ שבו:

- V היא קבוצה סופית של איברים הנקראים צמתים [nodes] או קודקודים [vertices].
- E היא קבוצה של זוגות לא סדורים מ־V הנקראים קשתות [edges].

באיור מקובל לתאר גרף במישור באופן הזה: צמתים מיוצגים על־ידי נקודות או עיגולים, וקשת בין זוג צמתים מיוצגת על־ידי קו המחבר את זוג הצמתים המתאים. לדוגמה, באיור 3.1 מצויר הגרף $G = (V, E)$ שבו

$$V = \{v_1, \ldots, v_{11}\}$$
$$E = \left\{ \begin{array}{c} \{v_1, v_2\}, \{v_1, v_5\}, \{v_1, v_6\}, \{v_2, v_3\}, \{v_2, v_7\}, \\ \{v_3, v_4\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_5, v_6\}, \\ \{v_8, v_9\}, \{v_8, v_{10}\}, \{v_9, v_{10}\}, \{v_{10}, v_{11}\} \end{array} \right\}$$

הערה: בהמשך נדון גם בגרפים מכוונים, המוגדרים באופן דומה. ההבדל הוא שבגרף מכוון, E היא קבוצה של זוגות סדורים של V. בתיאור הציורי של גרף מכוון תהיה כל קשת מסומנת בחץ המתאר את הכיוון שלה. ♣

צומת שכן [adjacent node]. צומת v הוא שכן של צומת u, אם $\{u, v\}$ היא קשת בגרף. ברור שאם u שכן של v, אז v שכן של u, כלומר, יחס השכנות (בגרף לא מכוון) הוא סימטרי. לדוגמה, באיור 3.1, הצמתים v_{11} ו־v_{10} הם שכנים, אך הצמתים v_{11} ו־v_8 אינם שכנים.

דרגה [degree]. הדרגה של צומת u שווה למספר הקשתות הסמוכות ל־u ומסומנת $\deg(u)$. לדוגמה, באיור 3.1, $\deg(v_1) = 3$, $\deg(v_2) = 3$.

מסלול [path]. מסלול בגרף הוא סדרה של צמתים $(v_1 \ldots, v_k)$ (בסדרה לפחות צומת אחד) כך שכל שני צמתים עוקבים בסדרה הם שכנים בגרף. לדוגמה, בגרף שבאיור 3.1, סדרת הצמתים $(v_2, v_7, v_2, v_1, v_6)$ מגדירה מסלול, בעוד שהסדרה (v_1, v_2, v_4, v_5) אינה מגדירה מסלול, כיוון ש $\{v_2, v_4\}$ אינה קשת בגרף.

מסלול פשוט [simple path]. מסלול פשוט הוא מסלול שבו שום צומת אינו מופיע יותר מפעם אחת.

אורך המסלול [length]. אורכו של המסלול $(v_1, \ldots, v_k)$ הוא $k - 1$, כלומר כמספר הקשתות (כולל כפילויות) שהמסלול עובר.

מרחק בין צמתים [distance] מוגדר כאורך המסלול הקצר ביותר המחבר אותם. נשים לב שהמרחק מקיים את אי־שוויון המשולש (ראו תרגיל 3.2). אם אין בגרף מסלול בין u ל־v, המרחק ביניהם יוגדר כאינסוף.

מעגל [cycle]. מעגל הוא מסלול המכיל לפחות שלושה צמתים שונים, ובו הצומת הראשון והאחרון זהים, בעוד כל שאר הצמתים במסלול מופיעים בדיוק פעם אחת. לדוגמה, בגרף באיור 3.1, המסלול $(v_6, v_1, v_2, v_7, v_4, v_6)$ הוא מעגל. אורך המעגל הוא אורך המסלול המתאר אותו.

גרף קשיר [connected graph]. גרף נקרא קשיר אם בין כל זוג צמתים בגרף יש מסלול.

תת־גרף [subgraph]**.** $H = (W, F)$ נקרא תת־גרף של הגרף $G = (V, E)$, אם H הוא גרף, $W \subseteq V$, ו־$F \subseteq E$.

נתרגל מעט את המושגים האלה.

תרגיל 3.1 עליכם למצוא בגרף שהוצג לעיל באיור 3.1:

1. הדרגה של v_4.
2. השכנים של v_1
3. תת־גרף קשיר עם מספר מקסימלי של של צמתים.
4. תת־גרף על 4 צמתים עם מספר מקסימלי של קשתות.
5. מספר מקסימלי של מסלולים פשוטים בין הצמתים v_1 ו־v_5, כך שכל קשת בגרף תופיע לכל היותר במסלול אחד (כלומר, מצאו מספר מקסימלי של מסלולים פשוטים זרים בקשתות בין v_1 ל־v_5).
6. מעגלים באורך 4, 5, ו־6.

פתרון בעמוד 44

תרגיל 3.2 הפונקציה $d : V \times V \to [0, \infty)$ על V מקיימת את אי־שוויון המשולש, אם לכל $u, v, w \in V$ מתקיים $d(u, w) \leq d(u, v) + d(v, w)$. הוכיחו שהמרחק בין צמתים בגרף קשיר מקיים את אי־שוויון המשולש.

פתרון בעמוד 44

נתעמק עתה מעט במבנה של גרפים לא קשירים. נתבונן באיור 3.1. הגרף באיור הזה מתחלק לשני "חלקים" הנקראים רכיבי קשירות [connected components]. רכיב קשירות אחד מכיל את הצמתים $\{v_1, \ldots, v_7\}$, והרכיב השני מכיל את הצמתים $\{v_8, \ldots, v_{11}\}$. כל רכיב קשירות מהווה גרף קשיר ואין מסלולים בין צמתים מרכיבי קשירות שונים. נגדיר זאת באופן מדויק באמצעות שני התרגילים הבאים.

תרגיל 3.3 הוכיחו שאם בין זוג צמתים בגרף קיים מסלול אז קיים ביניהם מסלול פשוט.

פתרון בעמוד 44

תרגיל 3.4 הוכיחו שהיחס[1] על הצמתים "קיים מסלול בין u ל־v" הוא יחס של שקילות.[2]

פתרון בעמוד 44

כפי שלמדנו בקורס "מתמטיקה דיסקרטית", יחס השקילות משרה פירוק של קבוצת הצמתים למחלקות שקילות. במקרה של יחס הקשירות, מחלקות השקילות הן רכיבי הקשירות של הגרף. כלומר, בין כל שני צמתים מאותו רכיב קשירות, קיים מסלול המחבר אותם, בעוד שבין כל שני צמתים מרכיבי קשירות שונים אין מסלול המחבר אותם.

עצים. עץ הוא גרף קשיר ללא מעגלים, ראו דוגמה באיור 3.2 משמאל. מחיקת קשת מעץ תהפוך אותו ללא קשיר, כי אילו היה הגרף נשאר קשיר לאחר הסרת

---

[1] יחס על קבוצה V הוא תת־קבוצה של $V \times V$. ראו הקורס "מתמטיקה דיסקרטית".

[2] יחס R על קבוצה V נקרא יחס שקילות אם הוא מקיים: 1. רפלקסיביות: לכל $v \in V$ מתקיים $(v, v) \in R$; 2. סימטריות: לכל $u, v \in V$ מתקיים שאם $(u, v) \in R$ אז $(v, u) \in R$; 3. טרנזיטיביות: לכל $u, v, w \in V$, אם $(u, v) \in R$ וגם $(v, w) \in R$, אז $(u, w) \in R$.

קשת $e = \{x, y\}$, משמעות הדבר הייתה שיש בגרף מסלול פשוט מ־x ל־y, שאינו מכיל את e.

$$P = (x = x_0, x_1, \ldots, x_s = y),$$

אם נצרף למסלול הזה את הקשת e נקבל את המעגל

$$C = (x_0, x_1, \ldots, x_s, x_0)$$

וזוהי סתירה להגדרת עץ.

התכונה הבאה של עצים היא שימושית מאוד, כי היא מאפשרת להפעיל אינדוקציה להוכחת טענות הקשורות בעצים.

טענה 3.1 *בכל גרף המכיל לפחות קשת אחת אך אינו מכיל מעגלים, קיים צומת u שדרגתו שווה לאחד, כלומר $\deg(u) = 1$.*

הוכחה. נניח על דרך השלילה, שאין צומת שדרגתו שווה ל־1. נסיר מהגרף את כל הצמתים מדרגה 0. כיוון שהסרת צומת מדרגה 0 אינה משנה את דרגתם של שאר הצמתים, וכיוון שיש בגרף לפחות קשת אחת, נותרנו עם גרף שכל צמתיו מדרגה 2 לפחות. נראה עתה שגרף כזה חייב להכיל מעגל. רעיון ההוכחה הוא להתחיל מצומת כלשהו ו"להתקדם" מצומת לצומת דרך קשת שעוד לא עברנו בה. תהליך זה בונה מסלול. נעצור כשנחזור לצומת שכבר ביקרנו בו. מסלול כזה, כפי שנראה להלן, בהכרח מכיל מעגל.

מבחינה פורמלית, אם יש בגרף צמתים, קיים בו צומת v_1. כיוון שדרגתו של v_1 אינה 0, הוא חייב להיות שכן של צומת אחר, ומכאן שבגרף הנותר יש לפחות 2 צמתים. כעת נבנה באופן הדרגתי את המסלול הבא. נתחיל ב־(v_1) ונניח שבנינו כבר את הרישא $(v_1, \ldots, v_i)$. אם v_i כבר הופיע במסלול שבנינו (כלומר קיים $1 \leq j < i$ כך ש־$v_i = v_j$), נסיים. אחרת, יהי v_{i+1} צומת שכן ל־v_i כך ש־$v_{i-1} \neq v_{i+1}$ (זה אפשרי כי לפי הנחת השלילה $\deg(v_i) \geq 2$). נצרף את v_{i+1} למסלול ונקבל $(v_1, \ldots, v_i, v_{i+1})$. תהליך זה חייב להיעצר כי מספר הצמתים בגרף הוא סופי, ולכן באיזשהו שלב נחזור בהכרח לצומת שכבר עברנו בו. נניח שעצרנו ב־v_s, אזי בהכרח קיים $r < s$ כך ש־$v_r = v_s$. שימו לב, כיוון שזאת הפעם הראשונה שצומת חוזר על עצמו במסלול, מובטח לנו שבמסלול $v_r, \ldots, v_{s-1}$ מופיע כל צומת פעם אחת בלבד. כמו־כן מהגדרת המסלול אנו יודעים כי $r \leq s - 3$. מכאן אפשר להסיק ש־$v_r, v_{r+1}, \ldots, v_s$ הוא מעגל, ומסקנה זו עומדת בסתירה להנחה שהגרף אינו מכיל מעגלים. □

תרגיל 3.5 הראו שכאשר מסירים מעץ צומת שדרגתו שווה ל־1, הגרף שיתקבל יהיה גם הוא עץ.

פתרון בעמוד 45

הנה דוגמה לשימוש בטענה 3.1.

טענה 3.2 *בגרף ללא מעגלים ועם קשת אחת לפחות, קיימים לפחות שני צמתים שדרגתם שווה ל־1.*

הוכחה. לצורך ההוכחה נשתמש באינדוקציה על מספר הקשתות בגרף. בגרף בעל קשת אחת, הקצוות של קשת זו הם צמתים מדרגה אחת, ויש שני צמתים כאלה.

נניח עתה שבגרף $G=(V,E)$ יש $m>1$ קשתות. לפי טענה 3.1 קיים ב־G צומת u שדרגתו שווה ל־1. תהי $\{u,v\}$ הקשת הסמוכה ל־u. נסיר את u ואת הקשת $\{u,v\}$ מ־G. מתקבל הגרף $G'=(V\setminus\{u\},E\setminus\{u,v\})$. בגרף הזה אין מעגלים והוא בעל $m-1$ קשתות. כיוון ש־$m\geq 2$, אנו מסיקים ש־$m-1\geq 1$. על־כן, אנו יכולים להחיל את הנחת האינדוקציה על הגרף המצומצם G', ולהסיק שהוא מכיל לפחות שני צמתים מדרגה 1; נסמנם ב־x ו־y. מכיוון שהגרף G מכיל בדיוק קשת אחת שאינה קיימת ב־G', וקצה אחד של קשת זו (הצומת u) אינו ב־G', אזי לפחות אחד מן הצמתים x או y אינו v (ולבטח אינו u, שאינו מופיע ב־G') ולכן הוא בעל דרגה 1 גם ב־G. בתוספת הצומת u, מצאנו ב־G שני צמתים שדרגתם שווה ל־1. □

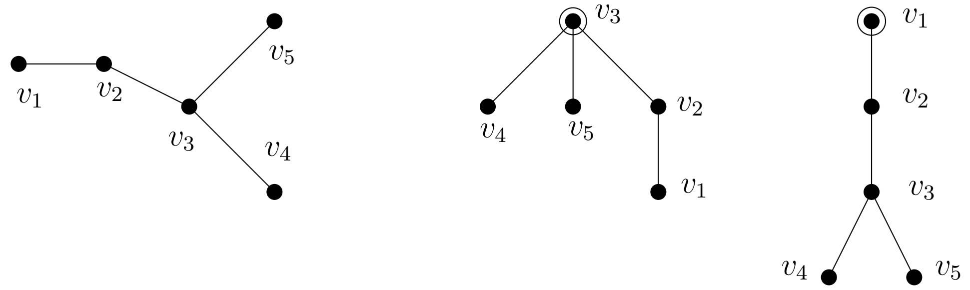
טענה 3.1 מאפשרת, לעיתים קרובות, להוכיח תכונות של עצים באינדוקציה על מספר הצמתים. זאת כיוון שכאשר מסירים מעץ צומת שדרגתו 1, הגרף המתקבל יהיה בעצמו עץ (תרגיל 3.5). נתרגל את השימוש בטענה 3.1 על־ידי הוכחה שונה של טענה (3.2) בספר.

תרגיל 3.6 הוכיחו באינדוקציה, תוך שימוש בטענה 3.1 ובתרגיל 3.5, שבעץ עם n צמתים יש $n-1$ קשתות.

פתרון בעמוד 45

תרגיל 3.7 השלימו את הוכחת טענה (3.2) בספר.

פתרון בעמוד 45



איור 3.2: עץ והשרשות שונות שלו. מצד שמאל מוצג עץ. במרכז זהו אותו העץ מושרש ב־v_3, ומימין אותו העץ מושרש ב־v_1.

עץ מושרש [rooted tree] הוא עץ שיש לו צומת מיוחד הנקרא שורש. בעץ מושרש מוגדר היחס צאצא/ אב־קדמון: צומת v הוא צאצא [descendant] של צומת u אם u מופיע על המסלול הפשוט (היחיד) המחבר את v לשורש העץ. שימו לב כי כל צומת הוא צאצא של עצמו. צומת v הוא בן של u אם הוא צאצא של u ושכן שלו. אב־קדמון [ancestor] הוא היחס ההפוך לצאצא. צומת u נקרא עלה בעץ מושרש T, אם u הוא עלה בעץ ואיננו השורש. באיור 3.2 מוצג מצד שמאל עץ. במרכז ומימין מוצגות שתי השרשות שונות של אותו העץ. בעץ המושרש מצד ימין באיור 3.2 העלים הם v_4 ו־v_5, בעוד שבעץ המושרש במרכז איור 3.2 העלים הם v_1, v_4, ו־v_5.

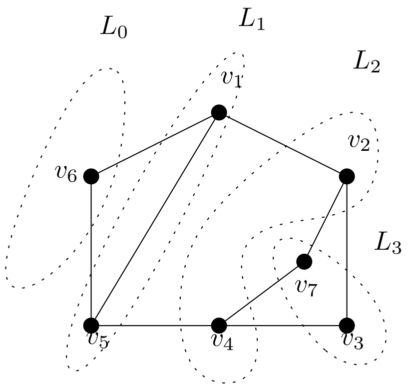
3.2 סריקה־לרוחב

קראו מתחילת סעיף 3.2 בספר, עד תחילת תת־הסעיף "סריקה־לעומק תחילה" (עמוד 90)

נסכם את אלגוריתם הסריקה־לרוחב [Breadth First Search — BFS] של גרף $G = (V, E)$ מצומת s:

> הסריקה מחלקת את הגרף לשכבות L_i. $L_0 = \{s\}$, שכבה L_1 מכילה את כל השכנים של s. L_2 מכילה את כל השכנים של צמתים ב־L_1 שאינם מופיעים ב־$L_0 \cup L_1$. בהינתן שבנינו כבר את השכבות $L_0, L_1, \ldots, L_i$, נגדיר את L_{i+1} כקבוצת הצמתים שהינם שכנים של צמתים ב־L_i ואינם מופיעים ב־$L_0 \cup \cdots \cup L_i$.

לדוגמה, איור 3.3 מתאר את השכבות בסריקה־לרוחב של הגרף שהוצג באיור 3.1, המתחילה מצומת v_6.



איור 3.3: השכבות בסריקה־לרוחב מ־v_6.

תכונות הסריקה־לרוחב מצומת s נתון:

- הקבוצה L_i היא קבוצת הצמתים במרחק i מ־s (טענה (3.3) בספר).
- כל קשת $e = \{u, v\}$ בגרף, מחברת שני צמתים משכבות עוקבות או מאותה שכבה (טענה (3.4) בספר).
- כמו־כן, אם מחברים כל צומת $u \in L_i$, $(i > 0)$, בקשת לבדיוק שכן אחד מ־L_{i-1} (חייב להיות שכן כזה, מהגדרת הסריקה־לרוחב), מתקבל עץ מושרש ב־s המהווה תת־גרף של הגרף המקורי. עץ זה נקרא "עץ BFS", ולמעשה הוא "עץ המרחקים הקצרים ביותר מ־s ב־G", כלומר לכל צומת u, המסלול בעץ מ־s ל־u הוא המסלול הקצר ביותר ב־G בין s ל־u.

תרגיל 3.8 הריצו את אלגוריתם הסריקה־לרוחב על הגרף באיור 3.1 כאשר הצומת ההתחלתי הוא v_1.

1. תארו את השכבות L_i.
2. בנו עץ סריקה־לרוחב מתאים.
3. הוסיפו לעץ את שאר הקשתות בגרף כקשתות מקווקוות.

פתרון בעמוד 46

3.3 סריקה־לעומק

המשיכו לקרוא עד סוף סעיף 3.2 בספר

שיטת סריקה נוספת של גרפים היא סריקה־לעומק – [Depth First Search DFS]. נחזור על תיאור הסריקה־לעומק בגרף קשיר $G=(V,E)$ מצומת r:

> הסריקה־לעומק בונה עץ. בצומת הנוכחי u הסריקה מחפשת שכן v של u ב־G שעדיין לא נסרק, מגדירה את v כבן של u, ועוברת ל־v. אם לא קיים שכן כזה v, הסריקה חוזרת לאב של u. אם לא קיים אב ל־u (כלומר $u=r$), הסריקה נעצרת.

התכונה החשובה של סריקה־לעומק של גרף G היא: ביחס לעץ הסריקה המתקבל, כל קשת של G (כולל קשתות העץ) מחברת שני צמתים שאחד מהם הוא אב קדמון של השני (טענה (3.7) בספר). קשתות מצאצא לאב קדמון שאינן קשתות בעץ הסריקה־לעומק נקראות קשתות אחורה.

כדי להבין טוב יותר את מבנה עץ הסריקה־לעומק, נגדיר לכל צומת u "זמן כניסה" b_u ו"זמן יציאה" f_u. זמן הכניסה b_u הוא נקודת הזמן בתחילת ביצוע $\text{DFS}(u)$, וזמן היציאה f_u היא נקודת הזמן בסיום ביצוע $\text{DFS}(u)$. באופן מדויק יותר, נשנה במקצת את שגרת הסריקה־לעומק המוצגת בסעיף 3.2 בספר כדלקמן: נגדיר משתנה גלובלי t שישמש כמונה ומאותחל ל־0, ונוסיף לשגרה רישום של זמני כניסה ויציאה במשתנים b_u ו־f_u. האלגוריתם המתקבל מתואר כאלגוריתם 3.1.

Algorithm 3.1 DFS(u)

$b_u \leftarrow t;\quad t \leftarrow t+1$
Mark u as "Explored".
for each edge $\{u,v\}$ incident to u **do**
    if v is not marked "Explored" **then**
        Recursively invoke DFS(v)
$f_u \leftarrow t;\quad t \leftarrow t+1$

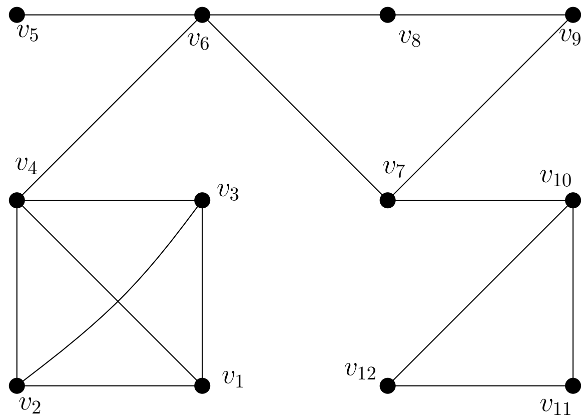
תרגיל 3.9 הריצו סריקה־לעומק על הגרף באיור 3.4, החל מצומת v_6. בנו את עץ הסריקה־לעומק וציינו עליו "קשתות אחורה". ליד כל צומת u ציינו את b_u ואת f_u.

פתרון בעמוד 46

טענה (3.5) בספר מראה שסריקה־לעומק סורקת את רכיב הקשירות של צומת ההתחלה. אפשר להרחיב את הסריקה־לעומק לכל הגרף כדלקמן: בלולאה, כל עוד קיים בגרף צומת u שלא נסרק, הריצו $\text{DFS}(u)$. התהליך הזה מריץ את שגרת $\text{DFS}(\cdot)$ על כל רכיבי הקשירות. האלגוריתם המתקבל יהיה:

Algorithm 3.2 DFS-Loop($G=(V,E)$ a graph)

Initialize all vertices to "Not-yet-Explored"
Set $t \leftarrow 0$
while there exists a "Not-Yet-Explored" node u **do**
    Call DFS(u)



איור 3.4: גרף קשיר

תרגיל 3.10 נסמן $[a,b] = \{x:\ a \leqslant x \leqslant b\}$. הוכיחו שלכל שני צמתים u ו־v מתקיים בדיוק אחד משני המקרים:

1. $[b_u, f_u] \subseteq [b_v, f_v]$ או $[b_v, f_v] \subseteq [b_u, f_u]$
2. $[b_v, f_v] \cap [b_u, f_u] = \emptyset$

הוכיחו שבמקרה הראשון היחס בין u ל־v הוא היחס צאצא/אב־קדמון.

פתרון בעמוד 46

תרגיל 3.11

1. הוכיחו שלאחר הרצת DFS-Loop נקבל:

$$\{b_u, f_u:\ u \in V\} = \{0, \ldots, 2n-1\}.$$

כלומר, אוסף זמני הכניסה והיציאה של צמתי הגרף הינו בדיוק כל הזמנים (הבדידים) בין 0 ל־$2n-1$.

2. הוכיחו כי $f_u - b_u + 1$ הוא מספר זוגי השווה לפעמיים מספר הצמתים בתת־העץ של u.

פתרון בעמוד 46

3.4 מימושים של אלגוריתמי סריקה

קראו את סעיף 3.3 בספר

במהלך כל הפרק הנוכחי, כל עוד ברור מההקשר איזה גרף הוא הגרף המדובר, אנו נסמן ב־n את מספר הצמתים בגרף וב־m את מספר הקשתות. שימו לב, מספר הקשתות הוא לכל היותר כמספר הזוגות הסדורים של n צמתים, לכן $m \leq \binom{n}{2} = O(n^2)$. גרפים דלילים הם גרפים שבהם m "קטן בהרבה" מ־n^2.

מימושי הסריקה שנראה בסעיף הנוכחי יהיו בעלי סיבוכיות $O(m+n)$, כלומר לינארים ב"גודל הטבעי" של הגרף.

כתבנו "גודל טבעי" במרכאות, מפני שזה תלוי באופן שבו מייצגים גרפים. ישנם שני ייצוגים עיקריים לגרף $G=(V,E)$.

1. **מטריצת שכנויות** [adjacency matrix]. מטריצה A ששורותיה ועמודותיה מייצגות צמתים, ואיבריה הם

$$A_{u,v} = \begin{cases} 0 & \text{if } \{u,v\} \notin E \\ 1 & \text{if } \{u,v\} \in E. \end{cases}$$

יתרונו העיקרי של ייצוג זה הוא: בהינתן זוג צמתים, מאפשר הייצוג לבדוק בזמן קבוע אם יש ביניהם קשת. חסרונותיו העיקריים הם:

- לייצוג דרוש $\Theta(n^2)$ זיכרון, גם בגרפים דלילים.
- סריקת הקשתות הסמוכות לצומת נתון נמשכת $\Theta(n)$ זמן, גם כאשר דרגת הצומת קטנה.

ראו להלן דוגמה של ייצוג על־ידי מטריצת השכנויות עבור הגרף שהובא לעיל באיור 3.1 (הכניסות הריקות מסמלות 0).

$$\left(\begin{array}{c|ccccccccccc}
 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} \\
\hline
v_1 & & 1 & & & 1 & 1 & & & & & \\
v_2 & 1 & & & & & 1 & 1 & & & & \\
v_3 & & & & 1 & & & & & & & \\
v_4 & & & 1 & & 1 & & 1 & & & & \\
v_5 & 1 & & & 1 & & 1 & & & & & \\
v_6 & 1 & 1 & & & 1 & & & & & & \\
v_7 & & 1 & & 1 & & & & & & & \\
v_8 & & & & & & & & & 1 & 1 & \\
v_9 & & & & & & & & 1 & & 1 & \\
v_{10} & & & & & & & & 1 & 1 & & 1 \\
v_{11} & & & & & & & & & & 1 &
\end{array}\right)$$

2. **רשימת שכנויות** [adjacency list]. הצמתים מוחזקים במערך, ולכל צומת יש רשימה מקושרת של שכניו בגרף (לכן רשימה מקושרת זו גם מכילה מידע על כל הקשתות הסמוכות לצומת). חסרונו העיקרי של ייצוג זה הוא: בהינתן זוג צמתים, כדי להחליט אם הם שכנים, נדרש לעיתים זמן יחסי לדרגת אחד הצמתים. היתרונות העיקריים של ייצוג זה הם:

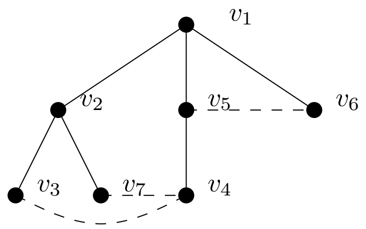
- גודל בזיכרון הדרוש הוא $\Theta(m+n)$.
- סריקת הקשתות השכנות של צומת נתון מדרגה d ניתנת לביצוע בזמן $O(d)$.

לדוגמה, ראו להלן ייצוג על־ידי רשימת שכנויות עבור הגרף שהובא באיור 3.1.

$$\begin{array}{lcllll}
v_1 & \to & (v_2, & v_5, & v_6) \\
v_2 & \to & (v_1, & v_3, & v_7) \\
v_3 & \to & (v_2, & v_4) & \\
v_4 & \to & (v_3, & v_7) & \\
v_5 & \to & (v_1, & v_4, & v_6) \\
v_6 & \to & (v_1, & v_5) & \\
v_7 & \to & (v_2, & v_4) & \\
v_8 & \to & (v_9, & v_{10}) & \\
v_9 & \to & (v_8, & v_{10}) & \\
v_{10} & \to & (v_8, & v_9, & v_{11}) \\
v_{11} & \to & (v_{10}) & &
\end{array}$$

מימוש של סריקות הגרפים (לרוחב ולעומק) הוא יעיל יותר בייצוג של רשימת שכנויות, וזה יהיה הייצוג שנבחר עבורם בפרק זה. המימוש של סריקה־לרוחב הוא פשוט בעזרת תור [queue]. סריקה־לעומק ניתנת למימוש פשוט בעזרת מחסנית [stack] או בצורה שקולה על־ידי רקורסיה (כפי שראינו בסעיף הקודם).

נבחן עתה את המימוש של סריקה־לרוחב של גרף הנתון כרשימת שכנויות בעזרת תור. כדוגמה ניקח את רכיב הקשירות השמאלי באיור 3.1 ונקרא לסריקה־לרוחב מצומת v_1. כמו כן נניח שבכל צומת, רשימת השכנויות מסודרת בסדר מילוני [לקסיקוגרפי]. עץ הסריקה המתקבל מתואר באיור 3.5.

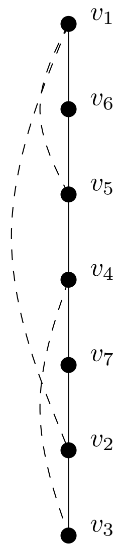


איור 3.5: עץ BFS המושרש ב־v_1 של הגרף שהובא באיור 3.1.

התור Q יתואר להלן כסדרה, שראשיתה (צידה השמאלי) הוא ראש התור. Q מתנהל כדלקמן:

1. התור מאותחל ל־$Q = (v_1)$.
2. לאחר סריקת השכנים של v_1, $Q = (v_2, v_5, v_6)$.
3. לאחר סריקת השכנים של v_2, $Q = (v_5, v_6, v_3, v_7)$.
4. לאחר סריקת השכנים של v_5, $Q = (v_6, v_3, v_7, v_4)$.
5. בשלב זה מעובדים הצמתים v_6, v_7, v_8, ו־v_4. כיוון שכל שכניהם כבר נסרקו, לא נכנסים ל־Q צמתים חדשים.

נריץ עתה את $\text{DFS}(v_1)$ על אותו גרף. עץ הסריקה־לעומק המתקבל מתואר באיור 3.6. נתאר כיצד תיראה המחסנית S של אלגוריתם הסריקה־לרוחב על הגרף הזה.



איור 3.6: עץ DFS המושרש ב־v_1 של הגרף שהובא באיור 3.1

1. תחילה המחסנית מאותחלת ל־$S = (v_1)$.
2. לאחר סריקת השכנים של v_1 מתקבל $S = (v_6, v_5, v_2)$.
3. הצומת v_6 נשלף, ובמקומו נדחף v_5, (השכן היחיד של v_6 שעדיין לא נסרק), לכן מצב המחסנית לאחר שלב זה הוא $S = (v_5, v_5, v_2)$.
4. צומת v_5 נשלף, ובמקומו נדחף שכנו v_4; מצב המחסנית $S = (v_4, v_5, v_2)$.
5. צומת v_4 נשלף, ונדחפים v_3 ו־v_7; מצב המחסנית $S = (v_7, v_3, v_5, v_2)$.
6. צומת v_7 נשלף, ונדחף v_2; מצב המחסנית $S = (v_2, v_3, v_5, v_2)$.
7. צומת v_2 נשלף, ונדחף v_3; מצב המחסנית $S = (v_3, v_3, v_5, v_2)$.
8. צומת v_3 נשלף; מצב המחסנית $S = (v_3, v_5, v_2)$
9. בשלב זה כל הצמתים נשלפו מהמחסנית לפחות פעם אחת, ולכן שאר השליפות לא מזהות צמתים חדשים.

טענה (3.13) בספר מראה שזמן הריצה של DFS(u) הוא $O(m_1)$, כאשר m_1 הוא מספר הקשתות ברכיב הקשירות של u. כדי לנתח את זמן הריצה של DFS־Loop(G) (אלגוריתם 3.2), נציע מימוש יעיל של התכונה Explored/Not-Yet-Explored: נשמור רשימה מקושרת כפולה של Not-Yet-Explored ואז נוכל למצוא צומת Not-Yet-Explored בזמן קבוע (על־ידי שליפת הצומת שבראש הרשימה) בלולאת While שבשגרה DFS-Loop. במימוש הזה האלגוריתם משקיע $O(n)$ זמן באתחול הרשימה, וזמן קבוע לכל רכיב קשירות. בסה"כ זמן הריצה על גרף בעל t רכיבי קשירות הוא:

$$O\left(n + t + \sum_{v\in V} \deg(v)\right) = O(m + n)$$

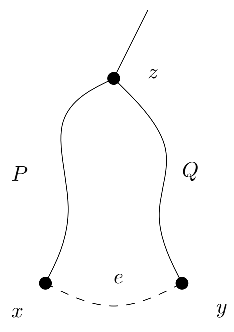
3.5 גרף דו־צדדי

קראו את סעיף 3.4 בספר

גרף דו־צדדי הוא גרף שניתן לחלק את צמתיו לשתי קבוצות כך שלא תהיה קשת שתחבר שני צמתים מאותה קבוצה. גרף דו־צדדי נקרא גם דו־צביע, כי אפשר לצבוע את צמתיו בשני צבעים כך שקצותיה של כל קשת יהיו צבועים בצבעים שונים. טענה (3.14) בספר מציעה הגדרה שקולה לגרף דו־צדדי: גרף ללא מעגלים באורך אי־זוגי. לעתים נוח יותר להשתמש באפיון זה. לדוגמה, בעזרת אִפיון זה קל לראות שעצים הם גרפים דו־צדדיים, כיוון שאינם מכילים מעגלים כלל.

סריקה־לרוחב מאפשרת לבדוק אם גרף קשיר נתון הוא דו־צדדי, ואם כן, היא מאפשרת לבנות את החלוקה המתאימה של הצמתים. טענה (3.15) בספר מוכיחה זאת. נסביר פה בקצרה את רעיון ההוכחה. נזכור כי לפי טענה (3.4) בספר, קשתות הגרף תמיד מקשרות בין צמתים בשכבות סמוכות או בין צמתים בתוך אותה שכבה של הסריקה־לרוחב. אם אין קשתות בין צמתים באותה שכבה, אז חלוקת הצמתים לשכבות זוגיות ושכבות אי־זוגיות, מהווה חלוקה לגרף דו־צדדי.

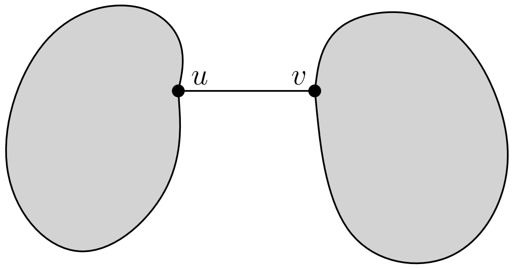
אם יש קשת $e = \{x, y\}$ בין זוג צמתים באותה שכבה, אז יש בגרף מעגל שאורכו אי־זוגי (ולכן הגרף אינו דו־צדדי) כדלקמן (ראו איור 3.7): יהי z האב־הקדמון הנמוך ביותר של x ו־y בעץ הסריקה־לרוחב T. נסמן ב־P את המסלול ב־T מ־z ל־x, וב־Q את המסלול ב־T מ־z ל־y. כיוון ש־x ו־y נמצאים באותה שכבה, אורכי P ו־Q שווים. כמו־כן, כיוון ש־z הוא האב הקדמון הנמוך ביותר, P ו־Q זרים בצמתים, למעט צומת ההתחלה z. נסמן ב־Q^{rev} את היפוך המסלול Q. מהאמור לעיל, אם משרשרים את $P \setminus \{z\}$ ל־Q^{rev}, מתקבל מסלול פשוט, באורך זוגי מ־y ל־x. בתוספת הקשת e מתקבל מעגל באורך אי־זוגי.



איור 3.7: עץ סריקה־לרוחב בגרף המכיל מעגל אי־זוגי.

תרגיל 3.12 הוכיחו כי בגרף דו־צדדי קשיר קיימת חלוקה ל"צדדים" (מחלקות צבע) אחת בלבד. האם טענה זו נכונה גם בגרף דו־צדדי לא קשיר? הסבירו.

פתרון בעמוד 47



איור 3.8: תיאור סכמטי של גשר בגרף

3.6 דו־קשירות בקשתות

בסעיף זה נראה כיצד משתמשים בסריקה־לעומק לצורך זיהוי "גשרים" בגרף. נגדיר:

הגדרה 3.1 גשר [bridge] הוא קשת $e = \{u, v\}$ שהסרתה מהגרף מנתקת את הקשירות בין u ל־v. $\diamondsuit$

ברשתות תקשורת, לדוגמה, גשרים הם נקודות־כשל פוטנציאליות: נפילת גשר משמעותה ניתוק הרשת. לעומת זאת נפילת חיבור (קשת) שאינו גשר אינה מנתקת את הקשירות (אך יכולה לפגוע בביצועי הרשת).

באופן סכמטי, גשר $e = \{u, v\}$ מחלק את צומתי הגרף לשני חלקים לא ריקים; האחד מכיל את u והשני את v, ובין חלקים אלה עוברת קשת יחידה – הגשר e, ראו איור 3.8. נחזור לאיור 3.4, ונציין בו את כל הקשתות שהן גשרים בגרף: $\{v_5, v_6\}$, $\{v_4, v_6\}$, $\{v_7, v_{10}\}$.

התרגיל הבא מתאר אפיון נוסף של גשרים.

תרגיל 3.13 הוכיחו שקשת היא גשר אם ורק אם אין מעגל המכיל אותה.

פתרון בעמוד 47

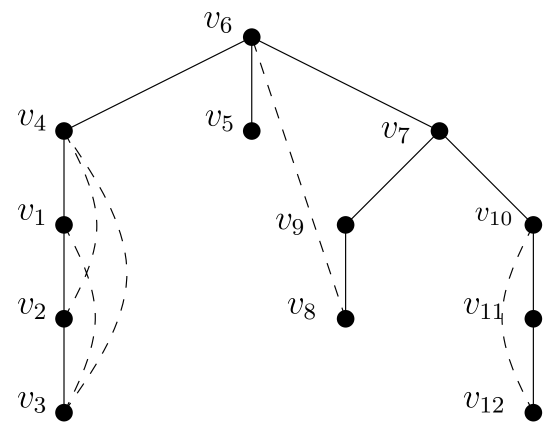
לשם הפשטות נתמקד עתה בפיתוח אלגוריתם לזיהוי גשרים בגרף קשיר – קל להרחיב את האמור להלן לגרפים לא קשירים. אלגוריתם פשוט יסרוק את קשתות הגרף, ועבור כל $e \in E$, יסיר אותה ויריץ סריקה על הגרף הנוצר כדי לבדוק אם הוא קשיר. סיבוכיות זמן הריצה של אלגוריתם זה היא $O(m^2)$. בסעיף הנוכחי אנו נפתח אלגוריתם המזהה גשרים בגרפים קשירים בזמן $O(m)$.

נתבונן בעץ הסריקה־לעומק של הגרף שהוצג באיור 3.4 כפי שהוא מתואר כעת באיור 3.9. נשים לב שבדוגמה הזו כל הגשרים – $\{v_5, v_6\}$, $\{v_4, v_6\}$, $\{v_7, v_{10}\}$ – הם קשתות של עץ שעבורן אין קשת אחורה מן הצאצא של הקשת אל האב הקדמון של הקשת. זו אינה מקריות, אלא אפיון של גשרים, כפי שנוכיח עתה.

טענה 3.3 יהי $G = (V, E)$ גרף קשיר ו־T עץ סריקה־לעומק שלו. קשת $e \in E$ היא גשר אם ורק אם $e = \{x, y\}$ היא קשת עץ ב־T, כך שאם־x אב של y ב־T אזי לא קיימת קשת אחורה המחברת צאצא (ב־T) של v עם אב קדמון (ב־T) של u.

הוכחה. נחלק את קשתות הגרף לשלושה סוגים (ראו גם איור 3.10):

A – קשתות שאינן נמצאות ב־T;



איור 3.9: עץ סריקה־לעומק מ־v_6 וקשתות אחורה של הגרף מאיור 3.4. קשתות העץ שאין קשת אחורה מצאצא שלהן לאב קדמון ממש שלהן (כדוגמת $\{v_4, v_6\}$) הן הגשרים בגרף.

B – קשתות שנמצאות ב־T אך קיימת קשת(ות) מן הצאצא אל האב הקדמון שלהן;

C – קשתות שנמצאות ב־T אך אין קשת מן הצאצא אל האב הקדמון שלהן.

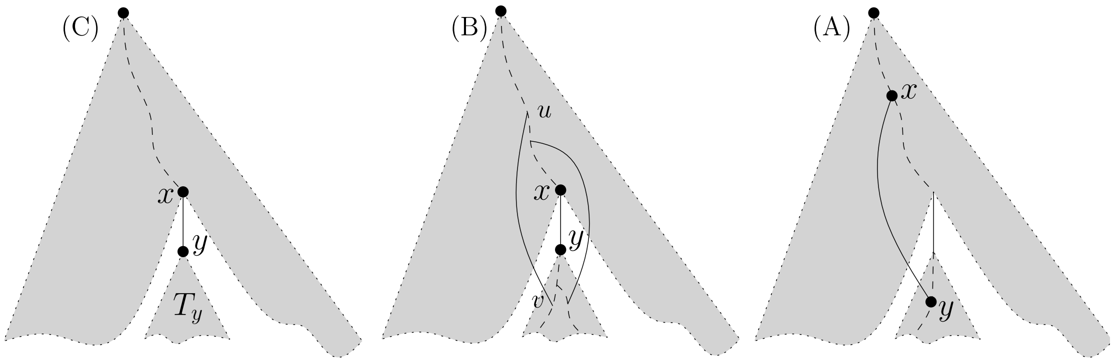
נבחן כל מקרה לגופו.

נבחן תחילה את הקשת $e = \{x, y\} \in E$ שאינה קשת ב־T (מקרה A, ראו גם איור 3.10). כיוון ש־T קשיר, קיים מסלול פשוט ב־T המחבר את x ל־y, ולכן הקשת e נמצאת על מעגל ב־G. מתרגיל 3.13 אנו למדים ש־e אינה גשר.

נבחן כעת קשת בעץ הסריקה־לעומק $e = \{x, y\} \in T$. בלי הגבלת הכלליות נניח ש־x הוא אב של y בעץ T. תחילה נניח שקיימת קשת אחורה $f = \{u, v\}$ מן הצאצא v של y לאב הקדמון u של x (מקרה B, ראו גם איור 3.10). קשת זו מגדירה מעגל ב־G: נתחיל ב־u ונרד דרך קשתות T לעבר v דרך הקשת e. בהגיענו לצומת v נחזור ל־u דרך הקשת f. התקבל מעגל ב־G המכיל את הקשת e ולכן e אינה גשר.

נותר המקרה האחרון (מקרה C, ראו גם איור 3.10): $e = \{x, y\} \in T$, היא קשת עץ ולא קיימת קשת אחורה מן הצאצא של y לאב הקדמון של x. מטענה (3.7) בספר אנו יודעים כי בהינתן עץ סריקה־לעומק T של גרף G, קשתות G נחלקות לשניים: קשתות עץ וקשתות (אחורה) מצאצא לאב קדמון. נתבונן בגרף $T \setminus \{e\}$. בגרף הזה x ו־y אינם קשירים. נסמן ב־T_y את העץ ששורשו y ב־T. נוסיף עתה לגרף $T \setminus \{e\}$ את שאר הקשתות ב־$E \setminus \{e\}$. כיוון שקשת העץ היחידה המחברת את T_y אל מחוץ ל־T_y היא e, כל קשת אחרת חייבת להיות קשת אחורה מן הצאצא של y לאב הקדמון של x, אך הנחנו שאין קשת כזו. לפיכך T_y מנותק בגרף $(V, E \setminus \{e\})$ משאר הצמתים, ובפרט מ־x. משמעות הדבר ש־e היא גשר.

שימו לב, בזאת הוכחנו את הטענה: אם e היא קשת עץ ואין קשת מצאצא שלה לאב הקדמון שלה, אז זה מתאים למקרה C לעיל ולכן היא גשר. בכיוון ההפוך, אם e היא גשר אז לפי הניתוח לעיל היא אינה יכולה להיות אחד מהמקרים A או B. כיוון שחילקנו את כל הקשתות לשלושה סוגים, היא חייבת להיות מסוג C, כלומר



איור 3.10: שלושה סוגי קשתות בגרף: (A) הקשת $\{x,y\}$ אינה קשת בעץ הסריקה־לעומק. במקרה זה היא חלק ממעגל המבוסס על המסלול בעץ בין x ל־y. (B) הקשת $\{x,y\}$ היא קשת עץ הסריקה־לעומק, אך יש קשת(ות) מצאצא של y לאב קדמון של x ולכן היא חלק ממעגל. (C) הקשת $\{x,y\}$ היא קשת עץ הסריקה־לעומק ואין קשת מצאצא של y לאב קדמון של x למעט $\{x,y\}$. במקרה זה $\{x,y\}$ היא גשר.

קשת העץ שעבורה אין קשת מן הצאצא שלה לאב הקדמון שלה. □

טענה 3.3 היא הבסיס לאלגוריתם מציאת כל הגשרים בזמן $O(m)$. נקבע גרף $G=(V,E)$ ועץ סריקה־לעומק T. נקבע צומת y (שאינו השורש), ונסמן ב־x את ההורה של y ב־T. נרצה לדעת אם יש קשת שאינה קשת עץ מן הצאצא של y לאב הקדמון של x. אם־כן, נדע שקשת העץ $\{x,y\}$ אינה גשר, ואם לא, אז $\{x,y\}$ היא גשר.

כזכור, באלגוריתם 3.1 (בסעיף 3.3) הגדרנו לכל צומת $w \in V$ את זמן הכניסה בסריקה־לעומק b_w. מתרגיל 3.10 אנו יודעים ש־w הוא אב הקדמון של u אם ורק אם $b_w < b_u < f_u < f_w$. לכן עבור y נתון נוכל לחשב את הערך b_w הקטן ביותר עבור קשת $\{v,w\}$ שאינה קשת עץ ושעבורה v הוא הצאצא של y. במלים אחרות, לכל צומת y (שאינו שורש) נחשב את הערך הזה:

$$L_y = \min\{b_w :\ \exists(v,w) \in E \setminus T,\ \text{and } v \text{ is a descendant of } y\}, \tag{3.1}$$

לפי המוסכמה ש־$\min\emptyset = \infty$.

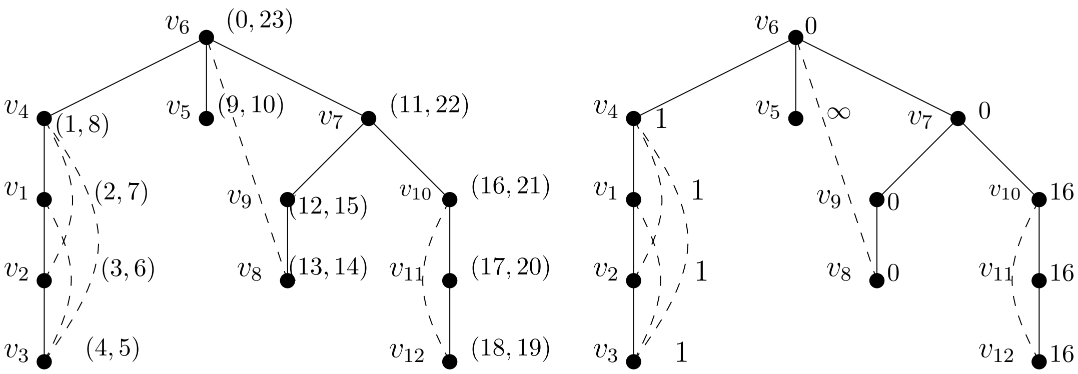
כדי להמחיש מהו L_u עבור צומת u, התבוננו באיור 3.11 שבו תוכלו לראות את עץ הסריקה לעומק של הגרף מאיור 3.4, בתוספת ערכי הכניסה והיציאה (b_u, f_u) מצד שמאל, וערכי L_u מצד ימין. הערך L_u מתקבל ציורית מהפעולה הבאה: ננסה להגיע הכי "גבוה שניתן" ע"י הילוך כלפי מטה בעץ מהצומת u, ואז "קפיצה" אחת כלפי מעלה בעזרת קשת אחורה. לדוגמה, באיור 3.11, $L_{v_1} = 1$ כיוון שהכי גבוה שתהליך זה מאפשר הוא ירידה מ־v_1 ל־v_2 (או v_3) ואז קפיצה ל־v_4, וזמן הכניסה ל־v_4 הוא 1. דוגמה אחרת: $L_{v_7} = 0$, כיוון שניתן לרדת מ־v_7 ל־v_8, ומשם לקפוץ בקשת אחורה ל־v_6 (השורש), שזמן הכניסה אליו הוא 0.

L_y הוא זמן הכניסה לצומת w (אם קיימת כזאת) הגבוה ביותר שניתן להגיע אליו בעזרת קשת אחורה היוצאת מצאצא של y (כולל y עצמו). כפי שראינו

בטענה 3.3, קשת העץ $\{x,y\}$ איננה גשר אם ורק אם הצומת w שלעיל הינו אב קדמון של x (האב של y). כיוון שקל לבדוק יחס אב־קדמון בעזרת זמני הכניסה, אנו משתמשים בהם ובודקים האם $b_w < b_y$, כלומר האם $L_y < b_y$.

נסכם את הדיון שלעיל בטענה הבאה:

טענה 3.4 יהי T עץ סריקה־לעומק של גרף G. קשת העץ $\{x,y\} \in T$, שבה x הוא האב של y (ב־T) היא גשר ב־G אם ורק אם $L_y \geq b_y$.



איור 3.11: עץ הסריקה־לעומק שהוצג באיור 3.9; בעץ שבצד שמאל מצוינים זמני כניסה ויציאה (b_u, f_u), ובעץ שבצד ימין מצוינים ערכי L_u.

ראינו כבר כיצד לחשב את b_y, לכן נותר כעת לראות כיצד לחשב את L_y. תרגום ישיר של הנוסחה (3.1) לאלגוריתם יתן זמן ריצה $O(nm)$, אך ניתן לשפר זאת בעזרת ההבחנה הבאה

טענה 3.5 נגדיר רקורסיבית את הערך $\bar{L}_u$, כדלקמן:

$$\bar{L}_u = \min\Big\{\min\{b_w : \{u,w\} \in E \setminus T\} \ , \ \min\{\bar{L}_v : v \text{ is a child of } u\}\Big\}.$$

לכל צומת u מתקיים $L_u = \bar{L}_u$.

תרגיל 3.14 הוכיחו את טענה 3.5.

פתרון בעמוד 47

להלן פרוצדורה רקורסיבית לחישוב L_u המבוססת על טענה 3.5. אנו מניחים ש־$G=(V,E)$, T, ו־$(b_u)_{u\in V}$ נתונים.

Algorithm 3.3 Calc-$L(u)$

$L_u \leftarrow \infty$
for every child v of u in T **do**
  Call recursively to Calc-$L(v)$
  $L_u \leftarrow \min\{L_u, L_v\}$
for every $\{u,w\} \in E \setminus T$ **do**
  $L_u \leftarrow \min\{L_u, b_w\}$

נדגים את הפעולה של אלגוריתם 3.3 על הסריקה־לעומק מאיור 3.9. התבוננו שוב באיור 3.11. חישוב $(L_u)_{u\in V}$ מתחיל מקריאה ל־Calc-$L(v_6)$. שגרה זו קוראת רקורסיבית ל־Calc-$L(v_4)$, Calc-$L(v_5)$, Calc-$L(v_7)$. בסיום הקריאות הרקורסיביות התקבלו הערכים $L_{v_4}=1$, $L_{v_5}=\infty$, $L_{v_7}=0$. המינימום של ערכים אלה הוא 0. כמו־כן נבדקות קשתות היוצאות מ־v_6 שאינן קשתות עץ. ישנה רק קשת אחת כזו, $\{v_6,v_8\}$ ולכן מחושב המינימום של $\{b_{v_8}\}$ שהוא 13. המינימום על כל הערכים הללו הוא 0, וזה הערך שמקבל L_{v_6}.

תרגיל 3.15 הוכיחו את נכונות האלגוריתם Calc-L והראו שזמן הריצה שלו הוא $O(m)$.

פתרון בעמוד 48

Algorithm 3.4 Bridges($G=(V,E)$)

Require: G is a connected graph
  Let $r\in V$ be an arbitrary vertex in G.
  Call DFS(r).
  Let T be the resulting DFS tree.
  For $u\neq r$, denote by $p(u)$ the parent of u in T.
  Call Calc-$L(r)$
  for every $u\in V$, $u\neq r$ **do**
    if $L_u<b_u$ **then**
      print the edge $\{p(u),u\}$

נסכם: אלגוריתם 3.4 מוצא בזמן $O(m)$ את הגשרים בגרף קשיר נתון G. נכונות אלגוריתם 3.4 נובעת מיידית מטענה 3.4 וממה שהוכחנו בתרגיל 3.15. זמן הריצה שלו מורכב משלושה חלקים: א. הזמן הדרוש לביצוע סריקה־לעומק שהוא $O(m)$; ב. הזמן הדרוש לביצוע אלגוריתם 3.3 שהוא $O(m)$; ג. הזמן הדרוש לביצוע לולאה על הצמתים שהוא $O(n)$. בסה"כ, $O(m)$.

רכיבי דו־קשירות. נתון גרף קשיר $G=(V,E)$. זוג צמתים ב־G ייקרא "דו־קשיר בקשתות" אם קיים ביניהם מסלול שאינו מכיל גשר.

תרגיל 3.16 הוכיחו כי:

1. דו־קשירות בקשתות היא יחס של שקילות על הצמתים. (מחלקות השקילות נקראות רכיבי דו־קשירות בקשתות).
2. כאשר B היא קבוצת הגשרים של G, רכיבי הדו־קשירות בקשתות הם רכיבי הקשירות בגרף $(V,E\setminus B)$.

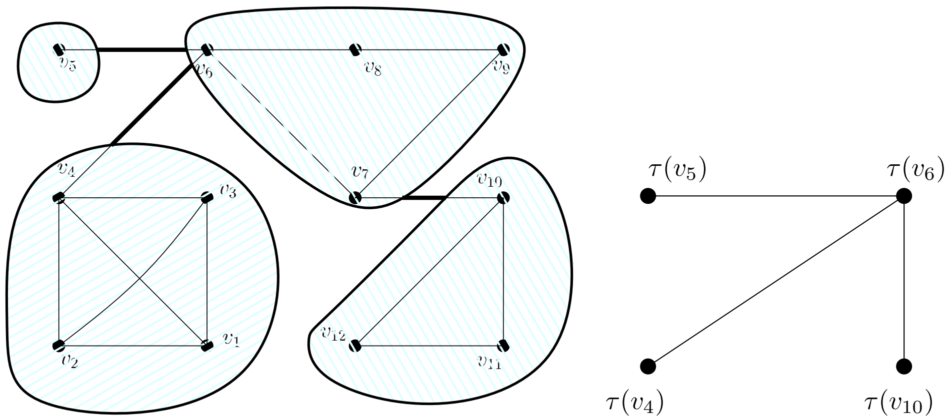
פתרון בעמוד 48

כפי שהזכרנו בתחילת הסעיף, גשרים הם קשתות בעלות תכונה מיוחדת: נפילה של קשת כזו מנתקת את הגרף. רכיבי דו־קשירות הם חלקים בגרף שקשירותם עמידה בפני נפילה של קשת אחת: אם הקשת הנופלת היא גשר, אזי, כפי שהוכחנו בתרגיל הקודם, רכיב הדו־קשירות נשאר קשיר. אם הקשת אינה גשר, אזי לפי ההגדרה, כל הגרף נשאר קשיר ובפרט הרכיב הדו־קשיר.

תרגיל 3.17 תארו אלגוריתם שבזמן $O(m)$ מחשב את רכיבי דו־הקשירות בקשתות של גרף קשיר. הדרכה: השתמשו באפיון רכיבי דו־קשירות שהוכחנו בתרגיל הקודם, ובאלגוריתם לחישוב גשרים.

פתרון בעמוד 48

עץ הגשרים של גרף קשיר נתון $G = (V, E)$ הוא גרף $H = (W, F)$ שבו צומתי W הם רכיבי הדו־קשירות של G, ובין זוג צמתים ב־W מחברת קשת, אם קיימת קשת (בהכרח גשר) אשר מחברת זוג צמתים ברכיבי הדו־קשירות המתאימים ב־G. לדוגמה, באיור 3.12 מוצג עץ הגשרים של הגרף שתואר באיור 3.4.



איור 3.12: עץ הגשרים של הגרף מאיור 3.4. כל צומת בעץ הגשרים מתאים לרכיב קשירות בגרף המקורי אחרי שמוציאים ממנו את הגשרים. הקשתות בעץ הגשרים מתאימות לגשרים בגרף המקורי. מימין, תאור סכמתי של עץ הגשרים.

פורמלית, יוגדר הגרף $H = (W, F)$ כדלקמן: תהי $B \subseteq E$ קבוצת הגשרים של G. צומתי H יהיו רכיבי הדו־קשירות של $(V, E \setminus B)$. נגדיר העתקה $\tau : V \to W$ בין צומתי G לרכיבי הדו־קשירות של G, כך ש־$\tau(u)$ הוא רכיב הדו־קשירות של u. כעת נגדיר את הקשתות של H:

$$F = \{\, \{\tau(u), \tau(v)\} : \ \{u, v\} \in B\}.$$

H נקרא עץ הגשרים של G.

תרגיל 3.18 הוכיחו שעץ הגשרים של גרף קשיר נתון הוא אכן עץ.

פתרון בעמוד 48

3.7 סריקות בגרפים מכוונים

בגרפים שדנו בהם עד כה, הקשתות היו זוג לא סדור $\{u, v\}$. גרפים אלה נקראים גם גרפים לא־מכוונים. בסעיף הזה נרחיב את הדיון לגרפים מכוונים – גרפים שהקשתות מהוות בהם זוג סדור (u, v). בגרפים האלה, לכל קשת יש כיוון, במקרה לעיל מ־u ל־v. כאשר הגרף מכוון – המושגים בהם דנו בסעיף הקודם,

משתנים במקצת. בסעיף הזה נדון בתכונות של גרפים מכוונים ובסריקות של גרפים מכוונים.

נפתח במילון מושגים לגרפים מכוונים:

גרף מכוון [**Directed Graph**] הוא זוג $G = (V, E)$ שבו:

- V היא קבוצה סופית של איברים הנקראים צמתים [nodes] או קודקודים [vertices].
- E היא קבוצה של זוגות סדורים בתוך V הנקראים קשתות [edges].

באיור מקובל לתאר גרף מכוון במישור באופן הזה: צמתים מיוצגים על־ידי נקודות או עיגולים, וקשת (u, v) מיוצגת על־ידי קו המחבר את u ל־v עם חץ לכיוון v. לדוגמה, באיור 3.13 מצויר הגרף $G = (V, E)$ שבו

$$V = \{v_1, \ldots, v_{10}\}$$
$$E = \left\{ \begin{array}{c} (v_1, v_7), (v_2, v_3), (v_3, v_4), (v_3, v_6), (v_4, v_1), \\ (v_4, v_5), (v_5, v_2), (v_6, v_1), (v_7, v_6), \\ (v_7, v_8), (v_8, v_{10}), (v_9, v_8), (v_9, v_{10}) \end{array} \right\}$$

דרגת כניסה של צומת v היא מספר הקשתות הנכנסות ל־v.

דרגת יציאה של צומת v היא מספר הקשתות היוצאות מ־v.

מסלול. מסלול בגרף הוא סדרה של צמתים (לפחות צומת אחד) כך שבין כל שני צמתים עוקבים בסדרה קיימת קשת מכוונת בגרף. שימו לב, בגרף מכוון ייתכן שקיים מסלול מ־u ל־v, אך לא בכיוון ההפוך.

מסלול מעגלי (או בקיצור מעגל) בגרף מכוון הוא מסלול מכוון שהצומת הראשון והאחרון שלו זהים.

גרף מכוון ללא מעגלים (גמ"ל) [Directed acyclic graph (DAG)] כשמו כן הוא – גרף מכוון ללא מסלולים מעגליים.

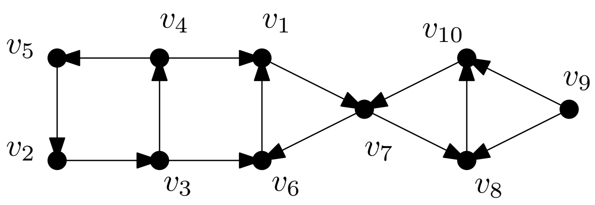
צמתים נגישים הדדית [**strongly connected**]. הצמתים u ו־v נקראים נגישים הדדית אם קיימים בגרף מסלול מ־u ל־v ומסלול מ־v ל־u.

סריקה־לרוחב. נגדיר כעת סריקה־לרוחב בגרף מכוון. הסריקה דומה לסריקה־לרוחב בגרף לא־מכוון אך מתחשבת בכיווני הקשתות. ביתר פירוט: $L_0 = s$; בהנחה שהגדרנו את השכבות $L_1, \ldots, L_j$, השכבה L_{j+1} מכילה צמתים שלא הופיעו בשכבה קודמת אך קיימות קשתות המכוונות אליהם מצמתים הנמצאים בשכבה L_j. שימו לב, בגרף מכוון ייתכנו קשתות משכבה j לשכבה i כאשר $i < j - 1$.

תרגיל 3.19 הריצו סריקה־לרוחב מצומת v_4 על הגרף באיור 3.13.

פתרון בעמוד 49

תרגיל 3.20 תהיינה $(L_i)_i$ השכבות שנוצרו בסריקה־לרוחב של גרף מכוון.



איור 3.13: גרף מכוון

1. אם קיימת קשת מצומת ב־L_j לצומת ב־L_i הוכיחו כי $i \leq j+1$.
2. תנו דוגמה לגרף מכוון על n צמתים שבסריקה־לרוחב שלו יש קשת מ־L_{n-1} ל־L_0.

פתרון בעמוד 49

סריקה־לעומק. נכתוב את אלגוריתם הסריקה־לעומק עבור גרפים מכוונים, בתוספת זמני כניסה וזמני יציאה. הסריקה־לעומק מתוארת באלגוריתם 3.5 ובאלגוריתם 3.6.

Algorithm 3.5 dDFS-Loop(G)

Require: G is a directed graph
  Initialize all vertices to "Not-yet-Explored"
  $t \leftarrow 0$
  while there exists a "Not-Yet-Explored" vertex u **do**
    Call dDFS(u)

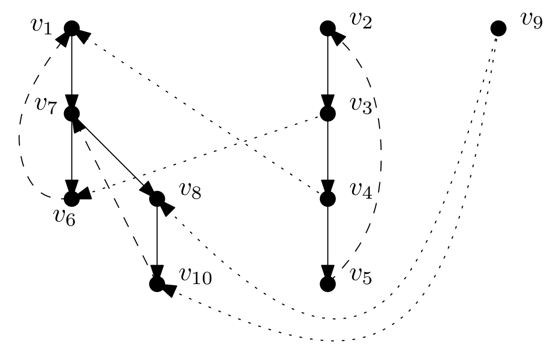
Algorithm 3.6 dDFS(u)

  $b_u \leftarrow t; \quad t \leftarrow t+1$
  Mark u as "Explored"
  for each directed edge (u, v) incident to u **do**
    if v is not marked "Explored" **then**
      Recursively invoke dDFS(v)
  $f_u \leftarrow t; \quad t \leftarrow t+1$

אלגוריתם 3.5 ואלגוריתם 3.6 דומים לאלגוריתמי סריקה־לעומק בגרפים לא־מכוונים (אלגוריתם 3.2 ואלגוריתם 3.1 בהתאמה), אך כעת הקשתות מכוונות. נריץ את אלגוריתם 3.5 על הגרף באיור 3.13, בהנחה שלולאת While באלגוריתם 3.5 סורקת את הצמתים בסדר מילוני. פלט הסריקה הוא אוסף עצים מושרשים המתוארים באיור 3.14.

תרגיל 3.21 בדקו שהטענות שהוכחנו בתרגיל 3.11 עודן נכונות בגרפים מכוונים.

שימו לב, בסריקה־לעומק של גרף מכוון יש ארבעה סוגי קשתות בגרף:



איור 3.14: העצים המושרשים ושאר קשתות הגרף לאחר סריקה־לעומק של הגרף המכוון מאיור 3.13. קו מלא מייצג קשתות עץ, קו מקווקוו מייצג קשתות אחורה, וקו מנוקד מייצג קשתות חוצות.

קשתות עץ – קשתות (u, v) שנסרקו בזמן הקריאה ל־$\mathrm{dDFS}(u)$, ובזמן הזה v לא היה מסומן כ־"Explored".

קשתות קדימה – קשתות מאב־קדמון לצאצא שאינן קשתות עץ (אינן מופיעות באיור 3.14).

קשתות אחורה – קשתות מצאצא לאב־קדמון (לדוגמה הקשתות המקווקוות באיור 3.14).

קשתות חוצות – כל השאר (לדוגמה הקשתות המנוקדות באיור 3.14).

הקשתות בשלושת הסוגים הראשונים הן קשתות שצורתן (u, v) כאשר u צאצא של v או להפך, כלומר $b_u < b_v < f_v < f_u$ או $b_v < b_u < f_u < f_v$. בקשתות מהסוג האחרון (קשתות חוצות) מתקיים בהכרח $[b_u, f_u] \cap [b_v, f_v] = \emptyset$.

טענה 3.6 אם $(u, v) \in E$ היא קשת בגרף המכוון ו־$[b_u, f_u] \cap [b_v, f_v] = \emptyset$, אז $b_v < f_v < b_u < f_u$.

הוכחה. כיוון ש־$[b_u, f_u] \cap [b_v, f_v] = \emptyset$, מספיק להוכיח את אי־השוויון האמצעי. נניח בשלילה ש־$b_u < f_v$. התנאי על אי־חיתוך המקטעים מחייב ש־ $f_u < b_v$. כלומר הצומת u נסרק לפני הצומת v וסריקתו הסתיימה לפני תחילת הסריקה של הצומת v. מזה אנו מסיקים שבסיום ביצוע השיגרה $\mathrm{dDFS}(u)$ הצומת v עדיין איננו במצב "Explored". אך זו סתירה לפעולת הסריקה־לעומק. במהלך ביצוע קריאה ל־$\mathrm{dDFS}(u)$ (אלגוריתם 3.6) תתבצע בהכרח קריאה ל־$\mathrm{dDFS}(v)$ כי (u, v) היא קשת בגרף, ובזמן שמתחילים את הביצוע של $\mathrm{dDFS}(u)$ הצומת v אינו במצב "Explored". מכאן שהצומת v בהכרח עוברת למצב "Explored" במהלך ביצוע $\mathrm{dDFS}(u)$. $\square$

תרגיל 3.22 הציגו דוגמה של גרף מכוון וסריקה לעומק שלו המכילה קשתות קדימה.

פתרון בעמוד 50

3.8 קשירות ומיון טופולוגי בגרפים מכוונים

קראו בספר סעיפים 3.5 ו־3.6

כפי שהגדרנו בתחילת סעיף 3.7, שני צמתים בגרף מכוון ייקראו נגישים הדדית אם קיים מסלול מן הראשון לשני, ולהפך. היחס "נגישים הדדית" הינו יחס שקילות על הצמתים [ראו טענה (3.17) בספר]. מחלקות השקילות נקראות רכיבי קשירות חזקה.

תרגיל 3.23 מצאו את רכיבי הקשירות החזקה בגרף המכוון באיור 3.13.

פתרון בעמוד 50

גרף מכוון נקרא קשיר היטב אם כל זוג צמתים בו נגישים הדדית, או בניסוח שקול, יש בו רק רכיב קשירות חזקה יחיד המכיל את כל צמתי הגרף. אפשר לבדוק אם גרף מכוון G הוא קשיר היטב בזמן לינארי על־ידי הרצת סריקה (לרוחב או לעומק) על הגרף, מצומת כלשהו s, והרצת סריקה מצומת s על הגרף G^{rev} שבו כיווני הקשתות הפוכים. אם בשתי הסריקות מתקבלים כל צומתי G, אזי G קשיר היטב. הסיבה לכך היא: אם u מופיע בסריקה של G מ־s, זה אומר שיש מסלול מ־s ל־u ב־G, ואם u מופיע בסריקה של G^{rev} מ־s, משמעות הדבר שיש מסלול מ־s ל־u ב־G^{rev}, כלומר, יש מסלול מ־u ל־s ב־G. לכן u נגיש הדדית ל־s. אם זה נכון לכל צומת u בגרף, אזי כל הצמתים נגישים ל־s, ומשמעות הדבר שהגרף קשיר היטב. בכיוון השני, אם G קשיר היטב, אפשר להסיק שיש מסלול מ־s לכל צומת אחר ב־G, ולכן בסריקת G מ־s יופיעו כל צומתי G. כמו־כן יש מסלול מכל צומת אחר ל־s ב־G, כלומר יש מסלול מ־s לכל צומת אחר ב־G^{rev}, ולכן כל הצמתים יופיעו בסריקה של G^{rev} המתחילה ב־s.

בעיה קשה יותר מבעיית ההחלטה האם גרף מכוון הוא קשיר היטב, היא בעיית מציאת רכיבי קשירות חזקה בגרף מכוון נתון. ישנו אלגוריתם פשוט אך "חכם" אשר מבצע פעולה זאת בזמן לינארי, אך הוא איננו חלק מחומר הלימוד של קורס זה.

מיון טופולוגי של צומתי גרף מכוון $G = (V, E)$, פירושו השתת סדר מלא על $V = \{v_1, \ldots, v_n\}$ כך שאם $(v_i, v_j) \in E$ אז $i < j$. בספר מוכח שבגרף מכוון G שלוש התכונות הבאות הן שקולות:

(א) G הוא גמ"ל;

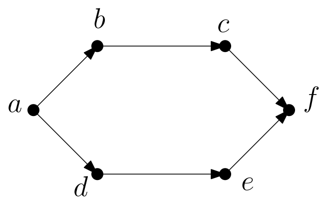
(ב) בכל תת־גרף של G יש צומת ללא קשת נכנסת;

(ג) קיים מיון טופולוגי של G.

תכונה (ב) לעיל מהווה בסיס לאלגוריתם לזיהוי גמ"ל ולמציאת מיון טופולוגי של גמ"ל: יש להתחיל מהגרף הנתון: כל עוד הגרף אינו ריק, חפשו צומת ללא קשת נכנסת. אם לא קיים צומת כזה, הגרף אינו גמ"ל. אם קיים צומת כזה, הסירו אותו וחִזרו על התהליך. אם התהליך הסתיים ויש בו גרף ריק, אזי הגרף המקורי הוא גמ"ל וסדר הסרת הצמתים הוא מיון טופולוגי. אפשר לממש את האלגוריתם הזה בזמן $O(m+n)$ על־ידי תחזוקה של רשימת הצמתים ללא קשת נכנסת, ותחזוקה של דרגת הכניסה בכל צומת בגרף הנוכחי, כפי שמפורט בסעיף 3.6 בספר.

תרגיל 3.24 הוכיחו שקצותיה של קשת (u, v) בגרף מכוון נמצאים באותו רכיב קשירות חזקה אם ורק אם (u, v) נמצאת על מסלול מעגלי.

תרגיל 3.25 תארו את כל המיונים הטופולוגיים השונים של הגרף הזה (איור 3.10 בספר):



פתרון בעמוד 50

נציג כעת אלגוריתם שונה לזיהוי גרף מכוון ללא מעגלים ולמציאת מיון טופולוגי בגמ"ל המבוסס על סריקה־לעומק. ניזכר בתכונות הסריקה־לעומק בגרף מכוון. אם קיימת קשת אחורה מצאצא u לאב קדמון v, אזי (u,v) בצירוף המסלול בעץ הסריקה־לעומק מ־v ל־u מהווה מעגל בגרף המכוון. אם לא קיימת קשת אחורה, נשים לב שקשתות הגרף הן קשתות עץ, קשתות מאב קדמון לצאצא, וקשתות חוצות (u,v) וכל אלה מקיימות $f_u > f_v$, (ראו טענה 3.6) , לכן במצב זה, סידור הצמתים בסדר f_v יורד מהווה מיון טופולוגי.

תרגיל 3.26 תארו אלגוריתם לזיהוי גרף מכוון ללא מעגלים וחישוב מיון טופולוגי בזמן $O(m+n)$ המבוסס על סריקה־לעומק ועל האבחנה שהובאה בפסקה לעיל.

פתרון בעמוד 50

תרגיל 3.27 יהי $G=(V,E)$ גרף מכוון. נגדיר גרף $H=(W,F)$ שבו צומתי H הם רכיבי קשירות חזקה של G. וקשתות H מוגדרות כדלקמן:

$$F = \{(a,b) \in W \times W :\ a \neq b,\ \exists u \in a,\ v \in b \text{ such that } (u,v) \in E\}.$$

הוכיחו ש־H הוא גרף מכוון ללא מעגלים.

פתרון בעמוד 52

3.9 פתרונות לתרגילים

פתרון תרגיל 3.1 מעמוד 23

1. 3.

2. v_6, v_5, v_2.

3. התת־גרף הנפרש על הצמתים $\{v_1,\ldots,v_7\}$, כלומר,

$$\left(\{v_1,\ldots,v_7\},\left\{\begin{array}{c}\{v_1,v_2\},\{v_1,v_5\},\{v_1,v_6\},\{v_2,v_3\},\{v_2,v_7\},\\ \{v_3,v_4\},\{v_4,v_5\},\{v_4,v_7\},\{v_5,v_6\},\end{array}\right\}\right)$$

4. תת־הגרף

$$(\{v_8,\ldots,v_{11}\},\{\{v_8,v_9\},\{v_8,v_{10}\},\{v_9,v_{10}\}\{v_{10},v_{11}\}\})$$

מכיל 4 קשתות.

5. יש שלושה מסלולים: (v_1,v_6,v_5), (v_1,v_5), (v_1,v_2,v_7,v_4,v_5). נשים לב שלא יכולים להיות יותר משלושה מסלולים זרים בקשתות, המתחילים מ־v_1, כי הדרגה של v_1 היא 3.

6. מעגל באורך 4: (v_2,v_3,v_4,v_7,v_2); מעגל באורך 5: $(v_1,v_2,v_7,v_4,v_5,v_1)$; מעגל באורך 6: $(v_1,v_2,v_7,v_4,v_5,v_6,v_1)$. ♣

:

פתרון תרגיל 3.2 מעמוד 23

יהי P מסלול קצר ביותר בין u ל־v, כלומר אורכו של P הינו $d(u,v)$. יהי Q המסלול הקצר ביותר בין v ל־w. כיוון שהצומת האחרון ב־P הוא v, שהינו הצומת הראשון ב־Q, ניתן לשרשר את שני המסלולים (תוך השמטת עותק אחד של v) ולקבל מסלול R מ־u ל־w שאורכו שווה לסכום אורכי P ו־Q, כלומר, $d(u,v)+d(v,w)$. כיוון ש־R הינו מסלול מ־u ל־v אורכו חוסם מלמעלה את המרחק מ־u ל־w, כנדרש. ♣

פתרון תרגיל 3.3 מעמוד 23

נניח שבין u ל־v קיים מסלול $u=u_0,u_1,\ldots,u_l=v$ ונבצע את תהליך המחיקה הזה: כל עוד קיימים $0\le i<j\le l$ כך ש $u_i=u_j$ נוכל למחוק את תת־המסלול $u_{i+1},\ldots,u_j$. לאחר מחיקת תת־המסלול, הסדרה הנותרת תהיה גם היא מסלול מ־u ל־v, אבל זה יהיה מסלול קצר יותר. לכן התהליך בהכרח מסתיים. בסיום התהליך לא מופיע אותו צומת במקומות שונים במסלול, לכן המסלול המתקבל בין u ל־v הוא מסלול פשוט. ♣

פתרון תרגיל 3.4 מעמוד 23

נראה כי שלוש התכונות המגדירות יחס שקילות מתקיימות עבור יחס הקשירות.

רפלקסיביות. המסלול (v) הוא מסלול מ־v ל־v.
סימטריות. אם $(v_1, \ldots, v_k)$ הוא מסלול בגרף, אז גם $(v_k, \ldots, v_1)$ הוא מסלול בגרף מפני שהגרף אינו מכוון.
טרנזיטיביות. אם P הוא מסלול מ־u ל־v ו־Q הוא מסלול מ־v ל־w , אזי השרשור של P ו־Q (כאשר צומת הסיום של P וצומת ההתחלה של Q מתמזגים), הוא מסלול מ־u ל־w.

♣

פתרון תרגיל 3.5 עמוד 24

יהי $T = (V, E)$ עץ. יהי u צומת בדרגה 1 ב־T, ו־$\{u, v\}$ היא הקשת הסמוכה ל־u ב־T. נסמן כי $T' = (V \setminus \{u\}, E \setminus \{u, v\})$. צריך להוכיח ש־$T'$ הוא עץ, כלומר קשיר וחסר מעגלים. תחילה נבדוק שאין ב־T' מעגלים. כיוון ש־T' הינו תת־גרף של T, אם היה בו מעגל, מעגל זה הוא גם מעגל ב־T, סתירה להיות T עץ.

נבדוק כעת ש־T' הוא קשיר. יהיו $x, y \in V \setminus \{u\}$ שני צמתים ב־T' ונראה כי יש מסלול המחבר אותם. כיוון שאלה צמתים ב־T, ו־T הוא קשיר, קיים (לפי תרגיל 3.3) מסלול פשוט $(x = x_0, x_1, \ldots, x_l = y)$ בין x ל־y ב־T. כעת נטען ש־u אינו מופיע במסלול הנ"ל. ואמנם מהנתון אנו יודעים כי $u \neq x_0$ ו־$u \neq x_l$. כיוון שהמסלול הנ"ל פשוט, x_{i-1}, x_i, x_{i+1} הם צמתים שונים עבור $0 < i < l$, ולכן דרגתו של x_i היא לפחות 2. מזה אנו מסיקים ש־$x_i \neq u$. כיוון ש־u אינו מופיע במסלול הנ"ל, זהו גם מסלול ב־T'.

לסיכום, הוכחנו ש־T' הוא קשיר ואינו מכיל מעגלים, ולכן הוא עץ. ♣

פתרון תרגיל 3.6 עמוד 25

ההוכחה באינדוקציה על n – מספר הצמתים בעץ. בסיס האינדוקציה: בעץ עם צומת אחד אין קשתות.

נניח את הנחת האינדוקציה עבור עצים שיש להם $n-1$ צמתים, ונוכיח זאת עבור עצים יש להם n צמתים. יהי T עץ יש לו n צמתים. לפי טענה 3.1 יש בו צומת בדרגה אחת. נסיר את הצומת הזה ואת הקשת הסמוכה לו. לפי תרגיל 3.5, הגרף המתקבל על $n-1$ צמתים הוא עץ. לפי הנחת האינדוקציה יש בעץ זה $n-2$ קשתות, ולכן ב־ T יש $1 + (n-2)$ קשתות. ♣

פתרון תרגיל 3.7 עמוד 25

$1.$, $2.$ $\Leftarrow$ $3.$ הוא תוכן טענה (3.1) בספר ותרגיל 3.6.

$1.$, $3.$ $\Leftarrow$ $2.$ נניח, בשלילה, ש־G קשיר ומכיל מעגל C, ונראה שמספר הקשתות בו גדול מ־$n-1$, הנחה זו עומדת בסתירה להנחה 3. נבחר קשת $\{x, y\} \in C$, ונסיר אותה מהגרף G. הגרף שנוצר נשאר קשיר, כי בכל מסלול ב־G שהשתמש בקשת $\{x, y\}$ אפשר להחליף את הקשת $\{x, y\}$ בתת־מסלול $C \setminus \{x, y\}$, ולקבל מסלול שיחבר בין אותם צמתים בלי להשתמש בקשת $\{x, y\}$. נמשיך בתהליך של מחיקת קשתות ממעגלים עד שנישאר עם גרף ללא מעגלים. כיוון שנשמרה הקשירות, הגרף שיתקבל הוא עץ ולפי תרגיל 3.6 הוא יכיל $n-1$ קשתות. כיוון שעץ זה יתקבל לאחר מחיקה של קשת אחת לפחות מ־G, נובע שיהיו ב־G יותר מ־$n-1$ קשתות, ותוצאה זו עומדת בסתירה להנחה 3.

2, 3. $\Leftarrow 1$. נניח, בשלילה, כי $G=(V,E)$ אינו קשיר, ונסיק שיש בו פחות מ־$n-1$ קשתות, בסתירה להנחה 3. אם G אינו קשיר, יש בו $t \geq 2$ רכיבי קשירות $V_1,\ldots,V_t$. תהי E_i קבוצת הקשתות של G, ששני קצותיהן ב־V_i. כיוון שאין קשת המחברת שני רכיבי קשירות שונים, $E_1 \cup \cdots \cup E_t = E$, לכן $\sum_i |E_i| = |E|$. שימו לב, כל אחד מהגרפים (V_i,E_i) הוא קשיר וללא מעגלים, ומכאן שהוא עץ, כלומר $|E_i| = |V_i| - 1$ ולכן

$$|E| = \sum\nolimits_i |E_i| = \sum\nolimits_i (|V_i| - 1) = n - t < n - 1.$$

♣

פתרון תרגיל 3.8 מעמוד 26

שימו לב, עץ הסריקה־לרוחב תלוי בסדר שבו אנו סורקים את הצמתים בכל שכבה. אנו מניחים שבתוך כל שכבה הצמתים מסודרים בסדר מספרי עולה. במקרה זה,

$$\begin{aligned} L_0 &= \{v_1\} \\ L_1 &= \{v_2, v_5, v_6\} \\ L_2 &= \{v_3, v_7, v_4\} \end{aligned}$$

העץ המתקבל מוצג באיור 3.5. ♣

פתרון תרגיל 3.9 מעמוד 27

בפתרון שלהלן מניחים שהסריקה־לעומק מתבצעת לפי האלגוריתם הרקורסיבי, וכי השכנים נסרקים בכל צומת לפי סדר לקסיקוגרפי. העץ המתקבל מוצג באיור 3.9, וערכי $(b_u)_{u\in V}$, $(f_u)_{u\in V}$ מוצגים באיור 3.11 בעץ מצד שמאל. ♣

פתרון תרגיל 3.10 מעמוד 28

יהי T עץ הסריקה־לעומק של גרף קשיר. נוכיח עתה כי אם v הוא בן של u ב־T, אז $b_u < b_v < f_v < f_u$. המשמעות של היות v בנו של u ב־T היא שהסריקה $\mathrm{DFS}(v)$ נקראת מתוך $\mathrm{DFS}(u)$, וזה מוכיח את הטענה. קל להראות כעת – באינדוקציה על המרחק בין v ל־u ב־T – כי אם v הוא צאצא של u אזי $b_u < b_v < f_v < f_u$.

נניח כעת שאין יחסי צאצא/אב־קדמון בין u ל־v. יהי w אב־קדמון משותף ל־u ול־v ב־T כך שאין ל־w בן שהוא אב־קדמון המשותף ל־u ול־v. הצומת w נקרא האב הקדמון המשותף הנמוך ביותר [lowest common ancestor]. ל־w יש בן v' שהוא אב קדמון של v (ייתכן ש־$v = v'$) ובן u' שהוא אב־קדמון של u (ייתכן ש־$u = u'$). שימו לב, התנאי על w מחייב ש־$u' \neq v'$. כיוון ששניהם בנים של w ב־T, מרגע הקריאה ל־$\mathrm{DFS}(w)$ ועד היציאה מהשגרה $\mathrm{DFS}(w)$ בוצעו קריאות ל־$\mathrm{DFS}(u')$ וגם ל־$\mathrm{DFS}(v')$. כיוון ש־u' ו־v' הם בנים של w ב־T, הקריאות האלה נקראו מתוך השגרה $\mathrm{DFS}(w)$. נניח שהקריאה ל־$\mathrm{DFS}(u')$ בוצעה ראשונה מבין השתיים (המקרה ההפוך הוא סימטרי). בקריאה ל־$\mathrm{DFS}(u')$ כל הצמתים שנסרקו הינם או צמתים שנסרקו כבר או צאצאים של u' ב־T. כיוון ש־v' איננו אף־אחד מהמקרים הללו, אנו מסיקים שהקריאה ל־$\mathrm{DFS}(v')$ לא החלה לפני שביצוע $\mathrm{DFS}(u')$ הסתיים. מכאן ש־$f_{u'} < b_{v'}$. כמו־כן, כיוון ש־u' הוא אב־קדמון של u, מהאמור לעיל נובע כי $f_u < f_{u'}$, ובאופן דומה $b_{v'} < b_v$ ולכן $f_u < b_v$. ♣

פתרון תרגיל 3.11 מעמוד 28

1. כל f_u ו־b_u מקבלים ערך בדיוק פעם אחת, וזהו ערכו של המשתנה t. המשתנה

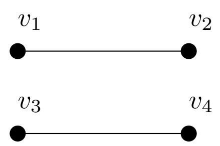
t גדל ב־1 לאחר כל השמה כזאת, ולכן הוא מקבל בדיוק $2n$ ערכים. כיוון שהוא מתחיל ב־0, טווח הערכים שהוא יקבל יהיה $0,..,2n-1$.

2. בתרגיל 3.10 למדנו שקבוצת הצמתים v המקיימים $b_u < f_v < f_u$ שווה לקבוצת הצמתים v המקיימים $b_u < b_v < f_u$ השווה לקבוצת הצאצאים של u בעץ הסריקה לעומק. לפי הפסקה הקודמת אנו יודעים שלכל מספר שלם s בין $b_u + 1$ ל־$f_u - 1$ קיים צומת v יחיד כך ש־b_v או f_v מקבל את הערך s. מכאן שמספר הצאצאים של u הוא בדיוק מחצית מ־$|\{b_u + 1, \ldots, f_u - 1\}|$. ♣

פתרון תרגיל 3.12 מעמוד 32

נקבע צומת s כלשהי. כפי שכבר ראינו, סריקה־לרוחב מ־s בגרף קשיר סורקת את כל הגרף. קיימת רק אפשרות אחת לחלק לשכבות: שכבה L_i היא בדיוק קבוצת הצמתים במרחק i מ־s. כיוון שכל חלוקה לצדדים בגרף דו־צדדי מתאימה לחלוקה לשכבות זוגיות/אי־זוגיות בסריקה־לרוחב מהצומת s, וחלוקה כזאת היא יחידה, יש רק אפשרות אחת לחלק לצדדים בגרף דו־צדדי קשיר.

שימו לב, הטיעון שלעיל אינו תקף בגרף לא־קשיר, כי חלק מן הצמתים אינם מופיעים בסריקה־לרוחב מצומת נתון. להלן דוגמה נגדית:



זהו גרף דו־צדדי, לא קשיר, שיש לו שתי חלוקות שונות לצדדים: האחת היא $\{v_1, v_3\}$, $\{v_2, v_4\}$ והשנייה היא $\{v_1, v_4\}$, $\{v_2, v_3\}$. ♣

פתרון תרגיל 3.13 מעמוד 33

נתון גרף קשיר $G = (V, E)$ וקשת $e = \{x, y\} \in E$. תחילה נניח שקיים מעגל C המכיל את e ונראה ש־e אינה גשר. אכן, המסלול $C \setminus \{e\}$ מקשר את x ו־y ב־$(V, E \setminus \{e\})$, לכן $(V, E \setminus \{e\})$ נשאר קשיר, כלומר e אינה גשר.

בכיוון השני, אם e אינה גשר, אז הצמתים x ו־y קשירים בגרף $(V, E \setminus \{e\})$. כלומר, ב־$(V, E \setminus \{e\})$קיים מסלול פשוט P בין x ל־y. מכאן ש־$P \cup \{e\}$ הוא מעגל ב־G המכיל את e. ♣

פתרון תרגיל 3.14 מעמוד 36

נסמן ב־T_u את תת־העץ של T המושרש ב־u. נוכיח ש־$\bar{L}_u = L_u$ באינדוקציה על $|T_u|$ (מספר הצמתים ב־T_u).

אם $|T_u| = 1$ אזי u הוא עלה של T, ומתקיים:

$$\bar{L}_u = \min\{b_w : \exists (u, w) \in E \setminus T\} = L_u.$$

אם $|T_u| > 1$, יהיו $v_1, \ldots, v_\ell$ הבנים של u ב־T. מהנחת האינדוקציה $\bar{L}_{v_i} = L_{v_i}$, ומתקיים:

$$\begin{aligned}
\bar{L}_u &= \min\{\min\{b_w : \exists(u,w) \in E \setminus T\}, \min_{1\le i\le \ell} \bar{L}_{v_i}\} \\
&= \min\{\min\{b_w : \exists(u,w) \in E \setminus T\}, \min_{1\le i\le \ell} L_{v_i}\} \\
&= \min\left\{\begin{array}{c} \min\{b_w : \exists(u,w) \in E \setminus T\}, \\ \min\limits_{1\le i\le \ell} \min\{b_w : \exists(v,w) \in E \setminus T, \text{ and } v \text{ descendant of } v_i\} \end{array}\right\} \\
&= \min\{b_w : \exists(v,w) \in E \setminus T, \text{ and } v \text{ descendant of } u\} \\
&= L_u.
\end{aligned}$$

♣

פתרון תרגיל 3.15 מעמוד 37

נכונות האלגוריתם Calc־L נובעת מההבחנה המיידית שהוא מחשב את ערכי $(\bar{L}_u)_{u\in V}$, ולפי טענה 3.5 גם את ערכי $(L_u)_{u\in V}$. ננתח את זמן הרי־teSKT llac:(emaNelifXeT) צה. הקריאה הרקורסיבית מתבצעת פעם אחת לכל צומת. במהלך הקריאות הרקורסיביות, כל קשת נסרקת פעם אחת. כיוון שזמן הריצה עבור כל צומת הוא קבוע, נקבל בסה"כ $O(m)$. ♣

פתרון תרגיל 3.16 מעמוד 37

1. נוכיח שדו־קשירות בקשתות היא יחס של שקילות. היחס סימטרי מפני שהגרף אינו מכוון, לכן, בהינתן מסלול ללא גשרים, מצומת u לצומת v, גם המסלול ההפוך מ־v ל־u אינו מכיל גשרים. היחס הוא רפלקסיבי כי המסלול שמכיל רק צומת אחד אינו מכיל קשתות כלל, ובפרט אינו מכיל גשרים. היחס הוא גם טרנזיטיבי כי בהינתן מסלול ללא גשרים, מ־a ל־b, ומסלול ללא גשרים מ־b ל־c, שרשורם של המסלולים הוא מסלול מ a ל־c ללא גשרים.

2. כיוון ש־B היא קבוצת הגשרים, נוכל לומר כי שני צמתים נמצאים באותו רכיב קשירות ב־$(V, E \setminus B)$ אם ורק אם יש ביניהם מסלול ללא גשרים, כלומר הם דו־קשירים בקשתות. ♣

פתרון תרגיל 3.17 מעמוד 38

האלגוריתם יהיה כדלקמן: נפעיל אלגוריתם למציאת גשרים; נבצע סריקה נוספת לעומק לצורך הסרת הגשרים מהגרף. לבסוף נריץ סריקה של כל הגרף החדש, ונמצא את רכיבי הקשירות שלו. נכונות האלגוריתם נובעת מהסעיפים הקודמים. זמן הריצה הוא $O(m+n)$, כי זהו הזמן הדרוש לכל אחת משלוש הסריקות המבוצעות באלגוריתם. ♣

פתרון תרגיל 3.18 מעמוד 38

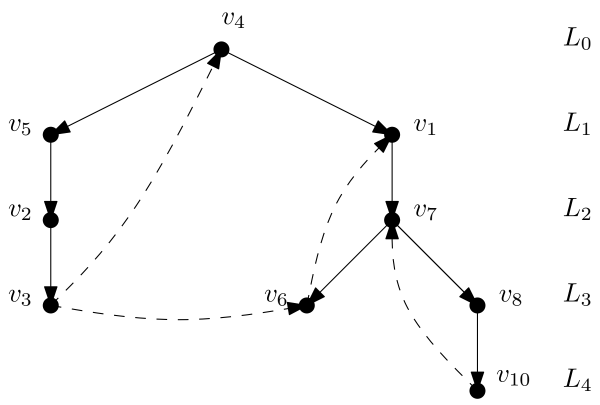
יהי $H = (W, F)$ עץ הגשרים של הגרף הקשיר $G = (V, E)$. נוכיח תחילה ש־H קשיר. הקשת $\{u, v\} \in E$ היא גשר או שאיננה גשר. אם היא איננה גשר אז $\tau(u) = \tau(v)$; ואילו אם היא גשר אז $\{\tau(u), \tau(v)\} \in F$. לכן, עבור $A, B \in W$, יהיו $a \in A$, $b \in B$ צמתים כלשהם ברכיבי דו־קשירות בקשתות A, B. כיוון ש־G קשיר, קיים מסלול $a = a_0, \ldots, a_l = b$. מהאמור לעיל, כל זוג צומתי H עוקבים

בסדרה $(A=\tau(a_0),\tau(a_1),\ldots,\tau(a_l)=B)$ הוא קשת ב־H או שזוג הצמתים זהה. לכן, לאחר מחיקה של צמתים עוקבים זהים בסדרה שלעיל, מתקבל מסלול ב־H מ־A ל־B. כיוון שזה נכון לכל $A,B\in W$, המסקנה היא ש־H קשיר.

נניח עתה, בשלילה, שב־H יש מעגל $A_0,A_1,\ldots,A_{l-1},A_l=A_0$. תחילה נניח ש־$\ell\geq 3$ מכאן נוכל להסיק שלכל $l-1\geq i\geq 0$, קיימת קשת $(b_i,a_{i+1})\in E$ כך ש־$b_i\in A_i$, $a_{i+1}\in A_{i+1}$. נגדיר $a_0=a_l$. כיוון ש־A_i הוא רכיב קשירות ב־$(V,E\setminus B)$, קיים מסלול פשוט p_i ב־G בין a_i ל־b_i המכיל רק צמתים מ־A_i. לכן, לכל $i\neq j$ יש צומתי p_i שהינם זרים בזוגות לצומתי p_j. מכאן ש־$p_0,p_1,\ldots,p_{l-1},a_0$ הוא מעגל ב־G. ובפרט הגשר $\{b_0,a_1\}$ נמצא על מעגל, בסתירה לתרגיל 3.13. ♣

פתרון תרגיל 3.19 מעמוד 39

איור 3.15 מתאר את עץ הסריקה־לרוחב של הגרף שהוצג באיור 3.13 עבור סריקה המתחילה ב־v_4. השכבות גם הן מתוארות. שימו לב שהצומת v_9 אינו מופיע בסריקה, מפני שאינו נגיש מ־v_4. ♣



איור 3.15: הסריקה־לרוחב של הגרף (שהוצג באיור 3.13) מתחילה מ־v_4.

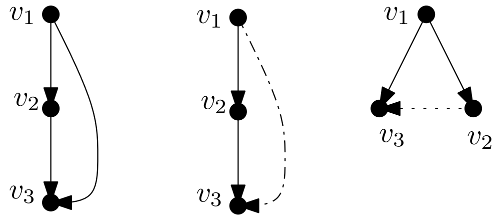
פתרון תרגיל 3.20 מעמוד 39

1. ההוכחה דומה להוכחה של טענה (3.4) בספר. נניח, בשלילה, שקיימת קשת (u,v), מן הצומת u בשכבה L_i, אל הצומת v בשכבה L_j, ו־$j>i+1$. משמעות הדבר שבזמן סריקת השכבה L_i, הצומת v לא התגלה עדיין (אחרת v היה מופיע בשכבה $j\leq i+1$) וזו סתירה להגדרת סריקה־לרוחב, כי בשכבה L_i, הצומת u נשלף מהתור ואז נסרקים כל הצמתים הנגישים מ־u דרך קשת מכוונת, ביניהם v.

2. יהי G מעגל מכוון על n צמתים:

$$G=(\{v_0,v_1\ldots,v_{n-1}\},\ \{(v_0,v_1),(v_1,v_2),\ldots,(v_{n-2},v_{n-1}),(v_{n-1},v_0)\}).$$

נניח כי הסריקה־לרוחב מתחילה בצומת v_0. עץ הסריקה־לרוחב המתקבל הוא מסלול שיש בו שכבה $L_i=\{v_i\}$, לכן יש קשת מצומת v_{n-1} בשכבה L_{n-1} אל הצומת v_0 בשכבה L_0. ♣



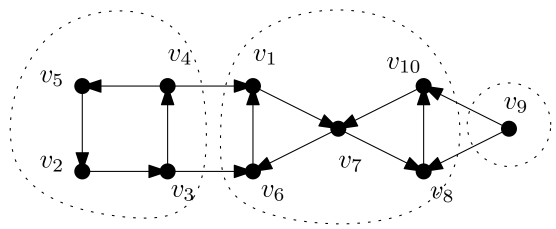
איור 3.16: מצד שמאל מאויר גרף. במרכז – סריקה לעומק שלו המכילה קשת קדימה. מצד ימין מאוירת סריקה לעומק שונה של אותו הגרף שאיננה מכילה קשת קדימה (ומכילה קשת חוצה במקומה).

פתרון תרגיל 3.22 מעמוד 41

דוגמא לסריקה לעומק עם קשת קדימה מוצגת באיור 3.16 ♣

פתרון תרגיל 3.23 מעמוד 42

רכיבי הקשירות החזקה של הגרף באיור 3.13 מסומנים באיור 3.17 על־ידי הקווים המנוקדים. ♣



איור 3.17: רכיבי קשירות חזקה של הגרף שהוצג באיור 3.13.

פתרון תרגיל 3.25 מעמוד 43

כיוון ש־a הוא הצומת היחיד ללא קשת נכנסת, הוא חייב להופיע ראשון במיון הטופולוגי. באופן דומה, כיוון ש־f הוא הצומת היחיד ללא קשת יוצאת, הוא חייב להופיע אחרון במיון הטופולוגי. הקשת (b,c) מחייבת את c להופיע אחרי b, והקשת (d,e) מחייבת את e להופיע לאחר d. לכן התמורות הבאות הן כל המיונים הטופולוגיים של הגרף הנתון:

$$\begin{array}{cccccc} a & b & d & c & e & f \\ a & d & b & e & c & f \\ a & d & b & c & e & f \end{array} \qquad \begin{array}{cccccc} a & b & c & d & e & f \\ a & b & d & e & c & f \\ a & d & e & b & c & f \end{array}$$

בסה"כ 6 מיונים טופולוגיים שונים. ♣

פתרון תרגיל 3.26 מעמוד 43

נציג תחילה פתרון שבו נשתמש בכלים שלמדנו. בפתרון זה נריץ סריקה־לעומק

על הגרף המכוון ונחשב את b_u, ו־f_u. כעת נסרוק את הגרף שוב, (לדוגמה, על־ידי סריקה־לרוחב) ונבדוק אם קיימת קשת אחורה מהסריקה־לעומק הקודמת. נוכל לבצע את הבדיקה הזאת תוך הקדשת זמן קבוע לכל קשת, כיוון שקשת (u,v) היא קשת אחורה אם ורק אם $b_v < b_u < f_u < f_v$. אם לא קיימת קשת אחורה, אזי הגרף הוא גרף מכוון ללא מעגלים. במקרה זה, נבצע מיון בזמן לינארי של f_u כדי למצוא מיון טופולוגי.

הפתרון השני שנציע ישנה במקצת את הסריקה־לעומק (אלגוריתם 3.5) נסביר להלן את החישובים שיבוצעו בסריקה הזו: כיוון שההשמה לערכי f_u מתבצעת ב־DFS בסדר זמנים עולה, כל אשר נדרש הוא להדפיס את הצמתים בסדר הפוך לסדר שבו מושמים ערכי $(f_u)_{u\in V}$. לצורך כך, u מוכנס למחסנית בזמן השמת הערך f_u, ובסיום הסריקה מוצאים איברי המחסנית ומודפסים; בדרך הזו מובטח שהַסדר יהיה הפוך לסדר ההשמה. בנוסף יש לבדוק שהגרף הוא גמ"ל. בדיקה זו נעשית במהלך הסריקה, בעזרת התנאי $b_v < b_u$: אם הקשת (u,v) הנסרקת אינה קשת עץ, בודקים אם היא קשת אחורה, וזאת כאשר f_v עדיין לא נקבע (כלומר, הוא גדול מ־b_u). את כל הפרטים אפשר לקרוא באלגוריתם 3.7 ובאלגוריתם 3.8. הוכחת הנכונות נובעת באופן מיידי מהאמור לעיל.

Algorithm 3.7 Topological-Sort-via-DFS(G)

Require: G is a directed graph
  Initialize all vertices to "Not-yet-Explored"
  Initialize the stack TopSortStack to empty.
  IsDAG $\leftarrow$ **true**
  $t \leftarrow 0$
  while there exists a "Not-Yet-Explored" vertex u **do**
    Call dDFS-Top-Sort(u)
  if IsDASG **then**
    print "G is a DAG. Topological sorting:" TopSortStack
  else
    print "G is not a DAG".

Algorithm 3.8 dDFS-Top-Sort(u)

  $b_u \leftarrow t; \quad t \leftarrow t+1$
  Mark u as "Explored"
  for each directed edge (u,v) incident to u **do**
    if v is not marked "Explored" **then**
      Recursively invoke dDFS(v)
    else
      if $b_v < b_u$ and f_v is not yet set **then**
        IsDAG $\leftarrow$ **false**
  $f_u \leftarrow t; \quad t \leftarrow t+1$
  Push u into TopSortStack.

♣

פתרון תרגיל 3.27 מעמוד 43
תחילה נשים לב שהקשת $e = (x, y)$ בגרף מכוון מקשרת שני רכיבים שונים הקשורים היטב אם ורק אם אינה חלק ממעגל מכוון. זהו תוכן תרגיל 3.24.

כעת, ההוכחה ש־H חסר־מעגלים דומה מאוד להוכחה שעץ הגשרים של גרף (לא־מכוון) הוא חסר־מעגלים, (ראו תרגיל 3.18). נניח, בשלילה, שיש ב־H מעגל $(A_0, A_1, \ldots, A_{l-1}, A_l = A_0)$, $l \geq 2$. מכאן נוכל להסיק שלכל $0 \leq i \leq l-1$, קיימת קשת $(b_i, a_{i+1}) \in E$ כך ש־$b_i \in A_i$, $a_{i+1} \in A_{i+1}$. נגדיר $a_0 = a_l$. כיוון ש־A_i הוא רכיב קשיר היטב, קיים מסלול p_i ב־G בין a_i ל־b_i. מכאן ש־$(p_0, p_1, \ldots, p_{l-1}, a_0)$ הוא מעגל ב־G. בפרט, הקשת (b_0, a_1) נמצאת על מעגל, בסתירה להנחה שהיא מחברת רכיבים שונים הקשורים היטב – A_0 ו־A_1. ♣

פרק 4

אלגוריתמים חמדניים

בפרק הזה אנו דנים לראשונה בבעיות אופטימיזציה, בהן יש למצוא מינימום או מקסימום של פונקציית מטרה, בכפוף לאוסף אילוצים. בלימודי חשבון דיפרנציאלי נתקלנו כבר בבעיות כאלה, עבור פונקציות של משתנים רציפים. למשל, בהינתן פונקציה של שני משתנים ממשיים, $x, y \in \mathbb{R}$, מהם ערכי הקיצון (מינימום ומקסימום) של הפונקציה המתקבלים בתחום כלשהו (נאמר, בעיגול היחידה שהמשתנים שלה x, y מוגבלים על־ידי האילוץ $x^2 + y^2 \leq 1$). בבעיות אופטימיזציה דיסקרטיות, לעומת זאת, הפונקציות הנדונות הן לרוב פונקציות של משתנים המקבלים ערכים על אוסף תת־הקבוצות של קבוצה סופית נתונה A. הבעיה היא כמובן שהמעבר על כל תת־הקבוצות דורש זמן חישוב מעריכי (אקספוננציאלי) בגודל הקלט.

אחת האסטרטגיות המובילות לתכנון אלגוריתמים פולינומיאליים היא האסטרטגיה החמדנית, במסגרתה אנו בונים את הפתרון באופן סדרתי, כאשר בכל שלב אנו מוסיפים לפתרון החלקי הנוכחי (או מחסירים ממנו) קבוצת איברים לפי קריטריון חמדני פשוט. כפי שתקראו בספר, קשה לאפיין באופן מדויק מהו "אלגוריתם חמדני". עבור בעיה ספציפית ייתכנו כמה קריטריונים חמדניים "סבירים", אך לא מובטח לנו כי אפילו אחד מהם אכן יניב אלגוריתם שיפיק פתרון אופטימלי לבעיה. מציאת הכלל הנכון (אם קיים כזה) הוא הקושי העיקרי בתכנון אלגוריתמים חמדניים. כדי שהאלגוריתם יחשב פתרון אופטימלי, הכלל "הנכון" צריך למצוא איבר השייך לאיזשהו פתרון אופטימלי (ואז נוכל להוסיף איבר זה לפתרון החלקי) או למצוא איבר שאינו שייך לאיזשהו פתרון אופטימלי (ואז נוכל להשמיט אותו מהקבוצה).

4.1 תזמון מקטעים

קראו בספר את סעיף 4.1

בסעיף זה אנו דנים בשתי בעיות אופטימיזציה, העוסקות בתזמון מקטעים. מקטע הוא תת־קבוצה קשירה חסומה בישר ממשי. בסעיף זה נדון רק במקטעים הסגורים משמאל ופתוחים מימין, כלומר במקטעים שצורתם

$$[a, b) = \{x \in \mathbb{R} : \ a \leq x < b\}.$$

הקלט לבעיות אלה הוא אוסף מקטעים [intervals]

$$\mathcal{I} = \{[s_1, f_1), [s_2, f_2), \ldots, [s_n, f_n)\}.$$

מקטע $[s_i, f_i)$ מייצג "משימה" [job] או "בקשה" [request] i המאופיינת על־ידי זמן התחלה s_i וזמן סיום f_i. הפלט יהיה תת־קבוצה או חלוקה לתת־קבוצות של מקטעים זרים בזוגות. תת־קבוצה $\mathcal{J} \subseteq \mathcal{I}$ של מקטעים מכילה מקטעים זרים בזוגות כאשר לכל $I_1, I_2 \in \mathcal{J}$, מתקיים התנאי הזה: אם $I_1 \neq I_2$ אז $I_1 \cap I_2 = \emptyset$. לדוגמה: אם

$$\mathcal{I} = \{[0,2),\ [1,4),\ [2,6),\ [3,5)\},$$

אזי תת־הקבוצות $\{[0,2),\ [2,6)\}$ ו־$\{[0,2),\ [3,5)\}$ מכילות מקטעים זרים בזוגות, בעוד שהמקטעים ב־$\mathcal{I}$ עצמה וגם המקטעים בקבוצה $\{[0,2),\ [1,4)\}$ אינם זרים בזוגות.

הבעיה הראשונה היא בעיית מקסימיזציה:

בעיה אלגוריתמית: תזמון מקטעים [Interval Scheduling].
הקלט: אוסף מקטעים $\mathcal{I} = \{[s_1, f_1), [s_2, f_2), \ldots, [s_n, f_n)\}$.
הפלט: תת־קבוצה $\mathcal{I}' \subseteq \mathcal{I}$ של מקטעים זרים בזוגות.
המטרה: להביא למקסימום את מספר המקטעים ב־$\mathcal{I}'$.

הבעיה השנייה היא בעיית מינימיזציה:

בעיה אלגוריתמית: חלוקת מקטעים [Interval Partitioning].
הקלט: אוסף מקטעים $\mathcal{I} = \{[s_1, f_1), [s_2, f_2), \ldots, [s_n, f_n)\}$.
הפלט: חלוקה של $\mathcal{I}$ ל־t קבוצות, כך שבכל קבוצה המקטעים יהיו זרים בזוגות.
המטרה: למזער את t.

תזמון מקטעים. בבעיה זאת פתרון אפשרי הוא תת־קבוצה של מקטעים מהקלט אשר זרים בזוגות, וערכו של פתרון אפשרי הוא מספר המקטעים בתת־קבוצה זאת. בספר מודגמות בפרק זה כמה אסטרטגיות חמדניות "כושלות", שהשימוש בהן אינו מוביל לאלגוריתם המפיק פתרון אופטימלי לבעיה. האסטרטגיה הטבעית ביותר, שגם פותרת את הבעיה, היא לבחור בין המקטעים האפשריים, שעדיין לא נבחרו, את המקטע שזמן הסיום שלו הוא הקטן ביותר. כלומר, מתחילים מהפתרון $\mathcal{I}' = \emptyset$, ובכל שלב מוסיפים ל־$\mathcal{I}'$ את המקטע שיש לו זמן סיום מינימלי מבין כל המקטעים שהם בעלי זמן התחלה גדול או שווה לזמני סיום של המקטעים שכבר נבחרו ל־$\mathcal{I}'$.

בבירור, האלגוריתם מחזיר פתרון חוקי שבו כל המקטעים זרים בזוגות. הסיבה שהאלגוריתם אכן מחשב פתרון אופטימלי נובעת מהעובדה הפשוטה הבאה (הבהירו לעצמכם מדוע זה אכן כך):

טענה 4.1 אם I_1 הוא מקטע ב־$\mathcal{I}$ שזמן הסיום שלו מינימלי, אז החלפת המקטע הראשון[1] של כל פתרון חוקי, במקטע I_1 יתן גם הוא פתרון אפשרי. בפרט, קיים פתרון אופטימלי המכיל את המקטע I_1 כמקטע ראשון.

אם כן, הבחירה של האלגוריתם החמדן במקטע I_1 היא "נכונה". כעת, אחרי שכללנו את I_1 בפתרון $\mathcal{I}'$, אנו צריכים לפתור את הבעיה השיורית של בחירת קבוצה אופטימלית $\mathcal{I}''$ מתוך המקטעים שזמן ההתחלה שלהם גדול או שווה לזמן הסיום של I_1. מהטענה לעיל נובע כי גם הבחירה השנייה של האלגוריתם החמדן, היא צעד נכון בכיוון פתרון הבעיה השיורית. ובאופן דומה, נובע כי בכל צעד וצעד, האלגוריתם החמדן בוחר מקטע "נכון", במובן הזה: קיים פתרון אופטימלי לבעיה השיורית, המכיל את המקטע הנבחר כמקטע שבא מיד אחרי המקטעים שכבר נבחרו. אנו מבחינים גם כי מתקיים הכלל הבא:

> בכל צעד, ערך הפתרון החלקי $\mathcal{I}'$ גדל ב־1, ואילו ערך הפתרון האופטימלי לבעייה השיורית קטן ב־1.

מכאן נובע כי בהגיענו למצב שבו ערך הפתרון האופטימלי לבעיה השיורית הוא אפס, ערך הפתרון החלקי $\mathcal{I}'$ שצברנו יהיה זהה לערך הפתרון האופטימלי.

תרגיל 4.1 נניח כי ברשותנו פרוצדורה המקיימת את התכונה הזו: בכל צעד, ערך הפתרון החלקי $\mathcal{I}'$ גדל ב־1, וערך הפתרון האופטימלי לבעיה השיורית קטן ב־2 לכל היותר. מה תוכלו להגיד אז על "טיב" הפתרון שהאלגוריתם מחשב?

פתרון בעמוד 73

תרגיל 4.2 נתבונן באלגוריתם שבו משתמשים באסטרטגיה "הכושלת" השנייה שהוצגה בספר: בכל שלב בוחרים את המקטע הקצר ביותר, מבין המקטעים שלא חופפים אף מקטע בפתרון החלקי. הראו כי אלגוריתם זה מחשב פתרון שערכו לפחות מחצית הערך האופטימלי.

פתרון בעמוד 73

חלוקת מקטעים. בבעיה זאת פתרון אפשרי הוא חלוקה של הקלט לתתי־קבוצה של מקטעים כל שבכל תת־קבוצה המקטעים זרים בזוגות. וערכו של הפתרון אפשרי הוא מספר תתי־הקבוצות בחלוקה.

אנו נתקלים כאן לראשונה בשיטה הבאה להוכחת אופטימליות. ראשית, יש לנו חסם תחתון על ערך הפתרון. במקרה שלנו חסם זה הוא המספר המקסימלי של מקטעים שמכילים נקודה, כלומר, אם $d(t)$ הוא מספר המקטעים ב־$\mathcal{I}$ שמכילים את t, אזי החסם התחתון לכל ערך t הוא $d(\mathcal{I}) = \max_t d(t)$. בספר חסם תחתון זה מכונה "העומק" של קבוצת הקלט $\mathcal{I}$.

תרגיל 4.3 בהינתן אוסף מקטעים $\mathcal{I}$ הציעו אלגוריתם פשוט לחישוב $d(\mathcal{I})$ שאינו משתמש בשוויון בין $d(\mathcal{I})$ לבין הפתרון האופטימלי של בעיית חלוקת המקטעים. הוכיחו את נכונות האלגוריתם.

פתרון בעמוד 73

אחת הדרכים הפופולריות לפיתוח אלגוריתמים חמדניים, בהם משתמשים בחסם תחתון (במקרה של בעיות מינימיזציה) היא כדלקמן: האלגוריתם יתחיל

[1] שימו לב, מקטעים זרים בזוגות מסודרים בסדר מלא המושרה על־ידי זמן ההתחלה שלהם (או באופן שקול, על־ידי זמן הסיום שלהם).

מהפתרון הריק ובכל פעם יוסיף לפתרון החלקי אלמנט חדש, בצורה חמדנית. כדי להראות שהפתרון המתקבל הוא מינימלי, נתכנן שגרה שתגדיל את ערך הפתרון החלקי באותה כמות שבה היא מקטינה את החסם התחתון של הבעיה השיורית. פעולה זו מבטיחה כי בסוף האלגוריתם, ערך הפתרון יהיה שווה לחסם התחתון.

כעת ננסה לתכנן שגרה שתמצא את החלק הראשון $\mathcal{I}_1$ בחלוקה שאנו מחפשים. המקטעים ב־$\mathcal{I}_1$ צריכים להיות זרים בזוגות, וגם התנאי הבא צריך להתקיים:

העומק של $\mathcal{I} \setminus \mathcal{I}_1$ יהיה קטן ב־1 מהעומק של $\mathcal{I}$, כלומר:

$$d\left(\mathcal{I} \backslash \mathcal{I}_1\right) = d\left(\mathcal{I}\right) - 1.$$

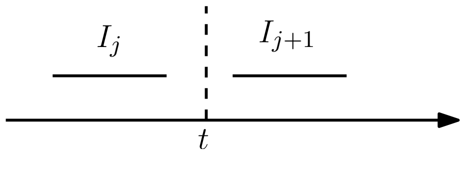
נניח שנצליח לתכנן פרוצדורה כזו. בשלב הראשון נמצא את החלק הראשון $\mathcal{I}_1$ בחלוקה שאנו מחפשים. כדי למצוא את החלק השני $\mathcal{I}_2$, נריץ את הפרוצדורה על $\mathcal{I} \setminus \mathcal{I}_1$ פעם נוספת, וכך נמשיך הלאה. כדי למצוא את החלק הבא, נריץ את הפרוצדורה על קבוצת המקטעים שעדיין לא נכללו בחלקים הקודמים. אחרי $d = d(\mathcal{I})$ שלבים, נקבל:

- d תת־קבוצות זרות $\mathcal{I}_1, \ldots, \mathcal{I}_d$ של $\mathcal{I}$, כך שהמקטעים בכל קבוצה זרים בזוגות;
- העומק של קבוצת המקטעים שלא שובצו יהיה אפס, ולכן כל מקטע יהיה שייך לאחת הקבוצות.

בדרך הזו קיבלנו חלוקה של $\mathcal{I}$ ל־d קבוצות כנדרש.

אך כיצד נתכנן אלגוריתם שימצא את $\mathcal{I}_1$ כנ"ל, אשר "יכסה" את כל הנקודות בהן העומק הוא d? לכל t עבורו $d(t) = d$ צריך להיות מקטע ב־$\mathcal{I}_1$ המכיל את t. יש כמה דרכים למצוא $\mathcal{I}_1$ כזה. הספר משתמש באלגוריתם הזה:

> מתחילים מהפתרון הריק $\mathcal{I}_1 = \emptyset$ ובכל שלב מוסיפים ל־$\mathcal{I}_1$ את המקטע שזמן ההתחלה שלו הוא המינימלי מבין כל המקטעים שזמן ההתחלה שלהם גדול או שווה לזמן הסיום הגדול ביותר של המקטעים ב־$\mathcal{I}_1$.



איור 4.1: אילוסטרציה לטענה 4.2

האבחנה המרכזית היא (ראו איור 4.1):

טענה 4.2 תהי t נקודה שאינה מכוסה על־ידי $\mathcal{I}_1$ המחושב כפי שתואר לעיל. נניח ש־I_j הוא המקטע ב־$\mathcal{I}_1$ שזמן הסיום שלו הוא הגדול ביותר מבין המקטעים שזמן הסיום שלהם קטן מ־t. במצב כזה, כל מקטע I שמכיל את t נחתך עם I_j. בפרט, I_j מכיל נקודה $x \in I_j$ שהעומק שלה גדול מזה של t, כלומר $d(x) > d(t)$.

הוכחה. זמן ההתחלה של כל מקטע I שמכיל את t, קטן יותר מהמקטע I_{j+1} שהוא המקטע ב־$\mathcal{I}_1$ שבא אחרי I_j. לכן, אם I לא היה חותך את I_j, האלגוריתם היה כולל את I ב־$\mathcal{I}_1$.

ישנם שני מקרי קצה נוספים הדורשים ניתוח. אם t היא נקודה לפני המקטע הראשון ב־$\mathcal{I}_1$, אז הטענה עדיין נכונה. יתרה מכך, העומק של t הוא אפס, כי אין מקטע המתחיל לפני המקטע הראשון ב־$\mathcal{I}_1$.

אם I_j הוא המקטע האחרון, אז I_{j+1} לא קיים, אבל הטענה עדיין נכונה; אחרת, האלגוריתם היה יכול להוסיף ל־$\mathcal{I}_1$ את המקטע I. □

תרגיל 4.4 לפתרון בעיית תזמון המקטעים השתמשנו באלגוריתם המאפשר לבחור, מבין כל המקטעים האפשריים, את המקטע שזמן הסיום שלו הוא הקרוב ביותר; האם האלגוריתם הזה מחשב גם את קבוצת המקטעים שאיחודם מכיל את כל הנקודות בהן העומק הוא העומק המקסימלי? הסבירו.

פתרון בעמוד 74

תרגיל 4.5 הציעו אלגוריתם פולינומיאלי לבעיה הזו: הקלט הוא קבוצה של מקטעים $\{I_1,\ldots,I_k,I'_1,\ldots,I'_k\}$, באורך של לפחות 2 כל אחד, כאשר לכל j מתקיים: אם $I_j=[a_j,b_j)$ אז $I'_j=[a_j+1,b_j+1)$ או $I'_j=[a_j-1,b_j-1)$. יש למצוא תת־קבוצה $\mathcal{I}$ של מקטעים זרים בזוגות, כך שלכל j, יהיה לכל היותר אחד מתוך I_j,I'_j השייך ל־$\mathcal{I}$. המטרה היא להביא למקסימום את מספר המקטעים $|\mathcal{I}|$ ב־$\mathcal{I}$.

פתרון בעמוד 74

4.2 תזמון כדי למזער איחור: טיעון החלפה

קראו בספר את סעיף 4.2

בסעיף זה אנו דנים בבעיית התזמון הזו:

בעיה אלגוריתמית: תזמון מקטעים לשם מזעור האיחור.

הקלט: n משימות, כאשר לכל משימה $i\in\{1,\ldots,n\}$ יש זמן ביצוע $t_i>0$ ומועד אחרון $d_i>0$.

הפלט: סדרה $s_1,\ldots,s_n$ כך שמתקיים: $s_i\geq 0$ וגם אוסף המקטעים הבא אינו נחתך בזוגות:

$$\{[s_i,s_i+t_i):\ i\in\{1,\ldots n\}\}.$$

המטרה: נסמן ב־$\ell_i=\max\{s_i+t_i-d_i,0\}$ את האיחור בביצוע המשימה ה־i. יש למזער את האיחור המקסימלי $L=\max_i \ell_i$.

תזמון ללא מרווחים הוא תזמון המקיים ש־$\bigcup_i[s_i,s_i+t_i)$ הוא מקטע. ברור כי לכל תזמון קיים תזמון ללא מרווחים שהינו טוב לא פחות – אפשר פשוט ל"מחוק" כל מרווח בגודל x על־ידי הזזת המקטעים, שמימין למרווח, x צעדים שמאלה. בכך הערך L לא יגדל. מכאן שקיים תזמון אופטימלי ללא מרווחים, ולכן נוכל להגביל עצמנו לתזמונים כאלה בלבד.

האלגוריתם שבספר מתעלם לחלוטין מזמני הביצוע t_i ומתזמן את המשימות בזו אחר זו, ללא רווחים, לפי סדר המועדים האחרונים d_i. ההוכחה כי האלגוריתם מחשב תזמון אופטימלי מבוססת על הרעיון הזה: מראים שאפשר להמיר כל פתרון אופטימלי לפתרון שמייצר האלגוריתם, בלי להעלות את ערך הפתרון. בהינתן תזמון ללא מרווחים, היפוך הוא מצב שבו יש שתי משימות (לאו דווקא עוקבות) המקיימות:

- משימה i משובצת לפני משימה j;
- $d_i > d_j$.

מספר ההיפוכים המקסימלי שיכול להיות הוא $\binom{n}{2}$ (חסם זה יכול להתקבל כאשר המשימות מסודרות בסדר ההפוך למיונן לפי הזמנים האחרונים). אמנם, היפוך לא מחייב משימות עוקבות, אבל אם קיים היפוך, אזי קיים לפחות גם היפוך אחד של משימות עוקבות i, j בסידור הנוכחי. בספר מוכח שאם נחליף את סדר הביצוע של i, j, אזי:

- ערך הפתרון לא יעלה;
- מספר ההיפוכים יקטן בלפחות אחד.

ומכאן נסיק כי קיים פתרון אופטימלי ללא היפוכים כלל. גם הפתרון שמייצר האלגוריתם הוא ללא היפוכים, אבל עדיין לא נובע מכך שזהו פתרון אופטימלי, כי במקרה שהמועד האחרון משותף לכמה משימות, ישנם כמה סדרים ללא היפוכים. הטיעון שמסיים את ההוכחה הוא: כל הפתרונות ללא היפוכים הם בעלי אותו ערך.

תרגיל 4.6 נניח כי במקום להביא למינימום את האיחור המקסימלי $L = \max_i \ell_i$, היינו מעוניינים להביא למינימום את סכום האיחורים $\sum_i \ell_i$.

1. האם גם אז האלגוריתם שלעיל ייחשב בהכרח לפתרון אופטימלי?
2. הראו כי כל אלגוריתם המחשב פתרון אופטימלי לבעיה הזאת, אינו יכול להתעלם מזמני ביצוע המשימות t_i.

פתרון בעמוד 74

4.3 שמירה אופטימלית בזיכרון מטמון*

סעיף זה אינו חובה בחומר הלימוד. אם ברצונכם להעמיק בשיטות חמדניות

קראו בספר את סעיף 4.3

הפרק הזה עוסק בבעיה שאפשר לנסח אותה בקצרה באופן הזה (הבהירו לעצמכם מדוע):

בעיה אלגוריתמית: ניהול בזיכרון מטמון [Cache management].
הקלט: סדרה $D = d_1 \ldots d_m$ מעל קבוצה U של n עצמים, מספר $n > k$ ותת־קבוצה U_0 שגודלה לכל היותר k.
הפלט: סדרה של תתי־קבוצות $U_1, \ldots, U_m \subset U$ (U_t מייצג את תוכן המטמון בזמן t), המקיימות $|U_i| \leq k$ ו־$d_i \in U_i$ לכל $i \in \{1, \ldots, m\}$
המטרה: להביא למינימום את $\sum_{i=1}^{m} |U_i \setminus U_{i-1}|$.

בספר מתואר אלגוריתם פשוט מאוד; כיוון שגם ניתוחו אינו מסובך, לא נחזור עליו כאן.

4.4 מסלולים קצרים ביותר בגרף

קראו בספר את סעיף 4.4

הבעיה הנדונה היא:

בעיה אלגוריתמית: מסלולים קצרים ביותר ממקור יחיד [Single Source Shortest Path].
הקלט: גרף מכוון $G=(V,E)$, פונקציית מרחק אי־שלילית על הקשתות $\ell : E \to [0,\infty)$, וצומת מקור $s \in V$.
הפלט: לכל צומת $v \in V$ מסלול קצר ביותר P_v מ־s ל־v.

כפי שמוסבר בספר, נוח לתכנן אלגוריתם שמחשב את אורכי המסלולים הקצרים ביותר, ואת המסלולים עצמם אפשר לחשב מהערכים האלה בקלות (ועוד נרחיב על כך בהמשך). נוכל להניח גם כי הגרף G הוא גרף שלם (כלומר גרף בו יש קשת בין כל זוג צמתים), על־ידי הגדרת $\ell(u,v)=\infty$ כאשר $(u,v) \notin E$. נסמן ב־$d(v)$ את אורך המסלול הקצר ביותר מ־s ל־v. אלגוריתם דייקסטרה מתבסס על הטענה הבאה [בספר זוהי טענה (4.14)].

טענה 4.3 תהי S קבוצת צמתים שמכילה את s, ולכל $y \in V \setminus S$ נגדיר

$$d'(y) = \min_{u \in S} \left(d(u) + \ell(u,y) \right) .$$

אם $v \in V \setminus S$ הוא צומת שממזער את ערכי $d'(\cdot)$, כלומר

$$d'(v) = \min_{y \in V \setminus S} d'(y),$$

אזי $d(v) = d'(v)$.

הוכחה. שימו לב: $d'(v)$ הוא אורך של מסלול אפשרי מ־s ל־v (המסלול הקצר ביותר לצומת $u \in S$ ומשם בקשת ל־v), ראו איור 4.2 משמאל. לכן $d'(v) \geq d(v)$, כיוון ש־$d(v)$ הוא אורך המסלול הקצר ביותר ל־v.

איור 4.2: המחשה להוכחת טענה 4.3.

עתה נוכיח את אי־השיוויון בכיוון השני. יהי P מסלול קצר ביותר מ־s ל־v, כלומר $d(v) = \ell(P)$, כאשר $\ell(P)$ הינו אורך המסלול P. תהי (x,y) הקשת

הראשונה במסלול הזה, העוברת מ־S ל־$V \setminus S$ (כלומר $x \in S$, $y \in V \setminus S$), ראו איור 4.2 מימין; קשת כזו קיימת כי $s \in S$ ואילו $v \in V \setminus S$. כיוון ש־P עובר דרך הקשת (x, y), מתקיים

$$\ell(P) \geq d(x) + \ell(x, y). \tag{4.1}$$

כמו־כן מתקיים

$$d(x) + \ell(x, y) \geq \min_{u \in S}[d(u) + \ell(u, y)], \tag{4.2}$$

כיוון ש־$x \in S$ והאגף בצד ימין של (4.2) הינו מינימום על u וניתן בפרט לבחור $u = x$. לבסוף נשים לב שאגף ימין של (4.2) הינו הגדרת $d'(y)$. מצד שני, v הוא צומת המביא למינימום את ערכי d'. לכן

$$\min_{u \in S}[d(u) + \ell(u, y)] = d'(y) \geq d'(v). \tag{4.3}$$

מצרוף אי־השוויונים (4.1), (4.2), ו־(4.3) אנו מסיקים ש־$d(v) = \ell(P) \geq d'(v)$. $\square$

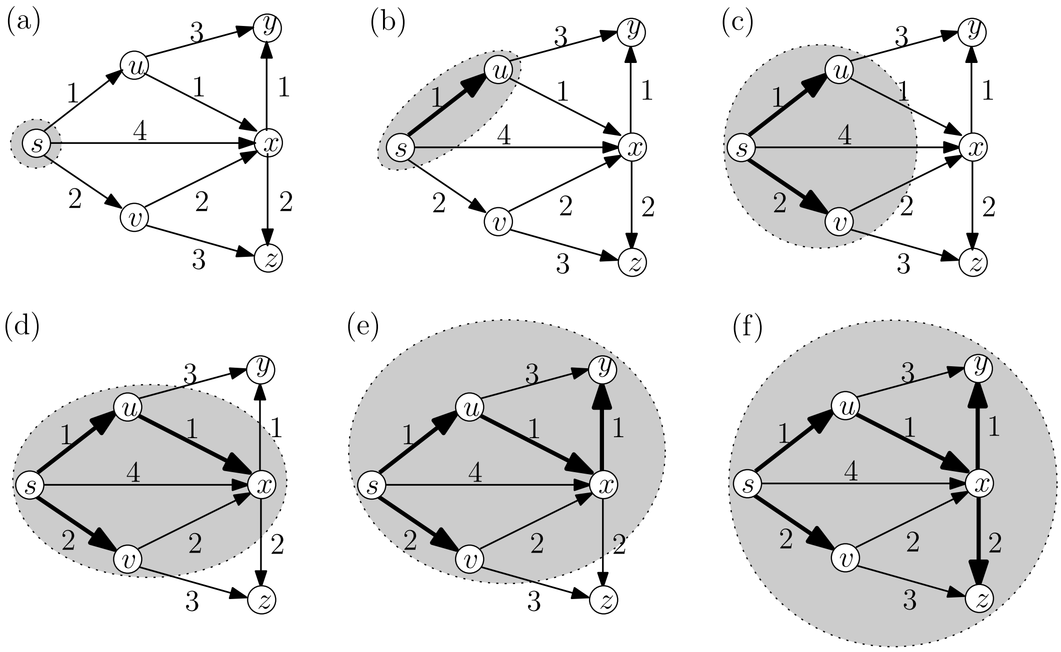
מטענה 4.3 נובע כי:

> אם $S \neq V$, וכבר חישבנו את המרחקים הקצרים ביותר מ־s לכל הצמתים בקבוצה S, אז נוכל לחשב (בזמן פולינומיאלי) את המרחק הקצר ביותר לצומת נוסף $v \in V \setminus S$.

אבחנה חשובה היא כי למעשה המסלולים המחושבים על־ידי אלגוריתם דייקסטרה יוצרים עץ מכוון T המושרש ב־s, שנקרא עץ המסלולים הקצרים ביותר. לכל צומת s, המסלול (היחיד) מ־s ל־v ב־T הוא מסלול קצר ביותר ב־G מ־s ל־v. עץ זה נבנה באופן טבעי במהלך האלגוריתם באופן הזה: נניח שחישבנו את המרחקים הקצרים ביותר עבור קבוצת הצמתים S ובנינו עבורם את עץ המרחקים הקצרים ביותר. כאשר אנו מוסיפים צומת v ל־S, אנו מוסיפים לעץ T קשת אחת (u, v) עבורה מתקבל המינימום של $\min_{u \in S}(d(u) + \ell(u, v))$. נדגים זאת על הגרף באיור 4.3, שהעתקנו מן הספר (שם הוא היה איור 4.7).

- **(a)** אנו מתחילים מהגרף בחלק a של איור 4.3 כאשר $S = \{s\}$ והעץ $T = (\{s\}, \emptyset)$ (כלומר T מכיל רק את הצומת s).
- **(b)** הצומת הקרוב ביותר ל־s, u, מצטרף ל־S וגם ל־T.
- **(c)** הצומת v מצטרף ל־S וגם ל־T, אבל יכולנו לבחור לצרף גם את x.
- **(d)** הצומת x מצטרף ל־S וגם ל־T דרך הקשת (u, x).
- **(e)** הצומת y מצטרף ל־S וגם ל־T דרך הקשת (x, y).
- **(f)** הצומת z מצטרף ל־S וגם ל־T דרך הקשת (x, z).

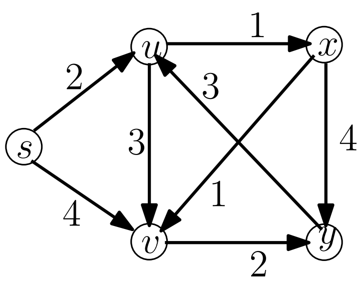
הערה: כפי שמוזכר בתחילת סעיף 4.4 בספר, אפשר להשתמש באלגוריתם דייקסטרה לחישוב מסלולים קצרים ביותר גם בגרפים לא־מכוונים. לשם כך מריצים את האלגוריתם על הגרף המכוון, המתקבל על־ידי החלפת כל קשת לא־מכוונת בשתי קשתות מנוגדות שהמחירים שלהן זהים לאלה של הקשת הלא־מכוונת. יותר מכך, אם T הוא עץ המסלולים הקצרים ביותר, ממקור s בגרף המכוון המתקבל, אזי גרף התשתית של T (כלומר, הגרף המתקבל מ־T לאחר "מחיקת" כיווני הקשתות) הוא עץ המסלולים הקצרים ביותר ממקור s בגרף הלא־מכוון. ♣



איור 4.3: הדגמת שלבים בריצת אלגוריתם דייקסטרה ובניית עץ המרחקים הקצרים ביותר. בכל שלב, הצמתים ב־S מסומנים באליפסה האפורה, וקשתות העץ T מודגשות.

תרגיל 4.7 הריצו את האלגוריתם של דייקסטרה על הגרף המכוון המוצג באיור 4.4. הדגימו את כל השלבים בריצת האלגוריתם ובנו את עץ המרחקים הקצרים ביותר.

פתרון בעמוד 75



איור 4.4: גרף מכוון ממושקל בתרגיל 4.7.

תרגיל 4.8 הציעו אלגוריתם יעיל לבעיה הזו: הקלט הוא גרף מכוון $G = (V, E)$ עם משקלים אי־שליליים על הקשתות ושתי תת־קבוצות זרות S, T של צמתים. יש למצוא מסלול מכוון P המתחיל מצומת ב־S ומסתיים בצומת ב־T. המטרה היא להביא למינימום את משקל המסלול P.

פתרון בעמוד 76

תרגיל 4.9 האלגוריתם של דייקסטרה בוחר בכל פעם צומת $v \in V \setminus S$ הממזער את הביטוי $\min_{u \in S}(d(u) + \ell(u,v))$ ומוסיף אותו ל־S ואת הקשת (u,v) הוא מוסיף ל־T (כאשר u הוא הצומת שהביא למינימום את הביטוי $d(u) + \ell(u,v)$). הוכיחו או הפריכו את הטענה הזו: אם נשנה את בחירת v לצומת הממזער את הביטוי "$\min_{u \in S} \ell(u,v)$", האלגוריתם שיתקבל גם כן יחשב את עץ המרחקים הקצרים ביותר.

פתרון בעמוד 76

תרגיל 4.10 הקוטר [diameter] של גרף לא־מכוון $G = (V,E)$ עם מרחקים אי־שליליים $\ell : E \to [0,\infty)$ על הקשתות, מוגדר על־ידי $D(G,\ell) = \max_{u,v \in V} d(u,v)$, כאשר $d(u,v)$ הוא המרחק (אורך המסלול הקצר ביותר) בין u ל־v ביחס לאורכי הקשתות $\ell(e)$. יהי T עץ המסלולים הקצרים ביותר ממקור s כלשהו, בגרף לא־מכוון G. הוכיחו כי הקוטר של T הוא לכל היותר פעמיים הקוטר של G, כלומר $D(T,\ell|_T) \le 2D(G,\ell)$. כאן אנו מסמנים ב־$\ell|_T$ את הצמצום של פונקציית אורכי הקשתות ℓ של G לקשתות T בלבד.

פתרון בעמוד 76

4.5 בעיית העץ הפורש המינימלי

קראו את סעיף 4.5 בספר

סעיף זה עוסק בגרפים (לא־מכוונים) $G = (V,E)$ קשירים וממושקלים, כלומר גרפים שיש להם פונקציית משקל אי־שלילית $c : E \to [0,\infty)$ על הקשתות. אנו נגדיר, עבור תת־קבוצה $E' \subseteq E$ של קשתות, את המשקל של E', באופן הזה:

$$c(E') = \sum_{e \in E'} c(e).$$

המטרה בסעיף זה תהיה לפתור את הבעיה הבאה.

בעיה אלגוריתמית: עץ פורש מינימלי [Minimum Spanning Tree].
הקלט: גרף לא־מכוון וקשיר $G = (V,E)$ עם משקלות אי־שליליים $c : E \to [0,\infty)$ על הקשתות.
הפלט: עץ פורש T של G.
המטרה: למזער את המשקל של T.

נעיר כי עץ "פורש" של G פירושו כי קבוצת הצמתים של העץ היא V, כלומר זהה לקבוצת הצמתים של G.[2] בספר מוצגות שתי שיטות לפתרון הבעיה. בשיטה הראשונה בונים את הפתרון "מלמטה" – החל מהפתרון הריק – ובכל שלב מוסיפים לפתרון קשת אחת, עד אשר מתקבל פתרון אפשרי. כלומר, משפרים את "הישימות" של הפתרון בכל שלב. כאן השאלה המכרעת היא:

> האם אפשר לתכנן שגרה שתמצא קשת, השייכת בהכרח לאיזשהו עץ פורש מינימלי?

[2] ובאופן כללי, גרף $G' = (V',E')$ הוא תת־גרף פורש של גרף $G = (V,E)$ אם $V' = V$ וגם $E' \subseteq E$.

בשיטה השנייה בונים את הפתרון "מלמעלה". מתחילים עם כל הקשתות, ובכל שלב מסירים קשת, עד אשר מתקבל עץ. בפרט, משפרים בכל שלב את העלות של הפתרון. כאן השאלה היא:

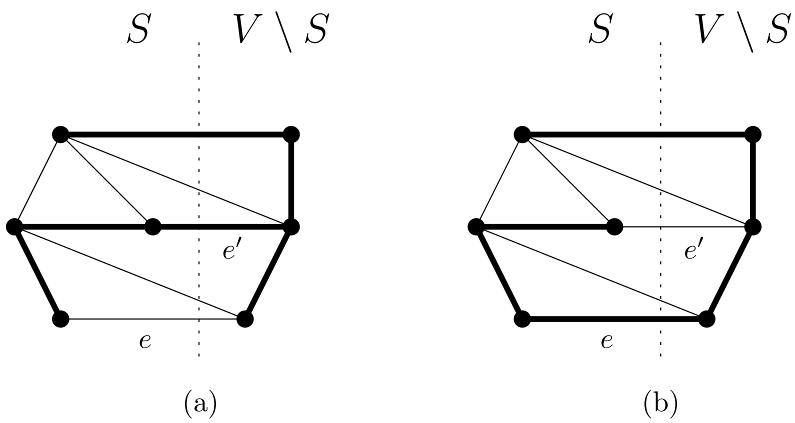
> האם אפשר לתכנן שגרה שתמצא קשת שבהכרח אינה שייכת לאיזשהו עץ פורש מינימלי?

בשני המקרים, נשתמש בתכונה פשוטה של עצים – טענה 4.4. כדי לתאר אותה נזדקק למושג החתך.

הגדרה 4.1 (חתך) . יהי $G = (V, E)$ גרף (לא־מכוון), ותהי $\emptyset \subsetneq S \subsetneq V$. החתך המוגדר על־ידי S הוא

$$E(S, V \setminus S) = \{\{u, v\} \in E : \ u \in S, \ v \in V \setminus S\}.$$

$\diamondsuit$



איור 4.5: הדגמה של הוכחת טענה 4.4. באיור (a) מתואר חתך בגרף ועץ פורש שקשתותיו מודגשות. הקשת e חוצה את החתך והוספתה לעץ הפורש יוצרת מעגל יחיד. מעגל זה חוצה את החתך בקשת (אחת או יותר) נוספת. באיור זו הקשת e'. זריקת הקשת e' מהעץ הפורש והוספת e במקומה יוצרת עץ פורש חדש המכיל את e, כמתואר באיור (b).

טענה 4.4 (טענת ההחלפה) יהי T עץ פורש של גרף לא מכוון $G = (V, E)$, תהי $\emptyset \subsetneq S \subsetneq V$, ותהי $e \in E(S, V \setminus S)$ קשת בחתך המוגדר על־ידי S. במקרה כזה T מכיל קשת e' בחתך המוגדר ע"י S, $e' \in T \cap E(S, V \setminus S)$, כך שגם הגרף $T' = (T \setminus \{e'\}) \cup \{e\}$ הוא עץ פורש של G.

הוכחה. אם $e \in T$, נבחר $e' = e$ והטענה ברורה.

נניח עתה ש־$e' \notin T$. הוכחת מקרה זה מודגמת באיור 4.5. הגרף $T \cup \{e\}$ מכיל מעגל יחיד $C \subset T \cup \{e\}$, ומעגל זה מכיל את הקשת e. כיוון של־e יש קצה אחד ב־S וקצה שני ב־$V \setminus S$, המעגל מכיל לפחות קשת אחת נוספת $e' = \{x, y\}$

השונה מ־e וגם לה יש קצה אחד ב־S וקצה שני ב־$V \setminus S$. אם נסיר את e' מ־$T \cup \{e\}$ יתקבל שוב גרף קשיר וחסר מעגלים, כלומר עץ. נוכיח זאת: הגרף המתקבל חסר מעגלים כי ב־$T \cup \{e\}$ יש מעגל יחיד, והסרנו קשת ממעגל זה. הגרף $(T \setminus \{e'\}) \cup \{e\}$ הוא גם קשיר כיוון שהגרף $T \cup \{e\}$ קשיר והסרנו ממנו קשת יחידה e', ובכל מסלול ב־$T \cup \{e\}$ אשר משתמש בקשת $e' = \{x, y\}$ אפשר להחליף את הקשת e' בתת־המסלול $C \setminus \{e'\}$ בין x ל־y (או בין y ל־x). □

טענה 4.5 (תנאי החתך) (דומה לטענה (4.17) בספר) תהי $\emptyset \subsetneq S \subsetneq V$ תת־קבוצה ממש לא ריקה של V. תהי e קשת זולה ביותר מבין הקשתות שיש להן קצה אחד ב־S וקצה שני ב־$V \setminus S$, כלומר

$$e = \underset{f \in E(S, V \setminus S)}{\arg\min}\ c(f).$$

במקרה כזה לכל עץ פורש T' קיימת קשת $e' \in T' \cap E(S, V \setminus S)$ כך ש־$T = (T' \setminus \{e'\}) \cup \{e\}$ הינו עץ פורש המכיל את e ומקיים $c(T) \leq c(T')$. בפרט, קיים עץ פורש מינימלי המכיל את e, ואם T' הוא עץ פורש מינימלי אז גם T הוא עץ פורש מינימלי.

הוכחה. לפי טענת ההחלפה (טענה 4.4), T מכיל קשת e' שיש לה קצה אחד ב־S וקצה שני ב־$V \setminus S$ כך שגם הגרף $T' = (T \setminus \{e'\}) \cup \{e\}$ הוא עץ. כיוון ש־e היא בעלת משקל מינימלי מבין הקשתות שיש להן קצה אחד ב־S וקצה שני ב־$V \setminus S$, $c(e) \leq c(e')$. אנו מסיקים כי

$$c(T') = c(T) + c(e) - c(e') \leqslant c(T),$$

כנדרש. □

הערה: בניגוד לטענה (4.17) בספר, איננו מניחים בטענה 4.5 שכל משקלי הקשתות שונים זה מזה. כתוצאה מזאת, מתקבלת תכונה חלשה יותר: e מופיעה אמנם באיזשהו עץ פורש מינימלי, אך היא אינה חייבת להופיע בכל עץ פורש מינימלי. ♣

כעת נוכל לתאר אלגוריתם מופשט לפתרון בעיית העץ הפורש המינימלי, הכולל את האלגוריתם של פרים ($Prim$) וגם את האלגוריתם של קרוסקל כמקרים פרטיים. אלגוריתם זה מתואר להלן באלגוריתם 4.1.

Algorithm 4.1 MST_Cut_Algorithm($G = (V, E), c : E \to [0, \infty)$)

Require: G is connected
  Intialize $F \leftarrow \emptyset$
  while F does not span G **do**
    Choose $\emptyset \subsetneq S \subsetneq V$ such that $F \cap E(S, V \setminus S) = \emptyset$.
    Add the chepeast edge $e \in E(S, V \setminus S)$ to F.
  return F

תרגיל 4.11 הוכיחו שאלגוריתם 4.1 מחשב עץ פורש.

פתרון בעמוד 77

טענה 4.6 אלגוריתם 4.1 מחשב עץ פורש מינמלי של גרף הקלט G.

הוכחה. מתרגיל 4.11 נובע כי האלגוריתם מחזיר עץ פורש של G. עבור איטרציה i של האלגוריתם, תהי S_i הקבוצה אותה בוחן האלגוריתם, תהי e_i קשת זולה ביותר בחתך $E(S_i, V \setminus S_i)$ אותה מוסיף האלגוריתם ל־F, ותהי $F_i = \{e_1, \ldots, e_i\}$. נוכיח באינדוקציה כי לכל $i = 1, \ldots, n-1$ $|F_i| = i$ וגם קיים עץ פורש מינימלי של G המכיל את F_i. מכאן נקבל כי העץ F_{n-1} אותו מחזיר האלגוריתם הוא עץ פורש מינימלי.

בסיס האינדוקציה $i = 1$ נובע מתנאי החתך (טענה 4.5). נניח עתה כי קיים עץ פורש מינימלי T' של G המכיל את F_{i-1}, $2 \leq i \leq n-1$. נוכיח בעזרת תנאי החתך כי אז קיים עץ פורש מינימלי T של G המכיל את $F_i = F_{i-1} \cup \{e_i\}$. על־פי תנאי החתך, קיימת קשת $e' \in T' \cap E(S_i, V \setminus S_i)$ כך ש־$T = T' \setminus \{e'\} \cup \{e_i\}$ גם הוא עץ פורש מינימלי. שימו לב כי $e' \notin F_{i-1}$, כי האלגוריתם בחר S_i כך שמתקיים $F_{i-1} \cap E(S_i, V \setminus S_i) = \emptyset$. לכן T הוא עץ פורש מינימלי המכיל את $F_i = F_{i-1} \cup \{e_i\}$. מאותה סיבה גם $e_i \notin F_{i-1}$, ולכן $|F_i| = |F_{i-1}| + 1 = i$, כנדרש. □

תרגיל 4.12 (האלגוריתם של בורובקה) נתבונן באלגוריתם 4.2 לחישוב עץ פורש מינימלי כאשר אין שתי קשתות עם משקל זהה.

1. הוכיחו שהאלגוריתם אמנם מחשב עץ פורש מינימלי.
2. הוכיחו שלולאת **while** מסתיימת לאחר $\log_2 n$ איטרציות לכל היותר.

פתרון בעמוד 77

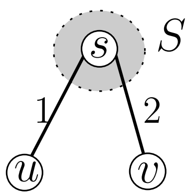
Algorithm 4.2 Boruvka$(G = (V, E), c : E \to [0, \infty))$

Require: G is a connected undirected graph.
Require: $c(e) \neq c(e')$ if $e, e' \in E$, and $e \neq e'$.
  $F \leftarrow \emptyset$
  while the number of connected components in $(V, F) > 1$ **do**
    Let $\mathcal{W}$ be the set of connected components of (V, F).
    for every $W \in \mathcal{W}$ **do**
      $e_W \leftarrow$ the cheapest edge in $E(W, V \setminus W)$
    $F \leftarrow F \cup \{e_W : W \in \mathcal{W}\}$
  return F

נעבור כעת לדון בשאלה מתי אנו יכולים להבטיח כי קשת אינה שייכת לאיזשהו פתרון אופטימלי? איור 4.6 מדגים כי תנאי דמוי תנאי החתך לא מספק לנו קריטריון שכזה.

טענה 4.7 (תנאי המעגל) (דומה לטענה (4.20) בספר) יהי C מעגל ב־G ותהי e' קשת כבדה ביותר ב־C. במקרה כזה, לכל עץ פורש T קיים עץ פורש T' שאינו מכיל את e' כך שמתקיים $c(T') \leq c(T)$. בפרט, קיים עץ פורש מינימלי שאינו מכיל את e'.

הוכחה. נסמן $e' = \{u, v\}$, ונניח ש־$e' \in T$ (אחרת אין מה להוכיח). הגרף $T \setminus \{e'\}$ מורכב משני רכיבי קשירות, נסמנם S ו־$V \setminus S$, כאשר $u \in S$ ו־$v \in V \setminus S$. תת־הגרף $C \setminus \{e'\}$ הוא מסלול ב־G בין u ל־v, ולכן יש בו קשת e שקצה אחד שלה



איור 4.6: יש שתי קשתות בחתך המושרה על־ידי S, ושתיהן נמצאות בכל פתרון.

ב־S והקצה השני ב־$V\setminus S$. הקשת e' היא קשת יחידה של T שקצה אחד שלה ב־S והקצה השני ב־$V\setminus S$, לכן, על־פי טענת ההחלפה (טענה 4.4), $T'=(T\setminus\{e'\})\cup\{e\}$ הוא עץ. שימו לב, $c(e)\le c(e')$, כי $e,e'\in C$, והקשת e' היא קשת כבדה ביותר ב־C. מכאן נקבל $c(T')=c(T)+c(e)-c(e')\le c(T)$, כנדרש. $\square$

אם־כן, האלגוריתם שמסיר קשתות – הנקרא אלגוריתם המחיקה לאחור – יהיה דומה לאלגוריתם החתך, אבל הוא ימחק קשתות במקום להוסיפן. בכל שלב אנו מוצאים מעגל ב־G, ומוחקים מ־G קשת כבדה ביותר במעגל זה. כשלא נותרים מעגלים, נקבל עץ פורש.

ברצוננו לטעון שהעץ הפורש המתקבל הוא בעל משקל מינימלי. שימו לב, לא מספיק לטעון כי היות שכל קשת שנמחקת אינה שייכת לאיזשהו עץ פורש מינימלי, מותר למחוק אותה, כי אז ייתכן (לכאורה) שבכל פעם נמחק קשת מעץ פורש מינימלי אחר ולבסוף נישאר עם אוסף קשתות שאינן מתאימות לאף עץ פורש מינימלי. זה אמנם לא קורה, אך הטיעון בפסקה הזו לא פוסל מקרה כזה.

במקום זאת נטען כך: נגדיר את סדרת הגרפים המתקבלת ממחיקת הקשתות. יהי $G_0=G$. תהי e_1 הקשת הראשונה שנמחקת, ויהי G_1 הגרף המתקבל מ־G_0 לאחר מחיקת e_1. באופן דומה, תהי e_i הקשת ה־i שנמחקת, ויהי G_i הגרף המתקבל מ־G_{i-1} לאחר מחיקת e_i. התהליך מסתיים בשלב ה־ℓ כאשר G_ℓ הוא עץ (פורש). נקבע כי T_0 הוא עץ פורש מינימלי של $G_0=G$. כעת נשתמש באינדוקציה, נניח ש־T_{i-1} הוא עץ פורש מינימלי של G ותת־גרף של G_{i-1}. לפי טענה 4.7, קיים ב־G_{i-1} עץ פורש – נסמנו T_i – שאינו מכיל את הקשת e_i, ולכן T_i הוא תת־גרף של G_i, וכמו־כן $c(T_i)\le c(T_{i-1})$ אך כיוון ש־T_{i-1} הוא עץ פורש מינימלי של G, וכיוון שגם T_i הוא עץ פורש של G, בהכרח $c(T_i)=c(T_{i-1})$, ו־T_i הוא עץ פורש מינימלי. בסיום התהליך קיבלנו כי T_ℓ הוא עץ פורש מינימלי של G ותת־גרף של העץ G_ℓ, ומכאן ש־ $G_\ell=T_\ell$.

תרגיל 4.13 הציגו דוגמה לגרף לא־מכוון שיש בו עץ פורש מינימלי יחיד ועץ מרחקים קצרים ביותר יחיד, והעצים האלה שונים.

פתרון בעמוד 79

תרגיל 4.14 נתון גרף (לא מכוון) קשיר $G=(V,E)$ עם משקלות כלשהן על הקשתות. עבור מסלול בגרף נסמן ב־$\max(P)$ את המשקל של קשת כבדה ביותר במסלול. עבור זוג צמתים נסמן:

$$B(s,t)=\min\{\max(P) : P \text{ is a simple } st-\text{path}\}\ .$$

הציעו אלגוריתם, יעיל ככל שתוכלו, שיחשב לכל זוג צמתים $s, t \in V$ את $B(s,t)$. הוכיחו את נכונות האלגוריתם ונתחו את סיבוכיותו.

פתרון בעמוד 79

4.6 מימוש האלגוריתמים של פרים ושל קרוסקל

קראו את סעיף 4.6 בספר

שימו לב, האלגוריתם של פרים הוא מקרה פרטי של אלגוריתם החתך, שבו אנו קובעים בתחילת האלגוריתם צומת $s \in S$, ובכל איטרציה בלולאה אנו בוחרים את S שיהיה רכיב הקשירות של היער (V,F) המכיל את s.

גם האלגוריתם של קרוסקל הוא מקרה פרטי, שבו אנו בוחרים, בכל צעד בלולאה, את הקשת הזולה ביותר המחברת שני רכיבי קשירות של היער (V,F), ובמקרה זה S הוא אחד משני רכיבי הקשירות של היער (V,F), שממנו יוצאת קשת זו.

בניגוד לגירסה הכללית של אלגוריתם החתך (אלגוריתם 4.1), אפשר לממש את האלגוריתמים של פרים ושל קרוסקל בזמן $O(m \log n)$, אך לא תמיד ניתן להבטיח זמן ריצה כזה לגרסאות אחרות של אלגוריתם החתך.

מימוש האלגוריתם של פרים. בכל צעד בלולאה, כל מה שעלינו לעשות זה למצוא את הקשת הזולה ביותר מבין הקשתות שקצה אחד שלהן נמצא ב־S והקצה השני ב־$V \setminus S$. בספר מוסבר איך לממש את זה בדומה לאלגוריתם של דייקסטרה.

מימוש האלגוריתם של קרוסקל. האלגוריתם מיושם בשני שלבים.

שלב 1: ממיינים את הקשתות לפי סדר עולה של מחירים. סיבוכיות הזמן: $O(m \log m) = O\left(m \log n^2\right) = O(m \log n)$.

שלב 2: מאתחלים $F \leftarrow \emptyset$. לפי סדר המיון, בודקים לכל קשת אם היא מחברת רכיבי קשירות שונים של הפתרון הנוכחי (V,F); אם כן, אנו מוסיפים קשת זו ל־F.

סיבוכיות הזמן של האלגוריתם תלויה בסיבוכיות הזמן הכוללת של m השאילתות "האם הקשת (u,v) מחברת שני רכיבי קשירות שונים?" כדי לממש ביעילות את שלב 2 משתמשים במבנה הנתונים **איחוד־חיפוש** [Union-Find]. מבנה הנתונים הזה מתחזק חלוקה של הקבוצה V, כלומר, אוסף תת־קבוצות זרות שאיחודן הוא V, ותומך בשתי הפעולות האלה:

עדכון : איחוד של שתי קבוצות בחלוקה, כלומר מיזוגן לקבוצה אחת.

שאילתה: בהינתן שני איברים, האם הם שייכים לאותה תת־קבוצה?

בטענה 4.24 בספר מוסבר כי ניתן לממש מבנה נתונים זה בסיבוכיות הזמן הבאה: **אתחול:** $O(n)$. **עדכון:** $O(1)$. **שאילתה:** $O(\log n)$.

במהלך שלב 2 יש $n-1$ עדכונים, ו־m שאילתות. לכן אפשר לממש את שלב 2 בסיבוכיות הזמן $O(m \log n)$.

תרגיל 4.15 הראו כי אפשר לממש את אלגוריתם המחיקה־לאחור בסיבוכיות זמן של $O(mn)$.

פתרון בעמוד 79

תרגיל 4.16 הציגו מימוש של אלגוריתם 4.2 (האלגוריתם של בורווקה) בסיבוכיות זמן של $O(m \log^2 n)$.

פתרון בעמוד 79

4.7 הצברה

קראו בספר את סעיף 4.7

בפרק זה דנים בבעיית הצברה [Clustering] שאפשר לנסח אותה בקצרה כך:

בעיה אלגוריתמית: הצברה בעלת מרווח מרבי.
הקלט: גרף $G = (V, E)$ עם מרחקים $d : E \to [0, \infty)$ על הקשתות, ומספר שלם $k \in \mathbb{N}$.
הפלט: חלוקה של V ל־k תת־קבוצות.
המטרה: המרווח (המרחק בין שתי קבוצות הכי קרובות בחלוקה) הוא מקסימלי.

בספר מוכיחים את הטענה הזו: אם T הוא עץ פורש מינימלי ב־G המחושב על־ידי האלגוריתם של קרוסקל (שימו לב כי הוכחת (4.26) אכן מסתמכת על האלגוריתם של קרוסקל), ו־K היא קבוצת k הקשתות הארוכות ביותר ב־T, אזי רכיבי הקשירות של $T \setminus K$ מהווים את החלוקה האופטימלית. כיוון שהאלגוריתם של קרוסקל יכול לייצר כל עץ פורש מינימלי (ראו תרגיל 4.11 בספר), נובע מכך שהטענה נכונה לכל עץ פורש מינימלי.

4.8 קודי הופמן ודחיסת נתונים

קראו את סעיף 4.8 בספר

בפרק זה דנים בבעיה הזו:

בעיה אלגוריתמית: קוד תחיליות אופטימלי II.
הקלט: קבוצה S ושכיחויות $f : S \to (0, \infty)$.
הפלט: מיפוי $\gamma : S \to \{0,1\}^*$ מ־S לקבוצת המחרוזות הבינאריות, כך שלכל $x, y \in S$ שונים, מתקיים **תנאי התחיליות**: אף אחת מהמחרוזות הבינאריות $\gamma(x)$, $\gamma(y)$ אינה תחילית של האחרת.
מטרה: להביא למינימום את הסכום $\sum_{x \in S} f_x |\gamma(x)|$.

הערה: בספר מניחים ש־ $\sum_{x \in S} f_x = 1$. אפשר להניח זאת ללא הגבלת הכלליות על־ידי הנרמול $f'_x = f_x / A$, כאשר $A = \sum_{x \in S} f_x$.

האבחנה הראשונה בתכנון האלגוריתם היא כי כל פתרון אפשרי לבעיה ניתן לייצג על־ידי עץ בינארי שקבוצת העלים שלו היא S. כדי "לקרוא" מהעץ את המחרוזות $\gamma(x)$, רושמים "0" על כל קשת המובילה מאב לבן שמאלי, ורושמים "1"

על כל קשת המובילה מאב לבן ימני. אם כן, לכל $x \in S$, המחרוזת שרשומה על המסלול P_x מהשורש לעלה x היא $\gamma(x)$. איור 4.16 בספר ממחיש זאת היטב. תנאי התחיליות מתקיים כי לכל שני עלים x, y, אף אחד מהמסלולים P_x, P_y איננו תת־מסלול של האחר. כל עץ בינארי המכיל קבוצת עלים S, מגדיר באופן זה קוד γ המקיים את תנאי התחיליות. למעשה, לא קשה לראות כי לכל קוד המקיים את תנאי התחיליות מתאים עץ אחד ויחיד כזה. כלומר, יש התאמה חד־ערכית בין קודים של S המקיימים את תנאי התחיליות לבין העצים הבינאריים שקבוצת העלים שלהם היא S.[3]

נסמן ב־$\mathrm{depth}_T(x)$ את אורך המסלול P_x מהשורש ל־x בעץ T.

תרגיל 4.17 הוכיחו שלכל קוד $\gamma : S \to \{0,1\}^*$ המקיים את תנאי התחיליות, מתקיים: $\sum_{x\in S} 2^{-|\gamma(x)|} \le 1$.

פתרון בעמוד 80

כיוון שהאורך של המחרוזת $\gamma(x)$ שווה ל־$\mathrm{depth}_T(x)$, אנו מקבלים:

$$\sum_{x\in S} f_x|\gamma(x)| = \sum_{x\in S} f_x \cdot \mathrm{depth}_T(x).$$

לכן, חישוב קוד אופטימלי לתחיליות שקול לבעיה הזו:

בעיה אלגוריתמית: קוד תחיליות אופטימלי.
הקלט: קבוצה S ושכיחויות $f : S \to (0,\infty)$.
הפלט: עץ בינארי T המכיל קבוצת עלים S.
המטרה: למזער את הביטוי $\sum_{x\in S} f_x \,\mathrm{depth}_T(x)$.

האבחנה השנייה היא: בהינתן עץ בינארי T שיש לו $k = |S|$ עלים, אפשר למצוא השמה $\tau : S \to T$ של איברי S לעלים של העץ, כך שהסכום $\sum_{x\in S} f_x \,\mathrm{depth}_T(\tau(x))$ יהיה מינימלי בעזרת האסטרטגיה החמדנית הזו: ממיינים את העלים של העץ בסדר עולה לפי המרחק מהשורש, נניח $v_1, \ldots, v_k$, כאשר v_1 הוא עלה קרוב ביותר לשורש ו־v_k הוא עלה רחוק ביותר מהשורש. ממיינים את איברי S בסדר יורד לפי השכיחויות, נניח $x_1, \ldots, x_k$. קל לוודא כי ההשמה האופטימלית לכל i עבור העץ הנתון היא $\tau(x_i) = v_i$. כלומר, האסטרטגיה החמדנית כאן קובעת כי "ככל שהמסלול ארוך יותר, השכיחות קטנה יותר". ובפרט:

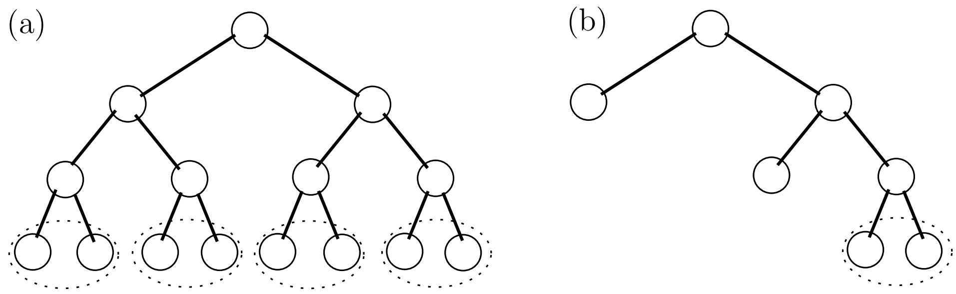
> שני האיברים שהשכיחויות שלהם הן הקטנות ביותר, יותאמו לשני העלים הרחוקים ביותר מהשורש.

האבחנה השלישית היא כי כל פתרון אופטימלי לבעיה לעיל הוא בהכרח עץ בינארי מלא, כלומר, לכל צומת שאינו עלה יש בדיוק שני בנים. אחרת, אם יש צומת u שיש לו בן יחיד, נוכל למחוק את u ולחבר את הבן היחיד של u ישירות לאב של u. בכך יוקטן ערך הפתרון, כי האורך של מסלול אחד לפחות יקטן ב־1, בעוד שאר המסלולים לא ישתנו.

האבחנה האחרונה מאפשרת לנו להפעיל רקורסיה. נניח שנתונים שני איברים $x, y \in S$ כך שקיים עץ אופטימלי, שבו x ו־y הם עלים אחים, כלומר יש להם אב

[3] הכוונה לעצים בינאריים עם סדר (שמאל/ימין) על בניו של כל צומת, כמו בעצי חיפוש בינאריים.

משותף. שימו לב שהעומק שלהם ב־T שווה. ייתכן שיש הרבה זוגות של אחים כאלה, למשל בעץ בינארי מאוזן (ראו איור 4.7(a)), אבל ייתכן גם שיש רק זוג אחד, למשל בעץ בינארי שהוא כמעט "שרוך" (ראו איור 4.7(b)).



איור 4.7: דוגמה לזוגות עלים שהם אחים

התכונה הבאה של עצים בינאריים מלאים מתקיימת בכל עץ בינארי מלא: הבנים של הצומת הפנימי הרחוק ביותר מהשורש הם עלים אחים. לכן, אלגוריתם ההשמה החמדן יוכל להתאים את האיברים x ו־y עם שתי השכיחויות הקטנות ביותר לעלים־אחים הללו. כעת נוכל לאחד את x ו־y לאיבר אחד (שיחליף את שניהם), נסמן אותו ב־w, והוא יהיה בעל שכיחות $f_w = f_x + f_y$. בדרך הזו קיבלנו בעיה קטנה יותר שניתן לפתור אותה ברקורסיה.

תרגיל 4.18 הריצו את האלגוריתם תוך פירוט כל השלבים על הקלט הבא (זוהי למעשה דוגמה מהספר):

$$S = \{a, b, c, d, e\}\,, \ \ (f_a, f_b, f_c, f_d, f_e) = (32, 25, 20, 18, 5)\,.$$

פתרון בעמוד 81

תרגיל 4.19 יהי T עץ בינארי שקבוצת עליו היא S, והשכיחויות על העלים הן $f : S \to (0, \infty)$. נרחיב את f ל־T, $f : T \to (0, \infty)$, כך ש־f_u יהיה סכום השכיחויות של כל העלים שהם צאצאים של u ב־T.

הוכיחו את הטענה הזו: אם קיימים $u, v \in T$ כך ש־$f_v < f_u$, וגם $\mathrm{depth}_T(u) < \mathrm{depth}_T(v)$ אז T מייצג קוד תחיליות שאינו אופטימלי ביחס לשכיחויות f.

פתרון בעמוד 81

תרגיל 4.20 בהינתן קבוצת תווים S שהשכיחויות שלהם הן $f : S \to (0, \infty)$, פתחו אלגוריתם ליצירת קוד תחיליות $\gamma : S \to \{0, 1, 2\}^*$ שימזער את $\sum_{x \in S} f_x |\gamma(x)|$. הוכיחו את הנכונות ונתחו את הסיבוכיות. כדי לקצר את התשובה במקצת, אתם יכולים להניח ש־$|S| \geq 3$ ואי־זוגי.

פתרון בעמוד 84

4.9 עץ מושרש זול ביותר בגרף מכוון: אלגוריתם חמדן דו־שלבי*

סעיף זה הוא רשות – אם ברצונכם להיחשף לארגומנט חמדני מתוחכם יותר

קראו בספר את סעיף 4.9

הסעיף הנוכחי עוסק בבעיה הזו:

בעיה אלגוריתמית: עץ מכוון פורש בעל עלות מינימלית [Minimum Cost Arborescence].
הקלט: גרף מכוון $G = (V, E)$ עם מחירים אי־שליליים $c : E \to [0, \infty)$ על הקשתות, ושורש $r \in V$.
הפלט: תת־עץ T מכוון פורש המושרש ב־r של G.
המטרה: למזער את העלות של T.

נזכיר כי גרף מכוון T הוא עץ מכוון המושרש ב־r אם:
(A) קיים ב־T מסלול מכוון מ־r לכל צומת אחר;
(B) גרף התשתית של T (הגרף הלא־מכוון המתקבל על־ידי התעלמות מכיווני הקשתות) הוא עץ.
כפי שמוכיחים בטענה (4.34) בספר, אפשר להחליף את תנאי (B) בתנאי הזה:
(C) ב־T, דרגת הכניסה של כל צומת שאינו r היא בדיוק 1.

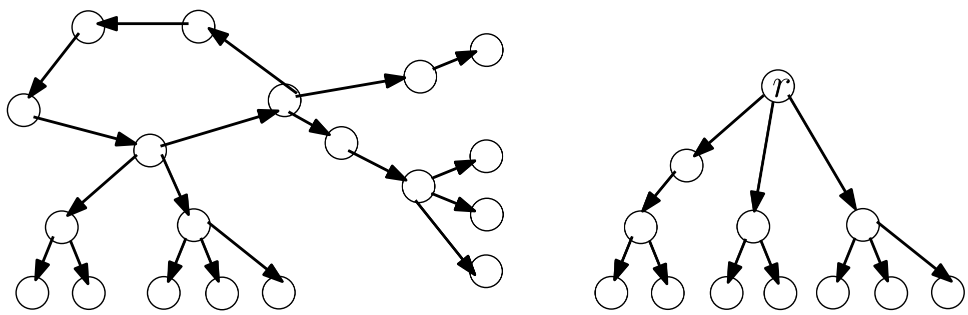
מתנאי (C) נובע כי עלינו לבחור לכל צומת $v \in V \setminus \{r\}$ בדיוק קשת אחת מבין הקשתות הנכנסות ל־v. אם לכל צומת $v \in V \setminus \{r\}$ נבחר את הקשת הזולה ביותר שנכנסת ל־v, נקבל גרף שעלותו אינו עולה על זו של הפתרון האופטימלי. הבעיה שגרף כזה אינו מכיל בהכרח מסלול מ־r לצמתים האחרים. למעשה, אפשר לאפיין גרפים כאלה באופן הבא:

טענה 4.8 יהי $H = (V, F)$ גרף מכוון, יהי $r \in V$, ונניח שב־H דרגת הכניסה של כל צומת $v \in V \setminus \{r\}$ היא 1 ודרגת הכניסה של r היא אפס. בתנאים אלה H הוא איחוד של תת־גרפים זרים בצמתים ובקשתות וכל תת־גרף הוא אחד משני הטיפוסים האלה:

- מעגל מכוון שעליו תלויים עצים מכוונים, שהשורש שלהם הוא אחד מצומתי המעגל, בדומה לחלק השמאלי של איור 4.8; או
- עץ פורש מכוון שהשורש שלו הוא r, בדומה לחלק הימני שלאיור 4.8.

תרגיל 4.21 הוכיחו את טענה 4.8.

הרעיון באלגוריתם הוא להשתמש ברקורסיה. זה נעשה בשני שלבים. בשלב הראשון מעדכנים את המחירים באופן הזה: לכל צומת $v \in V \setminus \{r\}$, מעדכנים את המחיר של כל קשת e שנכנסת ל־v, כך שהוא יהיה $c'(e) \leftarrow c(e) - y_v$, כאשר y_v הוא המחיר המינימלי של קשת ב־G שנכנסת ל־v. כפי שמוסבר בספר, מתקבלת בעיה שקולה, מפני שהאופטימום קטן ב־$\sum_{v \in V \setminus \{r\}} y_v$ בדיוק. כלומר, T הוא פתרון אופטימלי ביחס למחירים החדשים $c'(e)$ אם ורק אם הוא היה פתרון



איור 4.8: דוגמה לגרף שבו דרגת הכניסה של כל צומת היא 1, והוא אינו עץ פורש מכוון.

אופטימלי ביחס למחירים המקוריים $c(e)$. אם הקשתות שמחירן אפס מכילות עץ מכוון כנדרש, אזי סיימנו. אחרת, בשלב השני, הקשתות שמחירן אפס מכילות מעגל מכוון. מכווצים את המעגל לצומת אחד, ומפעילים את אותו אלגוריתם על הגרף המתקבל. בשלב שבו האלגוריתם מחזיר עץ מכוון, קיימת אפשרות שזהו עץ פורש מכוון בגרף הקלט, ואז סיימנו, או שהיה מעגל שכווץ בשלב הקודם. במקרה האחרון, פותחים את המעגל שכווץ בשלב הקודם, ומשמיטים ממנו קשת אחת (יש קשת יחידה כזו) כדי לקבל עץ מכוון.

תרגיל 4.22 הריצו את האלגוריתם על הדוגמה המוצגת בספר באיור 4.19.

פתרון בעמוד 85

4.10 פתרונות לתרגילים

פתרון תרגיל 4.1 מעמוד 55

ערך הפתרון שהאלגוריתם מחשב הוא לפחות מחצית הערך האופטימלי, ובאופן כללי נניח כי ברשותנו פרוצדורה המקיימת את התכונה הבאה: בכל צעד, ערך הפתרון החלקי $\mathcal{I}'$ גדל ב־1, ואילו ערך הפתרון האופטימלי לבעיה השיורית קטן ב־$\alpha \geq 1$ לכל היותר. במקרה זה, ערך הפתרון שהאלגוריתם מחשב הוא לפחות $1/\alpha$ פעמים הערך האופטימלי. ♣

פתרון תרגיל 4.2 מעמוד 55

נשתמש באבחנה הפשוטה הזו: אם מקטע J חותך שלושה מקטעים מתואמים I_1, I_2, I_3 (כלומר, $I_i \cap I_j = \emptyset$ לכל $1 \leq i < j \leq 3$), אזי J מכיל ממש לפחות מקטע אחד מהמקטעים I_1, I_2, I_3. ובפרט, אם $\mathcal{O}$ הוא פתרון אפשרי כלשהו לבעיית תזמון המקטעים, אזי המקטע הקצר ביותר בקלט $\mathcal{I}$ חותך לכל היותר שני מקטעים של $\mathcal{O}$ (אחרת הוא צריך להכיל ממש מקטע מ־$\mathcal{O}$, אך זה בלתי אפשרי עבור מקטע קצר ביותר. לכן:

> בכל צעד, ערך הפתרון החלקי $\mathcal{I}'$ גדל ב־1, ואילו ערך הפתרון האופטימלי לבעיה השיורית קטן ב־2 לכל היותר.

מכאן, שעל־פי תרגיל 4.1, האלגוריתם מחשב פתרון שערכו לפחות מחצית מהאופטימום. ♣

פתרון תרגיל 4.3 מעמוד 55

כפי שנראה בהמשך, האלגוריתם שפותר את בעית חלוקת המקטעים מחשב גם את $d(\mathcal{I})$, אבל החישוב של $d(\mathcal{I})$ בלבד הוא פשוט יותר.

Algorithm 4.3 Compute_$d(\mathcal{I} = \{(s_1, f_1), \ldots, (s_n, f_n)\})$

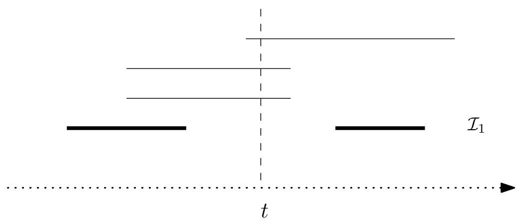
Require: $s_i < f_i, \forall i \in \{1, \ldots, n\}$
  $(t_1, \ldots, t_{2n}) \leftarrow$ Sort $\{s_i, f_i : \ i \in \{1, \ldots, n\}\}$ breaking ties such that finish events are placed before starting events.
  $d \leftarrow 0$
  $curr\_d \leftarrow 0$
  for $i \leftarrow 1, \ldots, 2n$ **do**
    if t_i is a beginning of interval **then**
      $curr\_d \leftarrow curr\_d + 1$
      $d \leftarrow \max\{d, curr\_d\}$
    else
      $curr\_d \leftarrow curr\_d - 1$
  return d

נכונות האלגוריתם ברורה. אפשר להוכיח זאת פורמלית על־ידי הוכחה באינדוקציה על i של הטענה הבאה:

> נניח שערכי t_i שונים בזוגות. אז בסיום ביצוע האיטרציה ה־i באלגוריתם 4.3 ערכו של $curr\_d$ שווה ל־$d(t_i)$.

אנו משאירים לקורא להוכיח טענה זאת. ♣

פתרון תרגיל 4.4 מעמוד 57
התשובה היא שלילית. האלגוריתם החמדן, המחשב קבוצה מקסימלית של מקטעים זרים בזוגות, אינו מכסה בהכרח את קבוצת המקטעים בעומק מקסימלי. ראו דוגמה נגדית באיור 4.9.



איור 4.9: קבוצת המקטעים המקסימלית $\mathcal{I}_1$ אינה מכילה מקטעים הנמצאים בעומק מקסימלי.

באיור 4.9, הנקודה t היא בעלת העומק המקסימלי 3, אבל היא אינה מכוסה על־ידי $\mathcal{I}_1$ המורכב משני המקטעים המודגשים. ♣

פתרון תרגיל 4.5 מעמוד 57
נפעיל את האלגוריתם החמדן הרגיל שבספר. קבוצת המקטעים $\mathcal{I}$ שהוא יחזיר תהיה מקסימלית בגודלה. נשים לב כי I_j, I'_j נחתכים, מפני שהאורך של כל אחד מהם הוא לפחות 2 ומפני שהאחד מתקבל מהשני על־ידי הזזה ביחידה אחת. לכן יתקיים התנאי "לכל j מקטע אחד לכל היותר מתוך I_j, I'_j שייך ל־$\mathcal{I}$". ♣

פתרון תרגיל 4.6 מעמוד 58
1. התשובה היא שלילית, ראו דוגמה נגדית באיור 4.10. האלגוריתם ישבץ תחילה את המשימה 1 ואחריה את המשימה 2. האיחורים במקרה זה הם

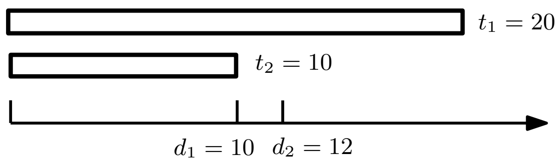
$$\ell_1 = 20 - 10 = 10, \qquad \ell_2 = 30 - 12 = 18,$$

והסכום הוא 28.
אם נשבץ תחילה את המשימה 2 ואחריה את המשימה 1, האיחורים יהיו

$$\ell_2 = \max\{10 - 12, 0\} = 0, \qquad \ell_1 = 30 - 10 = 20,$$

והסכום יהיה 20 בלבד.

2. תחילה עלינו להבהיר מה הכוונה באלגוריתם המתעלם מזמני הביצוע, כיוון שרשמית האלגוריתם צריך להחזיר את הזמנים של תחילת הריצה, אך כל עוד זמני הביצוע אינם ידועים, האלגוריתם לא יוכל להשיב עם זמני ביצוע בתזמון אפשרי. למרות זאת השאלה היא הגיונית, כיוון שבלי הגבלת הכלליות ניתן להניח שהמשימות מתוזמנות (מזמן 0) ברציפות – כפי שהנחנו בבעיה המקורית (תזמון מקטעים לשם מזעור האיחור). מכאן שמספיק להגדיר את סדר ביצוע המשימות כדי להביא למינימום את האיחור המקסימלי.



איור 4.10: דוגמה לקלט שעבורו התזמון הממזער את האיחור המקסימלי אינו ממזער את סכום האיחורים.

כדי להוכיח שכל אלגוריתם המתעלם מזמני הביצוע t_i אינו יכול לחשב סדר משימות שימזער את $\sum_i \ell_i$, נציג שני מופעים של הבעיה שבהם מספר המשימות יהיה זהה, וגם המועדים האחרונים לביצוע המשימות יהיו זהים, אך זמני הביצוע יהיו שונים. אנו נראה שלמופעים השונים יש סדרי משימות אופטימליים שונים. מובן שאלגוריתם המתעלם מזמני הביצוע אינו יכול להבדיל בין המופעים, ולכן הוא מחויב להחזיר את אותו פתרון, מכאן שלפחות באחד מן המופעים, האלגוריתם יחזיר פתרון שאינו הפתרון האופטימלי.

- במופע הראשון $n = 2$, $d_1 = 1$, $d_2 = 2$, $t_1 = t_2 = 1$. בסידור שבו משימה 1 מבוצעת לפני משימה 2, נקבל $s_1 = 0$, $s_2 = 1$, ולכן

$$\ell_1 + \ell_2 = \max\{s_1 + t_1 - d_1, 0\} + \max\{s_2 + t_2 - d_2, 0\} = 0 + 0 = 0.$$

בסידור שבו משימה 2 מבוצעת לפני משימה 1 נקבל $s_1 = 1$, $s_2 = 0$, ולכן

$$\ell_1 + \ell_2 = \max\{s_1 + t_1 - d_1, 0\} + \max\{s_2 + t_2 - d_2, 0\} = 1 + 0 = 1.$$

- במופע השני $n = 2$, $d_1 = 1$, $d_2 = 2$, $t_1 = 3$, $t_2 = 2$. בסידור שבו משימה 1 מבוצעת לפני משימה 2, נקבל $s_1 = 0$, $s_2 = 3$, ולכן

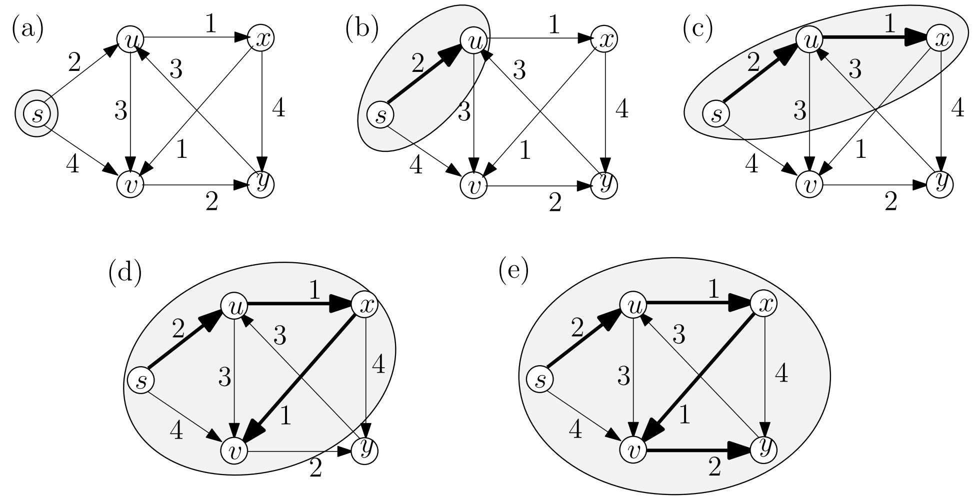
$$\ell_1 + \ell_2 = \max\{s_1 + t_1 - d_1, 0\} + \max\{s_2 + t_2 - d_2, 0\} = 2 + 3 = 5.$$

בסידור שבו משימה 2 מבוצעת לפני משימה 1 נקבל $s_1 = 2$, $s_2 = 0$, ולכן

$$\ell_1 + \ell_2 = \max\{s_1 + t_1 - d_1, 0\} + \max\{s_2 + t_2 - d_2, 0\} = 4 + 0 = 4.$$

♣

פתרון תרגיל 4.7 מעמוד 61
כל שלבי הבנייה של עץ המרחקים הקצרים ביותר מודגמים באיור 4.11. במקרה הזה נוכחנו שהעץ הוא מסלול. שימו לב, בשלב (d) יכולנו לצרף את הצומת v ישירות מ־s ולא דרך הצומת x (איזה עץ מרחקים קצרים ביותר היה מתקבל במקרה זה?). ♣



איור **4.11**: שלבי הריצה של אלגוריתם דייקסטרה מהצומת s בגרף שתואר באיור 4.4.

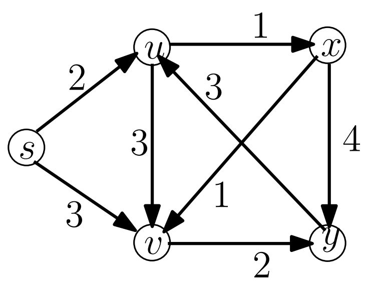
פתרון תרגיל 4.8 מעמוד 61
נוסיף שני צמתים חדשים s, t. נחבר את s לכל צומת ב־S על־ידי קשת במשקל 0, ונחבר כל צומת ב־T ל־t על־ידי קשת במשקל 0. בגרף המתקבל נשתמש באלגוריתם דייקסטרה לחישוב המסלול הקצר ביותר P' המוביל מ־s ל־t. קל לראות כי המסלול P המתקבל מ־P' ע"י זריקת הצומת הראשון (s) והאחרון (t), הוא המסלול הקצר ביותר, כנדרש. זמן הריצה של האלגוריתם הוא כמו זה של אלגוריתם דייקסטרה. ♣

פתרון תרגיל 4.9 מעמוד 62
האלגוריתם החדש אינו מחשב את המרחקים הקצרים ביותר מצומת נתון. לדוגמה, באיור 4.12 אפשר לראות את ריצת האלגוריתם הזה על הגרף, החל מצומת s. בהתחלה $S = \{s\}$. הצומת הקרוב ביותר ל־s הוא u; בשלב השני u מצטרף ל־S ול־$T = (\{s, u\}, \{(s, u)\})$. בשלב השלישי, הצומת הקרוב ביותר לצומת מ־S הוא x, שמצטרף ל־S. כעת $T = (\{s, u, x\}, \{(s, u), (u, x)\})$.

בשלב הרביעי הקשת הקלה ביותר היוצאת לצומת מ־S היא (x, v), לכן מתקבל העץ $T = (\{s, u, x, v\}, \{(s, u), (u, x), (x, v)\})$. בעץ הזה המרחק מ־$s$ ל־v הוא 4, בעוד שהמרחק המינימלי מ־s ל־v בגרף המקורי היה 3. ♣

פתרון תרגיל 4.10 מעמוד 62
נסמן ב־$d_T(u, v)$ את המרחק בין הצמתים $u, v \in V$ בעץ T, ונסמן ב־$d_G(u, v)$ את המרחק ביניהם בגרף G. יהיו u, v הצמתים שהמרחק ביניהם ב־T הוא מקסימלי. אזי מתקיים:

$$\begin{aligned} D(T, \ell|_T) = d_T(u, v) &\le d_T(u, s) + d_T(v, s) \\ &= d_G(u, s) + d_G(v, s) \le 2D(G, \ell). \end{aligned}$$



איור 4.12:

אי־השוויון הראשון הוא למעשה אי־שוויון המשולש שמקיימת כל פונקציית מרחקים; השוויון הרשום אחריו נובע מהעובדה שהמרחקים ב־G, מ־s לכל צומת, שווים למרחקים האלה ב־T. ♣

פתרון תרגיל 4.11 מעמוד 64

ברור כי האלגוריתם מחשב תת־גרף פורש קשיר (V,F), ונותר רק להוכיח כי גרף זה הוא עץ. למעשה נטען כי במהלך האלגוריתם (V,F) הוא יער, כלומר, אין ב־F מעגלים. עובדה זו נובעת משתי האבחנות הבאות.

1. בכל איטרציה באלגוריתם אנו מוסיפים קשת המחברת שני רכיבי קשירות שונים של (V,F),
2. אם נוסיף ליער כלשהו קשת המחברת שני רכיבי קשירות שונים נקבל שוב יער.

את הוכחת אבחנה 2 אנו משאירים לקורא. אבחנה 1 נכונה בגלל שבכל איטרציה S מקיימת $F \cap E(S, V \setminus S) = \emptyset$. לכן (הבהירו לעצמכם מדוע), כל רכיב קשירות של (V,F) או שכולו ב־S, או שכולו ב־$V \setminus S$. מכאן נובע כי כל קשת בחתך $E(S, V \setminus S)$, ובפרט הקשת שמוסיף האלגוריתם, מחברת שמי רכיבי קשירות שונים של (V,F). ♣

פתרון תרגיל 4.12 מעמוד 65

1. על־פי הגדרתו, אלגוריתם 4.2 מחזיר קבוצת קשתות F כך שהגרף $T = (V,F)$ הוא קשיר. אנו נוכיח כי בתת־הגרף הזה אין מעגלים. לשם כך מספיק להוכיח כי בכל איטרציה של לולאת **while**, אם F הוא יער (בתחילת האיטרציה) אזי גם $F \cup \{e_W : W \in \mathcal{W}\}$ הוא יער; כאן $\mathcal{W}$ היא קבוצת רכיבי הקשירות של היער F, ולכל $W \in \mathcal{W}$ הקשת e_W היא הקשת הקלה ביותר היוצאת מ־W, כלומר הקשת הקלה ביותר מבין קשתות החתך $E(W, V \setminus W)$.

נניח, בשלילה, כי $F \cup \{e_W : W \in \mathcal{W}\}$ מכיל מעגל C. כיוון שכל קשת בקבוצה $\{e_W : W \in \mathcal{W}\}$ מחברת בין רכיבי קשירות שונים של F, המעגל C לא יכול להיות תחום בתוך רכיב קשירות יחיד של F. נסמן ב־$(W_1, W_2, \ldots, W_\ell, W_1)$, $\ell \geq 2$ את סדרת רכיבי הקשירות ב־F שהמעגל C עובר בהם. נתבונן בקשת e המחברת בין רכיבי הקשירות W_1 ו־W_2; ייתכן שיש שתי קשתות כאלה ב־C אם $\ell = 2$. הקשת e היא בהכרח הקשת הקלה ביותר מבין הקשתות היוצאות מ־W_1 או הקשת הקלה

ביותר מבין הקשתות היוצאת מ־W_2 (או משניהם). נניח, ללא הגבלת הכלליות, ש־e היא הקשת הקלה ביותר היוצאת מ־W_2.

במקרה $\ell = 2$ יש עוד קשת $e' = \{W_2, W_1\}$, וזו הקשת הקלה ביותר היוצאת מ־W_1. לכן $c(e') < c(e)$ (לא ייתכן שוויון, כי אין שתי קשתות בעלות אותו משקל). אבל e' יוצאת גם מ־W_2, וכיוון ש־e היא הקשת הקלה ביותר שיוצאת מ־W_2 נקבל $c(e) < c(e')$. זו סתירה.

נניח אם־כן ש־$\ell \geq 3$ ונתבונן בקשת $\{W_2, W_3\}$. כיוון שאין שתי קשתות בעלות אותו משקל, והיא אינה הקשת הקלה ביותר היוצאת מ־W_2, לכן היא בהכרח הקשת הקלה ביותר היוצאת מ־W_3, וכן $c(\{W_2, W_3\}) > c(\{W_2, W_1\})$. באופן דומה (באינדוקציה) אנו מקבלים ש־$\{W_i, W_{i+1}\}$ היא בהכרח הקשת הקלה ביותר היוצאת מ־W_{i+1} וש־$c(\{W_i, W_{i+1}\}) > c(\{W_i, W_{i-1}\})$ לכל $i \in \{2, \ldots, \ell\}$. בפרט $\{W_\ell, W_{\ell-1}\}$ היא הקשת הקלה ביותר היוצאת מ־W_ℓ וכן

$$c(\{W_\ell, W_{\ell-1}\}) > c(\{W_{\ell-1}, W_{\ell-2}\}) > \ldots > c(\{W_2, W_1\}).$$

כעת נעשה עוד צעד אחד: הקשת $\{W_\ell, W_1\}$ אינה הקשת הקלה ביותר היוצאת מ־W_ℓ, לכן היא הקשת הקלה ביותר היוצאת מ־W_1 וכן

$$c(\{W_\ell, W_1\}) > c(\{W_\ell, W_{\ell-1}\}) > c(\{W_2, W_1\}),$$

וזאת סתירה.

הוכחנו שאלגוריתם בורובקה מחזיר עץ פורש. נותר להוכיח שהוא מחזיר עץ פורש מינימלי. נוכיח כי כל קשת בעץ המתקבל חייבת להופיע בכל העצים הפורשים המינימליים של הגרף G. לשם כך נפעיל את טענה (4.17) בספר: נתבונן בקשת כלשהי $e = \{u, v\}$ שאלגוריתם 4.2 החזיר. זה קרה כי ללא הגבלת הכלליות, במהלך ריצת אלגוריתם 4.2 נוצר רכיב קשירות U שמכיל את u, והקשת e היא הקשת הקלה ביותר היוצאת מ־U, כלומר היא הקשת הקלה ביותר החוצה את החתך $E(U, V \setminus U)$; לפי טענה (4.17) בספר (וכיוון שהנחנו שאין שתי קשתות עם משקל שווה ב־G), e בהכרח נמצאת בעץ הפורש המינימלי. כיוון שכל קשתות העץ הפורש, שאלגוריתם בורובקה מחזיר, חייבות להיות בכל העצים הפורשים, אנו מסיקים שלגרף יש עץ פורש מינימלי יחיד, והוא העץ המוחזר על־ידי אלגוריתם בורובקה.

2. אנו נוכיח שלאחר כל איטרציה של לולאת **while**, מספר רכיבי הקשירות בגרף (V, F) קטן פי 2 לפחות. בתחילת האלגוריתם $F = \emptyset$ ומספר רכיבי הקשירות הוא n, לכן לאחר t איטרציות, מספר רכיבי הקשירות יהיה לכל היותר $n/2^t$ ולכן ברור שאלגוריתם 4.2 אינו מגיע לאיטרציה $\lfloor \log_2 n \rfloor + 1$ כיוון שאחרת מספר רכיבי הקשירות היה לכל היותר

$$\frac{n}{2^{\lfloor \log_2 n \rfloor + 1}} < \frac{n}{2^{\log_2 n}} = 1,$$

אך זה לא ייתכן, כי האלגוריתם נעצר ברגע שיש רכיב קשירות אחד.

נותר להוכיח כי בכל איטרציה של לולאת **while**, מספר רכיבי הקשירות ב־(V, F) קטן לפחות פי 2. לשם כך נטען כי כל רכיב קשירות של הגרף (V, F') כאשר $F' = F \cup \{e_W : W \in \mathcal{W}\}$ מכיל לפחות שני רכיבי קשירות מתוך קבוצת רכיבי הקשירות $\mathcal{W}$ של F. הסיבה לכך היא כי לכל רכיב $W \in \mathcal{W}$ יש בקבוצה

$F' \setminus F = \{e_W : W \in \mathcal{W}\}$ לפחות קשת אחת שיוצאת מ־W לרכיב קשירות נוסף $W' \in \mathcal{W} \setminus \{W\}$, ולכן בהכרח W, W' יהיו מוכלים שניהם באותו רכיב הקשירות של F'. ♣

פתרון תרגיל 4.13 מעמוד 66

הדוגמה הכי פשוטה היא מעגל באורך 3 ומשקלי קשתות 2, 3, ו־4, כאשר צומת המקור s אינו קצה של הקשת שמשקלה 2. קל לוודא כי העץ הפורש המינימלי מורכב מהקשתות שמשקליהן 2 ו־3, ואילו עץ המרחקים הקצרים ביותר מ־s מורכב מהקשתות שמשקליהן 3 ו־4. ♣

פתרון תרגיל 4.14 מעמוד 66

האלגוריתם מוצא עץ פורש מינימלי T ב־G ולכל זוג צמתים $s, t \in V$, $B(s,t)$ הוא המשקל של הקשת הכבדה ביותר במסלול היחיד בין s ל־t ב־T.

נוכיח את נכונות האלגוריתם. יהיו $s, t \in V$, ויהי P המסלול היחיד בין s ל־t ב־T. תהי e קשת כבדה ביותר ב־P. עלינו להראות כי $w(e) = B(s,t)$. נניח, בשלילה, שזה לא כך. במקרה כזה קיים ב־G מסלול P' בין s ל־t כך שמתקיים $\max(P') < w(e)$; כלומר, כל קשת $e' \in P$ מקיימת את התנאי $w(e') < w(e)$. עתה, $T \setminus \{e\}$ מכיל שני רכיבי קשירות, S ו־$V \setminus S$, כאשר $s \in S, t \in V \setminus S$. נתבונן בחתך $\delta(S)$ של G. כיוון ש־T הוא עץ פורש מינימלי, אזי e היא קשת קלה ביותר בחתך הזה (תנאי החתך). כמו־כן, לפחות קשת אחת של P', נניח e', נמצאת בחתך הזה. אבל אז $w(e) < w(e')$, בסתירה להנחה כי $w(e') < w(e)$ לכל קשת $e' \in P$.

סיבוכיות: הזמן הדומיננטי הוא הזמן הדרוש לחישוב ערכי $B(s,t)$ לאחר מציאת T; אפשר למצוא את הקשת הכבדה ביותר לכל זוג $s, t \in V$ במסלול שבין s ל־t ב־T בזמן $O(|V|)$. לכן הסיבוכיות הכוללת של האלגוריתם היא $O(|V|^3)$. ♣

פתרון תרגיל 4.15 מעמוד 67

תחילה מוצאים בגרף איזשהו עץ פורש T, למשל על־ידי אלגוריתם BFS. הסיבוכיות היא $O(m)$. לאחר־מכן, כל עוד $E \setminus T \neq \emptyset$, בוחרים קשת $e \in E \setminus T$, ומבצעים את הפעולות האלה:

1. הגרף $T \cup \{e\}$ מכיל מעגל יחיד; מוצאים את הקשת היקרה ביותר e' במעגל זה.
2. מעדכנים את T ואת E: $T \leftarrow (T \cup \{e\}) \setminus \{e'\}$, $E \leftarrow E \setminus \{e'\}$.

את הפעולות האלה אפשר לממש בזמן $O(n)$, וכיוון שיש לכל היותר m פעולות כאלה, הסיבוכיות הכוללת היא $O(mn)$. ♣

פתרון תרגיל 4.16 מעמוד 68

כפי שהוכחנו בתרגיל 4.12, לולאת **while** של אלגוריתם 4.2 מתרחשת לכל היותר $\log_2 n$ פעמים, ולכן מספיק להציג את המימוש של אלגוריתם 4.2 שבו הסיבוכיות של כל איטרציה בלולאת **while** היא $O(m \log n)$. המימוש יהיה דומה למימוש של אלגוריתם קרוסקל. נתחזק את רכיבי הקשירות של (V, F) במבנה הנתונים איחוד־חיפוש. מימוש האיטרציה יבוצע כמפורט באלגוריתם 4.4.

עיקר הזמן במימוש הזה מתרחש בלולאה על הקשתות ב־E (יש m קשתות כאלה) ובתוך הלולאה הזאת, הזמן הדרוש לפעולת החיפוש הוא $O(\log n)$. לכן קיבלנו את זמן הריצה הדרוש.

Algorithm 4.4 Boruvka-Iteration-Implementation

Let $G=(V,E)$ be the original graph
Let W_0 be the set of connected componenets of (V,F)
for every $\hat{w}\in W_0$ **do**
  set $e(\hat{w})\leftarrow \mathsf{nil}$ {Assume $c(\mathsf{nil})=\infty$}
for every $e=\{u,v\}\in E$ **do**
  $\hat{u}\leftarrow$ Find connected component of u
  $\hat{v}\leftarrow$ Find connected component of v
  if $\hat{u}\neq\hat{v}$ **then**
    if $c(e)<c(e(\hat{u}))$ **then**
      $e(\hat{u})\leftarrow e$
    if $c(e)<c(e(\hat{v}))$ **then**
      $e(\hat{v})\leftarrow e$
for $\hat{w}\in W_0$ **do**
  Let $e(\hat{w})=\{u,v\}$
  $\hat{u}\leftarrow$ Find connected component of u
  $\hat{v}\leftarrow$ Find connected component of v
  if $\hat{u}\neq\hat{v}$ **then**
    Add $e(\hat{w})$ to F
    Union$(\hat{u},\hat{v})$

הערה. למרות זמן הריצה של אלגוריתם בורובקה שלכאורה הוא פחות טוב, גרסאות שלו משמשות באלגוריתמים המהירים ביותר לחישוב עץ פורש מינימלי.
♣

פתרון תרגיל 4.17 מעמוד 69

אנו ניעזר בייצוג קוד תחיליות באמצעות עץ בינארי מושרש. לפי השקילות הזאת מספיק להוכיח כי לכל עץ בינארי T עם קבוצת עלים S מתקיים:

$$\sum_{x\in S}2^{-\operatorname{depth}_T(x)}\leq 1. \tag{4.4}$$

ההוכחה של (4.4) מתבצעת באינדוקציה על מספר הצמתים של T. כאשר $|T|=1$, יהיה ל־T עלה אחד בלבד, שהוא גם השורש ולכן עומקו 0, ואמנם $2^{-0}=1$.

כאשר $|T|>1$, נסמן ב־x_0 את העלה העמוק ביותר ב־T וב־w_0 את אביו של x_0; אם ל־w_0 יש בן נוסף נסמנו ב־y_0; אחרת נסמן $y_0=x_0$. שימו לב, גם y_0 הוא עלה ב־T (אחרת צאצא של y_0 היה עלה עמוק יותר מ־x_0). כעת נגדיר עץ T' הדומה ל־T, למעט העובדה שהעלים x_0 ו־y_0 נמחקו. נשים לב שקבוצת העלים של T' היא $(S\setminus\{x_0,y_0\})\cup\{w_0\}$, ושב־$T'$ יש פחות צמתים מאשר ב־T. לכן, מהנחת האינדוקציה

$$\sum_{x\in(S\setminus\{x_0,y_0\})\cup\{w_0\}}2^{-\operatorname{depth}_{T'}(x)}\leq 1.$$

נשים לב ש־$\operatorname{depth}_{T'}=\operatorname{depth}_T$ למעט העובדה ש $\operatorname{depth}_{T'}$ אינה מוגדרת על

$\{x_0, y_0\}$. כמו־כן $\operatorname{depth}_T(x_0) = \operatorname{depth}_T(y_0) = \operatorname{depth}_T(w_0) + 1$. נסכם:

$$\sum_{x\in S} 2^{-\operatorname{depth}_T(x)} \le \sum_{x\in S\setminus\{x_0,y_0\}} 2^{-\operatorname{depth}_T(x)} + \frac{1}{2}\cdot \sum_{x\in\{x_0,y_0\}} 2^{-\operatorname{depth}_T(w_0)}$$
$$\le \sum_{x\in(S\setminus\{x_0,y_0\})\cup\{w_0\}} 2^{-\operatorname{depth}_{T'}(x)} \le 1.$$

♣

פתרון תרגיל 4.18 מעמוד 70

איור 4.13 מדגים את כל השלבים של האלגוריתם: בניית העץ והסקת הקידוד מהעץ.

(א) שתי השכיחויות הקטנות הן $f_e = 5$, $f_e = 18$. מחברים את d ו־e לאב de בעל השכיחות $5 + 18 = 23$.

(ב) שתי השכיחויות הקטנות הן $f_c = 20$, $f_{de} = 23$. מחברים את ed ו־c לאב cde בעל השכיחות $20 + 23 = 43$.

(ג) שתי השכיחויות הקטנות הן $f_b = 25$, $f_a = 32$. מחברים את a ו־b לאב ab בעל השכיחות $32 + 25 = 57$.

(ד) שתי השכיחויות הן $f_{cde} = 43$, $f_{ab} = 57$. מחברים את שני הצמתים המתאימים לאותו אב.

(ה) הסקת הקידוד. רושמים 0 על קשתות המובילות מאב לבן שמאלי, רושמים 1 על קשתות המובילות מאב לבן ימני (ואפשר גם להפך). הקידוד של x רשום על המסלול המוביל מן השורש אל העלה x.

הקידוד המתקבל וחישוב פונקציית המטרה מסוכמים בטבלה הבאה:

x	$\gamma(x)$	$\lvert\gamma(x)\rvert f_x$
a	00	$2\cdot 32 = 64$
b	01	$2\cdot 25 = 50$
c	10	$2\cdot 20 = 40$
d	110	$3\cdot 18 = 56$
e	111	$3\cdot 5 = 15$
		total 225

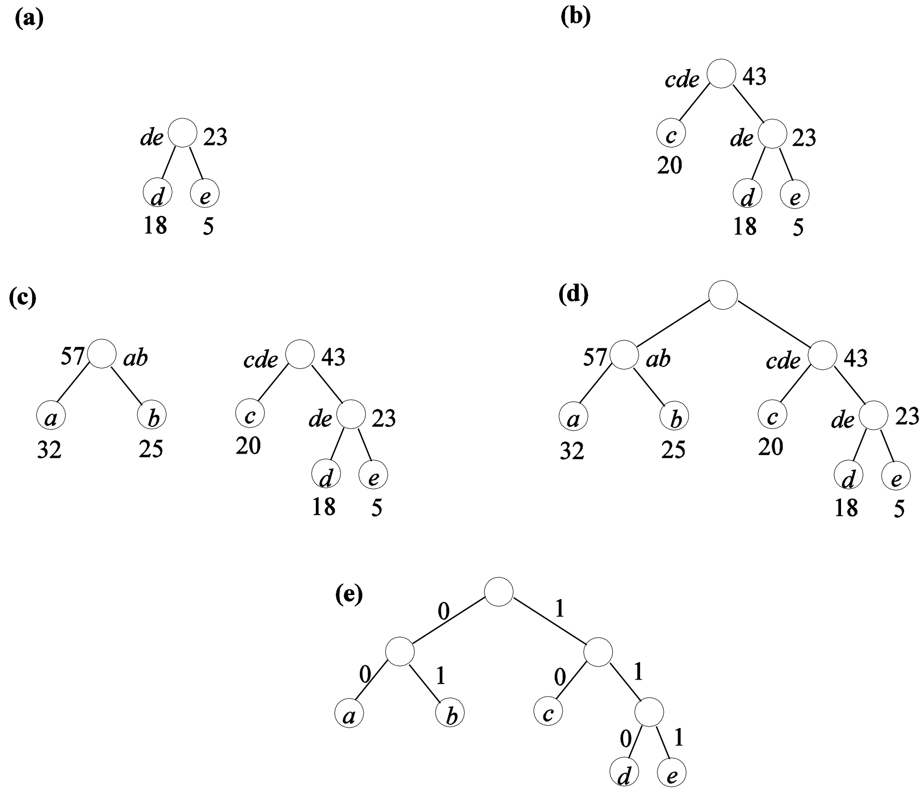
♣

פתרון תרגיל 4.19 מעמוד 70

נוכיח זאת על־ידי בניית עץ קידוד בינארי T' על קבוצת העלים S המקיים את התנאי הזה:

$$\sum_{x\in S} f_x \operatorname{depth}_{T'}(x) < \sum_{x\in S} f_x \operatorname{depth}_T(x).$$

נסמן ב T_1 את תת־העץ של T המושרש ב־u, וב־T_2 את תת־העץ של v ב־T, ראו איור 4.14(a). העץ T' מתקבל מהעץ T ע"י החלפת המקומות של T_1 ו־T_2 בעץ



איור 4.13: בניית העץ האופטימלי והסקת הקידוד.

T, ראו איור 4.14(b). נסמן ב־$S' = S \setminus (T_1 \cup T_2)$.

$$\sum_{x \in S} f_x \operatorname{depth}_{T'}(x) = \sum_{x \in S_1} f_x \operatorname{depth}_{T'}(x) + \sum_{x \in S \cap T_1} f_x \operatorname{depth}_{T'}(x) + \sum_{x \in S \cap T_2} f_x \operatorname{depth}_{T'}(x) \tag{4.5}$$

נשים לב שעבור $x \in S'$, $\operatorname{depth}_{T'}(x) = \operatorname{depth}_T(x)$ ולכן

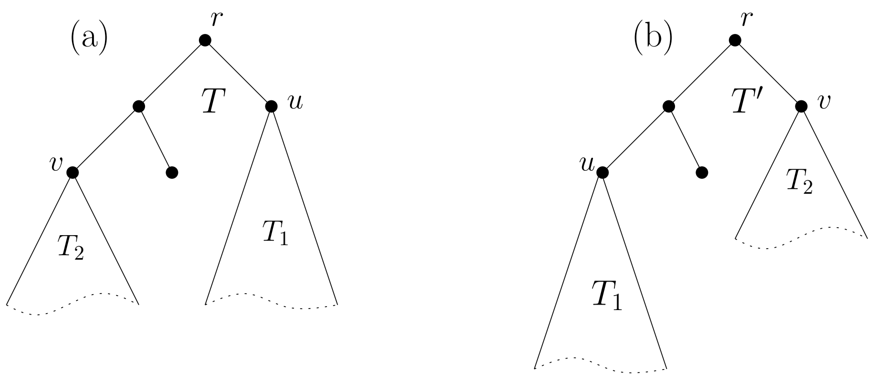
$$\sum_{x \in S_1} f_x \operatorname{depth}_{T'}(x) = \sum_{x \in S_1} f_x \operatorname{depth}_T(x).$$

לגבי הגורם השני ב־(4.5), עבור $x \in S \cap T_1$,

$$\operatorname{depth}_{T'}(x) = \operatorname{depth}_{T_1}(x) + \operatorname{depth}_{T'(u)} = \operatorname{depth}_{T_1}(x) + \operatorname{depth}_T(v)$$

ולכן

$$\sum_{x \in S \cap T_1} f_x \operatorname{depth}_{T'}(x) = \sum_{x \in S \cap T_1} f_x(\operatorname{depth}_{T_1}(x) + \operatorname{depth}_{T(v)}) = f_u \operatorname{depth}_T(v) + \sum_{x \in S \cap T_1} f_x \operatorname{depth}_{T_1}(x).$$



איור 4.14: (a) עץ קידוד בינארי T. (b) העץ הנוצר לאחר החלפת מקומם של תתי־העצים המושרשים ב־u וב־v.

באופן דומה הגורם השלישי ב־(4.5) מקיים:

$$\begin{aligned}\sum_{x\in S\cap T_2} f_x \operatorname{depth}_{T'}(x) &= \sum_{x\in S\cap T_2} f_x(\operatorname{depth}_{T_2}(x) + \operatorname{depth}_{T(u)}) \\ &= f_v \operatorname{depth}_T(u) + \sum_{x\in S\cap T_2} f_x \operatorname{depth}_{T_2}(x).\end{aligned}$$

שימו לב, מן ההנחות נובע כי $(f_v - f_u)(\operatorname{depth}_T(v) - \operatorname{depth}_T(u)) > 0$ ולכן

$$f_v \operatorname{depth}_T(v) + f_u \operatorname{depth}_T(u) > f_v \operatorname{depth}_T(u) + f_u \operatorname{depth}_T(v).$$

אם־כן, מ־(4.5) אנו מקבלים את השוויון הזה:

$$\begin{aligned}
&\sum_{x\in S} f_x \operatorname{depth}_{T'}(x)\\
&\quad= \sum_{x\in S_1} f_x \operatorname{depth}_{T'}(x)\\
&\qquad + f_u \operatorname{depth}_{T}(v) + \sum_{x\in S\cap T_1} f_x \operatorname{depth}_{T_1}(x)\\
&\qquad + f_v \operatorname{depth}_{T}(u) + \sum_{x\in S\cap T_2} f_x \operatorname{depth}_{T_2}(x)\\
&\quad< \sum_{x\in S_1} f_x \operatorname{depth}_{T}(x)\\
&\qquad + f_u \operatorname{depth}_{T}(u) + \sum_{x\in S\cap T_1} f_x \operatorname{depth}_{T_1}(x)\\
&\qquad + f_v \operatorname{depth}_{T}(v) + \sum_{x\in S\cap T_2} f_x \operatorname{depth}_{T_2}(x)\\
&\quad= \sum_{x\in S_1} f_x \operatorname{depth}_{T}(x) + \sum_{x\in S\cap T_1} f_x \operatorname{depth}_{T}(x) + \sum_{x\in S\cap T_2} f_x \operatorname{depth}_{T}(x)\\
&\quad= \sum_{x\in S} f_x \operatorname{depth}_{T}(x).
\end{aligned}$$

♣

פתרון תרגיל 4.20 מעמוד 70

נניח כי $|S| \geq 3$ ואי־זוגי. בעזרת אלגוריתם הופמן לקידוד מעל $\{0,1,2\}$, נבנה עץ טרינארי מלא, שקבוצת העלים שלו היא S; ראו אלגוריתם 4.5.

Algorithm 4.5 Trinary-Hufmman(S, $f : S \to [0,\infty)$)

Require: $|S| \geq 3$ and odd
  if $|S| = 3$ **then**
    Assume $S = \{x, y, z\}$.
    $T \leftarrow$ Root, and three children which are also leaves, labelled x_0, y_0, z_0.
  else
    Let x_0, y_0 and z_0 be the three lowest-frequency letters
    Let $w_0 \notin S$ a new letter
    Form $S' \leftarrow (S \setminus \{x_0, y_0, z_0\}) \cup \{w_0\}$
    Set $f_{w_0} \leftarrow f_{x_0} + f_{y_0} + f_{z_0}$
    Recursively call $T' \leftarrow$ Trinary-Hufmman(S', f)
    $T \leftarrow$ (Start with T'. Take the leaf labeled w_0 and add 3 children below it labeled x_0, y_0 and z_0.)
  return T

ההבדל העיקרי בין הוכחת הנכונות של אלגוריתם הופמן לקידוד טרינארי לבין אלגוריתם הופמן לקידוד בינארי, הוא בהוכחת הטענה הבאה:

טענה 4.9 כאשר $|S| \geq 3$ ואי־זוגי, עץ הקידוד האופטימלי הוא מלא.

הוכחה. יהיה T עץ קידוד טרינארי שאינו מלא. נראה כיצד לבנות עץ קידוד T' כך שיתקיים התנאי הזה:

$$\sum_{x \in S} f_x \operatorname{depth}_{T'}(x) < \sum_{x \in S} f_x \operatorname{depth}_T(x). \tag{4.6}$$

כיוון ש־T אינו מלא, קיים צומת שמספר בניו הוא 1 או 2. אם קיים צומת u שיש לו בן יחיד, אז בדומה לטיעון הנוגע לעצים בינאריים אפשר למחוק את u, ולחבר את הבן של u ישירות לאב של u וכך לקבל עץ T' המקיים את (4.6).

נניח אם־כן שקיים צומת u שיש לו רק שני בנים ולא קיים צומת שיש לו בן יחיד. במקרה זה, חייב להיות בעץ צומת נוסף v, $v \neq u$, שיש לו רק שני בנים, אחרת מספר העלים הוא זוגי (הוכיחו!). נניח בלי הגבלת הכלליות ש־$\operatorname{depth}_T(u) \geq \operatorname{depth}_T(v)$. במקרה זה נעביר את אחד מבניו של u להיות בן שלישי של v. פעולה זו לא מגדילה את מספר הסיביות הממוצע לאות בקוד. u נישאר כעת עם בן יחיד ונוכל להפעיל עליו את הפעולה שתוארה בפיסקה הקודמת ולקבל עץ T' המקיים את (4.6). □

המשך הוכחת הנכונות של אלגוריתם 4.5 זהה למעשה להוכחת הנכונות של אלגוריתם הופמן עבור קידוד בינארי, כפי שמופיע בספר.

טענה 4.10 אם x_0, y_0, z_0 הם התווים ב־S שהשכיחויות שלהם הן הנמוכות ביותר, אז קיים עץ קידוד טרינארי מלא שבו שלושתם עלים־אחים.

הוכחה. זהה להוכחת טענה (4.31) בספר. □

כעת הוכחת האופטימליות של קידוד הופמן הטרינארי באינדוקציה על $|S|$ זהה להוכחה בספר. ♣

פתרון תרגיל 4.22 עמוד 72

איור 4.15 מדגים את כל השלבים של האלגוריתם.

(a) גרף הקלט.

(b) המחירים $c'(e)$. כאן $y_a = 1, y_x = 2, y_z = 4, y_u = 2, y_v = 4$.

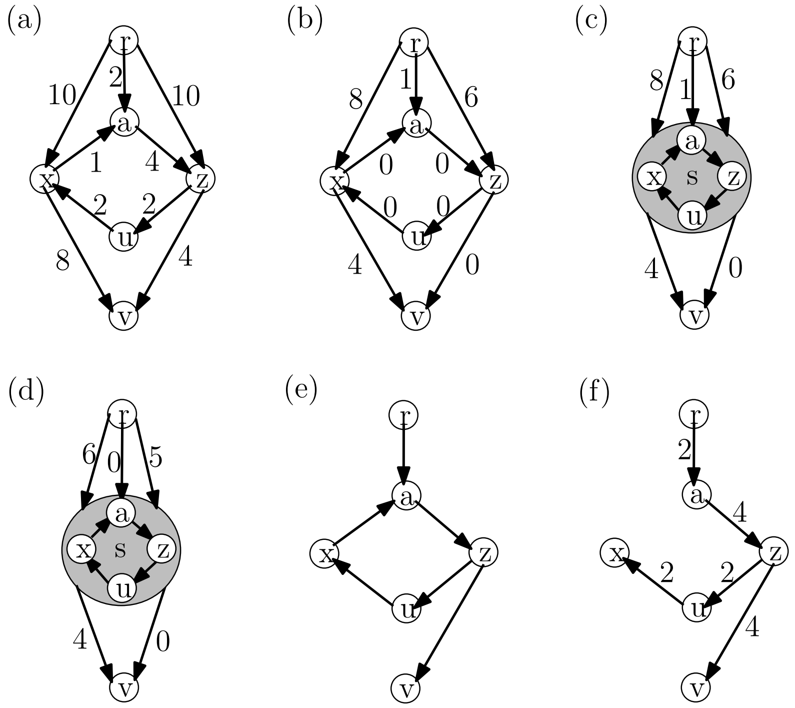
(c) מחירו של המעגל על הצמתים x, a, z, u הוא אפס, והוא מכווץ לצומת אחד s.

(d) עדכון מחירים שני, כאשר $y_v = 0,\ y_s = 1$. בשלב הזה, הגרף מכיל עץ מכוון.

(e) הגרף המתקבל מהעץ המכוון של שלב (d) ופתיחת המעגל של צומת s.

(f) העץ המכוון המוכל בגרף שחושב בשלב (e) הוא הפתרון האופטימלי.

♣



איור 4.15: הרצת האלגוריתם החמדן הדו־שלבי למציאת עץ־פורש מושרש בגרף מכוון.

פרק 5

הפרד ומשול

5.1 הצגת שיטת הפרד־ומשול

קראו בספר את ההקדמה לפרק 5

אלגוריתמים המבוססים על שיטת הפרד־ומשול מפרקים בעיה נתונה לתת־בעיות, פותרים את תת־הבעיות רקורסיבית (או ישירות אם תת־הבעיה "קטנה" מספיק), ואז מרכיבים מחדש את תת־הפתרונות לכדי פתרון לבעיה המקורית. האלגוריתם מיון־מיזוג [Merge Sort] הוא דוגמה פשוטה יחסית לגישה זו שנלמדה כבר בקורס "מבני נתונים ומבוא לאלגוריתמים". זהו אלגוריתם למיון סדרת מספרים הפועל כדלקמן: אם הסדרה מכילה רק איבר אחד – היא כבר ממוינת. אחרת, האלגוריתם מחלק את הסדרה לשני חלקים שווים (בערך) בגודלם, ממיינם רקורסיבית ואת שתי הסדרות הממוינות הוא ממזג בזמן לינארי לסדרה אחת ממוינת.

נשים לב שלמעט הקריאות הרקורסיביות זמן הריצה של האלגוריתם הוא לינארי. לכן מתקבל הביטוי הרקורסיבי הבא לזמן הריצה על סדרות באורך n:

$$T(n) \leq \begin{cases} O(1) & n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + cn & \text{otherwise} \end{cases}$$

כדי לפשט את הדיון, נתעלם מפעולת העיגול למספרים שלמים, שאינו משפיע על ההתנהגות האסימפטוטית של $T(n)$. במקרה שהצגנו לעיל, אנו נכתוב:

$$T(n) \leq \begin{cases} O(1) & n = 1 \\ 2T(n/2) + cn & \text{otherwise} \end{cases}$$

במהלך הפרק ניתקל בנוסחאות נסיגה שונות. סעיפים 5.1 ו־5.2 בספר מתרגלים פתרון נוסחאות נסיגה.

קראו בספר את סעיפים 5.1 ו־5.2 אם אתם זקוקים לרענון בפתרון נוסחאות נסיגה

תרגיל 5.1 מצאו חסמים הדוקים אסימפטוטית לחסמים הרקורסיביים האלה:

$$T(n) \le \begin{cases} 100 & n \le 6 \\ T(n/3) + T(2n/3) + n\log n & n > 6 \end{cases}$$

$$T(n) \le \begin{cases} 10 & n \le 2 \\ T(n/2)^2 + n^2 & n > 2 \end{cases}$$

$$T(n) \le \begin{cases} 10 & n \le 10 \\ T(n-\sqrt{n}) + 2T(2\sqrt{n}) + n & n > 10 \end{cases}$$

$$T(n) \le \begin{cases} 10 & n \le 10 \\ 2T(n/2) + cn\log n & n > 10 \end{cases}$$

$$T(n) \le \begin{cases} O(1) & n = 1 \\ T(7n/10) + cn & n > 1 \end{cases}$$

$$T(n) \le \begin{cases} O(1) & n \le 20 \\ T(n/5+1) + T(7(n+4)/10) & n > 20. \end{cases}$$

5.2 ספירת היפוכים

קראו בספר את סעיף 5.3

נתונה סדרת מספרים $s = (a_1, \ldots, a_n)$; מספר ההיפוכים בה הוא מספר הזוגות (i, j) בהם $1 \le i < j \le n$ וכן $a_i > a_j$. אנו מתעניינים כעת בבעיה הזו:

בעיה אלגוריתמית: ספירת היפוכים [Counting Inversions].
הקלט: סדרת מספרים $s = (a_1, \ldots, a_n)$.
הפלט: מספר ההיפוכים בסדרה הנתונה.

בסעיף הזה מוצג האלגוריתם מיון־ומנייה Sort-and-Count המבוסס על האלגוריתם מיון־מיזוג; האלגוריתם מיון־ומנייה סופר את מספר ההיפוכים בסדרה נתונה שאורכה n בזמן $O(n\log n)$, זאת למרות שמספר ההיפוכים יכול להיות עד $n(n-1)/2$.

האלגוריתם מיון־ומנייה עובד על בסיס הרעיון הזה: נתונה סדרה $(a_1, \ldots, a_n)$; נחלק אותה לשתי תת־סדרות שהאורך של כל אחת מהן יהיה כמחצית הסדרה הנתונה; המחצית הראשונה $A = (a_1, \ldots, a_{n/2})$ תיקרא תחתונה, והמחצית השנייה $B = (a_{n/2+1}, \ldots, a_n)$ תיקרא עליונה. את ההיפוכים שיתקבלו בפלט אפשר לחלק לשלושה סוגים:

- ספירת היפוך ששני איבריו ב־A תבוצע על־ידי קריאה רקורסיבית לאלגוריתם מיון־ומנייה על התת־סדרה A.

- הספירה של היפוך ששני איבריו ב־B, תבוצע על־ידי קריאה רקורסיבית לאלגוריתם מיון־ומנייה על התת־סדרה B.
- היפוך (a_i, a_j), $i < j$, $a_i \in A$, ו־$a_j \in B$. מספר ההיפוכים הללו ניתן לספירה בזמן לינארי כדלקמן: לכל $a_j \in B$, מספר ההיפוכים בהם מופיע $a_j \in B$ והאיבר האחר הוא מ־A, הוא בדיוק מספר האיברים ב־A הגדולים מ־a_j. סכימת מספר הזוגות הללו קלה לביצוע בשלב המיזוג שמבצע האלגוריתם מיון־מיזוג: יהיו A', ו־B' הרשימות הממוינות של A ו־B בהתאמה. בכל פעם שנבחר איבר מ־B' לרשימה הממוזגת, מספר האיברים ב A' שעדיין לא מוזגו הוא בדיוק מספר ההיפוכים הזה.

דוגמה: נריץ את האלגוריתם מיון־ומנייה על הסדרה $(3, 4, 2, 8, 7, 9, 6, 5, 1, 10)$.

- בשלב הראשון נחלק את הסדרה לשני חלקים שווים (בערך) בגודלם $A = (3, 4, 2, 8, 7)$, ו־$B = (9, 6, 5, 1, 10)$.
- נקרא רקורסיבית לאלגוריתם Sort-and-Count ונריץ אותו על הסדרה A. נקבל בחזרה את הסדרה $A' = \text{sort}(A) = (2, 3, 4, 7, 8)$, ואת מספר ההיפוכים ב־$A$, שהוא 3.
- נקרא רקורסיבית לאלגוריתם מיון־ומנייה ונריץ אותו על הסדרה B. נקבל בחזרה את הסדרה $B' = \text{sort}(B) = (1, 5, 6, 9, 10)$, ואת מספר ההיפוכים ב־$B$ שהוא 6.
- נריץ את Merge-and-Count על הקלט (A', B') בצורה הבאה:
  - נאפס את count$\leftarrow 0$
  - נמזג את A' ו־B'. המיזוג ייראה כך:

A'		2	3	4			7	8		
B'	1				5	6			9	10
Remaining in A'	5				2	2			0	0

  בכל פעם שממוזג איבר מ־B', מצוין בשורה השלישית מספר האיברים שעדיין לא מוזגו ב־A', לכן השגרה מיזוג־ומנייה (Merge-and-Count) מחזירה את הרשימה הממוזגת $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ ואת סכום ההיפוכים שהתגלו בשלב המיזוג: $5 + 2 + 2 + 0 + 0 = 9$.
- האלגוריתם מיון־ומנייה מחזיר את הרשימה הממוזגת, את סכום ההיפוכים שנספרו רקורסיבית ואת אלה שנספרו בשלב המיזוג, בסך־הכל: $3 + 6 + 9 = 18$.

♣

תרגיל 5.2 ספירת שחלופים [exhanges]. פעולת שחלוף על סדרת מספרים $\sigma = (a_1, \ldots, a_n)$ מחליפה את המקומות של a_i ו־a_{i+1} עבור $1 \le i < n$ כלשהו, ווצרת הסדרה $(a_1, \ldots, a_{i-1}, a_{i+1}, a_i, a_{i+2}, \ldots, a_n)$. מרחק השחלופים של סדרה σ הוא המספר המינימלי של פעולות שחלוף הנדרש כדי לעבור מן הסדרה הנתונה למיון העולה שלה. הוכיחו שמרחק השחלופים של סדרה שווה למספר ההיפוכים בה.

פתרון בעמוד 100

5.3 זוג נקודות קרובות ביותר במישור

קראו בספר את סעיף 5.4

הסעיף הזה עוסק בבעיה הזו:

בעיה אלגוריתמית: זוג נקודות קרובות ביותר במישור [Closest Pair of Points].
הקלט: n נקודות $P = \{p_1, \ldots, p_n\}$ במישור, $p_i = (x_i, y_i)$.
הפלט: זוג נקודות p_i, p_j, $i \neq j$.
המטרה: p_i, p_j הן זוג הנקודות הקרובות ביותר ברשימה P, לפי המרחק האוקלידי $d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

שימו לב, בעיה זו מכלילה את בעיית מציאת זוג הנקודות הקרוב ביותר על הישר (שאפשר למצוא בקלות על־ידי מיון), ושהאלגוריתם הסורק את כל הזוגות, רץ בזמן $O(n^2)$. הסעיף הנוכחי מציג אלגוריתם הפרד־ומשול מתוחכם, שפותר את הבעיה בזמן $O(n \log n)$. להלן תיאור מקוצר של האלגוריתם וניתוחו.

1. בשלב הראשון אנו מוצאים את החציון m לפי ציר־x של הנקודות ומחלקים אותן לשתי קבוצות בגודל $n/2$ (בערך):
$$P_L = \{p_i = (x_i, y_i) :\ x_i \leq m\} \quad P_R = \{p_i = (x_i, y_i) :\ x_i > m\}.$$
(תתי־הקבוצות באלגוריתם זה מתוחזקות בעזרת רשימות.)
2. אנו מפעילים רקורסיבית את האלגוריתם על P_L ומוצאים את זוג הנקודות הקרוב ביותר בקבוצה P_L, שהמרחק בין הנקודות הוא δ_L.
3. באופן דומה אנו פועלים על P_R ומוצאים את זוג הנקודות הקרוב ביותר ב־P_R, שהמרחק בין הנקודות הוא δ_R.
4. נסמן $\delta = \min\{\delta_L, \delta_R\}$.
5. בשלב ה"מיזוג" עלינו לבדוק את הזוגות "המעורבים" של הנקודות $p \in P_L$, $q \in P_R$, ולחפש אם יש זוג כזה המקיים $d(p,q) < \delta$. להלן תיאור החיפוש:
   א. תחילה אנו מבחינים כי מספיק לבדוק את הנקודות הקרובות לקו החציון, כלומר, מספיק לסרוק זוגות $p \in P'_L$ ו־ $q \in P'_R$ כאשר
$$P'_L = \{p_i = (x_i, y_i) :\ m - \delta \leq x_i \leq m\},$$
$$P'_R = \{p_i = (x_i, y_i) :\ m < x_i \leq m + \delta\}.$$
   מפני שהנקודות ב $P_L \setminus P'_L$ ו־ $P_R \setminus P'_R$ נמצאות רחוק מדי מקו ההפרדה ואינן יכולות להיות חלק מזוג מעורב במרחק הקטן מ־δ.
   ב. הקטנו את גודל הסריקה, אך לא במידה מספקת, כי עדיין ייתכן שאוסף הזוגות $\{(p,q) :\ p \in P'_L,\ q \in P'_R\}$ יכיל $\Omega(n^2)$ נקודות.
   ג. כאן אנו משתמשים בטענה המכריעה (שהובאה בספר כטענה (5.10), ראו גם תרגיל 5.3): נתון זוג נקודות (p,q) כאשר $p \in P'_L$, $q \in P'_R$; אם זוג הנקודות הללו במרחק קטן מ־δ – $d(p,q) < \delta$ – אז הן חייבות להיות עד 15 מקומות זו מזו במיון של $P'_L \cup P'_R$ לפי ציר־y.
   לכן כל שעלינו לעשות הוא לסרוק את $P'_L \cup P'_R$, ולכל איבר $p \in P'_L$ לסרוק את 30 האיברים הקרובים ל־p במיון לפי ציר־y של $P'_L \cup P'_R$, ולחפש ביניהם $q \in P'_R$ שמביא למינימום את $d(p,q)$.

6. האלגוריתם מחזיר את המינימום בין δ לערך שנמצא בסריקה לעיל של $P'_L \cup P'_R$.

הוכחת הנכונות הפורמלית תהיה, כנהוג באלגוריתמים רקורסיביים, באינדוקציה, ולמעשה הדרך כבר נרמזה בביאור לעיל. ננתח כעת את זמן הריצה. בניתוח נאיבי, בשלב הפירוק ובשלב המיזוג אנו נדרשים למיין, לכן זמן הריצה במימוש נאיבי של האלגוריתם מקיים $T(n) \leq 2T(n/2) + cn \log n$. כפי שראינו בתרגיל 5.1, נוסחת הנסיגה הזו מקיימת $T(n) = O(n \log^2 n)$.

אפשר לשפר במקצת את האלגוריתם לפי ההבחנה הזו: הצעדים היחידים שדרוש להם זמן $O(n \log n)$ במהלך הפירוק וההרכבה, הם מיון קבוצת הנקודות המקורית P לפי ציר־x ולפי ציר־y של תת־קבוצות. אפשר אם־כן בתחילת האלגוריתם, ולפני ביצוע השגרה הרקורסיבית שלעיל, למיין את כל הנקודות לפי ציר־x ולפי ציר־y, ואז הקלט לקריאות הרקורסיביות יהיה תת־קבוצה של הנקודות שכבר מוינו לפי ציר־x ולפי ציר־y. כעת נותר לנו רק להבחין שהקריאות הרקורסיביות לא צריכות כבר למיין אלא רק לשלוף תת־רשימות מתוך רשימות ממוינות, ואת זה אפשר לבצע בזמן לינארי. זמן הריצה המשופר מקיים אם־כן $T(n) \leq 2T(n/2) + cn$. נוסחת הנסיגה הזו מקיימת $T(n) = O(n \log n)$.

תרגיל 5.3

1. שפרו את החסם העליון על מספר האיברים המפרידים ברשימה הממוינת של $P'_L \cup P'_R$ בין איברי הזוג הקרוב ביותר, כך שיהיה קטן מ־15.
2. תנו דוגמה לקבוצת נקודות שבמהלך ריצת האלגוריתם עליה, בשלב המיזוג, בין זוג הנקודות הקרובות ביותר נמצאות 2 נקודות אחרות ברשימה הממוינת של $P'_L \cup P'_R$.

פתרון בעמוד 100

5.4 כפל מספרים שלמים

קראו בספר את סעיף 5.5

תרגיל 5.4 כתבו את האלגוריתם שמבצע כפל של שני מספרים בני n סיביות, שנלמד בבית־הספר בשם "כפל ארוך". הראו שמספר הפעולות הבסיסיות הדרושות לו (על סיביות) הוא $\Theta(n^2)$.

פתרון בעמוד 101

בסעיף זה אנו לומדים אלגוריתם הפרד־ומשול פשוט וחכם שמשפר (אסימפטוטית) את מספר הפעולות הבסיסיות הדרושות. הרעיון: בהינתן זוג מספרים x ו־y בני n סיביות, אפשר לכותבם כך: $x = 2^{n/2}x_1 + x_0$ ו־$y = 2^{n/2}y_1 + y_0$, ואז מספר הסיביות במספרים x_0, x_1, y_0, y_1 יהיה $n/2$ סיביות. לפיכך:

$$xy = (x_1 2^{n/2} + x_0) \cdot (y_1 2^{n/2} + y_0) = x_1 y_1 2^n + (x_1 y_0 + y_1 x_0) 2^{n/2} + x_0 y_0. \quad (5.1)$$

(שימו לב, כפל ב־2^n או ב־$2^{n/2}$ הוא בסך־הכל הזזת סיביות ואין צורך בקריאה רקורסיבית לפעולת הכפל). אנו מקבלים נוסחה עם ארבע מכפלות (רקורסיביות). ההבחנה החשובה היא שאפשר לחשב את שלושת הגורמים ב־(5.1) על־ידי חישוב

של שלוש המכפלות האלה:

$$a = (x_1 + x_0) \cdot (y_1 + y_0) \qquad b = x_1 \cdot y_1 \qquad c = x_0 y_0$$

הפעולה מבוצעת כדלקמן: הגורם שמכפיל את 2^n הוא b, הגורם שמכפיל את 1 הוא c, והגורם שמכפיל את $2^{n/2}$ הוא $a - b - c$. שימו לב שאפשר לבצע חיבור וחיסור בזמן לינארי (האלגוריתם שלמדנו בבית־הספר). לכן אנו מקבלים אלגוריתם רקורסיבי המבצע שלוש מכפלות רקורסיביות על מספרים שמספר הסיביות שלהם הוא מחצית ולכן זמן הריצה מקיים:

$$T(n) \leq 3T(n/2) + cn.$$

נוסחה רקורסיבית זו מקיימת $T(n) = O(n^{\log_2 3})$.

תרגיל 5.5 מספר מרוכב הוא נקודה במישור $x = (a, b)$. חיבור מספרים מרוכבים זהה לחיבור וקטורים דו־ממדיים, כלומר $(a, b) + (c, d) = (a + b, c + d)$. כפל מוגדר כדלקמן:

$$(a, b) \cdot (c, d) = (ac - bd, ad + bc).$$

הראו כיצד אפשר לחשב כפל של מספרים מרוכבים (המיוצגים כמו לעיל), בעזרת שלוש מכפלות ממשיות בלבד.

פתרון בעמוד 102

תרגיל 5.6 נתונים זוג פולינומים במשתנה אחד מעל השלמים $f, g \in \mathbb{Z}[x]$. כל פולינום נתון כסדרת מקדמים. הציגו ונתחו אלגוריתם לחישוב פולינום המכפלה $f \cdot g$ בזמן $O(n^{\log_2 3})$, כאשר $n - 1$ הוא הגבוה מבין דרגות הפולינומים (דרגת הפולינום היא החזקה הגבוהה ביותר שהמקדם שלה אינו 0).

לדוגמה: אנו מזהים את הקלט $f = (1, 2, 3, 4)$, $g = (0, -2, 4)$ עם הפולינומים $f(x) = 4x^3 + 3x^2 + 2x + 1$ ו־ $g(x) = 4x^2 - 2x$. המכפלה שלהם היא:

$$\begin{aligned}(f \cdot g)(x) &= (4x^3 + 3x^2 + 2x + 1) \cdot (4x^2 - 2x) \\ &= 16x^5 + (12 - 8)x^4 + (8 - 6)x^3 + (4 - 4)x^2 - 2x,\end{aligned}$$

ולכן הפלט צריך להיות $(0, -2, 0, 2, 4, 16)$. בסעיף הבא נראה אלגוריתם יעיל יותר לחישוב כפל פולינומים.

פתרון בעמוד 102

5.5 קונוולוציה והתמרת פורייה המהירה

קראו בספר את סעיף 5.6

הערה: כיוון שבסעיף זה נרבה להשתמש במספרים מרוכבים. הסימן "i" יהיה שמור בסעיף זה למספר המדומה $i=\sqrt{-1}$, ולא ישמש כאינדקס. ♣

סעיף זה דן בחישוב מהיר של כפל פולינומים. ניזכר: אם

$$f(x)=a_{n-1}x^{n-1}+\ldots+a_0, \qquad g(x)=b_{m-1}x^{m-1}+\ldots+b_0,$$

אזי

$$h(x)=f(x)\cdot g(x)=c_{m+n-2}x^{m+n-2}+c_{m+n-3}x^{m+n-3}+\ldots+c_0,$$

כאשר המספר c_k מוגדר כדלקמן:

$$c_k=\sum_{j=0}^{k}a_jb_{k-j}$$

הפעולה על הסדרות $(a_0,\ldots,a_{n-1})$ ו־$(b_0,\ldots,b_{m-1})$ המייצרת את הסדרה $(c_0,\ldots,c_{m+n-2})$ נקראת קונוולוציה [Convolution] וסימנה הוא "$*$".

דוגמה:

$$f(x)=-4x^3+5x^2+2x-1 \qquad g(x)=8x^3+7x^2+6x+5$$

אזי

$$\begin{aligned}h(x)&=f(x)\cdot g(x)\\&=(-4\cdot 8)x^6+(-4\cdot 7+5\cdot 8)x^5+(-4\cdot 6+5\cdot 7+2\cdot 8)x^4\\&\quad+(-4\cdot 5+5\cdot 6+2\cdot 7-1\cdot 8)x^3+(5\cdot 5+2\cdot 6-1\cdot 7)x^2\\&\quad+(2\cdot 5-1\cdot 6)x-1\cdot 5\\&=-32x^6+12x^5+27x^4+16x^3+15x^2+4x-5.\end{aligned}$$

בסימון סדרות אנו מקבלים

$$(-1,2,5,-4)*(5,6,7,8)=(-5,4,15,16,27,12,-32).$$

♣

בתרגיל 5.6 התבקשתם לפתח אלגוריתם הפרד־ומשול ישיר לכפל פולינומים שסיבוכיותו היא $O(n^{\log_2 3})$. בסעיף זה אנו לומדים על דרך מעט עקיפה לחישוב כפל פולינומים, שבסופה נקבל אלגוריתם מהיר ויעיל יותר, המבוסס על התמרת פורייה המהירה [Fast Fourier Transform (FFT)].

כצעד ראשון לפתוח אלגוריתם כפל פולינומים מהיר נבחין שכל פולינום f מדרגה $n-1$ נקבע באופן יחיד על־ידי ערכו ב־n נקודות שונות, ולכן אפשר לייצג את f בייצוג אלטרנטיבי על־ידי:

$$(x_0,f(x_0)),\ (x_1,f(x_1)),\ \ldots,(x_{n-1},f(x_{n-1})),$$

כאשר $x_0, \ldots, x_{n-1}$ הוא אוסף של נקודות זרות בזוגות. בשני התרגילים הבאים נוכיח זאת ונתייחס לאספקט האלגוריתמי של ייצוג זה.

תרגיל 5.7 בהינתן סדרת המקדמים $a_0, \ldots, a_{n-1} \in \mathbb{C}$ ו־$x_0 \in \mathbb{C}$, הראו כיצד לחשב את $f(x_0)$ בזמן $O(n)$, כאשר $f(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$.

פתרון בעמוד 103

תרגיל 5.8 (אינטרפולציה פולינומיאלית) תהי $(x_0, y_0),\ (x_1, y_1),\ \ldots, (x_{n-1}, y_{n-1})$ סדרה של זוגות מספרים מרוכבים המקיימת $x_k \neq x_j$ לכל $k \neq j$. הוכיחו שקיים פולינום יחיד $f(x)$ (עם מקדמים מרוכבים) מדרגה $n-1$ כך ש־ $f(x_k) = y_k$. פולינום זה נקרא פולינום האינטרפולציה מדרגה $n-1$ של אוסף הזוגות $(x_0, y_0),\ (x_1, y_1),\ \ldots, (x_{n-1}, y_{n-1})$. הראו כיצד ניתן לחשב את מקדמי פולינום האינטרפולציה בזמן $O(n^2)$.

פתרון בעמוד 103

עתה, בהינתן זוג פולינומים f ו־g בייצוג ערכיהם בנקודות $x_0, \ldots, x_m$, קל לייצר את מכפלתם:

$$(x_0, f(x_0)g(x_0)),\ \ldots, (x_m, f(x_m)g(x_m)).$$

אנו מקבלים גישה מעניינת לחישוב כפל פולינומים: בהינתן זוג הפולינומים f ו־g, כל אחד מדרגה $n-1$, עליכם לבצע פעולות אלה:

1. בחרו ℓ נקודות, $\ell \geq 2n-1$, $x_0, \ldots, x_{\ell-1} \in \mathbb{C}$.
2. חשבו את ערכי f ו־g בנקודות האלה.
3. חשבו את $h_k = f(x_k)g(x_k)$.
4. מיצאו את מקדמי פולינום האינטרפולציה h לזוגות $(x_0, h_0), \ldots, (x_{\ell-1}, h_{\ell-1})$ המקיים $h(x_j) = h_j$.

בגישה זאת, פעולת הכפל (צעד 3) מתבצעת בקלות בזמן לינארי. מובן שהסטנו את הקושי לחישוב ערכי הפולינום בנקודות, ולחישוב אינטרפולציה עבור שניהם ראינו בתרגיל 5.7 ובתרגיל 5.8 אלגוריתמים העובדים בזמן $O(n^2)$, מה שכמובן איננו משפר את זמן הריצה. הרעיון הנוסף שמאפשר את שיפור זמן הריצה הוא בחירה חכמה של הנקודות $x_0, \ldots, x_{\ell-1}$ שמאפשרת לקבל אלגוריתם הפרד־ומשול יעיל לביצוע חישוב הערכים והאינטרפולציה – זהו ה־FFT.

אם־כן, מטרתנו כעת היא למצוא סדרה של נקודות $x_0, \ldots, x_{n-1}$, שקל לחשב ערכי פולינומים ממעלה עד $n-1$ בהן. יהי $f(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0$. נניח ש־$n$ הוא חזקה של 2. נפרק את f לחזקות זוגיות וחזקות אי־זוגיות, ונכתוב $f(x) = xf_o(x^2) + f_e(x^2)$. כלומר,

$$\begin{aligned} f(x) &= a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \\ &= x\left(a_{n-1}x^{n-2} + a_{n-3}x^{n-4} + \ldots + a_1x^0\right) \\ &\quad + \left(a_{n-2}x^{n-2} + a_{n-4}x^{n-4} + \ldots a_2x^2 + a_0x^0\right) \\ &= xf_o(x^2) + f_e(x^2) \end{aligned}$$

כאשר,

$$\begin{aligned} f_e(x) &= a_nx^{\frac{n}{2}} + a_{n-2}x^{\frac{n}{2}-1} + \ldots + a_2x + a_0, \\ f_o(x) &= a_{n-1}x^{\frac{n}{2}-1} + a_{n-3}x^{\frac{n}{2}-2} + \ldots + a_3x + a_1. \end{aligned}$$

המטרה היא לחשב את ערכי f בנקודות $x_0,\dots x_{n-1}$ בשיטת הפרד־ומשול. נניח שחישבנו את ערכי f_e ו־f_o על מחצית מקבוצת הנקודות $x_0, x_2, \dots x_{n-2}$. לצורך חישוב f אנו צריכים לחשב את f_e ו־f_o בנקודות $x_0^2, x_1^2, \dots, x_{n-1}^2$. כדי ששיטת הפרד־ומשול תהיה יעילה, עלינו להוסיף כמה שפחות עבודה רקורסיבית לצורך חישוב f. במקרה זה, המשמעות היא שלצורך חישוב f אין אנו רוצים לבצע הערכות נוספות של פולינומים בנקודות. יהיה מועיל מאד אם־כן ששתי הקבוצות שלעיל יתלכדו. כלומר אנו רוצים שיתקיים

$$\{x_0^2, x_1^2, \dots, x_{n-1}^2\} = \{x_0, x_2, x_4, \dots, x_{n-2}\}.$$

אנו גם רוצים שתכונה זאת תישמר רקורסיבית, כלומר, שיתקיים גם

$$\{x_0^2, x_2^2, x_4^2, \dots, x_{n-2}^2\} = \{x_0, x_4, x_8, \dots, x_{n-4}\},$$

וכן הלאה. האם לכל n שהוא חזקה של שתיים, קיימת קבוצת מספרים המקיימת תנאים אלה? מסתבר שכן, בתנאי שאנו מאפשרים גם מספרים מרוכבים: אלה שורשי היחידה מסדר n.

לפני שנפנה להצגת שורשי היחידה נסביר את הסימון "$e^{i\theta}$". בספר, וגם במדריך אנו נשתמש בסימון $e^{i\theta} = \cos\theta + i\sin\theta$, כלומר e בחזקת $i\theta$, כאשר e הינו בסיס הלוגאריתם הטיבעי, ו־$i = \sqrt{-1}$. ישנה הצדקה מתמטית חזקה לזהות את הביטוי $\cos\theta + i\sin\theta$ עם פעולת החזקה במספר מדומה. אנא לא נרחיב בנקודה זאת פה. לצרכינו מספיק לחשוב על זה כסימון נוח. עיקר הנוחות נובעת מהקיצור ברישום ושנוסחת דה־מואבר,

$$(\cos\theta + i\sin\theta)^n = \cos(n\theta) + i\sin(n\theta),$$

מתלכדת עם פעולת החזקה:

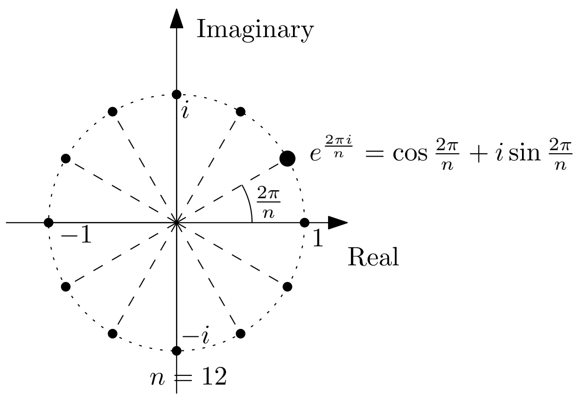
$$(\cos\theta + i\sin\theta)^n = (e^{i\theta})^n = e^{in\theta} = \cos(n\theta) + i\sin(n\theta),$$

שורשי היחידה. שורש יחידה מסדר n הוא מספר מרוכב x המקיים $x^n = 1$. כפי שלמדתם בקורס "אלגברה לינארית", שורשי היחידה הם קבוצת המספרים

$$\begin{aligned}\Big\{\cos\theta + i\sin\theta :\ \theta \in \{0, \tfrac{2\pi}{n}, \tfrac{2\pi}{n}\cdot 2, \tfrac{2\pi}{n}\cdot 3, \dots, \tfrac{2\pi}{n}\cdot(n-1)\}\Big\}\\ = \{1 = e^{i\frac{2\pi\cdot 0}{n}}, e^{i\frac{2\pi}{n}}, e^{i\frac{2\pi\cdot 2}{n}}, \dots, e^{i\frac{2\pi(n-1)}{n}} = e^{-i\frac{2\pi}{n}}\}.\end{aligned}$$

לדוגמא, עבור $n = 4$ שורשי היחידה מסדר 4 (כלומר הפתרונות של במשוואה $x^4 = 1$) הינם

$$\begin{aligned} e^{i0} &= \cos 0 + i\sin 0 = 1 && \text{for } j = 0\\ e^{i\frac{2\pi}{4}} &= \cos\tfrac{\pi}{2} + i\sin\tfrac{\pi}{2} = i && \text{for } j = 1\\ e^{i\frac{2\pi\cdot 2}{4}} &= \cos\pi + i\sin\pi = -1 && \text{for } j = 2\\ e^{i\frac{2\pi\cdot 3}{4}} &= \cos\tfrac{3\pi}{2} + i\sin\tfrac{3\pi}{2} = -i && \text{for } j = 3.\end{aligned}$$



איור 5.1: שורשי היחידה מסדר n הם נקודות על מעגל היחידה, והזווית שהן יוצרות עם ציר המספרים הממשיים, היא כפולה שלמה של $\frac{2\pi}{n}$; העלאה בחזקת k משמעותה הכפלת הזווית ב־k. בזווית $2\pi/n$ נמצא שורש יחידה פרימיטיבי. האיור מדגים את שורשי היחידה מסדר $n = 12$.

ראו גם המחשה של שורשי היחידה מסדר 12 באיור 5.1.

שורש יחידה פרימיטיבי מסדר n, המסומן ב־$\omega = \omega_n$, מקיים בנוסף לאמור לעיל, גם $\omega^k \neq 1$ לכל $n > k > 0$. לדוגמה, אפשר לבחור $\omega = e^{i2\pi/n}$ (כאמור, בסעיף זה $i = \sqrt{-1}$). כעת נבחר $x_k = w^k$. נשים לב שאמנם

$$\begin{aligned}
\{x_0^2, x_1^2, \ldots, x_{n-1}^2\} &= \{\omega^{0\cdot 2}, \omega^{1\cdot 2}, \ldots, \omega^{(n-1)2}\} \\
&= \{\omega^{0\cdot 2}, \omega^{1\cdot 2}, \ldots, \omega^{(\frac{n}{2}-1)\cdot 2}, \omega^{\frac{n}{2}\cdot 2}, \omega^{(\frac{n}{2}+1)\cdot 2}, \ldots, \omega^{(n-1)2}\} \\
&= \{\omega^{0\cdot 2}, \omega^{1\cdot 2}, \ldots, \omega^{n-2}, \omega^{0}, \omega^{2}, \ldots, \omega^{n-2}\} \\
&= \{x_0, x_2, x_4 \ldots, x_{n-2}\},
\end{aligned}$$

כנדרש.

אם כן, מטרתנו לחשב ההתאמה המתאימה לסדרת המספרים $(a_0, \ldots, a_{n-1})$ את סדרת המספרים $(f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1}))$, כאשר $f(x) = a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$ ו־ω שורש יחידה פרימיטיבי מסדר־n. התאמה זו נקראת התמרת פורייה הבדידה [Discrete Fourier Transform (DFT)].

עתה נתאר אלגוריתם מהיר לחישוב DFT. אלגוריתם זה נקרא התמרת פורייה המהירה [Fast Fourier Tansform (FFT)] כאשר n חזקה של 2:

1. רקורסיבית נחשב את $f_o(x) = a_{n-1}x^{\frac{n}{2}} + a_{n-3}x^{\frac{n}{2}-1} + \ldots + a_1$ בנקודות $1, \omega^2, \omega^4, \ldots, \omega^{n-4}, \omega^{n-2}$.
2. רקורסיבית נחשב את $f_e(x) = a_{n-2}x^{\frac{n}{2}} + a_{n-4}x^{\frac{n}{2}-1} + \ldots + a_0$ בנקודות $1, \omega^2, \omega^4, \ldots, \omega^{n-4}, \omega^{n-2}$.
3. "נרכיב" את החישובים שלעיל כדי לקבל את ערכי $f(x)$ בנקודות $1, \omega, \omega^2, \ldots, \omega^{n-1}$ כדלקמן:

$$f(\omega^k) = \omega^k f_o((\omega^k)^2) + f_e((\omega^k)^2),$$

תוך שאנו שמים לב שאת $f_o(\omega^{2k})$ חישבנו כבר בצעד 1., ואת $f_e(\omega^{2k})$ חישבנו כבר בצעד 2.

נרשום את אלגוריתם, הנקרא $\mathrm{FFT}((a_0, \ldots, a_{n-1}), \omega)$ (ראו אלגוריתם 5.1), המחשב את התמרת פורייה הבדידה של $(a_0, \ldots, a_{n-1})$ תחת ההנחה ש־ω הוא שורש יחידה פרימיטיבי מסדר n וש־n הוא חזקה של 2.

Algorithm 5.1 $\mathrm{FFT}((a_0, \ldots, a_{n-1}), \omega)$

Require: $\omega^n = 1$
Require: $\forall 0 < k < n$, $\omega^k \neq 1$.
Require: n is a power of 2.
  if $n = 1$ **then**
    return $(f(1) = a_0)$
  $(f_e(1), f_e(\omega^2), f_e(\omega^4), \ldots, f_e(\omega^{n-2})) \leftarrow \mathrm{FFT}((a_0, a_2 \ldots, a_{n-4}, a_{n-2}), \omega^2)$
  $(f_o(1), f_o(\omega^2), f_o(\omega^4), \ldots, f_o(\omega^{n-2})) \leftarrow \mathrm{FFT}((a_1, a_3, \ldots, a_{n-3}, a_{n-1}), \omega^2)$
  for $k \leftarrow 0, \ldots, n-1$ **do**
    $f(\omega^k) \leftarrow f_e(\omega^{2k}) + \omega^k f_o(\omega^{2k})$
  return $(f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1}))$

נכונות האלגוריתם תוארה בקצרה לעיל. סיבוכיות הזמן $T(n)$ מקיימת

$$T(n) \leq 2T(n/2) + cn$$

כלומר $T(n) = O(n \log n)$.

תרגיל 5.9 אלגוריתם 5.1 נכתב תוך שימוש ב"אינדקס" ω^k, ומנצל את העובדה ש־$\omega^{2k} = \omega^{2(k-\frac{n}{2})}$ עבור $k \geq n/2$. כיתבו את האלגוריתם מחדש בעזרת אינדקסים רגילים בטווח $0, \ldots, n-1$ הניחו ש־$\omega = e^{2\pi i/n}$ וש־n הינו חזקה של 2.

פתרון בעמוד 104

כדי לסיים את תיאור האלגוריתם המהיר לכפל פולינומים (או בניסוח שקול, חישוב קונוולוציות) נותר לנו רק לתאר את שלב האינטרפולציה בשורשי היחידה. זוהי התמרה ההפוכה ל־DFT. אנו רוצים למצוא כיצד לחשב את הפעולה ההפוכה, ולשם כך נתחיל בהבנה טובה יותר של מהי DFT. התמרת פורייה של סדרת מספרים $(a_0, \ldots, a_{n-1})$, הינה הווקטור הבא:

$$\begin{pmatrix} a_{n-1}\omega^{0(n-1)} + a_{n-2}\omega^{0(n-2)} + \ldots + a_1\omega^0 + a_0 \\ a_{n-1}\omega^{n-1} + a_{n-2}\omega^{n-2} + \ldots + a_1\omega + a_0 \\ a_{n-1}\omega^{2(n-1)} + a_{n-2}\omega^{2(n-2)} + \ldots + a_1\omega^2 + a_0 \\ \vdots \\ a_{n-1}\omega^{(n-1)(n-1)} + a_{n-2}\omega^{(n-1)(n-2)} + \ldots + a_1\omega^{n-1} + a_0 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{n-2} & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(n-2)} & \omega^{2(n-1)} \\ \vdots & & & & & \\ 1 & \omega^{n-1} & \omega^{(n-1)2} & \ldots & \omega^{(n-1)(n-2)} & \omega^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix}$$

כלומר, התמרת פורייה הבדידה היא התמרה (טרנספורמציה) לינארית שהמטריצה שלה $\mathcal{D}(\omega)$ מוגדרת כ- $\mathcal{D}(\omega)_{k\ell} = \omega^{k\ell}$ (האינדקסים הם בתחום $0,\ldots,n-1$).
אם־כן, האינטרפולציה היא גם כן התמרה לינארית שמטריצה שלה היא המטריצה ההפוכה ל־$\mathcal{D}(\omega)$.

טענה 5.1 המטריצה ההפוכה ל $\mathcal{D}(\omega)$ היא $\frac{1}{n}\mathcal{D}(\omega^{-1})$.

הוכחה. ניזכר שהמטריצה $\mathcal{D}(\omega)$ מוגדרת כ־$\mathcal{D}(\omega)_{k\ell} = \omega^{k\ell}$. כמו־כן, $\mathcal{D}(\omega^{-1})_{k\ell} = \omega^{-k\ell}$. נסמן את מכפלת המטריצות הללו כך: $\mathcal{I} = \mathcal{D}(\omega)\cdot\frac{1}{n}\mathcal{D}(\omega^{-1})$.
אנו צריכים להראות ש־$\mathcal{I}$ היא מטריצת הזהות. נחשב תחילה את ערכי האלכסון של $\mathcal{I}$:

$$\mathcal{I}_{kk} = \frac{1}{n}\sum_{\ell=0}^{n-1}\mathcal{D}(\omega)_{k\ell}\mathcal{D}(\omega^{-1})_{\ell k}$$

$$= \frac{1}{n}\sum_{\ell=0}^{n-1}\omega^{k\ell}\cdot\omega^{-k\ell} = \frac{1}{n}\sum_{\ell=0}^{n-1}\omega^{k\ell-k\ell} = \frac{1}{n}\sum_{\ell=0}^{n-1}1 = 1.$$

כעת נראה שמחוץ לאלכסון הראשי, ערכי $\mathcal{I}$ הם כולם 0. כמובן שלצורך כך אנו יכולים להניח ש־$n>1$ (עבור $n=1$ המטריצות הן מגודל 1×1 ואין איברים מחוץ לאלכסון הראשי). נקבע $k \neq k'$

$$\mathcal{I}_{kk'} = \frac{1}{n}\sum_{\ell=0}^{n-1}\mathcal{D}(\omega)_{k\ell}\mathcal{D}(\omega^{-1})_{k'\ell} = \frac{1}{n}\sum_{\ell=0}^{n-1}\omega^{k\ell}\cdot\omega^{-\ell k'} = \frac{1}{n}\sum_{\ell=0}^{n-1}\omega^{(k-k')\ell}$$

נותר לראות ש־$\sum_{\ell=0}^{n-1}\widehat{\omega}^{\ell} = 0$, כאשר $\widehat{\omega} = \omega^{k-k'}$. נשים לב $\widehat{w}$ הינו שורש יחידה (לא בהכרח פרמיטיבי) כיוון ש־

$$\widehat{w}^n = (\omega^n)^{k-k'} = 1^{k-k'} = 1.$$

כמו־כן, $\widehat{\omega} \neq 1$ כיוון ש־ω שורש יחידה פרמיטיבי ו־$0 < |k-k'| < n$.

כדי להוכיח ש־$\sum_{\ell=0}^{n-1}\widehat{\omega}^{\ell} = 0$, ניזכר ששורש יחידה מסדר n מקיים את המשוואה $x^n = 1$, כלומר $\widehat{\omega}$ מאפס את הפולינום $x^n - 1$. אנו יכולים לרשום את הפולינום $x^n - 1$ בעזרת נוסחת טור גיאומטרי כ:

$$x^n - 1 = (x-1)\cdot(1 + x + x^2 + \ldots + x^{n-1}) = (x-1)\cdot\left(\sum_{\ell=0}^{n-1}x^{\ell}\right)$$

נציב בחזרה $\widehat{\omega}$ בפולינום זה ונקבל

$$0 = \widehat{\omega}^n - 1 = (\widehat{\omega}-1)\cdot\left(\sum_{\ell=0}^{n-1}\widehat{\omega}^{\ell}\right). \tag{5.2}$$

נשים לב: באגף ימין של (5.2) יש מכפלה של שני איברים השווה ל־0. האיבר הראשון – $(\widehat{\omega}-1)$ – איננו 0 כיוון שהראינו ש־$\widehat{\omega} \neq 1$, ולכן האגף השני הוא 0, כלומר:

$$\sum_{\ell=0}^{n-1}\widehat{\omega}^{\ell} = 0,$$

וזה בדיוק מה שנדרשו להוכיח. □

כלומר, כדי לחשב את מקדמי פולינום האינטרפולציה, עלינו להפעיל את הטרנספורמציה הלינארית $\frac{1}{n}\mathcal{D}(\omega^{-1})$. כיוון שגם ω^{-1} הוא שורש יחידה פרמיטיבי מסדר n (ראו תרגיל 5.10), אפשר לחשב את הטרנספורמציה $\mathcal{D}(\omega^{-1})$ בעזרת השגרה $\mathrm{FFT}(\cdot, \omega^{-1})$. בזאת סיימנו את תיאור האלגוריתם המהיר לחישוב כפל פולינומים.

תרגיל 5.10 הוכיחו שאם ω הוא שורש פרימיטיבי מסדר n, אזי גם ω^{-1} הוא שורש פרימיטיבי מסדר n.

פתרון בעמוד 105

תרגיל 5.11 הריצו את $\mathrm{FFT}((-1, 2, 5, -4), \omega)$. ואחרי־כן הריצו את $\mathrm{FFT}(\cdot, \omega^{-1})$ על הפלט מההרצה הקודמת. השוו את התשובה שקיבלתם לקלט המקורי, $(-1, 2, 5, -4)$.

פתרון בעמוד 105

תרגיל 5.12 חשבו את התמרת פורייה הבדידה של הסדרה $(-1, -i, 2, 2i, 5, 5i, -4, -4i)$.

פתרון בעמוד 105

תרגיל 5.13 פתחו אלגוריתם שיחשב במהירות את התמרת פורייה הבדידה לסדרה באורך n כאשר n הוא חזקה של 3.

פתרון בעמוד 106

5.6 פתרונות לתרגילים

פתרון תרגיל 5.2 מעמוד 89

נסמן ב־$d_{\text{ex}}(\sigma)$ את מרחק השיחלופים של סדרה σ, וב־$\text{rev}(\sigma)$ את מספר ההיפוכים בה. עלינו להוכיח כי $d_{\text{ex}}(\sigma) = \text{rev}(\sigma)$ לכל סדרה σ.

נגדיר "היפוך צמוד" כזוג $(i, i+1)$ עבורו $a_i > a_{i+1}$. כעת שימו לב, אם בסדרה נתונה יש היפוכים (כלומר היא אינה מונוטונית עולה), אז יש בה לפחות "היפוך צמוד" אחד. כדי לראות זאת, נבחר היפוך (i,j), $i < j$, כלומר $a_i > a_j$. אם $j = i+1$, אז סיימנו (זה היפוך צמוד), אחרת, בהכרח $j > i+1$, ו־ $(i+1, j)$ הוא היפוך, ובאינדוקציה על $j - i$ סיימנו גם כן.

כמו־כן נשים לב ששיחלוף של היפוך צמוד $(i, i+1)$ מקטין את מספר ההיפוכים בסדרה באחד: לאחר השיחלוף, הזוג $(i, i+1)$ כבר אינו היפוך וכל היפוך אחר שבו הופיעו i או $i+1$ נשאר היפוך (עם החלפת i ב־$i+1$, והחלפת $i+1$ ב־i).

שילוב שתי הטענות לעיל מאפשר להוכיח באינדוקציה על $\text{rev}(\sigma)$ את אי־השוויון $d_{\text{ex}}(\sigma) \leq \text{rev}(\sigma)$: אם $\text{rev}(\sigma) = 0$, הטענה ברורה (הסדרה היא מונוטונית עולה ולכן גם $d_{\text{ex}}(\sigma) = 0$). אחרת, אם $\text{rev}(\sigma) > 0$, אזי לפי האמור לעיל, קיים שיחלוף (של היפוך צמוד) שמייצר סדרה σ' עבורה $\text{rev}(\sigma') = \text{rev}(\sigma) - 1$. מהנחת האינדוקציה, $d_{\text{ex}}(\sigma') \leq \text{rev}(\sigma')$ ולכן

$$d_{\text{ex}}(\sigma) \leq d_{\text{ex}}(\sigma') + 1 \leq \text{rev}(\sigma') + 1 = \text{rev}(\sigma).$$

בכיוון השני נוכיח באינדוקציה על $d_{\text{ex}}(\sigma)$ ש־ $\text{rev}(\sigma) \leq d_{\text{ex}}(\sigma)$. אם $d_{\text{ex}}(\sigma) = 0$ אז הסדרה מונוטונית עולה ולכן גם $\text{rev}(\sigma) = 0$. אם $d_{\text{ex}}(\sigma) > 0$, אזי מהגדרת d_{ex} קיים שיחלוף של σ המייצר סדרה σ' עבורה $d_{\text{ex}}(\sigma') = d_{\text{ex}}(\sigma) - 1$. שימו לב, $|\text{rev}(\sigma) - \text{rev}(\sigma')| \leq 1$: עבור שיחלוף $(i, i+1)$ כל הזוגות, למעט $(i, i+1)$ נשארים באותו היפוך או לא־היפוך (תוך שינוי אינדקס i ל־$i+1$ ולהפך). מהנחת האינדוקציה $\text{rev}(\sigma') \leq d_{\text{ex}}(\sigma')$ ולכן

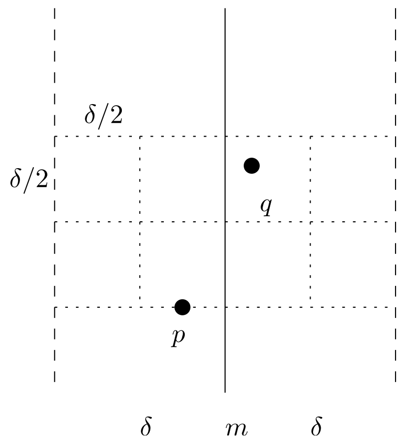
$$\text{rev}(\sigma) \leq \text{rev}(\sigma') + 1 \leq d_{\text{ex}}(\sigma') + 1 = d_{\text{ex}}(\sigma).$$

♣

פתרון תרגיל 5.3 מעמוד 91

1. הארגומנט בספר לא עבר אופטימיזציה, לכן קל מאוד לשפרו. אנו נשפר אותו ל־7 ללא מאמץ רב בצורה הזו (ראו גם איור 5.2): נניח כי קיים זוג נקודות $p = (x_1, y_1) \in P'_L$ ו־$q = (x_2, y_2) \in P'_R$, כך ש־ $d(p,q) < \delta$. בלי הגבלת הכלליות, נניח כי $y_1 < y_2$. נצייר ריבועים (כמו באיור 7.5 בספר) בגודל $\frac{\delta}{2} \times \frac{\delta}{2}$, ונדאג שהבסיס התחתון של הריבועים יהיה ב־y_1, (ראו איור 5.2). במקרה זה ברור ש־q יכול להימצא לכל היותר בשורה אחת גבוה יותר, אחרת יתקיים ש $d(p,q) \geq \delta$. כיוון שבכל שורה יש ארבעה ריבועים, ובכל אחד מהריבועים יכולה להיות לכל היותר נקודה אחת (ראו הטיעון בהוכחת (5.10) בספר), הנקודה q יכולה להופיע לכל היותר כנקודה השביעית לאחר p במיון עולה לפי ציר y של $P'_L \cup P'_R$.

2. נניח שהחציון m עובר בקו $x = 0$, $\delta = 1$, $p = (0,0)$, $q = (0.1, 0.99)$ וחוץ מנקודות אלה מופיעות גם הנקודות $(-0.999, 0)$, $(0.999, 0)$. במקרה זה, q תופיע רק כנקודה מספר שלוש לאחר p במיון לפי ציר־y של $P'_L \cup P'_R$. ♣



איור 5.2: שיפור החסם על ההפרש בין הנקודות הקרובות ביותר p, q לקו החציון m משני צדדיו: נצייר ריבועים בגודל $\frac{\delta}{2} \times \frac{\delta}{2}$ שבסיסם ב־p. כיוון ש־$d(p,q) < \delta$, הנקודה q חייבת להימצא בשורה השנייה לכל היותר, כלומר במסגרת של שמונה ריבועים, וכל ריבוע יכול להכיל נקודה אחת לכל היותר.

פתרון תרגיל 5.4 מעמוד 91

להלן דוגמה שתעזור לנו להיזכר באלגוריתם לביצוע כפל ארוך (בבסיס 2) כפי שנלמד בבית־הספר היסודי.

$$
\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
 & & & & & & & \times \\
 & & & & 1 & 1 & 0 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 0 & 0 & 0 & 0 & \\
 & & 1 & 1 & 0 & 1 & & \\
 & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1
\end{array}
$$

כדי לתאר את האלגוריתם נניח שנתונים לנו שני מערכי סיביות $(a_0, \ldots, a_{n-1})$ ו־$(b_0, \ldots, b_{n-1})$; וש־a_0 ו־b_0 הן הסיביות הפחות משמעותיות. האלגוריתם לביצוע כפל ארוך מתואר כדלקמן:

Algorithm 5.2 Long-Multiplication$((a_0,\ldots,a_{n-1}),(b_0,\ldots,b_{n-1}))$

```
Initialize c_0, ..., c_{2n-1} ← 0
for i ← 0, ..., n − 1 do
    carry ← 0
    if a_i = 1 then
        for j ← 0, ..., n − 1 do
            c_{i+j} ← c_{i+j} ⊕ b_j ⊕ carry {⊕ is the XOR operatoin}
            carry ← (c_{i+j} ∧ b_j) ∨ (c_{i+j} ∧ carry) ∨ (b_j ∧ carry)
        c_{i+n} ← carry
return (c_0, ..., c_{2n-1})
```

♣

פתרון תרגיל 5.5 מעמוד 92

אנו כותבים:

$$\alpha = ac, \qquad \beta = bd, \qquad \gamma = (a+b)\cdot(c+d),$$

ואז

$$(a,b)\cdot(c,d) = (\alpha-\beta, \gamma-\alpha-\beta).$$

♣

פתרון תרגיל 5.6 מעמוד 92

נניח (לשם הפשטות) ש־n היא חזקה של 2. בדומה לכפל מספרים שלמים, נסמן כך את הפונקציות:

$$f(x) = f_1(x)x^{n/2} + f_0(x), \qquad g(x) = g_1(x)x^{n/2} + g_0(x)$$

ואז

$$f(x)g(x) = f_1(x)g_1(x)x^n + ((f_1(x)g_0(x)) + g_1(x)f_0(x))x^{n/2} + f_0(x)g_0(x).$$

וכרגיל, נוכל להסתדר עם שלוש פעולות כפל רקורסיביות בלבד על־ידי הבחירה הזו:

$$\begin{aligned}
\alpha(x) &= f_1(x)g_1(x),\\
\beta(x) &= f_0(x)g_0(x),\\
\gamma(x) &= (f_1(x)+f_0(x))\cdot(g_1(x)+g_0(x)).
\end{aligned}$$

לאחר חישוב $\alpha(x), \beta(x), \gamma(x)$ נותר לחשב את הביטוי

$$f(x)g(x) = \alpha(x)x^n + (\gamma(x)-\beta(x)-\alpha(x))x^{n/2} + \beta(x)$$

תיאור מפורט מובא באלגוריתם 5.3.

Algorithm 5.3 Polynomial_Multiplication$((a_0,\dots a_{n-1}),(b_0,\dots,b_{n-1}))$

Require: n is a power of two (for simplicity of the presentation)

$(\alpha_0,\dots,\alpha_{n-1}) \leftarrow$Polynomial_Multiplication$((a_{n/2},\dots,a_{n-1}),(b_{n/2},\dots,b_{n-1}))$

$(\beta_0,\dots,\beta_{n-1}) \leftarrow$Polynomial_Multiplication$((a_0,\dots,a_{n/2-1}),(b_0,\dots,b_{n/2-1}))$

$(\gamma_0,\dots,\gamma_{n-1}) \leftarrow$Polynomial_Multiplication$((a_0+a_{n/2},\dots,a_{n/2-1}+a_{n-1}),(b_0+b_{n/2},\dots,b_{n/2-1}+b_{n-1}))$

$$\delta_i \leftarrow \begin{cases} \beta_i & 0 \le i \le n/2-1 \\ \beta_i + (\gamma_{i-n/2} - \alpha_{i-n/2} - \beta_{i-n/2}) & n/2 \le i \le n-1 \\ \alpha_{i-n} + (\gamma_{i-n/2} - \alpha_{i-n/2} - \beta_{i-n/2}) & n \le i \le 3n/2-1 \\ \alpha_{i-n} & 3n/2 \le i \le 2n-1 \end{cases}$$

return $(\delta_0,\dots,\delta_{2n-1})$

♣

פתרון תרגיל 5.7 מעמוד 94

האלגוריתם הנאיבי לחישוב $f(x_0)$ יקח $O(n^2)$ זמן. ההבחנה המאפשרת את שיפור זמן הריצה היא שניתן לכתוב את $f(x)$ כדלקמן:

$$f(x) = (\cdots((a_{n-1}x + a_{n-2})x + a_{n-2})x + \cdots + a_1)x + a_0.$$

נוסחה זאת מובילה ישירות לאלגוריתם הבא:

Algorithm 5.4 Poly–eval$((a_0,\dots,a_{n-1}),x_0)$

$y_0 \leftarrow a_{n-1}$

for $k \leftarrow n-2$ down to 0 **do**

  $y_0 \leftarrow fx_0 + a_k$

return y_0

♣

פתרון תרגיל 5.8 מעמוד 94

תחילה נוכיח שקיים פולינום אינטרפולציה. זוהי נוסחת לאגראנז' שמוגדרת כדלקמן:

$$f(x) = \sum_{k=0}^{n-1} y_k \cdot \frac{\prod_{j:j\neq k}(x-x_j)}{\prod_{j:j\neq k}(x_k-x_j)}. \tag{5.3}$$

נשים לב שבהגדרה אין חלוקה ב־0, כיוון שהנחנו ש־$x_j \neq x_k$ כאשר $j \neq k$.

נראה עתה שלכל $t \in \{0,\dots,n-1\}$, $f(x_t) = y_t$. ב־(5.3) מופיע סכום של n איברים. נתבונן באיבר ה־k בסכום. כאשר $k \neq t$ נקבל

$$y_k \cdot \frac{\prod_{j:j\neq k}(x_t-x_j)}{\prod_{j:j\neq k}(x_k-x_j)} = 0,$$

כיוון שבמכפלה במונה מופיע הביטוי $x_t - x_t = 0$, כאשר $j = t$. לעומת זאת כאשר $k = t$,

$$y_k \cdot \frac{\prod_{j:j\neq k}(x_t-x_j)}{\prod_{j:j\neq k}(x_k-x_j)} = y_t \cdot \frac{\prod_{j:j\neq t}(x_t-x_j)}{\prod_{j:j\neq t}(x_t-x_j)} = y_t,$$

כיוון שהמונה שווה למכנה. מכאן שסכום האיברים – $f(x_t)$ – שווה ל־y_t, כנדרש.

נוכיח עתה את יחידות הפתרון. לשם כך נשתמש באלגברה לינארית. נסמן ב־W את המטריצה מסדר $n\times n$ כך ש $W_{\ell k} = x_\ell^k$ (האינדקסים בטווח $\{0,\ldots,n-1\}$. כלומר

$$W = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{pmatrix}.$$

בפיסקה הקודמת, בהוכחת קיום פולינום אינטרפולציה, הוכחנו בעצם שלכל וקטור $y = (y_0,\ldots,y_{n-1})^T \in \mathbb{C}^n$, קיים וקטור מקדמים $a = (a_0,\ldots,a_{n-1})^T$ המקיים

$$W \cdot a = y.$$

במושגים של העתקות לינאריות ההעתקה $W : \mathbb{C}^n \to \mathbb{C}^n$, היא "על" ולכן היא חייבת להיות ממעלה n. לפי כללי האלגברה הלינארית אנו יודעים שזה ייתכן אם ורק אם W היא חד־חד ערכית, כלומר הפתרון a הוא יחיד, כנדרש.

נותר להציג אלגוריתם העובד בזמן $O(n^2)$ לחישוב מקדמי פולינום האינטרפולציה. אלגוריתם כזה נובע כמעט ישירות מנוסחת לאגראנז' (5.3). מספיק להראות שעבור k קבוע ניתן לחשב את הביטוי

$$y_k \cdot \frac{\prod_{j:j\neq k}(x - x_j)}{\prod_{j:j\neq k}(x_k - x_j)}$$

בזמן $O(n)$. זאת נעשה כדלקמן: המכנה הינה מכפלה של $n-1$ מספרים שמתבצעת בקלות בזמן $O(n)$. המונה הוא פולינום, שניתן לקבלו בייצוג לפי מקדמים בזמן לינארי בצורה הבאה: מחשבים פעם אחת את מקדמי הפולינום $g(x) = \prod_{j=0}^{n-1}(x - x_j)$. עתה נותר רק להבחין ש

$$\prod_{j:j\neq k}(x - x_j) = \frac{g(x)}{x - x_k}.$$

את אגף ימין ניתן לחשב בזמן לינארי בעזרת אלגוריתם לחילוק פולינומים שנלמד בבית־הספר. ♣

פתרון תרגיל 5.9 מעמוד 97

Algorithm 5.5 concrete-FFT$((a_0,\ldots,a_{n-1}))$

Require: n is a power of 2.
  if $n = 1$ **then**
    return $(f(1) = a_0)$
  $(b_0, b_1, \ldots, b_{\frac{n}{2}-1}) \leftarrow$concrete-FFT$((a_0, a_2 \ldots, a_{n-4}, a_{n-2}))$
  $(c_0, c_1, \ldots, c_{\frac{n}{2}-1}) \leftarrow$concrete-FFT$((a_1, a_3, \ldots, a_{n-3}, a_{n-1}))$
  for $k \leftarrow 0, 1, \ldots, \frac{n}{2} - 1$ **do**
    $d_k \leftarrow b_k + e^{2\pi ki/n} c_k$
    $d_{k+n/2} \leftarrow b_k - e^{2\pi ki/n} c_k$
  return $(d_0, \ldots, d_{n-1})$

♣

פתרון תרגיל 5.10 מעמוד 99

תחילה נראה ש-ω^{-1} הוא שורש יחידה מסדר n. ואמנם

$$(\omega^{-1})^n = \omega^{-n} = (\omega^n)^{-1} = 1^{-1} = 1.$$

שנית נראה שזהו שורש פרימיטיבי, כלומר לכל $n > k > 0$, $(\omega^{-1})^k \neq 1$. ואמנם,

$$(\omega^{-1})^k = \omega^{-k} = (\omega^k)^{-1} \neq 1^{-1} = 1.$$

♣

פתרון תרגיל 5.11 מעמוד 99

בתרגיל זה $n = 4$. נקבע $\omega = e^{2\pi i/4} = i$.

1. נחשב תחילה את $\mathrm{FFT}((-1, 2, 5, -4), \omega)$

1. Calling $\mathrm{FFT}((-1, 2, 5, -4), \omega = i)$
   - I. Calling $\mathrm{FFT}((-1, 5), \omega^2 = -1)$
     - a. Calling $\mathrm{FFT}((-1), \omega^4 = 1)$, returning (-1)
     - b. Calling $\mathrm{FFT}((5), \omega^4 = 1)$, returning (5)
     - c. Return $(-1 + 1 \cdot 5, -1 + \omega^2 \cdot 5) = (4, -6)$
   - II. Calling $\mathrm{FFT}((2, -4), \omega^2 = -1)$
     - a. Calling $\mathrm{FFT}((2), \omega^4 = 1)$, returning (2)
     - b. Calling $\mathrm{FFT}((-4), \omega^4 = 1)$, returning (-4)
     - c. Return $(2 + 1 \cdot (-4), 2 - 1 \cdot (-4)) = (-2, 6)$
   - III. Calculate $f(1) = 4 + 1 \cdot (-2) = 2$, $f(i) = -6 + i \cdot 6$, $f(-1) = 4 + (-1) \cdot (-2)$, $f(-i) = -6 - i \cdot 6$,
   - IV. Return $(2, -6 + 6i, 6, -6 - 6i)$

2. נריץ כעת את $\mathrm{FFT}((2, -6 + 6i, 6, -6 - 6i), \omega^{-1})$

1. Calling $\mathrm{FFT}((2, -6 + 6i, 6, -6 - 6i), \omega^{-1} = -i)$
   - I. Calling $\mathrm{FFT}((2, 6), \omega^{-2} = -1)$
     - a. Calling $\mathrm{FFT}((2), \omega^{-4} = 1)$, returning (2)
     - b. Calling $\mathrm{FFT}((6), \omega^{-4} = 1)$, returning (6)
     - c. Return $(2 + 1 \cdot 6, 2 + \omega^{-2} \cdot 6) = (8, -4)$
   - II. Calling $\mathrm{FFT}((-6 + 6i, -6 - 6i), \omega^{-2} = -1)$
     - a. Calling $\mathrm{FFT}((-6 + 6i), \omega^{-4} = 1)$, returning $(-6 + 6i)$
     - b. Calling $\mathrm{FFT}((-6 - 6i), \omega^{-4} = 1)$, returning $(-6 - 6i)$
     - c. Return $(-6 + 6i + 1 \cdot (-6 - 6i), -6 + 6i + \omega^{-2} \cdot (-6 - 6i) = (-12, 12i)$
   - III. Calculate $f(1) = 8 + 1 \cdot (-12) = -4$, $f(i) = -4 + \omega^{-1} \cdot 12i = 8$, $f(-1) = 8 + \omega^{-2} \cdot (-12) = 20$, $f(-i) = -4 + \omega^{-3} \cdot 12i = -16$,
   - IV. Return $(-4, 8, 20, -16)$

קיבלנו את הסדרה המקורית מוכפלת ב-$n = 4$, כפי שצריך להיות.

♣

פתרון תרגיל 5.12 מעמוד 99

נפתור שאלה זאת בעזרת FFT ופתרון תרגיל 5.11. בתרגיל זה $n = 8$,

ו־$\omega = e^{2\pi i/8} = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}}{2}i$. כדי לחשב את ה־FFT של הסדרה הנתונה, נפריד את סדרת המספרים לאלו במקומות הזוגיים ולאלו במקומות האי־זוגיים.

סדרת המקומות הזוגיים (כשמספור האינדקסים מתחיל ב־0!) היא $(-1, 2, 5, -4)$ ועלינו לפתור את $\mathrm{FFT}((-1, 2, 5, -4), \omega^2 = i)$ אך זו בדיוק הבעיה שפתרנו בתרגיל 5.11, ומשם אנו יודעים שהפתרון הוא

$$(b_0, \ldots, b_3) = (2, -6 + 6i, 6, -6 - 6i).$$

סדרת המקומות האי־זוגיים היא $(-i, 2i, 5i, -4i)$. ועלינו לפתור את $\mathrm{FFT}((-i, 2i, 5i, -4i), \omega^2 = i)$. כפי שכבר אמרנו FFT הינה פרוצדורה לחישוב DFT, וכפי שראינו DFT הינה התמרה לינארית. כיוון ש־

$$(-i, 2i, 5i, -4i) = i \cdot (-1, 2, 5, -4),$$

אנו מסיקים (מלינאיריות ההתמרה) ש־

$$\begin{aligned}(c_0, \ldots, c_3) &= \mathrm{DFT}((-i, 2i, 5i, -4i)) = i \cdot \mathrm{DFT}((-1, 2, 5, -4)) \\ &= i \cdot (2, -6 + 6i, 6, -6 - 6i) = (2i, -6 - 6i, 6i, 6 - 6i),\end{aligned}$$

השוויון השלישי הוא החישוב שעשינו לעיל.

נותר שלב "המיזוג" של תת־הפתרונות. תוך שימוש בסימונים של פתרון תרגיל 5.9,

$$\begin{aligned}
d_0 &= b_0 + c_0 = 2 + 2i, & d_4 &= b_0 - c_0 = 2 - 2i \\
d_1 &= b_1 + e^{2\pi i/8}c_1 = -6 + i(6 - 6\sqrt{2}) & d_5 &= b_1 - e^{2\pi i/8}c_1 = -6 + i(6 + 6\sqrt{2}) \\
d_2 &= b_2 + e^{2\cdot 2\pi i/8}c_2 = 0 & d_6 &= b_2 - e^{2\cdot 2\pi i/8}c_2 = 12 \\
d_3 &= b_3 + e^{3\cdot 2\pi i/8}c_3 = -6 + i(6\sqrt{2} - 6) & d_7 &= b_3 - e^{3\cdot 2\pi i/8}c_3 = -6 + i(-6 - 6\sqrt{2})
\end{aligned}$$

♣

פתרון תרגיל 5.13 מעמוד 99

אנו נפתח FFT המשתמש בחלוקות ל־3 במקום חלוקות ל־2 כפי שהיה באלגוריתם 5.1. פרט לשינוי הזה, הרעיונות יהיו זהים. נניח ש־n היא חזקה של 3. בהינתן סדרת מספרים $(a_0, a_1, \ldots, a_{n-1})$ נסתכל על הפולינום $f(x) = \sum_{k=0}^{n-1} a_k x^k$, ונכתוב אותו כך:

$$f(x) = f_0(x^3) + x f_1(x^3) + x^2 f_2(x),$$

כאשר

$$f_0(x) = \sum_{k=0}^{n/3-1} a_{3k} x^k$$

$$f_1(x) = \sum_{k=0}^{n/3-1} a_{3k+1} x^k$$

$$f_2(x) = \sum_{k=0}^{n/3-1} a_{3k+2} x^k$$

יהי ω שורש יחיד פרימיטיבי מסדר n. מכאן ש ω^3 הינו שורש יחידה פרמיטיבי מסדר $n/3$. לכן, ערכם של הפולינומים $f_h(x)$, $h \in \{0,1,2\}$, בנקודות $1, \omega^3, \omega^6, \ldots, \omega^{\frac{n}{3}-1}$ הם התמרת פורייה ההבדידה מסדר $n/3$ של הסדרות $(a_{3k+h})_{k\in\{0,1,\ldots,n/3-1\}}$, עבור $h \in \{0,1,2\}$ בהתאמה. קיבלנו אם כן נוסחה רקורסיבית לחישוב התמרת פורייה הבדידה, והיא מובילה ישירות לאלגוריתם 5.6.
♣

Algorithm 5.6 FFT_B3$((a_0, \ldots, a_{n-1}), \omega)$

Require: $\omega^n = 1$
Require: $\forall 0 < k < n$, $\omega^k \neq 1$.
Require: n is a power of 3.
  if $n = 1$ **then**
    return $(f(1) = a_0)$
  $(f_0(1), f_0(\omega^3), f_0(\omega^6), \ldots, f_0(\omega^{n-3}) \leftarrow$FFT_B3$((a_0, a_3, a_6, \ldots, a_{n-3}), \omega^3)$
  $(f_1(1), f_1(\omega^3), f_1(\omega^6), \ldots, f_1(\omega^{n-3}) \leftarrow$FFT_B3$((a_1, a_4, a_7, \ldots, a_{n-2}), \omega^3)$
  $(f_2(1), f_2(\omega^3), f_2(\omega^6), \ldots, f_2(\omega^{n-3}) \leftarrow$FFT_B3$((a_2, a_5, a_8, \ldots, a_{n-1}), \omega^3)$
  for $k \leftarrow 0, \ldots, n-1$ **do**
    $f(\omega^k) \leftarrow f_0(\omega^{3k}) + \omega^k f_1(\omega^{3k}) + \omega^{2k} f_2(\omega^3)$
  return $(f(\omega^0), f(\omega^1), \ldots, f(\omega^{n-1}))$

פרק 6

תכנון דינמי

6.1 תזמון מקטעים

קראו בספר מתחילת פרק 6 עד סוף סעיף 6.2

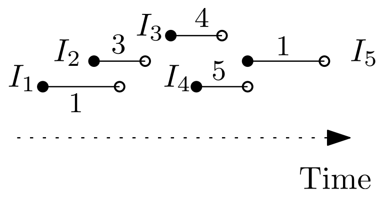
תכנון דינמי [Dynamic Programming] פותר בעיות אופטימיזציה על־ידי פתרון איטרטיבי של תת־בעיות ש"גודלן" הולך וגדל, כך שהמעבר מפתרון לתת־בעיה בגודל i לתת־בעיה בגודל $i+1$ הוא פשוט (יחסית). תיאור זה נכון גם לשיטות חמדניות ולשיטות הפרד־ומשול. ההבדל המשמעותי הוא שההחלטה בתכנון דינמי נעשית לרוב על־סמך "מידע רב יותר": בפתרון דינמי נוטים לחשב פתרון להרבה תת־בעיות בגודל i לפני שפותרים תת־בעיות בגודל $i+1$. נובע מכך שתכנון דינמי נוטה להיות בעל זמני ריצה ארוכים יותר, אך פותר מגוון רחב יותר של בעיות אלגוריתמיות. בהקשר זה אפשר לראות חלק גדול מהאלגוריתמים החמדנים כמקרים פרטיים – מנוונים – של תכנון דינמי.

הדוגמה הראשונה לאלגוריתם תכנון דינמי המובאת בספר היא פתרון בעיה של תזמון מקטעים ממושקלים.

בעיה אלגוריתמית: תזמון מקטעים ממושקלים.
הקלט: קבוצה $\mathcal{I} = \{I_1, I_2 \ldots, I_n\}$ של n מקטעים [Intervals], כאשר לכל מקטע $I_i = [s_i, f_i)$ יש משקל/ערך v_i.
הפלט: תת־קבוצה $\mathcal{J} \subseteq \mathcal{I}$ של מקטעים זרים בזוגות.
המטרה: מבין תת־הקבוצות של $\mathcal{I}$, אשר מכילים רק מקטעים זרים בזוגות, יהיה לתת־הקבוצה $\mathcal{J}$ משקל מקסימלי (השווה לסכום משקלי המקטעים שלה).

באיור 6.1 מוצגת דוגמה לקלט של בעיית תזמון מקטעים ממושקלים.

אלגוריתם תכנון דינמי לבעיה זו, בדומה לאלגוריתם החמדן לבעיה הלא־ממושקלת, מסדר את המקטעים לפי זמן סיום עולה, בוחן אותם אחד־אחד ומנסה לצרפם לפתרון האופטימלי. האלגוריתם החמדן ניסה "רק" להוסיף את המקטע I_i לפתרון האופטימלי של $I_1, \ldots, I_{i-1}$, בתנאי ש־I_i אינו חותך את הפתרון האופטימלי של $I_1, \ldots, I_{i-1}$. במקרה הממושקל, זה לא מספיק, כפי שרואים בספר באיור 6.1. במקום זאת, "זוכרים" בתכנון הדינמי את הפתרונות האופטימליים לכל תת־הבעיות שצורתן עבור $i > j$ היא $I_1, \ldots, I_j$, וכך אפשר לקבל החלטה טובה



איור 6.1: דוגמת קלט של בעיית תזמון קטעים ממושקלים. האלגוריתם החמדן לבעיה הלא־ממושקלת יחזיר $\{I_1, I_3, I_5\}$ שערכו 6, אבל הפתרון האופטימלי הוא $\{I_2, I_4, I_5\}$ שערכו 9.

יותר. למרות הפשטות לכאורה של גישה זאת, ההחלטה מה בדיוק צריך לזכור היא מהותית. כדי להדגים זאת, נבחן כמה דרכים שונות לזכור בעיות קטנות יותר:

1. לכל $i > j$ נזכור את ערך הפתרון האופטימלי לתת־הבעיה $I_1, \ldots, I_j$ (שיכול לכלול או לא לכלול את I_j). זו בחירה מוצלחת, המאפשרת להגדיר אלגוריתם יעיל, כפי שמודגם בסעיף 6.2 בספר. אבל זו אינה האפשרות היחידה.

2. נזכור לכל $i > j$ את ערך הפתרון האופטימלי לתת־הבעיה $I_1, \ldots, I_j$ מבין הפתרונות החוקיים שמכילים את I_j. גישה זו מובילה לאלגוריתם תכנון דינמי הדומה לאלגוריתם המוצג בספר, אך הוא פחות יעיל באופן משמעותי. בתרגיל 6.1 תתבקשו לכתוב ולנתח אלגוריתם המתבסס על גישה זאת.

3. בדומה לאפשרות 1, נזכור לכל $i > j$ את ערך הפתרון האופטימלי לתת־הבעיה $I_1, \ldots, I_j$ (עם או בלי I_j), ובנוסף נזכור גם את הפתרון האופטימלי עצמו. זו גישה "נאיבית". היא מאפשרת להחזיר גם את הפתרון האופטימלי ולא רק את ערכו, אך היא פחות יעילה באופן משמעותי. כפי שראינו בסעיף 6.1 בספר, לא צריך לשמור את הפתרונות האופטימליים לכל תתי־הבעיות הנוצרות במהלך האיטרציה כדי לבנות את הפתרון האופטימלי.

4. לכל תת קבוצה של $I_1, \ldots, I_{i-1}$ נזכור אם היא חוקית, ואם כן נזכור את ערכה. פיתוח גישה זו יוביל לאלגוריתם סריקה מקיף, הרץ בזמן מעריכי (אקספוננציאלי), כלומר, קל להתדרדר לפתרון לא יעיל עקב מעט חוסר תשומת לב.

תרגיל 6.1 כתבו אלגוריתם המבוסס על אפשרות 2. הוכיחו את נכונות האלגוריתם ונתחו את זמן הריצה וגודל הזיכרון הנדרש.

6.2 בעיית התרמיל

קראו בספר את סעיף 6.4

אם הקלטתם פעם שירים ל־CD יתכן שרציתם "לנצל" את הדיסק כמה שיותר, כלומר למלא אותו ככל האפשר בשירים מבלי לצרוב חלקי שיר. הבעיה הבאה הינה ניסוח אלגוריתמי כללי לבעיות מסוג זה.

בעיה אלגוריתמית: בעיית סכומי תת־קבוצות [Subset Sum Problem].
הקלט: n פריטים $\{1,\dots,n\}$, משקלות לפריטים $w_i \in \mathbb{N}$, וחסם $W \in \mathbb{N}$.
הפלט: תת־קבוצה $S \subseteq \{1,\dots,n\}$, כך ש־$\sum_{i\in S} w_i \le W$.
המטרה: להביא למקסימום את המשקל $w(S) = \sum_{i\in S} w_i$ של S.

אפשר לפתור את בעיית סכומי תת־קבוצות בעזרת תכנון דינמי: מגדירים את הטבלה $M(i,w)$ שמכילה את הסכום המקסימלי שניתן להגיע אליו מתוך תת־קבוצה של $\{w_1,\dots,w_i\}$ תחת האילוץ שהסכום הוא לכל היותר w. טבלה זאת מקיימת את הנוסחה הרקורסיבית הזו:

$$M(i,w) = \begin{cases} 0 & i = 0 \\ M(i-1,w) & w_i > w \\ \max\{M(i-1,w), w_i + M(i-1,w-w_i)\} & \text{otherwise.} \end{cases}$$

הסבר: נסתכל על הפתרון האופטימלי O של בעיית סכומי תת־קבוצות של $\{w_1.\dots,w_i\}$ שערכם לכל היותר w. יתכנו שני מקרים:

1. $i \in O$. במקרה זה $O \setminus \{i\}$ הוא פתרון אופטימלי של בעיית סכומי תת־קבוצות של $\{w_1.\dots,w_{i-1}\}$ שערכם לכל היותר $w - w_i$, ולכן מהנחת האינדוקציה ערך הפתרון האופטימלי הוא $w_i + M(i-1, w-w_i)$.
2. $i \notin O$. במקרה זה O הוא פתרון אופטימלי של בעיית סכומי תת־קבוצות של $\{w_1.\dots,w_{i-1}\}$ שערכם לכל היותר w, ולכן מהנחת האינדוקציה ערך הפתרון האופטימלי הוא $M(i-1,w)$.

כמובן שהפתרון האופטימלי הוא המקסימום מבין האפשרויות שלעיל.

הנוסחה שלעיל מגדירה אלגוריתם תכנון דינמי לחישוב הטבלה, והתשובה לשאלה המקורית היא כמובן $M(n,W)$. שימו לב, זמן הריצה וגודל הזיכרון הנדרש על־ידי האלגוריתם הזה הם $\Theta(nW)$. זה אינו חסם פולינומיאלי מפני ש־nW עלול להיות מעריכי במספר הסיביות $O(n \log W)$ הדרושות לייצוג הקלט. אלגוריתמים מסוג זה נקראים "פסאודו־פולינומיאלים".

תרגיל 6.2 הריצו את האלגוריתם תכנון דינמי לבעיית סכומי תת־קבוצות, על הקלט הזה:

$$n = 4, \quad \{w_1 = 2, w_2 = 3, w_3 = 5, w_4 = 6\},$$

ועבור כל $W \le 16$. מלאו את הטבלה M ופתרו את אוסף הבעיות הנ"ל.

פתרון בעמוד 126

להלן בעיה אשר מכלילה את הבעיה של סכומי תת־קבוצות.

בעיה אלגוריתמית: בעיית התרמיל [Knapsack Problem].
הקלט: n פריטים $\{1,\dots,n\}$. לכל פריט i יש משקל $w_i \in \mathbb{N}$ וערך $v_i \in \mathbb{N}$. בנוסף נתון חסם $W \in \mathbb{N}$.
הפלט: תת־קבוצה $S \subseteq \{1,\dots,n\}$ המקיימת $\sum_{i\in S} w_i \le W$.
המטרה: להביא למקסימום את הביטוי $\sum_{i\in S} v_i$.

בעיית סכומי תת־קבוצות היא מקרה פרטי של בעיית התרמיל כאשר $w_i = v_i$.

אפשר לפתור את הבעיה הזאת בעזרת תכנון דינמי הדומה מאוד לאלגוריתם עבור סכומי תת־קבוצות. נגדיר טבלה $M(i,w)$ המחושבת בעזרת הנוסחה הרקורסיבית הזו:

$$M(i,w) = \begin{cases} 0 & i = 0 \\ M(i-1,w) & w_i > w \\ \max\{M(i-1,w), v_i + M(i-1,w-w_i)\} & \text{otherwise.} \end{cases} \tag{6.1}$$

התשובה לבעיית התרמיל היא כמובן $M(n,W)$. הנוסחה לעיל מגדירה אלגוריתם תכנון דינמי שהוא כמעט זהה לאלגוריתם תכנון דינמי לבעיה של סכומי תת־קבוצות.

תרגיל 6.3 נוסחה (6.1) מחשבת רק את הערך האופטימלי. הציעו אלגוריתם יעיל שיחשב גם את תת־הקבוצה שמשיגה את הערך האופטימלי.

פתרון בעמוד 127

תרגיל 6.4 נבחן מקרה פרטי של בעיית התרמיל, בו ידוע שכל הערכים v_i מקיימים $v_i \leq 10n^3$. הציעו אלגוריתם פולינומיאלי לבעיה זאת. הוכיחו את טענותיכם.

פתרון בעמוד 127

תרגיל 6.5 במקום תרמיל אחד היכול לשאת משקל W, נתונים כעת שני תרמילים שיכולים לשאת את המשקלות W_1, ו־W_2 בהתאמה. הציעו אלגוריתמים מבוססי תכנון דינמי לבעיית התרמילים בתנאים האלה:

1. כאשר כל פריט יכול להתחלק בין התרמילים בכל צורה שהיא;
2. כאשר כל פריט חייב להיכנס במלואו לתרמיל אחד.

פתרון בעמוד 129

6.3 תכנון דינמי על־פני מקטעים

קראו בספר את סעיף 6.5

ספר הלימוד בחר להציג את שיטת התכנון הדינמי על פני מקטעים בעזרת בעיה אלגוריתמית שמגיעה מביולוגיה. למרות ההנחות המפשטות הרבות שהספר מניח כדי לקבל בעיה אלגוריתמית נקייה, הבעייה המתקבלת מכילה מספר פרטים ונושאים טכניים שאינם נחוצים כדי להבין את מהות השיטה. במקום זאת אנו נציג עתה בעיה דומה ברוחה, אך עם פחות פרטים טכניים.

בתוכנות מתמטיות רבות, הטקסט מקבל ביטויים מתמטיים המכילים סוגי סוגרים שונים. לדוגמה:

$$(abc\{x\{y[x]z\}\}[math])$$

נתון לנו אם־כן טקסט שמכיל, בין היתר, את סוגי הסוגרים האלה: 1. ( ) ; 2. [ ] ; 3. { } ; כל סימן פותח מתאזן על־ידי סימן סוגר ואפשר ל"סגור" את הביטוי רק כאשר אין בו סוגרים "פתוחים" מסוג כלשהו. לדוגמה: הביטוי '([ )]' אינו מאוזן

ולכן אינו חוקי. כיוון שכל התווים, למעט הסוגרים, אינם משפיעים על האיזון של הסוגרים, נניח שהמחרוזת מכילה רק סוגרים.

נגדיר כעת באופן פורמלי את התאמות הסוגרים. עבור המחרוזת הזו

$$X = x_1 \ldots x_n \in \{`(', `[', `\{', ')', ']', '\}'\}^*$$

התאמה חוקית היא אוסף של זוגות סדורים

$$\mathcal{M} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_t, j_t)\}$$

המקיימים את התנאים האלה:

1. לכל $\ell \in \{1, \ldots, t\}$, $1 \leq i_\ell < j_\ell \leq n$.
2. כל $a \in \{1, \ldots, n\}$ מופיע לכל היותר פעם אחת ב־$\mathcal{M}$.
3. לכל $\ell \in \{1, \ldots, t\}$ הזוג (x_{i_ℓ}, x_{j_ℓ}) הם זוג סוגרים תואמים, כלומר $x_{i_\ell} \in \{`(', `[', `\{'\}$, $x_{j_\ell} \in \{`)', `]', `\}'\}$ ואם $x_{i_\ell} = `('$ אז $x_{j_\ell} = `)'$ וכדומה.
4. ההתאמה היא לאמינרית [laminar], כלומר, לכל $\ell \neq k$ מתקיימת אחת מארבע האפשרויות האלה:
   א. $i_\ell < j_\ell < i_k < j_k$;
   ב. $i_k < j_k < i_\ell < j_\ell$;
   ג. $i_\ell < i_k < j_k < j_\ell$;
   ד. $i_k < i_\ell < j_\ell < j_k$.

לדוגמה, עבור המחרוזת הבאה:

(	(	[	[	{	{	}	[	)	]	]	)
1	2	3	4	5	6	7	8	9	10	11	12

ההתאמה הבאה היא התאמה חוקית:

$$\{(1, 12),\ (3, 11),\ (6, 7),\ (8, 10)\}$$

ואילו התווים במקומות 2, 4, 5, ו־9 נשארו ללא בן זוג.

מחרוזת נקראת מאוזנת אם קיימת התאמה חוקית אשר לא משאירה אף איבר במחרוזת ללא בן זוג.

תרגיל 6.6

1. תארו אלגוריתם בסיבוכיות זמן לינארית, אשר בודק אם מחרוזת נתונה

$$X \in \{\ `(', `)', `[', `]', `\{', `\}'\ \}^*$$

היא מאוזנת.

2. אם המחרוזת אינה מאוזנת, המשתמשים רוצים לגלות היכן "הסוגר החסר". לצורך זה, המערכת צריכה להחזיר התאמה שתכיל מספר מינימלי של תווים שאינם מותאמים. פתחו ונתחו אלגוריתם שיבצע משימה זאת.

פתרון בעמוד 130

6.4 יישור סדרות

קראו בספר את סעיף 6.6

בעיית יישור הסדרות [Sequence Alignment] מוגדרת כדלקמן: בהינתן אלפבית Σ (שאינו כולל את הסימן "-") ושתי סדרות (מחרוזות) $X, Y \in \Sigma^*$, מוסיפים את הסימן "-" לתוך X ו־Y במקומות שונים, כך שמתקבלות המחרוזות $X' \in (\Sigma \cup \{\text{"-"}\})^*$ ו־$Y' \in (\Sigma \cup \{\text{"-"}\})^*$ (בהתאמה) שהן באותו אורך. פעולה זו היא **פעולת יישור**: קיבלנו שתי מחרוזות באותו אורך ו"יישרנו" (כלומר התאמנו) את התו X'_i ל־Y'_i.

ברצוננו להביא למינימום את "עלות היישור" שתוגדר להלן. לצורך כך נגדיר "קנס" $\alpha(a,b)$ עבור התאמה של התו $a \in \Sigma$ לתו $b \in \Sigma$ (בדרך־כלל $\alpha(a,a) = 0$), וקנס פער $\delta > 0$ עבור שימוש בריווח, כלומר בתו "-". אפשר לאחד את הקנסות בצורה הזו: בהינתן עלות אי־התאמה בין סמלים $\alpha : \Sigma \times \Sigma \to \mathbb{R}$, וקנס פער $\delta > 0$. נרחיב את עלות האי־התאמה גם לסימן "-":

$$\overline{\alpha} : (\Sigma \cup \{\text{-}\}) \times (\Sigma \cup \{\text{-}\}) \to \mathbb{R},$$
$$\overline{\alpha}(a,b) = \begin{cases} \alpha(a,b) & a \neq \text{-} \text{ and } b \neq \text{-} \\ \delta & \text{otherwise.} \end{cases}$$

בהינתן יישור $X' = x_1 x_2 \dots x_n$, $Y' = y_1 y_2 \dots y_n$ של שתי סדרות $X, Y \in \Sigma^*$, עלות היישור מוגדרת כך:

$$\sum_{i=1}^{n} \overline{\alpha}(x_i, y_i).$$

בעיה אלגוריתמית: יישור הסדרות.
הנחות: אלפבית ידוע ועלויות קבועות לאי־התאמה ולפער.
הקלט: זוג סדרות $X = x_1 x_2 \dots x_m$ ו־$Y = y_1 y_2 \dots y_n$ ביחס (או מעל) Σ,
הפלט: יישור של הסדרות X ו־Y.
המטרה: מזעור עלות היישור.

קל יחסית לכתוב אלגוריתם תכנון דינמי שיחשב את עלות היישור המינימלית. נגדיר טבלה $A(i,j)$ שתכיל את עלות היישור האופטימלי בין $x_1 x_2 \dots x_i$ לבין $y_1 y_2 \dots y_j$. אפשר למלא את הטבלה A בעזרת הנוסחה הרקורסיבית הזו:

$$A(i,j) = \begin{cases} i\delta & j = 0 \\ j\delta & i = 0 \\ \min \left\{ \begin{array}{c} \alpha(x_i, y_j) + A(i-1, j-1), \\ \delta + A(i, j-1), \\ \delta + A(i-1, j) \end{array} \right\} & i, j > 0. \end{cases} \tag{6.2}$$

נוסחה (6.2) מגדירה באופן מיידי את האלגוריתם תכנון דינמי לחישוב $A(i,j)$ ועלות היישור המינימלית של המחרוזות המקוריות היא $A(m,n)$.

נותר להסביר מדוע נוסחה (6.2) באמת מחשבת את העלות המינימלית של היישור בין $x_1x_2\ldots x_i$ לבין $y_1y_2\ldots y_j$. עובדה זאת מוכחת בספר בטענה (6.16) ומבוססת על האבחנה הבאה [טענה (6.15) בספר]: ביישור של X ו־Y תתקיים אחת משלוש אפשרויות: א. התו האחרון ב־X מותאם לתו האחרון ב־Y; ב. התו האחרון ב־X אינו מותאם לאף תו ב־Y; ג. התו האחרון ב־Y אינו מותאם לאף תו ב־X.

נעיר שהאלגוריתם המתקבל ממלא טבלה בגודל $m \times n$ וזמן הריצה שלו הוא $O(mn)$.

תרגיל 6.7 שנו את האלגוריתם לחישוב הערך המינימלי של יישור סדרות נתונות באורך m ו־n, כך שגודל הזיכרון שיידרש לו יהיה רק $O(m+n)$. רמז: שימו לב שהאלגוריתם אינו זקוק בו־זמנית לכל הטבלה M.

פתרון בעמוד 132

הערה: אם ברצוננו למצוא את היישור האופטימלי (ולא רק את ערכו) תוך שימוש בגודל זיכרון לינארי, אזי האלגוריתם שהוצג בתרגיל 6.7 אינו מספק (ודאו שאתם מבינים מדוע תרגיל 6.7 אינו פותר בעיה זו). למרות זאת אפשר לחשב את היישור האופטימלי גם תוך שימוש בגודל זיכרון לינארי. תוכלו לקרוא על כך בסעיף 6.7 בספר, אך סעיף זה אינו חובה בקורס. ♣

מרחק עריכה

נדון כעת בבעיה הקרובה ברוחה לבעיית יישור סדרות – זוהי בעיית חישוב **מרחק העריכה** [Edit distance] בין מחרוזות. מרחק זה הוא מדד למידת השוני בין שתי מחרוזות. הוא מודד את מספר הפעולות המינימלי מסוג מחיקת תו והוספת תו הדרוש כדי לעבור ממחרוזת אחת לשנייה. מרחק העריכה משמש לעתים קרובות כדי להתגבר על טעויות הקלדה באפליקציות מילוניות.

דוגמה: נראה שמרחק העריכה בין המחרוזת sophisticated למחרוזת phisticats הוא לכל היותר 5; מוחקים את התווים 's' ו־'o' בהתחלה המחרוזת, מוחקים את התווים 'e' ו־'d' בסוף המחרוזת, ומוסיפים את התו 's' בסוף המחרוזת. ♣

באופן פורמלי נגדיר:

הגדרה 6.1 (פעולת מחיקה) המחרוזת $Y \in \Sigma^*$ מתקבלת מהמחרוזת $X = x_1x_2\cdots x_m \in \Sigma^*$ על־ידי פעולת מחיקה, אם קיים $i \in \{1,\ldots,n\}$ שעבורו $Y = x_1x_2\ldots x_{i-1}x_{i+1}\ldots x_n$. ♢

הגדרה 6.2 (פעולת הוספה) המחרוזת $Y \in \Sigma^*$ מתקבלת מהמחרוזת $X = x_1x_2\cdots x_m \in \Sigma^*$ על־ידי פעולת הוספה, אם קיימים אינדקס $i \in \{0,1,\ldots,n\}$ ואות $z \in \Sigma$ עבורם מתקיים $Y = x_1x_2\ldots x_izx_{i+1}\ldots x_n$. ♢

הגדרה 6.3 (מרחק עריכה) מרחק העריכה בין המחרוזת $X \in \Sigma^*$ למחרוזת $Y \in \Sigma^*$, הוא המספר המינימלי של פעולות מחיקה והוספה ההופכות את X ל־Y. מרחק זה מסומן ב־$d_{\text{ED}}(X,Y)$. ♢

תרגיל 6.8 הוכיחו שמרחק העריכה הוא סימטרי, כלומר מתקיים $d_{\mathrm{ED}}(X,Y) = d_{\mathrm{ED}}(Y,X)$, לכל $X,Y \in \Sigma^*$.

פתרון בעמוד 132

תרגיל 6.9 פתחו אלגוריתם לחישוב מרחק עריכה בין שתי מחרוזות נתונות. הוכיחו את נכונות האלגוריתם ונתחו את סיבוכיות זמן הריצה שלו.

פתרון בעמוד 133

6.5 מסלולים קצרים ביותר בגרף בעל משקלים חיוביים ושליליים

קראו בספר את סעיף 6.8

בפרק 4 למדנו שאלגוריתם דייקסטרה משמש לחישוב מרחקים מצומת נתון בגרף עם משקולות אי־שליליות על הקשתות. בסעיף הזה נבחן בעיה כללית יותר, כאשר מותרות קשתות עם משקלות שליליים. התבוננו באיור 6.21 בספר (שם יש לגרף קשתות שליליות), ותיווכחו כי במקרה כזה האלגוריתם של דייקסטרה לא תמיד עובד; יתרה מזאת, אם יש בגרף מעגל שלילי, אזי לא קיים מרחק קצר ביותר (סופי) בין זוג צמתים על מעגל זה.

כעת נניח כי אין בגרף מעגלים שליליים. תחת הנחה זאת קל לראות שקיימים מסלולים קצרים ביותר שהם פשוטים, ולכן מכילים $|V|-1$ קשתות לכל היותר. עובדות אלה מאפשרות לנו להגדיר אלגוריתם לתכנון דינמי שיחשב את המרחקים הקצרים ביותר מכל הצמתים לצומת נתון t. נגדיר את $\mathrm{OPT}(i,v)$ כאורך המסלול הקצר ביותר מ־v ל־t , המכיל לכל היותר i קשתות. קל להוכיח באינדוקציה על i ש־$\mathrm{OPT}(i,v)$ מקיים את נוסחת הנסיגה:

$$\begin{aligned}
&\mathrm{OPT}(0,t) = 0\\
&\mathrm{OPT}(0,v) = \infty \quad \text{when } v \neq t\\
&\mathrm{OPT}(i,v) = \min\bigl\{\mathrm{OPT}(i-1,v), \min_{(v,u)\in E} \mathrm{OPT}(i-1,u) + c(v,u)\bigr\}\\
&\qquad\qquad \text{when } i > 0
\end{aligned} \tag{6.3}$$

המרחק הקצר ביותר בין s ל־t הוא כמובן $\mathrm{OPT}(n-1,s)$. אנו נשתמש כאן (וגם בהמשך) בסימן האינסוף ∞ כדי לציין שאין מסלול המקיים את התנאים. הפעולות שמתבצעות, באלגוריתמים שנלמד, על מרחקים (שליליים או אי־שליליים) ועל אינסוף, הם: חיבור, מציאת מינימום והשוואה. באופן טבעי, $a+\infty=\infty$, ו־$\min\{a,\infty\}=a$.

נוסחה (6.3) מובילה באופן ישיר לאלגוריתם לתכנון דינמי. נגדיר טבלה $M(i,v)$ ונמלא אותה על־פי (6.3). האלגוריתם המתקבל נקרא אלגוריתם בלמן־פורד, והוא רשום באופן מפורט כאלגוריתם 6.1.

דוגמה: נריץ את אלגוריתם 6.1 (בלמן־פורד) אל הצומת t על הגרף מאיור 6.2. בצד ימין של איור 6.2 מתוארים ערכי הטבלה $M(i,v)$ כפי שממלא אותה

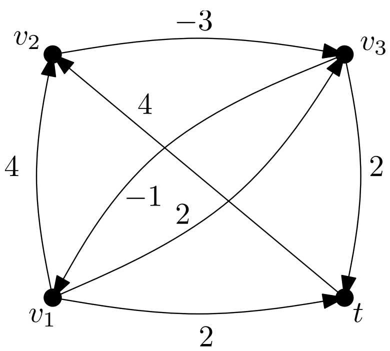
Algorithm 6.1 Bellman-Ford algorithm

Require: Weighted directed graph $G=(V,E,c)$ with n vertices, so that G has no negative cycles.
Require: $t \in V$.
  Initialize $M(0,t) \leftarrow 0$.
  Initialize $M(0,v) \leftarrow \infty$, for every $v \neq t$.
  for $i \leftarrow 1,\dots,n-1$ **do**
    for $v \in V$ **do**
      $M(i,v) \leftarrow \min\big\{M(i-1,v), \min_{u\in V:(v,u)\in E} M(i-1,u)+c(v,u)\big\}$
  return $M(n-1,u) \quad \forall u \in V$

אלגוריתם בלמן־פורד. שימו לב, $M(1,v_2)=\infty$ כיוון שאין קשת מ־v_2 ל־t, ואילו $M(2,v_2)=-1$ מפני שבסבב זה בוצע השיפור $M(1,v_3)+c(v_2,v_3)=2-3$. אבל באותו סבב $M(2,v_3)$ שופר ל־1 ולכן המצב בסבב האחרון הוא:

$$M(3,v_2)=M(2,v_2)+c(v_2,v_3)=1-3=-2.$$

♣



M	v_1	v_2	v_3	t
$i=0$	∞	∞	∞	0
$i=1$	2	∞	2	0
$i=2$	2	-1	1	0
$i=3$	2	-2	1	0

איור 6.2: בצד ימין מוצגת הטבלה $M(i,v)$ כפי שמולאה על־ידי אלגוריתם 6.1 שהורץ על הגרף שבצד שמאל.

ניתוח זמן הריצה של אלגוריתם בלמן־פורד. האלגוריתם מבצע לולאה על המשתנים i, ו־v; חישוב פעולת המינימום מתבצע d_v פעמים לכל v (כאשר מייצגים גרפים ברשימת שכנויות). זמן הריצה שיתקבל בסה"כ יהיה:

$$O\left(\sum_{i=1}^{n}\sum_{v\in V} d_v\right)=O(nm),$$

כאשר n הוא מספר הצמתים בגרף ו־m הוא מספר הקשתות בגרף.

תרגיל 6.10 בהינתן גרף ממושקל $G=(V,E,c)$ ללא מעגלים שליליים וצומת $s\in V$, כתבו אלגוריתם שימצא את המרחקים מצומת $s\in V$ לכל שאר הצמתים בגרף.

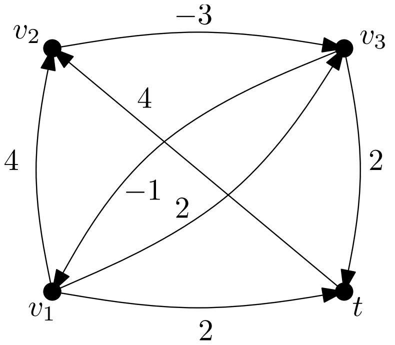
פתרון בעמוד 134

אלגוריתם חוסך־מקום. כמו במקרים קודמים, כדי לחשב את המסלולים הקצרים ביותר, אין צורך לשמור באופן מפורש את כל טבלה M. נבחין כי בשלב i האלגוריתם פונה רק לכניסות $(i-1,k)$ ו־(i,k) בטבלה M עבור $k \in \{1,\ldots,n\}$, לכן מספיק לשמור, בכל רגע נתון, רק את שתי השורות האחרונות ב־M. לאמיתו של דבר אפשר לפשט אף יותר – התבוננו באלגוריתם 6.2; גם הוא מחשב מרחקים קצרים ביותר.

Algorithm 6.2 Space efficient Bellman-Ford

Require: Weighted directed graph $G = (V, W, c)$ with n vertices, so that G has no negative cycles.
Require: $t \in V$.
Initialize $M(t) \leftarrow 0$.
Initialize $M(v) \leftarrow \infty$, for every $v \neq t$.
for $i \leftarrow 1, \ldots, n-1$ **do**
  for $v \in V$ **do**
    for $u \in V$, $(v,u) \in E$ **do**
      $M(v) \leftarrow \min\{M(v), M(u) + c(v,u)\}$
return $M(s)$ for every $s \in V$

דוגמה: באיור 6.3 אנו מריצים על הגרף (בצד שמאל) את אלגוריתם 6.2. באיור 6.2 הרצנו על אותו גרף את אלגוריתם 6.1. בריצת אלגוריתם 6.1 התקבל הערך $M(2,v_2) = -1$, בעוד שבריצת אלגוריתם 6.2, כאשר $i = 2$, התקבלה הערך $M(v_2) = -2$. הסיבה לכך היא שבזמן ביצוע הלולאה $i = 2$ באלגוריתם 6.2, $M(v_3)$ ירד ל־1 ומיד לאחר־מכן, בעדכון של v_2 , ירד ערכו של $M(v_2)$ ל־-2. אלגוריתם 6.1, לעומת זאת, מעדכן את $M(2,v_2)$ בעזרת $M(1,v_3)$ שנשאר 2. שימו לב: ערכי הטבלה M בזמן הריצה של אלגוריתם 6.2, רגישים לסדר שבו אנו עוברים על הצמתים (בניגוד לריצת אלגוריתם 6.1), ולכן בדוגמה המובאת באיור 6.3 קבענו שמקומו של v_3 לפני v_2 בסדר עדכון הצמתים. ♣



M	t	v_3	v_2	v_1
$i=0$	0	∞	∞	∞
$i=1$	0	2	-1	2
$i=2$	0	1	-2	2
$i=3$	0	1	-2	2

איור 6.3: בצד ימין מוצגת הטבלה $M(v)$ כפי שמולאה על־ידי אלגוריתם 6.2 שהורץ על הגרף שבצד שמאל. אנו מניחים שהצמתים הנסרקים על־ידי האלגוריתם נרשמים בטבלה משמאל לימין.

תרגיל 6.11 הוכיחו כי אלגוריתם 6.2 מחשב את המרחקים מכל הצמתים אל הצומת t.

פתרון בעמוד 134

תרגיל 6.12 הוכיחו שאם קיים $\ell \leq n-1$ כך ש־ $M_\ell(v) = M_{\ell-1}(v)$ לכל $v \in V$, אזי לכל $v \in V$, המרחק הקצר ביותר מ־v ל־t הוא $M_\ell(v)$.

פתרון בעמוד 135

תרגיל 6.12 מציע שיפור פרקטי לאלגוריתם 6.2: אין צורך להריץ את הלולאה על i עד $n-1$, מספיק לבדוק שהמערך M לא השתנה כלל במשך ביצוע איטרציה אחת של הלולאה על i.

חישוב מסלולים קצרים ביותר. נדון עתה בשאלה כיצד נוכל לשמור גם את המסלולים הקצרים ביותר ל־t באלגוריתם 6.2 (בלמן־פורד החוסך־מקום). בפרק 4 ראינו כבר כיצד מייצגים מסלולים קצרים ביותר לצומת נתון t: בעזרת תת־גרף מכוון שגרף התשתית שלו (כאשר מתעלמים מכיווני הקשתות) הינו עץ מושרש ששורשו הוא t. כמו־כן כיווני הקשתות (בגרף המקורי) הם מבן לאב. כדי לתחזק עץ כזה מספיק שכל צומת $u \in V \setminus \{t\}$ יצביע על $S(u)$ – הצומת העוקב במסלול קצר ביותר מ־u ל־t (שהוא אביו בעץ שהוגדר לעיל). עובדות אלו נכונה גם בגרף עם משקולות שלמים (חיובים או שליליים) – כל עוד אין מעגלים שליליים. הסיבה (שכבר ראינו) היא שאם

$$u = u_0, u_1, \ldots, u_k = t$$

הוא מסלול קצר ביותר מ־u ל־t, אזי בהכרח

$$u_1, \ldots, u_k = t$$

הוא מסלול קצר ביותר מ־u_1 ל־t. אחרת, לו היה מסלול קצר יותר מ־u_1 ל־t, היינו יכולים להשתמש בו כדי לקבל מסלול קצר יותר מ־u ל־t. משמעות הדבר שהסדרה

$$u,\ u_1 = S(u),\ u_2 = S(u_1), \ldots$$

תיעצר ב־t והמסלול שיתקבל יהיה המסלול הקצר ביותר מ־u ל־t.

כעת נוכל להתאים את אלגוריתם 6.2 (בלמן־פורד החוסך־מקום) כך שיחשב גם את פונקציית העוקב $S(\cdot)$ עבור כל צומת שאיננו t. התוצאה היא אלגוריתם 6.3.

לצורך הוכחת הנכונות של אלגוריתם 6.3 מספיק לטעון את הטענה הבאה:

טענה 6.1 בכל שלב במהלך אלגוריתם 6.3 ולכל $v \in V$, אם $M(v) < \infty$ אזי קיים מסלול מ־v ל־t שאורכו $M(v)$ והצומת השני בו הוא $S(v)$.

הוכחה. החלק הראשון של הטענה (קיום מסלול שאורכו $M(v)$) הוכח למעשה בפתרון של תרגיל 6.11. נחזור כאן על הוכחת הטענה, וההוכחה היא באינדוקציה על מספר הפעמים שהפקודה **if** מתבצעת. לאחר האתחול של M ו־S הטענה האינדוקטיבית נכונה בבירור. כאשר הפעולה **if** מתבצעת, משמעות הדבר היא שקיים צומת u והעדכונים האלה מתבצעים:

$$S(v) \leftarrow u, \qquad M(v) \leftarrow M(u) + c(v,u)$$

ונשמר התנאי האינדוקטיבי על S. $\square$

Algorithm 6.3 Bellman-Ford with shortest paths

Require: Weighted directed graph $G = (V, W, c)$ with n vertices, so that G has no negative cycles.
Require: $t \in V$.
  Initialize $M(t) \leftarrow 0$.
  Initialize $M(v) \leftarrow \infty$, for every $v \neq t$.
  Initialize $S(v) \leftarrow$nil, for every $v \in V$
  for $i \leftarrow 1, \ldots, n-1$ **do**
    for $v \in V$ **do**
      for $u \in V$, $(v, u) \in E$ **do**
        if $M(v) > M(u) + c(v, u)$ **then**
          $M(v) \leftarrow M(u) + c(v, u)$
          $S(v) \leftarrow u$
  return M and S

.

בתרגיל 6.11 כבר טענו כי בסיום ריצת האלגוריתם, $M(v)$ הוא המרחק הקצר ביותר, לכן נוכל להסיק מן הטענה שלעיל כי בסיום ריצת האלגוריתם, $S(v)$ מכיל את הצומת העוקב ל־v במסלול הקצר ביותר מ־v ל־t.

6.6 מסלולים קצרים בין כל זוגות הצמתים

נדון עתה בבעיית מציאת המרחקים הקצרים ביותר בין כל הזוגות בגרף מכוון וממושקל אבל ללא מעגלים שליליים. פורמלית:

בעיה אלגוריתמית: מציאת מסלולים קצרים ביותר בין כל זוגות הצמתים [All-pairs shortest paths].

הקלט: גרף מכוון וממושקל (כולל משקלים שליליים) $G = (V = \{v_1, \ldots, v_n\}, E, c)$ ללא מעגלים שליליים.

הפלט: מטריצת המרחקים M, אשר בה הכניסה M_{ij} מכילה את המרחק מ־v_i ל־v_j.

ניתן כמובן להריץ את אלגוריתם בלמן־פורד אל כל צומת לגרף. כיוון שכל הרצה של בלמן פורד חסומה בזמן $O(nm)$ וכיוון שיש n צמתים בגרף, סיבוכיות זמן הריצה של אלגוריתם זה היא $O(n^2m)$. בסעיף זה נתוודע לאלגוריתמים יעילים יותר אסימפטוטית כפונקציה של m ו־n.

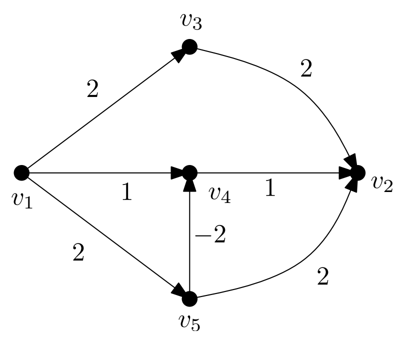
אלגוריתם פלויד־וורשאל [Floyd–Warshall] למציאת המרחקים בין כל זוגות הצמתים, מבוסס על תכנון דינמי שונה מזה שראינו באלגוריתם בלמן־פורד. לשם הנוחות נניח שצומתי הגרף ממוספרים, כלומר $V = \{v_1, \ldots, v_n\}$. עבור מסלול פשוט (כלומר, שאף צומת בו לא מופיעה יותר מפעם אחת) $(v_{i_1}, v_{i_2}, \ldots, v_{i_{\ell-1}}, v_{i_\ell})$ אנו נקרא לצמתים $v_{i_2}, \ldots, v_{i_{\ell-1}}$ צמתים פנימיים; ולצמתים v_{i_1}, v_{i_ℓ}, קצוות המסלול.

עתה נגדיר $\mathrm{OPT}(i, j, k)$ כאורך המסלול הקצר ביותר מבין כל המסלולים מ־v_i ל־v_j שכל צמתיהם הפנימיים שייכים לתת־הקבוצה $\{v_1, \ldots, v_k\}$. לדוגמא,

באיור 6.4,

$$\mathrm{OPT}(1,2,0)=\mathrm{OPT}(1,2,1)=\mathrm{OPT}(1,2,2)=\infty,$$

כיוון שאין מסלול מ־v_1 ל־v_2 שאיננו מכיל בפנימו אף צומת למעט v_1 או v_2 (כלומר אין קשת מ־v_1 ל־v_2). לעומת זאת, $\mathrm{OPT}(1,2,3)=4$ כיוון שיש מסלול יחיד – (v_1,v_3,v_2) – מ־v_1 ל־v_2 העובר רק דרך $\{v_1,v_2,v_3\}$. כמו־כן $\mathrm{OPT}(1,2,4)=2$ בעזרת המסלול (v_1,v_4,v_2) שמשקלו הוא 2. לבסוף $\mathrm{OPT}(1,2,5)=1$ בזכות המסלול (v_1,v_5,v_4,v_2) שמשקלו 1.



איור 6.4:

היתרון של הביטוי $\mathrm{OPT}(i,j,k)$ הוא שניתן לחשבו בקלות יחסית בעזרת $\mathrm{OPT}(i',j',k-1)$ בעזרת ההבחנה הבאה: במסלול הקצר ביותר מ־v_i ל־v_j שכל צמתיו הפנימיים הם מ־$v_1,\ldots,v_k$ מתקיים אחד מן השתיים: או ש־v_k איננו צומת פנימי, או שהוא כן צומת פנימי במסלול זה. במקרה הראשון, המסלול הינו גם מסלול קצר ביותר מ־v_i ל־v_j שכל צמתיו הפנימיים הם מ־$v_1,\ldots,v_{k-1}$ ומהנחת האינדוקציה זה $\mathrm{OPT}(i,j,k-1)$. במקרה השני, כיוון שהמסלול פשוט, v_k מופיע בו רק פעם אחת, ולכן ניתן לפרקו לשני תתי־מסלולים – מ־v_1 ל־v_k, ומ־v_k ל־v_j. בתתי מסלולי אלו v_k איננו מופיע כצומת פנימית ולכן סכום אורכיהם הוא $\mathrm{OPT}(i,k,k-1)+\mathrm{OPT}(k,j,k-1)$. פורמלית, נוכיח:

טענה 6.2

$$\begin{aligned}
&\mathrm{OPT}(i,i,0)=0 \\
&\mathrm{OPT}(i,j,0)=c(v_i,v_j) && (v_i,v_j)\in E \\
&\mathrm{OPT}(i,j,0)=\infty && (v_i,v_j)\notin E \\
&\mathrm{OPT}(i,j,k)=\min\big\{\mathrm{OPT}(i,j,k-1), && k>0 \\
&\qquad\qquad \mathrm{OPT}(i,k,k-1)+\mathrm{OPT}(k,j,k-1)\big\}
\end{aligned} \tag{6.4}$$

הוכחה. ההוכחה תהיה באופן טבעי באינדוקציה על k. עבור $k=0$, מסלול בין v_i ל־v_j שאינו עובר באף צומת אחר, יכול להיות רק קשת, לכן, אם אין קשת בין v_i ל־v_j לא קיים מסלול כזה (עלות אינסופית), ואם יש קשת כזאת – היא המסלול ועלותה היא עלות המסלול.

עבור $k > 0$, יהי P מסלול פשוט, שהוא המסלול הקצר ביותר בין v_i ל־v_j שכל צמתיו הפנימיים שייכים לקבוצה $\{v_1, \ldots, v_k\}$. ייתכנו שני מקרים:

- המסלול P אינו מכיל את הצומת v_k. במקרה זה כל צמתיו הפנימיים שייכים לקבוצה $\{v_1, \ldots, v_{k-1}\}$ ולפי הנחת האינדוקציה, אורכו $\mathrm{OPT}(i, j, k-1)$.
- המסלול P מכיל את הצומת v_k. כיוון ש־P הוא מסלול פשוט, הוא עובר דרך v_k רק פעם אחת, ואפשר לפרק אותו לשני מסלולים: מסלול P_1 מ־v_i ל־v_k, ומסלול P_2 מ־v_k ל־v_j. הצמתים הפנימיים של P_1 מהווים קבוצה חלקית של $\{v_1, \ldots, v_{k-1}\}$. לכן P_1 הוא המסלול הקצר ביותר מבין המסלולים המובילים מ־v_i ל־v_k שצמתיהם הפנימיים מוכלים ב־$\{v_1, \ldots, v_{k-1}\}$. אילו היה קיים מסלול P_1' קצר יותר, הייתה אפשרות להחליף את P_1 ב־P_1' במסלול P, ולקבל מסלול P' מ־v_i ל־v_j, קצר יותר מ־P העובר רק דרך הצמתים $\{v_1, \ldots, v_k\}$, בניגוד לאופטימליות של P. מהנחת האינדוקציה אנו יודעים כי אורכו של P_1 הוא $\mathrm{OPT}(i, k, k-1)$. באופן דומה, אורכו של P_2 הוא $\mathrm{OPT}(k, j, k-1)$. לכן אורכו של P, שהוא סכום אורכי P_1 ו P_2,יהיה:

$$\mathrm{OPT}(i, k, k-1) + \mathrm{OPT}(k, j, k-1).$$

□

נוסחה (6.4) מגדירה ישירות את האלגוריתם לתכנון דינמי, שאנו מפרטים באלגוריתם 6.4.

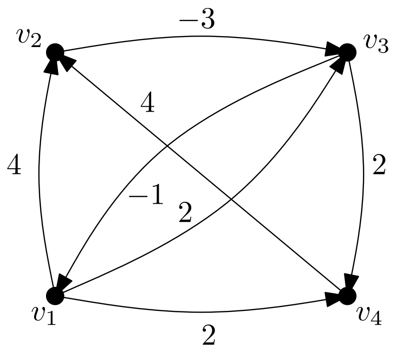
Algorithm 6.4 Floyd–Warshall algorithm

Require: Weighted graph $G = (\{1, \ldots, n\}, E, c)$ without negative cycles.
  Initialize $M(i, j, 0) \leftarrow c(v_i, v_j)$ if $(v_i, v_j) \in E$
  Initialize $M(i, j, 0) \leftarrow \infty$ if $(v_i, v_j) \notin E$
  Initialize $M(i, i, 0) \leftarrow 0$
  for $k \leftarrow 1, \ldots, n$ **do**
    for $i \leftarrow 1, \ldots, n$ **do**
      for $j \leftarrow 1, \ldots, n$ **do**
        $M(i, j, k) \leftarrow \min\{M(i, j, k-1), M(i, k, k-1) + M(k, j, k-1)\}$.
  return $M(i, j, n)$ for every $i, j \in \{1, \ldots, n\}$.

דוגמה: באיור 6.5 מודגמת הריצה של אלגוריתם פלויד-וורשאל על הגרף שבאיור. הטבלה $M(i, j, 0)$ היא למעשה מטריצת השכנויות של הגרף (עם '0' על האלכסון ו־'∞' כאשר אין קשת). ♣

טענה 6.3 אלגוריתם פלויד-וורשאל מחשב מרחקים בין כל זוגות הצמתים בגרף ממושקל בעל n צמתים, ללא מעגלים שליליים. זמן הריצה שלו הוא $O(n^3)$.

הוכחה. נכונות האלגוריתם נובעת ישירות מנוסחה (6.4). זמן הריצה נשלט על־ידי ביצוע שלוש לולאות מקוננות, כל אחת באורך n, ולכן הוא נמשך $O(n^3)$. □



$M(i,j,0)$

	v_1	v_2	v_3	v_4
v_1	0	4	2	2
v_2	∞	0	-3	∞
v_3	-1	∞	0	2
v_4	∞	4	∞	0

$M(i,j,1)$

	v_1	v_2	v_3	v_4
v_1	0	4	2	2
v_2	∞	0	-3	∞
v_3	-1	3	0	1
v_4	∞	4	∞	0

$M(i,j,2)$

	v_1	v_2	v_3	v_4
v_1	0	4	1	2
v_2	∞	0	-3	∞
v_3	-1	3	0	1
v_4	∞	4	1	0

$M(i,j,3)$

	v_1	v_2	v_3	v_4
v_1	0	4	1	2
v_2	-4	0	-3	-2
v_3	-1	3	0	1
v_4	0	4	1	0

$M(i,j,4)$

	v_1	v_2	v_3	v_4
v_1	0	4	1	2
v_2	-4	0	-3	-2
v_3	-1	3	0	1
v_4	0	4	1	0

איור 6.5: הטבלה $M(i,j,k)$ כפי שמולאה על־ידי אלגוריתם 6.4 שהורץ על הגרף המוצג מעליה.

תרגיל 6.13 גודל הזיכרון הדרוש לאלגוריתם 6.4 הוא $\Theta(n^3)$. כתבו אלגוריתם דומה שיספיק לו גודל זיכרון $O(n^2)$.

פתרון בעמוד 135

תרגיל 6.14 ברצוננו למצוא כעת את המסלולים הקצרים ביותר בין כל זוג צמתים.

1. הגדירו ייצוג של מסלולים קצרים ביותר שדרוש לו גודל זיכרון $O(n^2)$ ובהינתן זוג צמתים, הוא מאפשר להחזיר מסלול קצר ביותר ביו הצמתים בזמן $O(\ell)$ כאשר ℓ הוא מספר הצמתים במסלול המדווח.
2. שנו את האלגוריתם שכתבתם בתרגיל 6.13, כך שבהינתן הנתונים הרשומים לעיל (בסעיף 1), הוא יחזיר את ייצוג המסלולים הקצרים ביותר.

פתרון בעמוד 135

תרגיל 6.15 (האלגוריתם של ג'ונסון (תרגיל רשות)) בתרגיל זה נפתח אלגוריתם אחר לחישוב המרחקים בין כל הצמתים בגרף ממושקל ללא מעגלים שליליים, שיהיה יותר יעיל אסימפטוטית מפלויד־וורשאל בגרפים דלילים. יהי $G=(V,E,c)$ גרף מכוון, ממושקל, עם משקולות שלמים וללא מעגלים שליליים.

אלגוריתם	פרק	מרחקים מ...	ממושקלים?	זמן	מקום
סריקה־לרוחב	3	מצומת נתון	לא	$m+n$	$m+n$
דייקסטרה	4	"	טבעיים	$(m+n)\log n$	$m+n$
בלמן-פורד	6	"	שלמים	mn	$m+n$
פלויד-וורשאל	6	בין כל הזוגות	"	n^3	n^2

טבלה 6.1: אלגורתמים למציאת מרחקים קצרים ביותר בגרפים שנלמדו בקורס. סיבוכיות הזמן מבוטאת עד כדי סדר גודל. לאלגוריתם דייקסטרה יש מימוש יעיל יותר עם סיבוכיות זמן של $O(m + n\log n)$.

1. נניח שנתונה על הצמתים הפונקציה $h : V \rightarrow \mathbb{R}$ (הנקראת "פונקציית גובה"). נגדיר משקולות חדשות $\bar{c} : E \rightarrow \mathbb{R}$ על הקשתות באופן הזה: $\bar{c}(u,v) = c(u,v) + h(u) - h(v)$. בהינתן המרחקים הקצרים ביותר ב־$\overline{G} = (V, E, \bar{c})$ ופונקציית הגובה h, הראו כיצד אפשר לחשב ביעילות את המרחקים הקצרים ביותר ב־G.
2. נוסיף כעת ל־G צומת חדש q ונחבר אותו לכל הצמתים המקוריים של G בקשתות יוצאות שמשקלן 0. עבור $v \in V$ נגדיר את $h(v)$ כמרחק הקצר ביותר מ־q ל־v בגרף החדש שנוצר. הוכיחו שהמשקולות החדשים $\bar{c}$ הם אי־שליליים כאשר משתמשים בפונקציית הגובה h.
3. השתמשו בסעיפים הקודמים של התרגיל הזה כדי לפתח אלגוריתם לחישוב המרחקים בין כל זוגות הצמתים בגרף G, ללא מעגלים שליליים, עם סיבוכיות זמן $O(mn\log n)$.
רמז: השתמשו באלגוריתמים של בלמן־פורד ושל דייקסטרה.

פתרון בעמוד 136

6.7 סיכום

תכנון דינמי היא שיטה אלגוריתמית שימושית. בספר ובמדריך הוצגו דוגמאות רבות מתחומים מגוונים: הצברה, דמיון מחרוזות, מרחקים בגרפים עם משקלים שליליים ועוד. סביר שתתקלו בדוגמאות נוספות בהמשך לימודיכם.

כאן המקום לסכם את האלגוריתמים השונים לחישוב מרחקים בגרפים שנלמדו בקורס זה. טבלה 6.1 מסכמת את תכונותיהם העיקריות.

6.8 פתרונות לתרגילים

פתרון תרגיל 6.1 מעמוד 110

נכתוב אלגוריתם תכנון דינמי בשיטה האיטרטיבית. נניח שהמקטעים $I_1 = [s_1, f_1), \ldots, [s_n, f_n)$ מסודרים בסדר לא יורד של של זמני סיום – $f_i \le f_{i+1}$. האלגוריתם יחשב מערך $M(i)$ אשר מכיל את הערך המקסימלי שניתן להשיג מתת־קבוצה של המקטעים $S \subseteq \{I_1, \ldots, I_i\}$ כך ש $I_i \in S$.

לצורך חישוב $M(\cdot)$ אנו גם נזדקק לפונקציה

$$p(i) = \max\{k :\ 0 \le k < j \text{ and } f_k \le s_i\},$$

אשר מתאימה לכל מקטע את המקטע שמסתיים המאוחר ביותר מבין המקטעים שמסתיימים לפני תחילת I_i.

עתה ניתן לחשב אינדוקטיבית את $M(i)$ כדלקמן: מחשבים את המקסימום על פני כל $M(j)$ על־פני כל האינדקסים j כך ש־$j < i$ ואינם חותכים את I_i, כלומר $j \le p(i)$. לזה מוסיפים את הערך של המקטע I_i.

Algorithm 6.5 Inefficient-DP$(I_1 = [s_1, f_1), \ldots, I_n = (s_n, f_n))$

Require: $f_i \le f_{i+1}, \forall i \in \{1, \ldots, n-1\}$
Require: $p(i) = \max\{k :\ 0 \le k < j \text{ and } f_k \le s_i\}, \forall i \in \{1, \ldots, n\}$
  $M(0) \leftarrow 0$
  for $i = 1, \ldots, n$ **do**
    $M(i) \leftarrow \max\{M(j) :\ j \le p(i)\} + v_i$
  return $\max_i M(i)$

נשים לב שסיבוכיות הזמן של אלגוריתם 6.5 היא $O(n^2)$, בניגוד לסיבוכיות הלינארית של אלגוריתם התכנון הדינמי המתואר בספר. קל להוכיח את נכונות האלגוריתם באינדוקציה על j, ולכן לא נתאר אותה כאן.

נסיים בתאור האלגוריתם לחישוב הפונקציה p, ללא הוכחת נכונות. נניח שוב כי $I_1, I_2, \ldots, I_n$ מסודרים לפי זמן סיום עולה. נניח גם כי נתונה הפרמוטציה $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ כך שהמקטעים $I_{\pi(1)}, I_{\pi(2)}, \ldots, I_{\pi(n)}$ מסודרים לפי זמן התחלה עולה. שימו לב שאת שני הסדרים הללו ניתן לחשב בקלות בעזרת מיון בזמן $O(n \log n)$. עתה האלגוריתם הוא כדלקמן.

Algorithm 6.6 Compute-$p(I_1 = [s_1, f_1), \ldots, I_n = (s_n, f_n))$

Require: $f_i \leq f_{i+1}, \forall i \in \{1, \ldots, n-1\}$
Require: $s_{\pi(i)} \leq s_{\pi(i+1)}, \forall i \in \{1, \ldots, n-1\}$

```
p(π(1)) ← 0
i ← 2
j ← 1
while i ≤ n do
    if s_π(i) < f_j then
        i ← i + 1
    else
        p(π(i)) ← j
        j ← j + 1
return p
```

♣

פתרון תרגיל 6.2 מעמוד 111

שימו לב, אלגוריתם התכנון הדינמי עבור הערך $W = W_0$ פותר למעשה את בעיית סכומי תת־הקבוצות לכל $W \leq W_0$. מילוי הטבלה M מתבצע כדלקמן:

- $M(0, w) = 0$
- $M(1, 0) = M(1, 1) = 0$, ועבור $w \geq 2$, מתקיים

$$M(1, w) = \max\{0, 2 + 0\} = 2$$

- $M(2, w) = M(1, w)$, עבור $w_2 = 3 > w$.

$$M(2, 3) = \max\{M(1, 3), 3 + M(1, 0)\} = 3$$

וכנ"ל עבור $M(2, 4)$. עבור $w > 4$,

$$M(2, w) = 3 + M(1, w - 3) = 3 + 2 = 5.$$

- עבור $w < 5$, $M(3, w) = M(2, w)$. $M(3, 5) = \max\{5, 0 + 5\} = 5$.
עבור $w > 5$,

$$M(3, w) = \max\{5, 5 + M(2, w - 5)\} = 5 + M(2, w - 5).$$

לסיכום, הנה הטבלה M

$i \setminus w$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	0	0	2	3	3	5	5	5	5	5	5	5	5	5	5	5	5
3	0	0	2	3	3	5	5	7	8	8	10	10	10	10	10	10	10
4	0	0	2	3	3	5	6	7	8	9	10	11	11	13	14	14	16

הפתרונות עבור $W \le 16$ מוצגים בשורה האחרונה בטבלה. ♣

פתרון תרגיל 6.3 מעמוד 112
החישוב ייעשה באופן זהה לחישוב הסכומים של תת־הקבוצות, ונחזור עליו כאן: דרך ראשונה (פחות יעילה) היא להחזיק, בנוסף לטבלת הערכים $M(i, w)$, גם את טבלת תת־הקבוצות $S(i, w)$ שתכיל את תת־הקבוצה המשיגה את $M(i, w)$. האלגוריתם המתקבל הוא:

Algorithm 6.7 Compute the knapsack ver. I

Require: weights $(w_i)_{i\in\{1,\ldots,n\}}$, and values $(v_i)_{i\in\{1,\ldots,n\}}$.

```
Initialize M(0, w) ← 0; S(0, w) ← ∅
for i ← 1, ..., n do
  for w ← 0, ..., W do
    if w_i > w then
      M(i, w) ← M(i − 1, w)
      S(i, w) ← S(i − 1, w)
    else
      if M(i − 1, w) ≤ v_i + M(i − 1, w − w_i) then
        M(i, w) ← M(i − 1, w)
        S(i, w) ← S(i − 1, w).
      else
        M(i, w) ← v_i + M(i − 1, w − w_i)
        S(i, w) ← S(i − 1, w − w_i) ∪ {i}
return S(n, W)
```

חסרונו של הפתרון שקיבלנו לעיל הוא הגדלת הזיכרון וזמן הריצה שגדלו ל־$O(n^2W)$ (שימו לב שפעולת העתקה של קבוצות עלולה לקחת $O(n)$). אפשר לשפר זאת על־ידי שמירה על תת־הקבוצות S בצורה עקיפה, כדלקמן: לכל כניסה (i, w) בטבלה M, נחזיק גם את הכניסה $(i-1, w')$ שנעזרנו בה לעדכון הכניסה (i, w). כלומר $w' = w$ כאשר i אינו חלק מהפתרון, ו־$w' = w - w_i$ כאשר i הוא חלק מהפתרון. נוכל לשפר במקצת את הפתרון הזה אם נשים לב שבזוג $(i-1, w')$ הערך של $i-1$ ידוע מההקשר, ולכן מספיק לזכור את w'. נבנה טבלה $P(i, w)$ שתשמור את הערכים האלה. התוצאה היא אלגוריתם 6.8, שסיבוכיות גודל הזיכרון וזמן הריצה שלו הם $O(nW)$.

♣

פתרון תרגיל 6.4 מעמוד 112
לפני שניגש לפתרון התרגיל הנתון, נניח לרגע שהחסם $w_i \le 10n^3$ נתון על w_i במקום על v_i. במקרה זה אנו טוענים שהאלגוריתם הממשש את (6.1) הוא פולינומיאלי. ואמנם, עבור $W \ge 10n^4$ הבעיה היא קלה מאד, כי אז $W \ge \sum_i w_i$ וניתן לקחת את כל הפריטים. לכן אנו יכולים להניח ש־$W \le 10n^4$, ואז זמן הריצה הוא $O(nW) = O(10n^5)$. הנקודה החשובה פה היא שעל הערכים v_i אנו מבצעים חישובים (כגון חיבור ומקסימום) שהינם פולינומיאליים במספר הסיביות.

כעת נחזור לתרגיל המקורי. כדי לכתוב אלגוריתם עם סיבוכיות זמן פולינומיאלית עבור חסם פולינומיאלי על v_i, עלינו לכתוב תכנון דינמי שיחשב

Algorithm 6.8 Compute the knapsack ver. II

Require: weights $(w_i)_{i\in\{1,\dots,n\}}$, and values $(v_i)_{i\in\{1,\dots,n\}}$.

```
Initialize M(0, w) ← 0;
for i ← 1, ..., n do
  for w ← 0, ..., W do
    if w_i > w then
      M(i, w) ← M(i − 1, w); P(i, w) ← w.
    else
      if M(i − 1, w) ≤ v_i + M(i − 1, w − w_i) then
        M(i, w) ← M(i − 1, w); P(i, w) ← w.
      else
        M(i, w) ← v_i + M(i − 1, w − w_i); P(i, w) ← w − w_i.
S ← ∅
w ← W
for i ← n downto 1 do
  if P(i, w) < w then
    S ← S ∪ {i}
  w ← P(i, w)
return S
```

את $\text{Weight}\,\mathrm{OPT}(i,v)$ המוגדר כך:

$$\text{Weight}\,\mathrm{OPT}(i,v)=\min\left\{\sum_{j\in S} w_j:\ S\subseteq\{1,\dots,i\},\quad \sum_{j\in S} v_j=v\right\}.$$

כלומר $\text{Weight}\,\mathrm{OPT}_{\mathcal{W}}(i,v)$ הוא המשקל המינימלי של תת־קבוצה $S\subseteq\{1,\dots,i\}$ שערְכּה הוא v. לשם הנוחות נניח שערך זה הוא ∞ אם אין תת־קבוצה כזאת, כלומר $\min\emptyset=\infty$. שימו לב, בהינתן $\text{Weight}\,\mathrm{OPT}(n,v)$ לכל $v\le 10n^4$, אנו יכולים לפתור את בעיית התרמיל על־ידי החזרת הערך v הגדול ביותר עבור $\text{Weight}\,\mathrm{OPT}(n,v)\le W$.

נציג כעת נוסחה רקורסיבית לחישוב $\text{Weight}\,\mathrm{OPT}(i,v)$ שתוביל לאלגוריתם תכנון דינמי.

$$\text{Weight}\,\mathrm{OPT}(i,v)=\begin{cases}0 & i=v=0\\ \infty & v>i=0\\ \text{Weight}\,\mathrm{OPT}(i-1,v) & v_i>v.\end{cases}$$

When $i>0$ and $v_i\le v$,

$$\text{Weight}\,\mathrm{OPT}(i,v)=\min\{\text{Weight}\,\mathrm{OPT}(i-1,v),w_i+\text{Weight}\,\mathrm{OPT}(i-1,v-v_i)\}. \tag{6.5}$$

הוכחת (6.5) דומה מאוד להוכחת הנכונות של נוסחה (6.1), ולכן נתארה פה בקיצור. זו הוכחה באינדוקציה על i. תהי $S\subseteq\{1,\dots,i\}$ עם ערך v, ומשקל $\text{Weight}\,\mathrm{OPT}(i,v)$.

- כאשר $i=0$, הערך היחיד שתת־קבוצה של $\emptyset$ יכולה להשיג הוא 0 (במשקל 0).

- כאשר $0 < i$, ו־ $v < v_i$, אזי ברור ש־ $i \notin S$ ולכן $S \subseteq \{1, \ldots, i-1\}$ ולכן משקל S הוא $\text{Weight}\,\mathrm{OPT}(i-1, v)$.
- כאשר $0 < i$ וגם $v \geq v_i$ יש שתי אפשרויות:
  - $S \notin i$, ואז כמו בסעיף הקודם, משקלה של S הוא $\text{Weight}\,\mathrm{OPT}(i-1v)$.
  - $i \in S$. במקרה זה, ערכה של $S \setminus \{i\}$ הוא $v - v_i$, ול־ $S \setminus \{i\}$ משקל מינימלי מבין תת־הקבוצות של $\{1, \ldots, i-1\}$ שערכן $v - v_i$, ולכן משקלה של S הוא $w_i + \text{Weight}\,\mathrm{OPT}(i-1, v-v_i)$.

  כיוון שאנו מחפשים משקל מינימלי, המשקל המתקבל הוא המינימום בין שתי האפשרויות לעיל.

נוסחה (6.5) בתוספת האבחנה שהערך המקסימלי בבעיה הנתונה אינו יכול לעבור את $10n^4$, מובילה באופן ישיר לאלגוריתם תכנון דינמי לבעיית התרמיל, המתואר כאלגוריתם 6.9. זמן הריצה שלו הוא בבירור $O(n^5)$. ♣

Algorithm 6.9 Alternative Knapsack$(n, v_1, \ldots, v_n, w_1, \ldots, w_n, W)$

```
Require: v_i ≤ 10n^3
  M(0,0) ← 0.
  M(0,v) ← ∞ for every i ∈ {1, ..., 10n^4}.
  for i ← 1, ..., n do
      for v ∈ 0, 1 ..., 10n^4 do
          if v_i > v then
              M(i,v) ← M(i−1, v)
          else
              M(i,v) ← min{M(i−1, v), w_i + M(i−1, v−v_i)}
  return max{v : M(n,v) ≤ W}
```

פתרון תרגיל 6.5 מעמוד 112

הבעיה בסעיף הראשון שקולה לבעיית התרמיל, עם תרמיל אחד שיכול לשאת משקל $W_1 + W_2$.

הבעיה בסעיף השני מעניינת יותר. יהי $\mathrm{OPT}(i, w, x)$ הערך המקסימלי של פריטים מתוך $\{1, \ldots, i\}$ שאפשר לסחוב בשני התרמילים, כך שבתרמיל הראשון המשקל לא יעלה על w ובתרמיל השני המשקל לא יעלה על x. אזי מתקיימת

הנוסחה הרקורסיבית הזו:

$$
\begin{array}{ll}
\mathrm{OPT}(0,w,x)=0 & \forall\ w,x \\
\mathrm{OPT}(i,w,x)=\mathrm{OPT}(i-1,w,x) & \text{if } w_i>\max\{w,x\} \\
\mathrm{OPT}(i,w,x)=\max\left\{\begin{array}{c}\mathrm{OPT}(i-1,w,x),\\ v_i+\mathrm{OPT}(i-1,w-w_i,x)\end{array}\right\} & \text{if } w\ge w_i>x \\
\mathrm{OPT}(i,w,x)=\max\left\{\begin{array}{c}\mathrm{OPT}(i-1,w,x),\\ v_i+\mathrm{OPT}(i-1,w,x-w_i)\end{array}\right\} & \text{if } x\ge w_i>w \\
\mathrm{OPT}(i,w,x)=\max\left\{\begin{array}{c}\mathrm{OPT}(i-1,w,x),\\ v_i+\mathrm{OPT}(i-1,w-w_i,x),\\ v_i+\mathrm{OPT}(i-1,w,x-w_i)\end{array}\right\} & \text{if } \min\{x,w\}\ge w_i
\end{array}
$$
(6.6)

הוכחת (6.6) דומה להוכחת (6.1), ולכן נסקור אותה בקצרה. כרגיל ההוכחה תהיה באינדוקציה על i. יהיו $S_1,S_2\subseteq\{1,\ldots,i\}$ שתי תת־קבוצות זרות שמביאות למקסימום את הערך המשותף, תחת האילוץ שמשקל S_1 הוא לכל היותר w, ומשקל S_2 הוא לכל היותר x. כלומר, הערך המתקבל הוא $\mathrm{OPT}(i,w,x)$. אנו נבדוק רק מקרה אחד שבו $w_i\le\min\{w,x\}$. במצב הזה ייתכנו שלוש אפשרויות:

- $i\notin S_1\cup S_2$. במצב הזה קל לראות שהערך הוא $\mathrm{OPT}(i-1,w,x)$.
- $i\in S_1$. במצב הזה קל לראות שהערך הוא $v_i+\mathrm{OPT}(i-1,w-w_i,x)$.
- $i\in S_2$. במצב הזה קל לראות שהערך הוא $v_i+\mathrm{OPT}(i-1,w,x-w_i)$.

לכן ברור שהערך של $\mathrm{OPT}(i,w,x)$ הוא המקסימום בין שלוש האפשרויות. ♣

פתרון תרגיל 6.6 מעמוד 113

1. אלו מכם שלמדו את הקורס "אוטומטים ושפות פורמליות" יכולים לראות ששפת המחרוזות המאוזנות היא שפה חסרת הקשר, המתקבלת על־ידי הדקדוק

$$S\to(S)\mid[S]\mid\{S\}\mid SS\ \mid\varepsilon.$$

אפשר לזהות שפות חסרות הקשר על־ידי אוטומטי מחסנית (לא־דטרמיניסטיים). עבור שפת המחרוזות המאוזנות קיים אוטומט מחסנית דטרמינסטי פשוט המזהה אותה, ומשמעות הדבר היא שקיים אלגוריתם זיהוי עם סיבוכיות זמן לינארית.

2. נפנה כעת לפיתוח אלגוריתם למציאת התאמה חוקית עם מספר מינימלי של סוגרים לא מותאמים. בדומה לבעיית המבנה השניוני של רנ"א שהוצגה בספר (בסעיף 6.5) אנו נפתח אלגוריתם לתכנון דינמי על מקטעי המחרוזת X. מספר הסוגרים ללא בן־זוג בהתאמה חוקית $\mathcal{M}$ של מחרוזת הוא $n-2|\mathcal{M}|$. לכן מספיק להביא למקסימום את גודל ההתאמה החוקית. נסמן ב־$\mathrm{OPT}(i,k)$ את גודל ההתאמה החוקית המקסימלית של תת־המחרוזת $x_ix_{i+1}\ldots x_{i+k-1}$.

טענה 6.4 $\mathrm{OPT}(i,k)$ מקיים את הנוסחה הרקוסיבית

$$\begin{aligned}
&\mathrm{OPT}(i,0)=0\\
&\mathrm{OPT}(i,1)=0\\
&\qquad\qquad \text{When } k\geq 2, i \text{ and } (x_i,x_{i+k-1})\text{match,}\\
&\mathrm{OPT}(i,k)=\max_{0<t<k}(\mathrm{OPT}(i,t)+\mathrm{OPT}(i+t,k-t))\\
&\qquad\qquad \text{When } k\geq 2,\text{ and } (x_i,x_{i+k-1})\text{don't match,}\\
&\mathrm{OPT}(i,k)=\max\big\{\mathrm{OPT}(i+1,k-2)+1,\\
&\qquad\qquad \max_{0<j<k}(\mathrm{OPT}(i,j)+\mathrm{OPT}(i+j,k-j))\big\}
\end{aligned} \tag{6.7}$$

הוכחה. נוכיח כעת את טענה 6.4. עבור $k\in\{0,1\}$, ברור ש־ $\mathrm{OPT}(i,k)=0$. נניח כעת כי $k\geq 2$. תחילה נשים לב שאמנם

$$\mathrm{OPT}(i,k)\geq \max_{0<t<k}\{\mathrm{OPT}(i,t)+\mathrm{OPT}(i+t,k-t)\},$$

כי לכל $k>t>0$, וגם לכל התאמה חוקית $\mathcal{M}_1$ של $x_i\dots x_{i+t-1}$, ולכל התאמה חוקית $\mathcal{M}_2$ של $x_{i+t}\dots x_{i+k-1}$, מתקיים כי $\mathcal{M}_1\cup\mathcal{M}_2$ היא התאמה חוקית של $x_i\dots x_{i+k-1}$. כמו־כן, אם x_{i+k-1} תואם ל־x_i אז

$$\mathrm{OPT}(i,k)\geq \mathrm{OPT}(i+1,k-2)+2,$$

כיוון שלכל $\mathcal{M}_1$ התאמה חוקית של $x_{i+1}\dots x_{i+k-2}$, מתקיים כי $\mathcal{M}_1\cup\{(i,i+k-1)\}$ היא התאמה חוקית של $x_i\dots x_{i+k-1}$.

כדי להוכיח את אי־השוויונים בכיוון ההפוך, נקבע התאמה חוקית אופטימלית $\mathcal{M}$ עבור $x_i\dots x_{i+k-1}$, כלומר $|\mathcal{M}|=\mathrm{OPT}(i,k)$. ייתכנו שתי אפשרויות:

- אם $(i,i+k-1)\in\mathcal{M}$, אז $\mathcal{M}\setminus\{(x_i,x_{i+k-1})\}$ היא התאמה חוקית של $x_{i+1}\dots x_{i+k-2}$, ולכן

$$\mathrm{OPT}(i+1,k-2)+1\geq|\mathcal{M}|=\mathrm{OPT}(i,k).$$

- אם $(i,i+k-1)\notin\mathcal{M}$ אז אחת מן השתיים:
  - ל־x_i אין בן־זוג ב־$\mathcal{M}$. במקרה זה ברור ש־$\mathcal{M}$ היא התאמה חוקית של $x_{i+1}\dots x_{i+k-1}$, ולכן

$$\mathrm{OPT}(i,1)+\mathrm{OPT}(i+1,k-1)=\mathrm{OPT}(i+1,k-1)\geq|\mathcal{M}|=\mathrm{OPT}(i,k).$$

  - ישנו $0<t<k$ כך ש־$\mathcal{M}$ $(i,i+t-1)\in$. במקרה זה אפשר להסיק מתנאי הלמינריות כי לא קיים זוג $(i',j')\in\mathcal{M}$ כך ש־ $i\leq i'<i+t\leq j'$. משמעות הדבר היא שאפשר לכתוב $\mathcal{M}$ כאיחוד של $\mathcal{M}_1\cup\mathcal{M}_2$ כך ש־$\mathcal{M}_1$ תהיה התאמה חוקית של $x_i\dots x_{i+t-1}$ ו־$\mathcal{M}_2$ תהיה התאמה חוקית של $x_{i+t}\dots x_{i+k-1}$ ולכן

$$\mathrm{OPT}(i,t)+\mathrm{OPT}(i+t,k-t)\geq|\mathcal{M}_1|+|\mathcal{M}_2|=\mathrm{OPT}(i,k).$$

□

טענה 6.4 מאפשרת לכתוב אלגוריתם לתכנון דינמי שימלא את הטבלה $M(i,k)$ כך שיתקיים $M(i,k) = \mathrm{OPT}(i,k)$ כדלקמן:

Initialize $M(i,0) \leftarrow M(i,1) \leftarrow 0$ for every $i \in \{1,\ldots,n\}$
for $k \leftarrow 2,\ldots,n$ **do**
  for $i \leftarrow 1,\ldots,n-k+1$ **do**
    Calculate $M(i,k)$ according to (6.7)
return $M(1,n)$

לצורך חישוב ההתאמה עצמה, נתחיל, כרגיל מהתשובה הסופית $M(1,n)$ ו"נעקוב לאחור" אחר תת־הבעיות שעבורן מתקבל המקסימום; בכל פעם שהמקסימום מתקבל בביטוי $\mathrm{OPT}(i+1,k-1)+1$, נוכל לאסוף את הזוג $(i,i+k-1)$. אוסף הזוגות שנאספו הוא ההתאמה האופטימלית. ♣

פתרון תרגיל 6.7 מעמוד 115
נסתכל על האלגוריתם לתכנון דינמי המוצע בספר עבור הבעיה הזו.

Algorithm 6.10 Alignment(X,Y)

Array $A(0\ldots.m,0\ldots n)$
Initialize $A(i,0) \leftarrow i\delta$ for each i
Initialize $A(0,j) \leftarrow j\delta$ for each j
for $j \leftarrow 1,\ldots,n$ **do**
  for $i \leftarrow 1,\ldots,m$ **do**
    Use the recurrence (6.2) to compute $A(i,j)$
return $A(m,n)$

בחישוב האיברים בטור j בטבלה A, האלגוריתם מתייחס רק לאיברים בטור j ובטור $j-1$, לכן מספיק לשמור, בכל רגע נתון, את הטור "הנוכחי" ואת הטור "הקודם". מתקבל האלגוריתם הזה.

Algorithm 6.11 Space efficient Alignment(X,Y)

Array CurColA$(0\ldots.m)$, PrevColA$(0\ldots m)$
Initialize CurrColA$(i) \leftarrow i\delta$ for each i
for $j \leftarrow 1,\ldots,n$ **do**
  PrevColA $\leftarrow$ CurColA
  Initialize CurrColA$(0) \leftarrow j\delta$
  for $i \leftarrow 1,\ldots,m$ **do**
    Use the recurrence (6.2) to where we replace $A(t,j)$ with CurColA(t), and $A(t,j-1)$ with PrevColA(t).
return CurrColA(m)

♣

פתרון תרגיל 6.8 מעמוד 116
מספיק להראות כי אם אפשר לקבל את Y מ־X על־ידי k פעולות מחיקה

והוספה, אזי אפשר לקבל גם את X מ־Y על־ידי אותו מספר k של פעולות מחיקה והוספה. נניח כי Y מתקבל מ־X על־ידי k פעולות מחיקה והוספה. אז קיימת סדרת מחרוזות $X = X_1, X_2, \ldots, X_k = Y$ כך שלכל $i \in \{1, \ldots, k-1\}$ המחרוזת X_{i+1} מתקבלת מהמחרוזת X_i על־ידי פעולה אחת של מחיקה או הוספה. כעת נשים לב כי פעולות המחיקה וההוספה הן פעולות הפוכות, כלומר, אם מחרוזת אחת מתקבלת מהשניה על־ידי פעולת הוספה (מחיקה), אזי המחרוזת השנייה מתקבלת מהראשונה על־ידי פעולת מחיקה (הוספה). ולכן, בסדרה ההפוכה $X_k = Y, X_{k-1}, \ldots, X_1 = X$ כל מחרוזת עוקבת מתקבלת מהמחרוזת הקודמת על־ידי פעולה אחת של מחיקה או הוספה. ובכך הוכחנו שאפשר לקבל את X מ־Y על־ידי k פעולות מחיקה והוספה, כנדרש. ♣

פתרון תרגיל 6.9 מעמוד 116

בעיית מרחק העריכה מזכירה את בעיית יישור הסדרות, ולכן סביר לפתח עבורה אלגוריתם תכנון דינמי הדומה לאלגוריתם עבור יישור סדרות. לאמיתו של דבר, מחשבה מעמיקה יותר על הבעיה של חישוב מרחק עריכה, מביאה למסקנה שהיא מקרה פרטי של בעיית יישור הסדרות. נציג זאת כטענה פורמלית.

טענה 6.5 מרחק העריכה בין זוג מחרוזות X ו־Y הוא בדיוק עלות היישור המינימלית של שתי המחרוזות כאשר $\delta = 1$; עלות היישור לכל תו a היא $\alpha(a,a) = 0$ ולכל זוג תווים $a \neq b$ עלות היישור היא $\alpha(a,b) = \infty$.

הוכחה. תחילה נוכיח שאפשר למצוא לכל זוג מחרוזות X ו־Y יישור שעלותו תהיה לכל היותר $d_{\text{ED}}(X,Y)$. טענה זו מוכחת באינדוקציה על מרחק העריכה בין X ל־Y. אם $d_{\text{ED}}(X,Y) = 0$ אז $X = Y$ ולכן היישור $X' = X$ ו־$Y' = Y$ הוא יישור חוקי שעלותו 0. כעת נניח כי $d_{\text{ED}}(X,Y) > 0$. קיימת פעולת הוספה/מחיקה אשר מקטינה את המרחק; ללא הגבלת הכלליות הפעולה הזאת מתבצעת על X ויוצרת מחרוזת $\widehat{X}$, כך ש־ $d_{\text{ED}}(\widehat{X},Y) = d_{\text{ED}}(X,Y) - 1$. מהנחת האינדוקציה נובע שקיים יישור Y' של Y ויישור $\widehat{X}'$ של $\widehat{X}$ שעלותה $d_{\text{ED}}(\widehat{X},Y)$. נבחן כעת את הפעולה שיצרה את $\widehat{X}$ מן המחרוזת $X = x_1 \ldots x_n$.

- נניח תחילה שזו פעולת הוספה ולכן $\widehat{X} = x_1 \ldots x_i z x_{i+1} \ldots x_n$.
  - אם ביישור של הזוג $(\widehat{X},Y)$ התו z ב־X מותאם לתו זהה במחרוזת היישור של Y, אזי אפשר להגדיר מחרוזת יישור של X הזהה למחרוזת היישור של $\widehat{X}$ פרט להחלפת התו z בתו '-'. מתקבל יישור של X מול Y שעלותו גדולה בדיוק באחד מעלות היישור של $\widehat{X}$ ו־Y. ובזה מוכחת הנחת האינדוקציה.
  - אם ביישור של הזוג $(\widehat{X},Y)$ התו z ב־X מותאם לתו '-' במחרוזת היישור של Y, אזי נוכל להגדיר מחרוזות יישור של X ו־Y שתהיה זהות למחרוזות היישור של $\widehat{X}$ ו־Y , למעט העובדה שהתו z ביישור של $\widehat{X}$ והתו '-' המקביל לו ביישור של Y נמחקים. אנו מקבלים יישור של X ו־Y שעלותו קטנה באחד מעלות היישור של $\widehat{X}$ ו־Y.
- כעת נניח שזו פעולת מחיקה ולכן $\widehat{X} = x_1 \ldots x_{i-1} x_{i+1} \ldots x_n$. ניקח את היישור של $\widehat{X}$ ו־Y ונבנה ממנו יישור ל־X ו־Y על־ידי הוספת התו x_i במקום המתאים ביישור של $\widehat{X}$ ומולו הוספת התו '-' ביישור של Y. מתקבל יישור שעלותו גדולה באחד מעלות היישור של $\widehat{X}$ ו־Y.

כעת נוכיח, בכיוון ההפוך, שהעלות של כל יישור של X ו־Y, היא לפחות $d_{\text{ED}}(X,Y)$. ההוכחה תהיה באינדוקציה על עלות היישור, השווה בדיוק למספר

תווי '-' הנמצאים ביישור. אם עלות היישור היא 0, אז $X = Y$ וגם $d_{\text{ED}}(X,Y) = 0$. אחרת, נבחר תו '-', בלי הגבלת הכלליות מהיישור של Y. מול התו הזה עומד ביישור של X התו z אשר הופיע במקור גם ב־X. נייצר את $\widehat{X}$ מ־X על־ידי מחיקת התו z מ־X. כמו־כן נבנה יישור של $\widehat{X}$ ו־Y על־ידי מחיקת התו z ומולו נמחק גם את התו '-' ביישור של X ו־Y. עלות היישור של $\widehat{X}$ ו־Y קטנה באחד מעלות היישור של X ו־Y, ולפי הנחת האינדוקציה היא לפחות $d_{\text{ED}}(\widehat{X},Y)$. כיוון ש־$d_{\text{ED}}(X,Y) \le d_{\text{ED}}(\widehat{X},Y) + 1$, סיימנו בזה גם את הוכחת הכיוון השני. □

טענה 6.5 מובילה ישירות לאלגוריתם לתכנון דינמי המחשב את מרחק העריכה בסיבוכיות הזמן $O(n^2)$ – זהו האלגוריתם ליישור סדרות שכולל את העלויות המיוחדות שצוינו בטענה. עלויות אלה מפשטות במקצת את האלגוריתם; הפירוט רשום באלגוריתם 6.12.

Algorithm 6.12 Edit Distance($X = x_1 \dots x_n$, $Y = y_1 \dots y_m$)

```
Array A(0...m, 0...n)
Initialize A(i,0) ← i for each i
Initialize A(0,j) ← j for each j
for j ← 1,...,n do
    for i ← 1,...,m do
        A(i,j) ← 1 + min{A(i,j−1), A(i−1,j)}
        if x_i = y_j then
            A(i,j) ← min{A(i,j), A(i−1,j−1)}
return A(m,n)
```

הערת אגב: טענה 6.5 מוכיחה גם את הטענה שהוצגה בתרגיל 6.8, כיוון שעלות יישור הסדרות היא בבירור סימטרית. ♣

פתרון תרגיל 6.10 מעמוד 117

כדי לחשב את המרחקים מצומת נתון s בגרף G לשאר הצמתים בגרף, אנו יכולים לייצר בשלב הראשון, בזמן לינארי, גרף G^{rev} שבו הקשתות הפוכות לקשתות G, אך יש להן אותו משקל. כעת נפעיל על G^{rev} את אלגוריתם 6.1 ונמצא את המרחקים מכל הצמתים בגרף G^{rev} אל s, שהם המרחקים מ־s לכל הצמתים בגרף G. ♣

פתרון תרגיל 6.11 מעמוד 119

נסמן ב־$M_\ell(v)$ את הערך של $M(v)$ באלגוריתם 6.2 בסוף ביצוע הלולאה $i = \ell$. ההוכחה דומה להוכחת הנכונות של אלגוריתם 6.1 (בלמן־פורד הסטנדרטי) אך טיפה יותר גמישה. אם עבור אלגוריתם 6.1 הוכחנו כי $M(\ell,v) = \text{OPT}(\ell,v)$, עבור אלגוריתם 6.2 נטען רק ש־$M_\ell(v) \le \text{OPT}(\ell,v)$ ושהערך $M_\ell(v)$ מייצג מרחק של איזשהו מסלול מ־v ל־t (לאו דוקא מסלול שיש בו ℓ קשתות לכל היותר). הוכחת שתי הטענות הללו מתבצעות באינדוקציה על ℓ , בדומה להוכחת הטענה על $M(\ell,v)$.

שימו לב שבניגוד למצב באלגוריתם 6.1, הערכים ב־$M_\ell(v)$ תלויים בסדר בו נסרקים הצמתים, אבל בסיום האלגוריתם התוצאה איננה תלויה בסדר סריקת הצמתים (תמיד מתקבלים המרחקים המינימליים בגרף). ♣

פתרון תרגיל 6.12 מעמוד 119
בכל איטרציה עבור $\ell \geq 1$ המערך M מתעדכן רק על־סמך הערכים ב־M, לכן אם בלולאה על v לא השתנה אף ערך ב־M, גם אחר־כך לא ישתנה אף ערך ב־M ולכן $M_{n-1}(v) = M_\ell(v)$, לכל $v \in V$. ♣

פתרון תרגיל 6.13 מעמוד 123
כרגיל במקרים כאלה, האינדקס k בטבלה M מיותר למעשה, ולכן אפשר פשוט להשמיטו.

Algorithm 6.13 Space efficient Floyd–Warshall

Require: Weighted graph $G = (\{1, \ldots, n\}, E, c)$ without negative cycles.
  Initialize $M(i,j) \leftarrow c(v_i, v_j)$ if $(v_i, v_j) \in E$
  Initialize $M(i,j) \leftarrow \infty$ if $(v_i, v_j) \notin E$
  Initialize $M(i,i) \leftarrow 0$
  for $k \leftarrow 1, \ldots n$ **do**
    for $i, j \leftarrow 1, \ldots n$ **do**
      $M(i,j) \leftarrow \min\{M(i,j), M(i,k) + M(k,j)\}$.
  return $M(i,j)$ for every $i, j \in \{1, \ldots, n\}$.

♣

פתרון תרגיל 6.14 מעמוד 123
1. כפי שכבר ראינו (בדיון על מסלולים קצרים ביותר) באלגוריתם בלמן־פורד החוסך־מקום, לכל זוג צמתים $u, v \in V$ מספיק לשמור במערך $S(u,v)$ את הצומת העוקב ל־u במסלול קצר ביותר מ־u ל־v. כמו באלגוריתם בלמן־פורד ניתן גם לייצג את המסלולים הקצרים ביותר כ־n עצי מסלולים קצרים ביותר, עץ T_v עבור צומת $v \in V$. העץ T_v מוגדר כדלקמן:

$$T_v = \{(u, S(u,v)) |\; u \in V \setminus \{v\}\}.$$

2. נוסיף את ניהול S לאלגוריתם 6.13.

Algorithm 6.14 Space efficient Floyd–Warshall with shortest paths

Require: Wegihted graph $G = (\{1, \ldots, n\}, E, c)$ without negative cycles.
  Initialize $M(i,j) \leftarrow c(v_i, v_j)$, if $(v_i, v_j) \in E$
  Initialize $M(i,j) \leftarrow \infty$, if $(v_i, v_j) \notin E$
  Initialize $M(i,i) \leftarrow 0$
  Initialize $S(i,j) \leftarrow j$, if $(v_i, v_j) \in E$
  Initialize $S(i,j) \leftarrow$nil, if $(v_i, v_j) \notin E$
  for $k \leftarrow 1, \ldots n$ **do**
    for $i, j \leftarrow 1, \ldots n$ **do**
      if $M(i,j) > M(i,k) + M(k,j)$ **then**
        $M(i,j) \leftarrow M(i,k) + M(k,j)$
        $S(i,j) \leftarrow S(i,k)$
  return $M(i,j)$ and $S(i,j)$ for every $i, j \in \{1, \ldots, n\}$.

אנו כבר יודעים (מפתרון תרגיל 6.13) שבסיום ריצת האלגוריתם, המערך $M(u,v)$ יכיל את אורך המסלול הקצר ביותר מ־u ל־v. לכן מספיק להראות (בדומה לטענה 6.1) שבכל שלב של ריצת האלגוריתם, אם $M(u,v) < \infty$ אז התא $S(u,v)$ מכיל את העוקב ל־u במסלול מ־u ל־v שאורכו $M(u,v)$. הוכחת טענה זו מתבצעת באינדוקציה על מספר הפעולות שהאלגוריתם ביצע. בשלב האתחול הנכונות ברורה. בהמשך, $S(u,v)$ מתעדכן אם ורק אם $M(u,v)$ מתעדכן, והעדכון של $S(u,v)$ שומר על התכונה הנ"ל. ♣

פתרון תרגיל 6.15 מעמוד 123

1. הטענה הבאה מראה כיצד לחשב את d_G מתוך $d_{\overline{G}}$ ו־h.

טענה 6.6 לכל $u, v \in V$ מתקיים $d_G(u,v) = d_{\overline{G}}(u,v) + h(v) - h(u)$

הוכחה. יהי $u = u_0, u_1, \ldots, u_t = v$ המסלול הקצר ביותר ב־ G, אזי

$$\begin{aligned} d_{\overline{G}}(u,v) &\leq \sum_{i=1}^{t} \overline{c}(u_{i-1}, u_i) = \sum_{i=1}^{t} (c(u_{i-1}, u_i) + h(u_{i-1}) - h(u_i)) \\ &= h(u) - h(v) + \sum_{i=1}^{t} c(u_{i-1}, u_i) = d_G(u,v) + h(u) - h(v) \end{aligned}$$

באופן דומה, יהי $u = u_0, u_1, \ldots, u_t = v$ המסלול הקצר ביותר ב־ $\overline{G}$, אזי

$$\begin{aligned} d_{\overline{G}}(u,v) &= \sum_{i=1}^{t} \overline{c}(u_{i-1}, u_i) = \sum_{i=1}^{t} (c(u_{i-1}, u_i) + h(u_{i-1}) - h(u_i)) \\ &= h(u) - h(v) + \sum_{i=1}^{t} c(u_{i-1}, u_i) \geq d_G(u,v) + h(u) - h(v) \end{aligned}$$

□

2. תהי $(u,v) \in E$. אחד המסלולים בגרף המורחב, מ־q ל־v, הוא המסלול הקצר ביותר ל־u בתוספת הקשת (u,v). אורכו $h(u) + c(u,v)$. מכאן ש־$h(v) \leq h(u) + c(u,v)$, ומשמעות הדבר היא:

$$\overline{c}(u,v) = c(u,v) + h(u) - h(v) \geq 0$$

3. נסמן ב־ G' את הגרף המתקבל מ־G אחרי הוספת הצומת q ונסמן את הקשתות במשקל 0 לכל הצמתים ב־G. תחילה נבחין הבחנה פשוטה: אם אין ב־G מעגלים שליליים, אז גם ב־ G' אין מעגלים שליליים. הסיבה לכך פשוטה: כל הקשתות שנוספו אינן נמצאות על מעגלים, שהרי כולן יוצאת מ־q, ואף קשת אינה נכנסת אל q. משום־כך נקבל את האלגוריתם הבא לחישוב המרחקים בין כל הזוגות בגרף $G = (V,E,c)$ ללא מעגלים שליליים:

א. נבנה את הגרף G'.

ב. נריץ את אלגוריתם בלמן־פורד על G', החל מצומת מקור q ונקבל פונקציית גובה על V המוגדרת $h(v) = d_{G'}(q,v)$.

ג. נגדיר $\overline{c}(u,v) = c(u,v) + h(u) - h(v)$ ונסמן את הגרף עם המשקולות $\overline{c}$ כ־$\overline{G} = (V,E,\overline{c})$.

ד. ב־ $\overline{G}$ אין קשתות שליליות, לכן נוכל לחשב את המרחקים הקצרים ביותר $d_{\overline{G}}(u,v)$ בין כל זוגות הצמתים $u,v \in V$ על־ידי הפעלת האלגוריתם של דייקסטרה על כל צומת.

ה. נפלוט $d_G(u,v) = d_{\overline{G}}(u,v) - h(u) + h(v)$

נכונות האלגוריתם הוכחה בסעיפים הקודמים. הזמן הדרוש לביצוע האלגוריתם הזה נשלט על־ידי הרצת האלגוריתם של דייקסטרה n פעמים, ולכן הוא $O(mn \log n)$. ♣

פרק 7

זרימה ברשתות

7.1 בעיית הזרימה המקסימלית ואלגוריתם פורד־פולקרסון

קראו בספר את סעיף 7.1

נחזור בקצרה על המושגים המרכזיים בסעיף שקראתם.

הגדרה 7.1 רשת זרימה היא רביעייה (G, c, s, t), המקיימת:
- $G = (V, E)$ הוא גרף מכוון;
- $c : E \to [0, \infty]$ היא פונקציית קיבול על הקשתות המקיימת $c_e \geq 0$ לכל $e \in E$ (אפשר גם $c_e = \infty$);
- $s, t \in V$. הצומת s נקרא צומת מקור, ו־t נקרא צומת בור. ל־s לא נכנסות קשתות, ומ־t לא יוצאות קשתות.

$\diamondsuit$

הגדרה 7.2 זרימה (ממקור s לבור t) ברשת זרימה היא פונקציה $f : E \to [0, \infty)$ המתאימה מספר אי־שלילי לכל קשת, כך שמתקיימים:

(1) $\forall e \in E \qquad 0 \leq f(e) \leq c_e$ (אילוצי הקיבול)

(2) $\forall v \in V \setminus \{s, t\} \qquad f^{\text{in}}(v) = f^{\text{out}}(v)$ (חוק שימור הזרימה)

כאשר:

$f^{\text{in}}(v) = \sum_{e \text{ enters } v} f(e)$ היא הזרימה הנכנסת ל־v;

$f^{\text{out}}(v) = \sum_{e \text{ leaves } v} f(e)$ היא הזרימה היוצאת מ־v.

ערך/גודל הזרימה הוא

$$\nu(f) = f^{\text{out}} = f^{\text{out}}(s) - f^{\text{in}}(s).$$

זרימה בעלת ערך מקסימלי תיקרא זרימה מקסימלית. $\diamondsuit$

הבעיה שבה אנו דנים בפרק הנוכחי היא:

בעיה אלגוריתמית: זרימה מקסימלית.
הקלט: רשת זרימה (G, c, s, t) עם מקור s ובור t.
הפלט: זרימה חוקית f ברשת הזרימה.
המטרה: להביא למקסימום את ערך הזרימה $\nu(f)$.

אנו נראה שיטה אינקרמנטלית לפתרון בעיית הזרימה המקסימלית. בשיטה זאת משפרים את הזרימה החוקית על־ידי התקדמות בצעדים – בכל צעד מגדילים את הערך של הזרימה. בהינתן זרימה f ברשת זרימה (G, c, s, t) עם מקור s ובור t, אנו נדרשים לענות על השאלות האלה: האם קיימת דרך לקבוע ש־f היא הזרימה המקסימלית? ואם f אינה הזרימה המקסימלית – האם יש דרך למצוא בעזרת f זרימה בעלת ערך גדול יותר?

הגיוני יהיה להציע את הדרך הבאה: נמצא מסלול מכוון P מ־s ל־t בגרף G, כך שלכל קשת $e \in P$ יתקיים $c_e - f(e) > 0$, כלומר

$$\varepsilon = \varepsilon(P, f) = \min_{e \in P}(c_e - f(e)) > 0. \tag{7.1}$$

בתנאים אלה נוכל להגדיל את הזרימה ב־ε לאורך P באופן הזה:

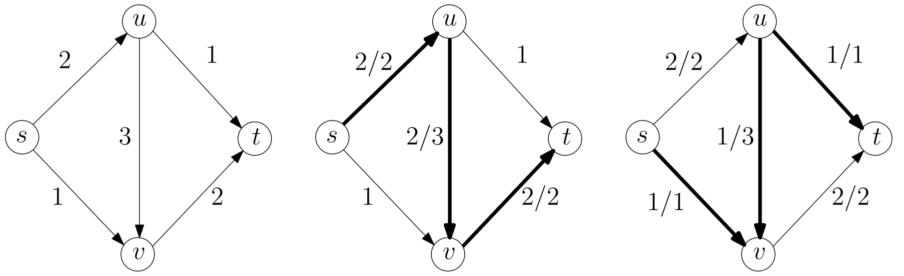
$$f'(e) = \begin{cases} f(e) + \varepsilon & e \in P \\ f(e) & \text{otherwise.} \end{cases}$$

קל לראות כי f' היא זרימה (כאשר f היא זרימה), וערכה גדול בדיוק ב־ε מהערך של f, כלומר $\nu(f') = \nu(f) + \varepsilon$. אבל לא יהיה נכון לומר כי אי־קיום מסלול P כנ"ל משמעותו ש־f היא זרימה מקסימלית. הסתכלו בדוגמה המוצגת באיור 7.1 (הדומה לאיור 7.3 בספר) האיור במרכז מציג את הזרימה $f(s, u) = f(u, v) = f(v, t) = 2$. במצב זה לא ניתן להגדיל את הזרימה בעזרת מסלול מהצורה האמורה לעיל. למרות זאת, הזרימה איננה זרימה מקסימלית. באיור 7.1 מצד ימין מוצגת זרימה גדולה יותר. הסיבה לכך היא: כדי להגדיל את ערך הזרימה, ייתכן שנצטרך להקטין את הזרימה דרך קשתות מסוימות כך שנוכל להגדילה באופן משמעותי דרך הקשתות האחרות. בדוגמה באיור 7.1 הקטנו את הזרימה בקשת האנכית כדי להגיע לזרימה מצד ימין.

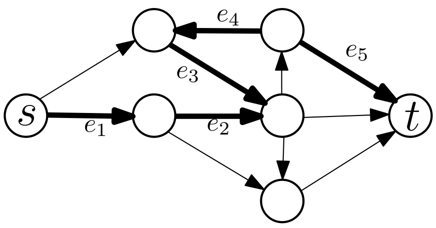
ננתח רעיון זה מזווית שונה במקצת מזו שבספר. נזכיר כי גרף התשתית של G הוא הגרף הלא־מכוון המתקבל מ־G על־ידי התעלמות מכיווני הקשתות. תהי $Q = (s = v_0, v_1, \dots, v_k = t)$ סדרת צמתים שמתאימה למסלול פשוט מ־s ל־t בגרף התשתית של G. תהי $P = (e_1, e_2, \dots, e_t)$ סדרת קשתות ב־G, כך שלכל $i \in \{1, \dots, k\}$ מתקיים $e_i = (v_{i-1}, v_i)$ או $e_i = (v_i, v_{i-1})$. אם $e_i = (v_{i-1}, v_i)$ אז e_i תיקרא קשת קדימה [forward edge] (ב־P). אחרת, e_i תיקרא קשת אחורה [backward edge] (ביחס ל־P). לדוגמה, ראו איור 7.2.

הגדרה 7.3 הקיבולת השיורית של קשת e_i במסלול P מהסוג שלעיל ביחס לזרימה f מוגדרת כ:

$$\varepsilon_{e_i}(P, f) = \begin{cases} c_{e_i} - f(e_i) & \text{if } e_i \text{ is forward edge} \\ f(e_i) & \text{if } e_i \text{ is backward edge.} \end{cases}$$



איור 7.1: דוגמה לרשת זרימה (מצד שמאל) וזרימת מסלול "חוסמת" באמצע $s \to u \to v \to t$ בערך 2. מצד ימין מוצגת זרימה גדולה יותר. ניתן לראות אותה כתוספת של זרימת המסלול $s \to v \to u \to t$ בערך 1 מעבר לזרימה הקודמת.



איור 7.2: גרף מכוון ו־$P = (e_1, e_2, e_3, e_4, e_5)$ הוא מסלול מ־s ל־t בגרף התשתית. במסלול זה, הקשתות e_1, e_2, e_5 הן קשתות קדימה, בעוד שהקשתות e_3, e_4 הן קשתות אחורה.

צוואר־הבקבוק [bottleneck] של P הוא:

$$\varepsilon(P, f) = \min_{i \in \{1,\dots,k\}} \varepsilon_{e_i}(P, f). \tag{7.2}$$

מסלול P של קשתות בגרף התשתית יקרא מסלול שיפור [augmenting path] ביחס לזרימה f אם $\varepsilon(P, f) > 0$. ♢

דוגמה: נתבונן שוב באיור 7.1 בזרימה המוצגת במרכז. נתבונן בסדרת הקשתות $P = \big((s,v),(u,v),(u,t)\big)$. במקרה זה,

$$\begin{aligned}
\varepsilon_{(s,v)}(P, f) &= c_{(s,v)} - f((s,v)) = 1 - 0 = 1;\\
\varepsilon_{(u,v)}(P, f) &= f((u,v)) = 2;\\
\varepsilon_{(v,t)}(P, f) &= c_{(v,t)} - f((v,t)) = 1 - 0 = 1.
\end{aligned}$$

ולכן $\varepsilon(P, f) = \min\{1, 2, 1\} = 1$. ♣

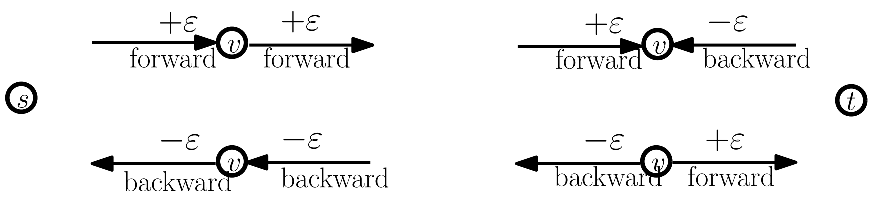
נשים לב שההבדל מהגדרה הנאיבית של צוואר הבקבוק ומסלול שיפור כפי שהיא מבוטאת ב־(7.1), הוא שבהגדרה הנוכחית (7.2) אנו מאפשרים להשתמש גם בקשתות אחורה, בתנאי שעוברת בהן זרימה. הטענה הבאה מראה שהגדרה (7.2) מוצדקת – ניתן להשתמש בה כדי להגדיל את הזרימה ברשת.

טענה 7.1 יהי P מסלול שיפור ביחס לזרימה f, ונסמן $\varepsilon = \varepsilon(P,f)$. הפונקציה $f' : E \to [0,\infty)$ המוגדרת על־ידי

$$f'(e) = \begin{cases} f(e) + \varepsilon & e \text{ is a forward edge in } P \\ f(e) - \varepsilon & e \text{ is a backward edge in } P \\ f(e) & \text{otherwise} \end{cases}$$

מייצגת זרימה חוקית מ־s ל־t והערך שלה גבוה מן הערך של f ב־ε.

הוכחה. מהגדרת ε נובע כי $0 \le f'(e) \le c_e$ לכל $e \in E$. נותר להראות כי f' מקיימת את חוק שימור הזרימה: $\sum_{e \text{ enters } v} f'(e) = \sum_{e \text{ leaves } v} f'(e)$ לכל $v \in V \setminus \{s,t\}$. ברור לנו שחוק שימור הזרימה ממשיך להתקיים לגבי כל צומת v שאינו שייך למסלול P וזאת על־סמך העובדה ש־f היא זרימה. נתבונן כעת בצומת $v \in P \setminus \{s,t\}$ ובשתי הקשתות הרצופות של P ש־v נמצא ביניהן. יש ארבעה מקרים:



איור 7.3: ארבעה מקרים של צומת השייך למסלול שיפור.

בכל אחד מהמקרים (באיור 7.3) חוק שימור הזרימה ממשיך להתקיים. שתי הזרימות (הנכנסת ל־v והיוצאת ממנו) עשויות לגדול או לקטון ב־ε, אך ייתכן גם שהן לא תשתנינה. במקרה המוצג למעלה, בצד שמאל – שתיהן גדלות; במקרה המוצג בצד שמאל למטה – שתיהן קטנות ב־ε, ובשני המקרים המוצגים בצד ימין של האיור – לא חל בשתיהן כל שינוי. □

דוגמה: נחזור שוב לדוגמת הזרימה שבמרכז איור 7.1 והתבוננו במסלול השיפור $P = \big((s,v),(u,v),(u,t)\big)$. כפי שחישבנו לעיל, $\varepsilon = \varepsilon(P,f) = 1$, והזרימה f' לאחר הוספת הזרימה לאורך P הינה

$$\begin{aligned} f'((s,v)) &= f(s,v) + 1 = 1, \\ f'((u,v)) &= f((u,v)) - 1 = 2 - 1 = 1, \\ f'((u,t)) &= f((u,t)) + 1 = 1, \end{aligned}$$

ובשאר הקשתות $f' = f$. הזרימה f' המתקבלת מוצגת בזרימה שמצד ימין באיור 7.1. ♣

אבל כיצד נמצא את מסלול השיפור שתיארנו לעיל? הדרך הקלה והיעילה לעשות זאת היא לחפש מסלול מכוון מ־s ל־t ברשת השיורית [residual network]. נזכיר כי הרשת השיורית G_f של זרימה f ברשת זרימה G מוגדרת כך: לכל קשת $e = (u,v)$ ברשת המקורית, תימצא ברשת השיורית הקשת הבאה:

- אם $c_e - f(e) > 0$, תהיה ברשת השיורית קשת (u, v) עם קיבול שיורי $c_e - f(e)$;
- אם $f(e) > 0$, תהיה ברשת השיורית קשת "הפוכה" (v, u) עם קיבול שיורי $f(e)$.

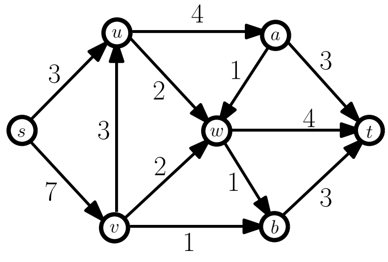
לחילופין, אפשר לבנות את הרשת השיורית גם בדרך הזו: לכל קשת $e = (u, v)$ בגרף המקורי, רושמים שתי קשתות ברשת השיורית: קשת (u, v) עם קיבול שיורי $c_e - f(e)$, וקשת הפוכה (v, u) עם קיבול שיורי $f(e)$. לאחר־מכן מוחקים את כל הקשתות שהקיבול השיורי שלהן הוא אפס.

כל זה מוביל לאלגוריתם הבא לחישוב זרימה מקסימלית ע"י מציאת מסלולי שיפור עוקבים:

Algorithm 7.1 Ford-Fulkerson(flow network G, c, s, t)

```
Initialize f ← feasible flow (can be f ≡ 0)
while there exists an augmenting path P with respect to f do
    Increase f by ε(P, f) along P.
return f
```

אם מניחים שכל הקיבולים הם מספרים שלמים, ערך הזרימה גדל לפחות ב־1 בכל שלב של הלולאה. לכן מספר הפעמים שהלולאה מתבצעת יהיה לכל היותר ν פעמים, כאשר ν הוא הערך המקסימלי של זרימה ברשת. בפרט, אם C הוא חסם כלשהו על ν (לדוגמה, C יכול להיות הסכום של קיבולי כל הקשתות) אזי הלולאה תתבצע C פעמים לכל היותר (טענה (7.4) בספר). בכל שלב של הלולאה בונים את הרשת השיורית, ומוצאים בה מסלול מ־s ל־t (אם קיים מסלול כזה), ומשפרים את הזרימה לאורך המסלול. אפשר לעשות זאת בזמן $O(m)$. לכן הסיבוכיות הכוללת של האלגוריתם היא $O(\nu m) = O(Cm)$ (טענה (7.5) בספר). בשלב זה עדיין לא ברור אם אלגוריתם פורד־פולקרסון מחשב זרימה אופטימלית; מיד נעסוק בכך, בסעיף 7.2.



איור 7.4: רשת זרימה

תרגיל 7.1 הריצו את אלגוריתם פורד־פולקרסון על הרשת המוצגת באיור 7.4. מהו ערך הזרימה שמייצר האלגוריתם?

פתרון בעמוד 163

7.2 זרימות מקסימליות וחתכים מינימליים ברשת

קראו בספר את סעיף 7.2

הגדרה 7.4 חתך s-t ברשת זרימה $(G=(V,E),c,s,t)$ הוא תת־קבוצה A של צומתי הגרף, המקיימת $s\in A$, $t\in V\setminus A$. נאמר כי הקשת $e=(u,v)\in E$ "יוצאת מ־A" אם $u\in A$ ו־$v\notin A$. קיבול החתך הוא

$$c(A)=\sum_{e\text{ leaves }A} c_e .$$

חתך s-t בעל קיבול מינימלי נקרא חתך s-t מינימלי, או פשוט חתך מינימלי אם s,t ברורים מההקשר. $\diamondsuit$

הערה: הזוג $(A,V\setminus A)$ נקרא בספר חתך, אבל ברור שדי בקבוצה A בלבד כדי לאפיין את החתך. $\clubsuit$

בסעיף 7.2 בספר מוכיחים את שני המשפטים החשובים האלה:

משפט 7.2 הזרימה f היא זרימה מקסימלית אם ורק אם אין מסלול שיפור ביחס ל־f.

ממשפט 7.2 הראשון משתמע כי אלגוריתם פורד־פולקרסון אכן מחשב זרימה אופטימלית (טענה (7.10) בספר).

משפט 7.3 (זרימה־מקסימלית וחתך־מינימלי [Max-Flow Min-Cut]) (טענה (7.13) בספר) בכל רשת זרימה, ערך הזרימה המקסימלית שווה לקיבול החתך המינימלי.

נתחיל בהוכחת החסם $\nu(f)\le c(A)$ לכל זרימה f ולכל חתך A. לצורך זה משתמשים בטענה הבאה – טענה (7.6) בספר:

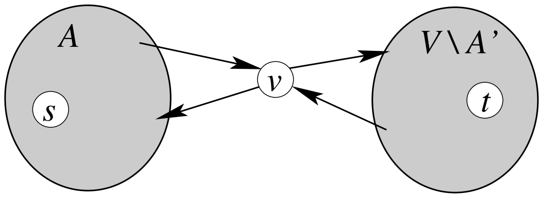
טענה 7.4 תהי f זרימה מ־s ל־t ויהי A חתך s-t כלשהו. אזי ערך הזרימה $\nu(f)$ מקיים:

$$\nu(f)=f^{\text{out}}(A)-f^{\text{in}}(A)\ .$$

הוכחה. ניתן הוכחה "ציורית" יותר מזו שבספר. נוכיח כי אם A חתך $s-t$ כלשהו, אזי העברת צומת $v\in V\setminus A$ $(v\neq t)$ ל־A לא משנה את "סך הזרימה" $f^{\text{out}}(A)-f^{\text{in}}(A)$ דרך A. כלומר, אם $A'=A\cup\{v\}$ אזי

$$f^{\text{out}}(A)-f^{\text{in}}(A)=f^{\text{out}}(A')-f^{\text{in}}(A')\ .$$

כיוון שניתן לקבל כל חתך $s-t$ A מהחתך $\{s\}$ על־ידי סדרה של "העברות" כאלה, נקבל $\nu(f)=f^{\text{out}}(\{s\})-f^{\text{in}}(\{s\})=f^{\text{out}}(A)-f^{\text{in}}(A)$, כנדרש.



איור 7.5: המחשה להוכחת טענה 7.4.

עבור $X, Y \subseteq V$ נסמן ב־$f(X,Y)$ את הזרימה בקשתות ההולכות מ־X ל־Y.
שימו לב כי $f^{\text{out}}(v) = f(v,A) + f(v, V\setminus A')$, $f^{\text{in}}(v) = f(V\setminus A', v) + f(A,v)$, ראו איור 7.5. ולכן על־פי חוק שימור הזרימה:

$$(f(v,V\setminus A') + f(v,A)) - (f(A,v) + f(V\setminus A',v)) = f^{\text{out}}(v) - f^{\text{in}}(v) = 0\ .$$

כמו כן, קל לראות כי (ראו איור 7.5):

$$\begin{aligned} f^{\text{out}}(A') &= f^{\text{out}}(A) + f(v,V\setminus A') - f(A,v) \\ f^{\text{in}}(A') &= f^{\text{in}}(A) + f(V\setminus A',v) - f(v,A) \end{aligned}$$

על־ידי חיסור השוויון השני מהראשון נקבל:

$$f^{\text{out}}(A') - f^{\text{in}}(A') = f^{\text{out}}(A) - f^{\text{in}}(A) + [f^{\text{out}}(v) - f^{\text{in}}(v)] = f^{\text{out}}(A) - f^{\text{in}}(A)$$

כפי שרצינו להוכיח. □

עתה קל לחסום מלמעלה את הזרימה המקסימלית בעזרת קיבולי חתכי s-t:

טענה 7.5 (טענה (7.8) בספר) $\nu(f) \le c(A)$ לכל זרימה f מ־s ל־t ולכל חתך s-t A.

הוכחה.

$$\nu(f) = f^{\text{out}}(A) - f^{\text{in}}(A) \le f^{\text{out}}(A) \le c(A).$$

השוויון הראשון נובע כמובן מטענה 7.4. אי השוויון האחרון הוא מיידי מההגדרות: הזרימה היוצאת מ־A הינה סכום הזרימות בקשתות היוצאות מ־A. זרימה בקשת היא לכל היותר קיבול הקשת, ולכן הזרימה היוצאת מ־A היא לכל היותר סכום קיבולי הקשתות היוצאות מ־A, כלומר קיבול החתך A. □

מסקנה 7.6 תהי f זרימה. אם ב־G יש חתך s-t, נניח A, שקיבולו $\nu(f)$, אזי f היא זרימה מקסימלית, ו־A הוא חתך s-t מינימלי.

משפט 7.7 תהי f זרימה מ־s ל־t ברשת זרימה (G,c,s,t). שלושת התנאים הבאים שקולים:

1. f היא זרימה מקסימלית.
2. לא קיים מסלול שיפור ביחס ל־f.

3. ב־G יש חתך s-t שקיבולו $\nu(f)$.

הוכחה. נוכיח .1 $\Leftarrow$ .2 $\Leftarrow$ .3 $\Leftarrow$ .1.

.1 $\Leftarrow$ .2 זה ברור, אחרת נוכל להגדיל את הזרימה.

.2 $\Leftarrow$ .3 תהי A קבוצת כל הצמתים $v \in V$ עבורם קיים מסלול מ־s ל־v ברשת השיורית G_f. כיוון שאין מסלול ברשת השיורית מ־s ל־t, אזי $s \in A$, $t \notin A$. לכל קשת $e = (u, v)$ היוצאת מ־A (כלומר $u \in A$, $v \notin A$) מתקיים $f(e) = c_e$, אחרת e הייתה גם קשת ברשת השיורית G_f והיינו יכולים "להאריך" ברשת השיורית את המסלול מ־s ל־u, עד ל־v, נובע מכך ש־$v \in A$, וזה עומד בסתירה להגדרת e. בדומה לכך, לכל קשת הנכנסת ל־A מתקיים $f(e) = 0$. מכאן:

$$\nu(f) = \sum_{e \text{ leaves } A} f(e) - \sum_{e \text{ enters } A} f(e) = \sum_{e \text{ leaves } A} c_e - 0 = c(A).$$

כלומר, A הוא חתך s-t בקיבול $\nu(f)$.

.3 $\Leftarrow$ .1 זוהי בדיוק מסקנה 7.6.

$\square$

נשים לב שהשקילות .2 $\Leftrightarrow$ .1 הינה תוכן משפט 7.2 כמו־כן, בהוכחת .2 $\Leftarrow$ .3 קיבלנו אלגוריתם לחישוב חתך מינימלי. למעשה, באופן דומה נוכל להסיק את הטענה הבאה:

טענה 7.8 תהי f זרימה מקסימלית כלשהי מ־s ל־t. ויהי A חתך s–t. אזי A חתך s–t מינימלי אם ורק אם אין קשת היוצאת מ־A ברשת השיורית G_f.

תרגיל 7.2

1. הוכיחו את טענה 7.8.
2. מצאו שני חתכי מינימום ברשת שאיור 7.4.

פתרון בעמוד 163

נחזור עתה על הוכחת משפט 7.3. הוכחת המשפט מורכבת משני צעדים: הצעד הראשון הוא להראות שבכל רשת זרימה, ערך כל זרימה הוא לכל היותר קיבול החתך המינימלי – זהו תוכן טענה 7.5.

הצעד השני הוא להראות שקיימת זרימה שערכה שווה לחתך המינמלי. את זה הראנו (עבור רשתות זרימה עם קיבולים שלמים), בהבחנה שהאלגוריתם של פורד־פלקרסון חייב לעצור, ועל־פי תנאי העצירה לא קיים מסלול שיפור ברשת בהשיורית, מכאן שעל־פי משפט 7.7 התקבלה זרימה שערכה שווה לקיבול החתך.

הערה: נשים לב שבניסוח משפט 7.3 לא הגבלנו את הרשת להיות בעלת קיבולים שלמים, ולכן המשפט נכון עבור כל קיבולים ממשיים אי־שליליים. הכללה זאת איננה חשובה לצרכי הקורס הנוכחי, אבל למשפט 7.3 ישנם גם שימושים לא אלגוריתמיים, בהם הצורה הכללית יותר הינה שימושית. ההוכחה שהוצגה לעיל מסתמכת על ההנחה שהקיבולים הם מספרים שלמים. בפרט, הטענה שהאלגוריתם של פורד־פלקרסון עוצר בזמן סופי איננה בהכרח נכונה כאשר הקיבולים הם מספרים ממשיים. בסעיף הבא נראה גירסה של האלגוריתם של פורד־פלקרסון שעוצר לאחר מספר סופי של צעדים שאיננו תלוי בקיבולים, ובזאת יוכח גם משפט 7.3 בניסוחו הכללי. ♣

תרגיל 7.3

1. הפונקציה $g : 2^V \to \mathbb{R}$ אשר מקנה לכל תת־קבוצה $X \subseteq V$ ערך ממשי $g(X) \in \mathbb{R}$ נקראת *תת־מודולרית* [submodular] אם לכל $X, Y \subseteq V$ מתקיים:

$$g(X) + g(Y) \geq g(X \cap Y) + g(X \cup Y)$$

הוכיחו כי פונקציית הקיבול $g(A) = c(A)$ היא תת־מודולרית.

2. הוכיחו כי אם X, Y הם חתכי s-t מינימליים, אז גם $X \cup Y$ ו־$X \cap Y$ הם חתכי s-t מינימליים.

פתרון בעמוד 165

תרגיל 7.4 נתונה רשת להעברת מידע, המיוצגת על־ידי גרף מכוון $G = (V, E)$; לכל כבל (קשת) e ברשת יש קיבול c_e של כמות המידע שאפשר להעביר דרכו ביחידת זמן. יש להעביר את המידע סימולטנית ממקור s לקבוצת יעדים T; לכל יעד $t \in T$ נתונה כמות הזרימה $b(t)$ שצריכה להגיע אליו ביחידת זמן. הציעו אלגוריתם פולינומיאלי שיחשב את זרימת המידע שיכולה לענות על הדרישות, או יקבע שזה בלתי אפשרי.

פתרון בעמוד 166

7.3 בחירת מסלולי שיפור טובים

קראו בספר את סעיף 7.3

בהנחה כי הקיבולים הם מספרים שלמים, אלגוריתם פורד־פולקרסון אמנם מחשב פתרון אופטימלי, אך זמן הריצה שלו אינו בהכרח פולינומיאלי, אלא פסאודו־פולינומיאלי בלבד (כלומר, זמן הריצה חסום על־ידי פולינום בערכי הקיבולים, במקום במספר הסיביות הנדרשות לייצוג הקיבולים). הדוגמה באיור 7.6 בספר מראה כי מצב זה אכן יכול לקרות. בסעיף הזה נדון בדרכים שונות לבחור מסלול שיפור באלגוריתם כדי לשפר את זמן הריצה שלו. נזכיר כי האלגוריתם הכללי של פורד־פולקרסון הוא:

Algorithm 7.2 Ford Fulkerson

```
Initialize f(e) ← 0, for all e ∈ E.
while the residual graph G_f contains an s → t path do
    Augment the flow along such a path P.
return f
```

התיאור לעיל אינו מפרט איזה מסלול P לבחור. הבחירה הטבעית ביותר (והחמדנית) היא למצוא בכל שלב בלולאה מסלול שיפור "רחב ביותר", כלומר, מסלול שצוואר־הבקבוק שלו $\varepsilon(P,f)$ הוא הגדול ביותר. לא קשה לתכנן אלגוריתם שבהינתן מספר Δ יבדוק אם יש מסלול שיפור P המקיים $\varepsilon(P,f) \geq \Delta$. כל שעלינו לעשות הוא פשוט לחפש מסלול בגרף $G_f(\Delta)$ המתקבל מהרשת השיורית G_f, על־ידי הורדת כל הקשתות עם קיבול שיורי שהוא קטן ממש מ־Δ. אם נמצא מסלול מ־s ל־t ב־$G_f(\Delta)$, אזי צוואר־הבקבוק של המסלול הזה יהיה לפחות Δ; אחרת – אין מסלול כזה. כמובן, כדי למצוא את הערך המקסימלי של Δ, אפשר לעבור על כל הערכים האפשריים בין 1 ל־C, כאשר C הוא הקיבול הגדול ביותר של קשת ברשת G. אבל זה שוב נותן אלגוריתם פסבדו־פולינומיאלי בלבד.

תרגיל 7.5 בהנחה כי כל הקיבולים הם מספרים שלמים, הראו כי אפשר למצוא מסלול שיפור עם צוואר־בקבוק מקסימלי בזמן $O(m \log \min\{m, C\})$, כאשר C הוא הקיבול הגדול ביותר של קשת ברשת, ו־m הוא מספר הקשתות ברשת.

פתרון בעמוד 166

אם כן, בכל שלב בלולאה של אלגוריתם פורד־פולקרסון, אנו יכולים למצוא מסלול שיפור P עם צוואר־בקבוק מקסימלי בזמן $O(m \log m)$. אבל בכך עוד לא הוכחנו כי זה נותן אלגוריתם פולינומיאלי, כי עדיין נראה שייתכן מצב שבו (לדוגמה) $\Delta = 1$ בכל שלב. השאלה היא, האם הגרסה המשופרת של אלגוריתם פורד־פולקרסון מאפשרת לחסום את מספר השלבים בלולאה? התשובה היא: כן, ואנו נוכיח זאת בתרגיל הבא.

תרגיל 7.6 נתבונן בגרסת "המסלול הרחב ביותר" של אלגוריתם פורד־פולקרסון. כלומר, בגרסה שבה האלגוריתם מוצא, בכל שלב בלולאה, את מסלול השיפור שיש לו צוואר־בקבוק מקסימלי. נסמן ב־ν_i את ערך הזרימה המקסימלית ברשת השיורית בשלב ה־i, ונסמן ב־ν את ערך הזרימה המקסימלית ברשת (שימו לב כי $\nu_0 = \nu$).

1. הראו כי $\nu_{i+1} \leq \nu_i(1 - 1/m)$.
2. הסיקו מהסעיף הקודם כי לאחר $\lceil m \log \nu \rceil$ שלבים $\nu_i < 1$.
3. הראו כי אם הקיבולים הם מספרים שלמים אזי מספר השלבים בלולאה הוא $O(m \log \nu)$.

פתרון בעמוד 166

מבחינת יעילותה, גרסת האלגוריתם של פורד־פולקרסון המוצגת בספר עולה על גרסת "המסלול הרחב ביותר". הטענה שבה משתמשים כדי לחסום את מספר השלבים היא:

טענה 7.9 (בספר זוהי טענה (7.18)) תהי f זרימה מ־s ל־t ברשת זרימה (G, c, s, t). אם צוואר־הבקבוק המקסימלי של מסלול שיפור הוא קטן ממש מ־Δ, אזי קיים ב־G חתך s-t A כך שמתקיים

$$\nu(f) \geq c(A) - \left[d^{\text{in}}(A) + d^{\text{out}}(A)\right] \cdot \Delta$$

כאשר $d^{\text{in}}(A)$ הוא מספר הקשתות הנכנסות ל־A ו־$d^{\text{out}}(A)$ הוא מספר הקשתות היוצאות מ־A. בפרט, ערך הזרימה המקסימלית ברשת הוא לכל היותר $\nu(f)+m\Delta$.

הוכחה. תהי A קבוצת כל הצמתים $v \in V$ עבורם קיים מסלול מ־s ל־v ב־$G_f(\Delta)$. כיוון שאין ב־$G_f(\Delta)$ מסלול מ־s ל־t, אזי $s \in A,\ t \in V \setminus A$. לכל קשת $e=(u,v)$ היוצאת מ־A מתקיים $c_e - f(e) < \Delta$, אחרת היינו יכולים "להאריך" ברשת השיורית את המסלול מ־s ל־u עד v. בדומה לכך, מתקיים $f(e) < \Delta$ לכל קשת הנכנסת ל־A . לכן:

$$\begin{aligned}\nu(f) &= \sum_{e \text{ leaves } A} f(e) - \sum_{e \text{ enters } A} f(e) > \sum_{e \text{ leaves } A} (c_e - \Delta) - \sum_{e \text{ enters } A} \Delta \\ &= c(A) - \left[d^{\text{in}}(A) + d^{\text{out}}(A)\right] \cdot \Delta.\end{aligned}$$

□

בתרגיל הבא נתאר גרסה נוספת של האלגוריתם שהוצג בספר.

תרגיל 7.7 נתבונן באלגוריתם הבא לחישוב הזרימה המקסימלית. נסמן $C = \max_{e \in E} c_e$. מתחילים עם $\Delta = C$ ו־$f = 0$; כל עוד יש מסלול מ־s ל־t ב־$G_f(\Delta)$, מעלים את ערך הזרימה ב־Δ לאורך מסלול זה. ברגע שאין מסלול כזה, אם $\Delta = 1$ מסיימים; אחרת מעדכנים $\Delta \leftarrow \lceil \Delta/2 \rceil$ וממשיכים באותו אופן.

1. הראו כי האלגוריתם נעצר ומחשב זרימה מקסימלית.
2. הראו כי מספר הערכים ש־Δ מקבל במהלך האלגוריתם הוא $O(\log C)$.
3. הראו כי מספר הפעמים שמחשבים מסלול שיפור, לכל ערך של Δ, הוא לכל היותר $2m$.

פתרון בעמוד 167

האלגוריתם שהובא בספר פועל לפי עיקרון דומה לזה שבתרגיל 7.7, בהבדל אחד: הוא מתחיל עם $\Delta_1 = 2^{\lfloor \log_2 C \rfloor}$, כלומר Δ_1 הוא המספר הגדול ביותר שהינו חזקה של 2 וגם קטן או שווה ל־$\max_{e \in E} c_e$.

7.3.1 אלגוריתם דיניץ/אדמונדס־קרפ

האלגוריתם שהוצג בסעיף הקודם הוא אמנם פולינומיאלי בגודל הקלט, אך הוא אינו בהכרח פולינומיאלי בגודל הרשת (כלומר במספר הצמתים והקשתות ברשת). בסעיף הנוכחי נדון באלגוריתם שזמן ריצתו הוא פולינומיאלי במספר הצמתים והקשתות ברשת. נזכיר כי גם באלגוריתמים שראינו לבעיית העץ הפורש המינימלי ולבעיית המסלולים הקצרים ביותר, היו מחירים/מרחקים על הקשתות, אבל מספר הפעולות הבסיסיות בהם היה תלוי במספר הצמתים והקשתות בלבד. אלגוריתמים כאלה נקראים אלגוריתמים פולינומיאליים במובן החזק.

הערה: טכנית, הזמן הדרוש לחיבור שני מספרים שלכל אחד מהם יש b סיביות, הוא $O(b)$, לכן פעולות חשבון דורשות זמן $O(\log C)$. כיוון שפעולות מסוג זה מופיעות תמיד באלגוריתמים בהם מעורבים מספרים, נהוג לספור רק את מספר הפעולות הבסיסיות באלגוריתם (חיבור, כפל, השוואה, וכולי) ולהתעלם ממשך הזמן הדרוש לכל פעולת חשבון בסיסית. ♣

אלגוריתם פולינומיאלי במובן החזק לבעיית זרימה מקסימלית מתואר בספר בסעיף 7.4. האלגוריתם הזה מדגים שיטות חשובות, אך הוא מסובך במקצת וגם שונה מהותית מאלגוריתם פורד־פולקרסון. במקומו, אנו רוצים להראות כי גם גרסה פשוטה יחסית של אלגוריתם פורד־פולקרסון יכולה להניב אלגוריתם פולינומיאלי במובן החזק. נציין כי סיבוכיות האלגוריתם הזה פחות טובה מהאלגוריתם שבסעיף 7.4.

אם נסתכל שוב על הדוגמאות בהן אלגוריתם פורד־פולקרסון מניב תוצאות גרועות, נבחין כי "ההתנהגות הרעה" של האלגוריתם נובעת משתי הסיבות הבאות:

- השתמשנו במסלולים שצוואר־הבקבוק שלהם קטן (ובכך טיפל כבר האלגוריתם שבספר);
- מסלולי השיפור היו יותר מדי ארוכים (ובכך ננסה לטפל בהמשך).

הרעיון הבא הוצע על־ידי פורד ופולקרסון כבר בשנות ה־50 של המאה הקודמת, ונותח באופן בלתי תלוי על־ידי שלושה חוקרים שונים. דיניץ [Dinitz] היה הראשון שהציע גרסה (משופרת) של האלגוריתם הרשום להלן; אדמונדס [Edmonds] יחד עם קרפ [Karp] ניתחו את האלגוריתם הזה, מאוחר יותר, באופן בלתי תלוי. הכלל באלגוריתם הזה הוא:

> בכל שלב באלגוריתם פורד־פולקרסון, מוצאים את מסלול השיפור הקצר ביותר ברשת השיורית G_f.

כאן האורך של המסלול הוא מספר הקשתות שיש בו (ללא קשר לקיבולים). כדי למצוא מסלול כזה ברשת השיורית אין צורך להשתמש באלגוריתם דייקסטרה, אפשר פשוט לבצע חיפוש לרוחב [BFS]. שימו לב כי האורך של כל מסלול הוא בין 1 ל־$n-1$. יותר מכך, האינטואיציה שלנו היא שאורכי מסלולי השיפור הקצרים ביותר הולכים ונעשים יותר ארוכים במהלך האלגוריתם, שהרי בכל שלב באלגוריתם נעלמת לפחות קשת אחת ממסלול קצר ביותר בגרף השיורי. בהמשך נוכיח את המשפט הבא.

משפט 7.10 אם בכל שלב באלגוריתם פורד־פולקרסון בוחרים מסלול שיפור קצר ביותר, אזי מספר השלבים הוא $O(mn)$, ולכן, גרסה זו של האלגוריתם ניתנת ליישום בזמן $O(m^2n)$.

הוכחת משפט 7.10 מסתמכת על שתי טענות שמראות כיצד אורכי מסלולי השיפור הקצרים ביותר גדלים במהלך האלגוריתם. תהי G_i הרשת השיורית בשלב ה־i, כאשר $G_0 = G$. עבור צומת v יהי $\mathrm{dist}_i(v)$ האורך של המסלול הקצר ביותר מבחינת מספר הקשתות מ־s ל־v ברשת השיורית G_i; כלומר, $\mathrm{dist}_i(v)$ היא הרמה של v בעץ הסריקה לרוחב של G_i המושרש ב־s. האבחנה הראשונה היא שהמרחקים $\mathrm{dist}_i(v)$ יכולים רק לעלות עם הזמן.

טענה 7.11 $\mathrm{dist}_{i+1}(v) \geq \mathrm{dist}_i(v)$ לכל $v \in V$ ולכל i.

הוכחה. ההוכחה היא באינדוקציה על $\mathrm{dist}_{i+1}(v)$.

בסיס האינדוקציה: אם $\mathrm{dist}_{i+1}(v) = 0$ אז $v = s$ ולכן גם $\mathrm{dist}_i(v) = 0$.

צעד האינדוקציה: נניח שהטענה נכונה עבור $\mathrm{dist}_{i+1}(v) \leq d-1$ ונוכיח אותה עבור $\mathrm{dist}_{i+1}(v) = d$, $d \geq 1$. יהי $s \to \ldots \to u \to v$ המסלול הקצר ביותר מ־s ל־v ב־G_{i+1} (אם אין מסלול כזה, אז $\mathrm{dist}_{i+1}(v) = \infty$, ובמקרה כזה ברור שהטענה

נכונה). כיוון שלקחנו מסלול קצר ביותר, נקבל $\mathrm{dist}_{i+1}(v) = \mathrm{dist}_{i+1}(u) + 1$. כמו כן, על־פי הנחת האינדוקציה $\mathrm{dist}_{i+1}(u) \geq \mathrm{dist}_i(u)$.

נתבונן בשני מקרים: אם (u,v) היא קשת גם ב־G_i, אז

$$\mathrm{dist}_i(v) \leq \mathrm{dist}_i(u) + 1 \leq \mathrm{dist}_{i+1}(u) + 1 = \mathrm{dist}_{i+1}(v),$$

כנדרש. מצד שני, אם אין קשת (u,v) ב־G_i, אזי הייתה קשת (v,u) במסלול השיפור שבו השתמשנו במעבר מ־G_i ל־G_{i+1}, והקיבול השיורי של הקשת (v,u) ב־G_i היה שווה לצוואר־הבקבוק של מסלול השיפור. במקרה זה, במעבר מ־G_i ל־G_{i+1} הקשת (v,u) "נעלמה" והקשת ההפוכה לה (u,v) "הופיעה". בפרט, (v,u) נמצאת על המסלול הקצר ביותר מ־s ל־t ב־G_i. מכאן נקבל

$$\mathrm{dist}_i(v) = \mathrm{dist}_i(u) - 1 < \mathrm{dist}_i(u) + 1 \leq \mathrm{dist}_{i+1}(u) + 1 = \mathrm{dist}_{i+1}(v).$$

□

שימו לב כי במקרה השני, שבו הקשת (v,u) נעלמה מ־G_i ובמקומה הופיעה הקשת (u,v) ב־G_{i+1}, יכולנו להוכיח כי $\mathrm{dist}_{i+1}(v) \geq \mathrm{dist}_i(v) + 2$. כלומר, במקרה הזה המרחק מ־s ל־v ממש גדל. האבחנה השנייה היא כי אותה קשת לא יכולה "להיעלם" יותר מדי פעמים מהרשת השיורית במהלך האלגוריתם.

טענה 7.12 במהלך האלגוריתם, כל קשת (u,v) יכולה להיעלם לכל היותר $n/2$ פעמים מהרשת השיורית.

הוכחה. נניח כי עבור האינדקסים $i < j$, הקשת (u,v) נמצאת ברשתות השיוריות G_i, G_{j+1} אבל היא אינה נמצאת באף אחת מרשתות הביניים $G_{i+1}, \ldots, G_j$. אז בהכרח:

- (u,v) הייתה במסלול השיפור שבו השתמשנו במעבר מ־G_i ל־G_{i+1}. לכן $\mathrm{dist}_i(v) = \mathrm{dist}_i(u) + 1$.
- (v,u) הייתה במסלול השיפור שבו השתמשנו במעבר מ־G_j ל־G_{j+1}. לכן $\mathrm{dist}_j(v) = \mathrm{dist}_j(u) - 1$.

מן מהטענה הקודמת נקבל:

$$\mathrm{dist}_j(u) = \mathrm{dist}_j(v) + 1 \geq \mathrm{dist}_i(v) + 1 = \mathrm{dist}_i(u) + 2.$$

לסיכום, בין ההיעלמות וההופעה של אותה קשת (u,v), המרחק בין s ל־u גדל לפחות ב־2. כיוון שהמרחק בין s לכל צומת רלוונטי u הוא לכל היותר $n-1$, מספר ההיעלמויות יכול להיות לכל היותר $n/2$. □

כעת אנו יכולים לחסום את מספר השלבים באלגוריתם. כיוון שכל קשת יכולה להיעלם לכל היותר $n/2$ פעמים, מספר ההיעלמויות יגיע לכל היותר ל־ $mn/2$. אבל בכל שלב, יש לפחות קשת אחת שנעלמת (הבהירו לעצמכם מדוע). לכן מספר השלבים הוא לכל היותר $O(mn)$, כפי שרצינו להוכיח.

7.4 האלגוריתם דחיפת קדם־זרימה למציאת זרימה מקסימלית*

סעיף זה אינו חלק מחומר הלימוד. קראו אותו אם ברצונכם ללמוד אלגוריתם שונה לגמרי לחישוב זרימה מקסימלית.

קראו בספר את סעיף 7.4

אלגוריתם פורד־פולקרסון שייך למשפחה של אלגוריתמים שמתחילים עם פתרון אפשרי ובכל צעד משפרים את "האופטימליות" שלו. לעומת זאת, באלגוריתם שנתאר בסעיף הזה, מתחילים עם "פתרון" בלתי אפשרי, שערכו גדול מהאופטימלי, ובכל צעד משפרים את "האפשריות" של הפתרון, על־ידי ויתור מסוים בערך שלו.

קדם־זרימה f היא "כמעט" זרימה, פרט לעובדה שחוק שימור הזרימה אינו מתקיים לכל צומת $v \in V \setminus \{s,t\}$; במקום "זרימה נכנסת שווה לזרימה יוצאת" מתקיים "זרימה נכנסת גדולה או שווה לזרימה היוצאת". לכל צומת v, ההפרש בין "הזרימה הנכנסת לבין הזרימה היוצאת" הוא העודף [excess] $e_f(v)$ של הזרימה בצומת v. הרשת השיורית של קדם־זרימה $G_f = (V, E_f)$ מוגדרת בדיוק באותו אופן כמו הרשת השיורית של הזרימה. בנוסף לקדם־זרימה f האלגוריתם מתחזק גם תגים/גבהים $h(v)$ של צומתי הגרף. הגבהים h וקדם־זרימה f תואמים אם מתקיימים שני התנאים האלה:

$$(1.)\ \ h(t) = 0, h(s) = n$$
$$(2.)\ \ h(v) \leqslant h(w) + 1 \ \forall (v,w) \in E_f$$

הסיבה לשני התנאים האלה מסוכמת בטענה הבאה – שבספר מופיעות כטענות (7.21), (7.22):

טענה 7.13 אם קדם־זרימה f והגבהים h הם תואמים, אזי אין מסלול מ־s ל־t ברשת השיורית. בפרט, אם f היא זרימה, אזי היא הזרימה המקסימלית.

אם כן, ברור כבר מה יעשה האלגוריתם. הוא יתחיל עם הקדם־זרימה והגבהים הטבעיים:

$$f(e) = \begin{cases} c_e & e \text{ leaves } s \\ 0 & \text{otherwise,} \end{cases}.$$

$$h(v) = \begin{cases} n & v = s \\ 0 & \text{otherwise.} \end{cases}$$

קל לוודא כי הזרימה הזו והגבהים האלה תואמים.

כל עוד יש צומת $v \in V \setminus \{s,t\}$ עם עודף $e_f(v) > 0$ חיובי ממש, האלגוריתם מפעיל אחת משתי הפעולות הבאות:

דחיפה [push]. אם יש קשת $(v,w) \in E_f$ עם $h(w) < h(v)$ אז "דוחפים" (קדימה או אחורה) זרימה בקשת המתאימה ב־E :

- אם (v,w) היא קשת קדימה, השינוי הוא העלאת $f(v,w)$ בשיעור $\min\{e_f(v), c_e - f(v,w)\}$.

- אם (v,w) היא קשת אחורה, השינוי הוא הורדת $f(v,w)$ בשיעור $\min\{e_f(v), f(w,v)\}$.

תיוג מחדש [**relabel**]**.** אם קיים $v \in V$ עבורו $h(w) \geq h(v)$ לכל קשת $(v,w) \in E_f$ אז מעדכנים $h(v) \leftarrow h(v)+1$.

לא קשה לוודא כי f ו־h ממשיכים להיות תואמים במהלך האלגוריתם. כמו־כן, כאשר האלגוריתם נעצר, $e_f(v) = 0$ לכל $v \in V \setminus \{s,t\}$, כלומר f היא זרימה. מכאן נקבל, על־סמך הטענה הקודמת את הטענה הזו – טענה (7.24) בספר:

טענה 7.14 כאשר האלגוריתם נעצר f היא זרימה מקסימלית.

אם־כן, השאלה היחידה היא כמה צעדי דחיפה ותיוג מחדש מתבצעים עד שהאלגוריתם נעצר. זה תלוי, בין היתר, בבחירת הצומת v עליו מפעילים את הפעולות האלה. אבל גם במקרה הגרוע, אפשר לחסום זאת בדרך הזו – בספר אלה הטענות (7.26)-(7.29):

טענה 7.15 במהלך האלגוריתם מתבצעות לכל היותר $O(n^2)$ פעולות תיוג מחדש ו־$O(n^2m)$ פעולות דחיפה.

לא נחזור כאן על ההוכחות, שהן פשוטות למדי, אבל שימו לב כי בספר, ב־(7.29) מוכח החסם $4n^2m$ ולא $2n^2m$.

חסמים טובים יותר מתקבלים כאשר בכל צעד בוחרים מבין הצמתים שבהם $e_f(v) > 0$ את הצומת שהגובה שלו מקסימלי. במקרה זה נקבל כי מספר פעולות הדחיפה הוא $O(nm + n^3) = O(n^3)$ (נובע מ־(7.28) ו־(7.30) בספר).

7.5 יישום ראשון: בעיית הזיווג הדו־צדדי

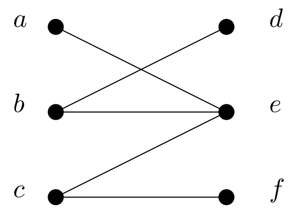
קראו בספר את סעיף 7.5

גרף דו־צדדי $G = (X \cup Y, E)$ (X ו־Y קבוצות זרות) הוא גרף לא־מכוון שבו יש לכל קשת $e \in E$, קצה אחד ב־X וקצה שני ב־Y. זיווג M בגרף לא־מכוון $G = (V,E)$, הוא אוסף של קשתות M ($M \subseteq E$) כך שאין ב־M שתי קשתות בעלות קצה משותף. ראו דוגמה באיור 7.6.

בעיה אלגוריתמית: בעיית זיווג מקסימלי בגרף דו־צדדי.
הקלט: גרף דו־צדדי G.
הפלט: זיווג M ב־G.
המטרה: להביא למקסימום את גודלו של M.

פותרים את בעיית הזיווג המקסימלי בגרף דו־צדדי $G = (X \cup Y, E)$ על־ידי רדוקציה פשוטה לבעיית הזרימה המקסימלית ברשת G' עם קיבולי קשתות 1, באופן הבא:

אלגוריתם למציאת זיווג מקסימלי בגרפים דו־צדדיים



איור 7.6: גרף דו־צדדי שבו $X = \{a, b, c\}$ ו־$Y = \{d, e, f\}$. קבוצת הצמתים $\{\{b, d\}, \{b, e\}\}$ אינה זיווג, מפני שהצומת b הוא קצה של שתי קשתות בקבוצה זו. הקבוצה $\{\{a, e\}, \{b, d\}, \{c, f\}\}$ והקבוצה $\{\{b, e\}, \{c, f\}\}$ הן זיווגים.

1. מכוונים את כל הקשתות מ־X ל־Y.
2. מוסיפים: מקור s וקשת מכוונת מ־s לכל צומת ב־X; בור t וקשת מכוונת מכל צומת ב־Y ל־t.
3. ברשת הזרימה המתקבלת G', שבה קיבולי כל הקשתות הם 1, מחשבים את הזרימה המקסימלית f במספרים שלמים (לדוגמה, בעזרת פורד־פולקרסון).
4. הזיווג המקסימלי הוא $M = \{e \in E : f(e) = 1\}$.

הוכחת הנכונות של האלגוריתם מסתמכת על האבחנה כי יש התאמה בין זרימות שלמות מ־s ל־t ב־G' לבין זיווגים ב־G. כלומר, לכל זיווג M ב־G יש זרימה שערכה $|M|$ ב־G', וגם לכל זרימה שלמה f מ־s ל־t ב־G' יש זיווג M ב־G בגודל $\nu(f)$. ההוכחה לכך היא פשוטה מאוד ולא נחזור עליה. שימו לב, האלגוריתם הוא עבור גרפים דו־צדדיים בלבד. נעיר כי לבעיית הזיווג המקסימלי יש אלגוריתם פולינומיאלי גם בגרפים כלליים, אבל אלגוריתם זה והוכחתו מסובכים, ולא ננתח אותו כאן.

הגדרה 7.5 (כיסוי־בקשתות [edge-cover]**)** כיסוי־בקשתות בגרף לא־מכוון G הוא קבוצת קשתות F כך שכל צומת של G הוא קצה של איזושהי קשת ב־F. ♢

הגדרה 7.6 (כיסוי־בצמתים [vertex-cover]**)** כיסוי־בצמתים בגרף לא־מכוון G הוא קבוצת צמתים U כך שאחד מקצותיה של כל קשת ב־G, נמצא ב־U. ♢

תרגיל 7.8 נסמן ב־$\eta(G)$ את הגודל המקסימלי של זיווג ונסמן ב־$t(G)$ את הגודל המינימלי של כיסוי־בקשתות בגרף G.

1. הוכיחו כי לכל גרף $G = (V, E)$ ללא צמתים מבודדים מתקיים: $t(G) = |V| - \eta(G)$.
2. הציעו אלגוריתם בעל סיבוכיות $O(nm)$ המחשב כיסוי־בקשתות שיהיה הקטן ביותר בגרף דו־צדדי.

פתרון בעמוד 167

תרגיל 7.9 הוכיחו כי לכל זיווג M ולכל כיסוי־בצמתים U בגרף G כלשהו (לאו דווקא דו־צדדי) מתקיים $|M| \leq |U|$.

פתרון בעמוד 168

מטרת התרגיל הבא היא להוכיח את המשפט הזה:

משפט 7.16 (משפט קניג [Konig's Theorem]) בגרף דו־צדדי G, הגודל המקסימלי של זיווג שווה לגודל המינימלי של כיסוי־בצמתים, כלומר:

$$\max\{|M| : M \text{ is a matching in } G\} = \min\{|U| : U \text{ is a vertex-cover in } G\}.$$

תרגיל 7.10 יהי $G = (X \cup Y, E)$ גרף דו־צדדי. ברשת G' שנבנית באלגוריתם למציאת הזיווג המקסימלי, נשנה את הקיבולים של הקשתות ב־E כך שיהיו $m+1$. כל אחת מן הקשתות היוצאות מ־s ומן הקשתות הנכנסות ל־t היא בעלת קיבול 1, כמו בבנייה המקורית. הראו כי ערך הזרימה המקסימלית ב־G' שווה לגודל הזיווג המקסימלי בגרף הדו־צדדי המקורי G. כעת יהי A חתך s-t מינימלי ב־G'. הוכיחו כי:
1. אין קשת מכוונת מ־$X \cap A$ ל־$Y \setminus A$.
2. קבוצת הצמתים $U = (X \setminus A) \cup (Y \cap A)$ היא כיסוי־בצמתים ב־G.
3. בגרף G', מספר הקשתות שיוצאות מ־A הוא בדיוק $|U|$.
4. הוכיחו את משפט 7.16.

פתרון בעמוד 168

תרגיל 7.11 באי טריגמיה מותר לכל גבר להתחתן עם 3 נשים לכל היותר (אבל לכל אישה מותר להתחתן רק עם גבר אחד). נתונה רשימת זוגות E (גבר־אישה) של אנשי האי שמוכנים להתחתן עם זולתם. הציעו אלגוריתם פולינומיאלי המביא למקסימום את מספר הנשים שאפשר לחתן. כלומר, הפלט של האלגוריתם יהיה רשימת זוגות $E' \subseteq E$ שבה כל גבר יהיה רשום ב־3 זוגות לכל היותר ב־E' ואילו כל אישה תירשם בזוג אחד לכל היותר.

פתרון בעמוד 169

7.6 מסלולים זרים בגרפים מכוונים ובלתי מכוונים

קראו בספר את סעיף 7.6

אחת מטענות המפתח בסעיף זה היא טענה (7.42) בספר: "ברשת זרימה עם קיבולי 0/1 וזרימה במספרים שלמים שערכה ν, קבוצת הקשתות עם זרימה 1, מכילה ν מסלולים זרי־קשתות מ־s ל־t." אנו נוכיח במדריך את משפט 7.19, המכליל את הטענה שלעיל.

נקבע רשת זרימה (G, c, s, t). וזרימה $f : E \to [0, \infty)$ ברשת. אנו נשאף להראות שכל זרימה ניתנת לפרוק (כלומר לכתיבה כסכום) של זרימות לאורך מסלולי s–t ועוד זרימות לאורך מעגלים, שאינן תורמות לערך הזרימה. באיור 7.7 מודגמות זרימת מסלול וזרימת מעגל. נגדיר עתה פורמלית את סוגי הזרימות הללו.

הגדרה 7.7 (זרימת מסלול) יהי P מסלול פשוט ומכוון ברשת הזרימה מ־s ל־t, ונניח ש־$0 < \varepsilon \le c_e$ לכל $e \in P$. במצב הזה, הזרימה

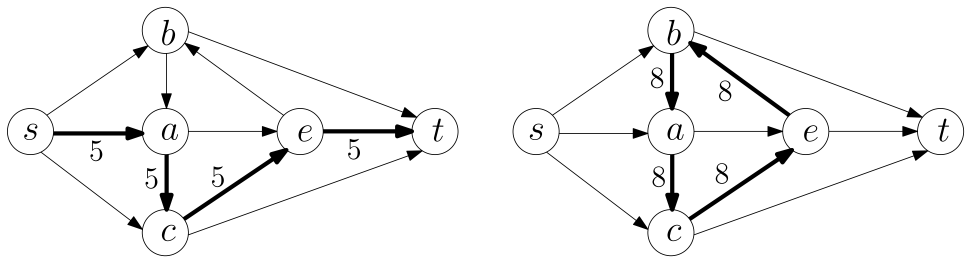
$$f(e) = \begin{cases} \varepsilon & \text{if } e \in P \\ 0 & \text{otherwise.} \end{cases}$$

נקראת זרימת מסלול (עם ערך ε). ראו זרימת מסלול לדוגמה באיור 7.7 משמאל.
$\diamondsuit$

הגדרה 7.8 (זרימת מעגל) יהי C מעגל מכוון ברשת הזרימה, ונניח ש־$c_e \geq \varepsilon > 0$ לכל $e \in C$. במצב הזה, הזרימה

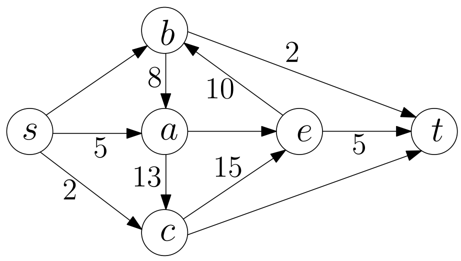
$$f(e) = \begin{cases} \varepsilon & \text{if } e \in C \\ 0 & \text{otherwise.} \end{cases}$$

נקראת *זרימת מעגל* (עם ערך ε). ראו זרימת מעגל לדוגמה באיור 7.7 מימין. ♢



איור 7.7: רשת זרימה (הקיבולים אינם מצויינים באיור). משמאל מוצגת זרימת מסלול שערכה 5, ומימין מוצגת זרימת מעגל שערכה 8.

הגדרה 7.9 (פרוק זרימה למעגלים ומסלולים) נאמר כי הזרימה f *ניתנת לפירוק לזרימות מסלול וזרימות מעגל*, אם קיימת משפחה Π של מסלולים פשוטים מ־s ל־t ומעגלים מכוונים, ומשקלות חיוביים ממש $\varepsilon : \Pi \to (0, \infty)$ כך שלכל קשת $e \in E$ מתקיים: סכום משקלי המעגלים והמסלולים העוברים דרך e הוא בדיוק $f(e)$. כלומר: $\sum_{P \in \Pi,\, e \in P} \varepsilon_P = f(e)$. לדוגמה, באיור 7.8 מוצגת זרימה הניתנת לפירוק לזרימות המסלול והמעגל מאיור 7.7 ועוד זרימת מסלול אחת. ♢



איור 7.8: רשת זרימה (הקיבולים אינם מצויינים באיור) וזרימה שערכה 7 ברשת. זרימה זאת ניתנת לפירוק לזרימת המסלול וזרימת המעגל המופיעות באיור 7.7; ולזרימת מסלול שערכה 2 במסלול (s, c, e, b, t).

טענה 7.17 אם זרימה f ניתנת לפירוק לזרימות מסלול וזרימות מעגל Π כמו בהגדרה 7.9, אזי ערך הזרימה f הוא בדיוק סכום ערכי המסלולים ב־Π, ואילו תרומת המעגלים לערך הזרימה היא אפס.

הוכחה. ערך הזרימה מוגדר כדלקמן:

$$\nu(f) = f^{\text{out}}(s) - f^{\text{in}}(s) = \sum_{e \text{ leaves } s} f(e) - \sum_{e \text{ enters } s} f(e).$$

מסלול $P \in \Pi$ עם ערך ε_P, הוא מסלול פשוט מ־s ל־t ולכן יש בו בדיוק קשת אחת היוצאת מ־s, ואין בו קשת הנכנסת ל־s. לכן המסלול תורם ε_P לערך הזרימה. במעגל, לעומת זאת, מספר הקשתות הנכנסות ל־s שווה למספר הקשתות היוצאות מ־s ולכן המעגל תורם 0 לערך הזרימה. $\square$

עבור זרימה f נסמן ב־$E(f) = \{e \in E : \ f(e) > 0\}$ את קבוצת הקשתות עם זרימה חיובית ממש.

טענה 7.18 תהי f זרימה. אם $E(f) \neq \emptyset$ אז בגרף $(V, E(f))$ יש מעגל פשוט או מסלול פשוט מ־s ל־t.

הוכחה. נתבונן בגרף G' המתקבל מהגרף $(V, E(f))$ על ידי הוספת הקשת (t, s). על־פי חוק שימור הזרימה, הגרף G' מקיים את התכונה הבאה. לכל $(u, v) \in E(f)$ יש ב־G' קשת הנכנסת ל־u וגם יש קשת היוצאת מ־v. מכך ניתן להסיק (עשו זאת) כי G' מכיל מעגל פשוט C. אם C מכיל את הקשת (t, s) שהוספנו, אז $C \setminus \{(t, s)\}$ הוא מסלול מ־s ל־t ב־$E(f)$. אחרת, C הוא מעגל ב־$E(f)$. $\square$

משפט 7.19 (פירוק זרימה [Flow Decomposition]) כל זרימה f ניתנת לפרוק ל־$|E(f)|$ זרימות מסלול וזרימות מעגל לכל היותר. בנוסף, אם הזרימה היא במספרים שלמים, אזי קיים פירוק כזה שבו ערכי הזרימות הם מספרים שלמים.

הוכחה. ההוכחה היא באינדוקציה על $|E(f)|$ – מספר הקשתות ב־$E(f)$. אם $E(f) = \emptyset$, אזי מוזרם 0 בכל הקשתות והטענה ברורה. נניח אם כן ש־$E(f) \neq \emptyset$. על פי טענה 7.18 יש בגרף $(V, E(f))$ מעגל או מסלול פשוט מ־s ל־t, נניח P. יהי $\varepsilon_P = \min_{e \in P} f(e)$. תהי f' הזרימה המתקבלת על־ידי הקטנת $f(e)$ ב־ε_P לכל $e \in P$, כלומר

$$f'(e) = \begin{cases} f(e) - \varepsilon_P & \text{if } e \in P \\ f(e) & \text{otherwise.} \end{cases}$$

הפונקציה f' הינה זרימה חוקית:

- **אילוצי הקיבול:** לכל $e \in E$, $f'(e) \leq f(e) \leq c_e$. כמו־כן, לכל $e \in E \setminus P$, $f'(e) = f(e) \geq 0$, ועבור $e \in P$,

$$f'(e) = f(e) - \min_{e' \in C} f(e') \geq f(e) - f(e) = 0.$$

- **שימור הזרימה:** יהי $v \in V \setminus \{s, t\}$. אם $v \notin P$, אזי $f'(e) = f(e)$ לכל קשת e הנכנסת ל־v או יוצאת מ־v, ולכן הזרימה נשמרת כפי שהיא נשמרת ב־f. אם $v \in P$, אזי הזרימה הנכנסת ל־v והזרימה היוצאת מ־v יורדות שתיהן ב־ε_P. בשני המקרים חוק שימור הזרימה נשמר.

עתה נוכיח ש־$|E(f')| \le |E(f)| - 1$. כיוון ש־$0 \le f'(e) \le f(e)$ לכל $e \in E$, אנו מסיקים ש $E(f') \subseteq E(f)$. כמו כן קיימת קשת $e_0 \in P$, המקיימת $e_0 = \arg\min_{e \in P} f(e)$, כלומר $f(e_0) = \varepsilon_P$. מכאן ש־$f'(e_0) = f(e_0) - \varepsilon_p = 0$, ולכן $e_0 \in E(f) \setminus E(f')$. אנו מסיקים ש־$|E(f')| \le |E(f)| - 1$.

לפי הנחת האינדוקציה, אפשר לפרק את f' ל־$|E(f')|$ זרימות מסלול וזרימות מעגל לכל היותר. נוסיף את P עם המשקל ε_P ונקבל פירוק של f כנדרש. □

באיור 7.9 מודגם תהליך הפירוק של זרימה למסלולים ומעגלים כפי שמתואר בהוכחה שלעיל. נשים לב שהוכחה איננה מגדירה פירוק יחיד – הפירוק תלוי במעגל או המסלול הנבחר בכל רגע. בפרט הפירוק המתקבל באיור 7.9 שונה מהפירוק המתואר באיורים 7.8 ו־ 7.7.

תרגיל 7.12 הוכיחו את הטענה הבאה: נניח כי מתקיים $c_e = 1$ לכל קשת $e \in E$ ברשת זרימה $(G = (V, E), c, s, t)$. בתנאים אלה קיימת זרימה בגודל k, מ־s ל־t, אם ורק אם יש k מסלולים זרים בקשתות מ־s ל־t.

פתרון בעמוד 169

כל הטענות בהמשך הסעיף נובעות כמעט באופן מיידי ממשפט פירוק הזרימה שהוכחנו לעיל. המשפט הכי חשוב שמוכח בסעיף הזה הוא:

משפט 7.20 (משפט מנגר [Menger's Theorem]) (בספר זוהי טענה (7.45)) *בגרף מכוון, המספר המקסימלי של מסלולים מ־s ל־t, שכל שניים מהם זרים בקשתות, שווה למספר הקשתות המינימלי שלאחר השמטתן מהגרף לא קיים מסלול מ־s ל־t.*

הוכחה. נייחס לכל קשת קיבול 1. המשפט נובע משתי האבחנות האלה:

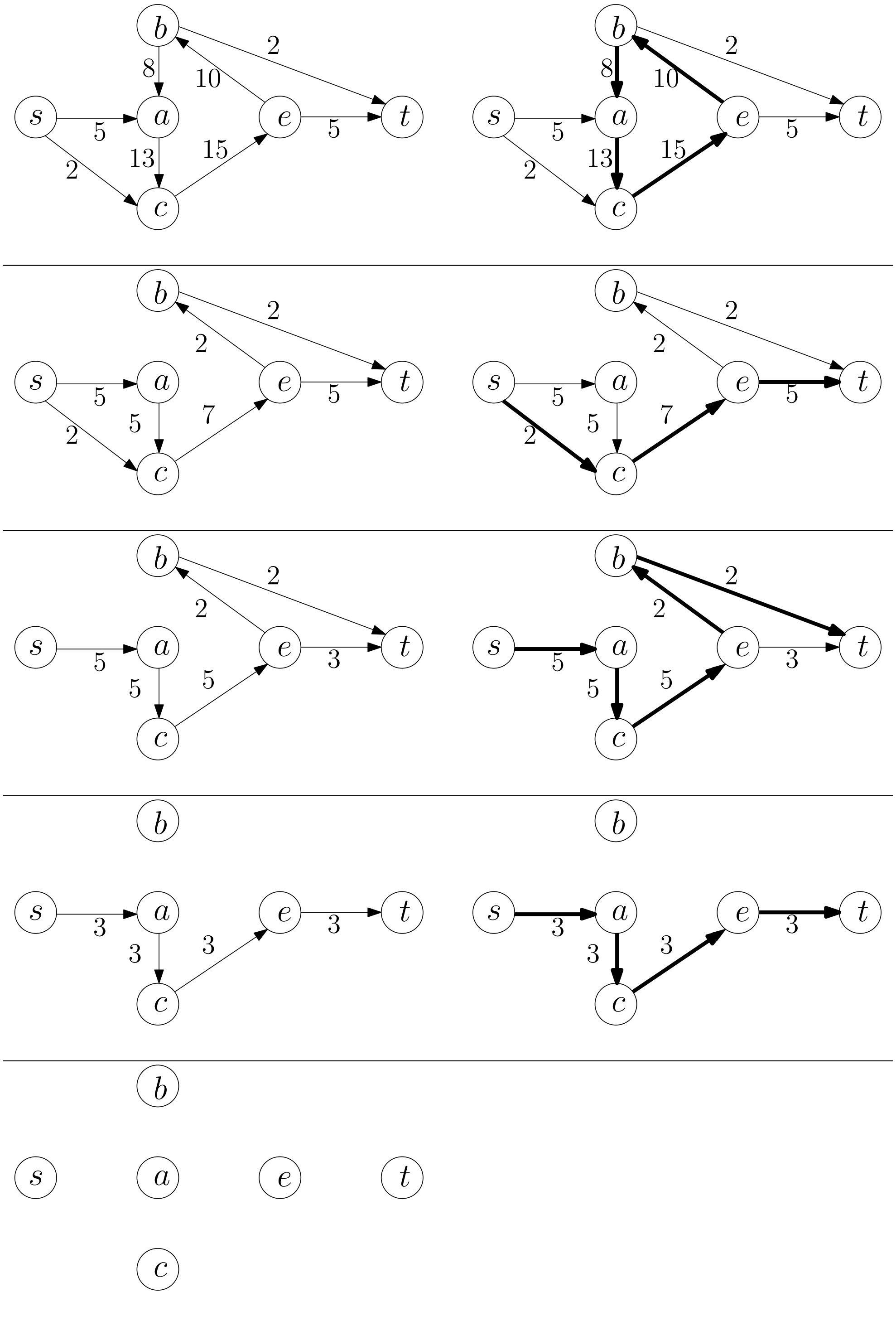
(א) המספר המינימלי של קשתות שלאחר השמטתן מהגרף לא יהיה מסלול מ־s ל־t, שווה למספר הקשתות בחתך המינימום.

(ב) המספר המקסימלי של מסלולים זרים בקשתות מ־s ל־t, שווה לגודל הזרימה המקסימלית.

המשפט נובע מיידית מאבחנות (א) ו־(ב) וממשפט זרימה־מקסימלית וחתך־מינימלי. את אבחנה (ב) כבר הוכחנו בתרגיל 7.12.

נוכיח את אבחנה (א). יהי $G = (V, E)$ הגרף הנתון. תהי F קבוצת קשתות בעלת גודל מינימלי כך שבגרף $(V, E \setminus F)$ לא קיים מסלול מ־s ל־t. נתבונן בגרף $(V, E \setminus F)$ בקבוצת הצמתים A אליהם קיים מסלול מ־s. במקרה כזה $s \in A,\ t \in V \setminus A$, ובגרף $(V, E \setminus F)$ אין קשת היוצאת מ־A. לכן בגרף G מתקיים $c(A) \le |F|$. בכיוון השני, יהי A חתך s-t מינימלי ב־G, ותהי F קבוצת הקשתות היוצאות מ־A. אזי לפי ההגדרה $|F| = c(A)$. כמו־כן, לפי ההגדרה, אין קשת היוצאת מ־A בגרף $(V, E \setminus F)$, ולכן אין מסלול מ־s ל־t בגרף $(V, E \setminus F)$. משום־כך, $c(A)$ הוא לפחות מספר הקשתות המינימלי הנדרש, שהסרתן מנתקת את כל המסלולים מ־s ל־t. לכן $c(A) \ge |F|$. □

המשפט נכון גם לגרפים לא־מכוונים – הוכחת הטענה בגרף לא מכוון נובעת ישירות מהטענה לגרפים מכוונים והבניה הסטנדרטית של החלפת כל קשת לא מכוונת בשתי קשתות אנטי מקבילות. להלן גרסת "הצמתים" של משפט מנגר:



איור 7.9: תהליך פרוק זרימה של הזרימה מאיור 7.8. משמאל מתוארת הזרימה שעוד יש לפרק (רק הקשתות עם זרימה גדולה מ־0), ומימין מודגש המעגל או המסלול הנבחר.

משפט 7.21 (גרסת הצמתים של משפט מנגר) בגרף מכוון שאין בו קשת מ־s ל־t, המספר המקסימלי של מסלולים מ־s ל־t, שכל שניים מהם זרים בצמתים, שווה למספר המינימלי של צמתים ב־$V \setminus \{s,t\}$ שלאחר השמטתם מהגרף אין מסלול מ־s ל־t (והשמטת צומת גוררת גם השמטת כל הקשתות הסמוכות אליו).

תרגיל 7.13 תארו אלגוריתם פולינומיאלי הפותר את בעיית הזרימה המקסימלית מ־s ל־t עם קיבולים $c : V \setminus \{s,t\} \to (0,\infty)$ על הצמתים (ראו גם תרגיל 7.13 בספר). כאן אילוצי הקיבול הם כי $f^{\text{in}}(v) \le c_v$ לכל $v \in V \setminus \{s,t\}$.

פתרון בעמוד 170

תרגיל 7.14 הוכיחו את משפט 7.21. (רמז: היעזרו בתרגיל 7.13).

פתרון בעמוד 170

תרגיל 7.15 (ראו גם תרגיל 7.32 בספר) נתון גרף לא־מכוון $G = (V,E)$. עבור $s,t \in V$ נסמן ב־$\lambda(s,t)$ את המספר המקסימלי של מסלולים זרים (בזוגות) בקשתות בין s ל־t.

1. הוכיחו כי לכל שלושה צמתים שונים $u,v,w \in V$ מתקיים אי־השוויון הזה: $\lambda(u,w) \ge \min\{\lambda(u,v), \lambda(v,w)\}$.
2. הראו דוגמה שבה מתקיים אי־שוויון ממש.
3. יהי k מספר טבעי כלשהו; נגדיר את היחס הבא בין צומתי V: $(u,v) \in R$ אם קיימים k מסלולים זרים בקשתות בין u ל־v. הוכיחו כי היחס R הוא טרנזיטיבי.

פתרון בעמוד 171

7.7 הרחבות לבעיית הזרימה המקסימלית

קראו בספר את סעיף 7.7

בעיה אלגוריתמית: בעיית ההפצה [Circulation Problem].
הקלט: רשת זרימה G,c וביקושים (מספרים שלמים, שיכולים להיות שליליים) $d : V \to \mathbb{R}$.
שאלה: האם קיימת פונקציה $f : E \to \mathbb{R}$ כך שמתקיים:

$$0 \le f(e) \le c_e, \quad \forall e \in E \qquad \text{(אילוצי הקיבול)} \qquad (7.3)$$

$$f^{\text{in}}(v) - f^{\text{out}}(v) = d_v, \quad \forall v \in V \qquad \text{(תנאי ביקוש)} \qquad (7.4)$$

פונקציה המקיימת את שני התנאים האלה נקראת הפצה עבור הביקושים d.
שימו לב, הבעיה לעיל היא בעיית החלטה ולא בעיית אופטימיזציה. קל לראות כי תנאי הכרחי לקיום הפצה אפשרית הוא $\sum_{v \in V} d_v = 0$, אך זה אינו תנאי מספיק.

באמצעות רדוקציה קלה אפשר להמיר את בעיית ההפצה לבעיית הזרימה המקסימלית. נחלק את צומתי הגרף לשלוש קבוצות:

- $S = \{s \in V :\ d_s > 0\}$ הם צומתי הביקוש;

- $T = \{t \in V : \ d_t < 0\}$ הם צומתי ההיצע;
- $V \setminus (S \cup T) = \{v \in V : \ d_v = 0\}$ הם צומתי המעבר.

נוסיף צומת חדש s^* (המקור) וקשת (s^*, s) בקיבול d_s לכל צומת ביקוש $s \in S$;
נוסיף צומת חדש t^* (הבור) וקשת (t, t^*) בקיבול $-d_t$ לכל צומת היצע $t \in T$.
ברשת הזרימה המתקבלת עם מקור s^* ובור t^* מחשבים את הזרימה המקסימלית.
אם ערך הזרימה המקסימלית הוא $\sum_{s \in S} d_s = -\sum_{t \in T} d_t$, אזי יש הפצה אפשרית;
אחרת – אין הפצה כזו. ההוכחה והניתוח פשוטים מאד, ולא נחזור עליהם כאן.

תרגיל 7.16 הוכיחו כי לבעיית ההפצה יש פתרון אפשרי אם ורק אם $\sum_{s \in S} d_s = -\sum_{t \in T} d_t$, ולכל $A \subseteq V$ מתקיים

$$c(A) \geq \sum_{s \in S \cap A} d_s + \sum_{t \in T \cap A} d_t.$$

פתרון בעמוד 172

"סיבוך" נוסף מתקבל כאשר לכל קשת $e \in E$ יש גם חסם תחתון על הזרימה שצריכה לעבור בה, כלומר, תנאי הקיבול הם $\ell_e \leq f(e) \leq c_e$ לכל $e \in E$. גם למקרה זה יש רדוקציה קלה לבעיית הזרימה המקסימלית שלא נחזור עליה כאן.

7.8 יישומים: תכנון סקרים

קראו בספר את סעיף 7.8

הבעיה והרדוקציה שלה לבעיית הזרימה המקסימלית הם פשוטים מאוד, ולא נחזור עליהם כאן.

7.9 תזמון טיסות

קראו בספר את סעיף 7.9

הבעיה והרדוקציה שלה לבעיית הזרימה המקסימום הם פשוטים מאוד, ולא נחזור עליהם כאן.

7.10 הקטעת תמונות

קראו בספר את סעיף 7.10

הבעיה שצריך לפתור כאן שונה מהבעיות שהוצגו בסעיפים 7.8 ו־7.9, כי כאן הרדוקציה היא לבעיה של חתך המינימום. למרות זאת איננו רואים צורך לחזור על תוכן הסעיף.

אך בתרגיל הבא תוכלו לראות יישום חשוב נוסף לבעיית חתך המינימום.

הגדרה 7.10 צפיפות של גרף (לא־מכוון) היא היחס בין מספר הקשתות לבין מספר הצמתים בו, כלומר מחצית הדרגה הממוצעת של הגרף. $\diamondsuit$

מטרת התרגיל הבא היא לפתח אלגוריתם פולינומיאלי לבעיה הבאה:

בעיה אלגוריתמית: תת־גרף צפוף ביותר [Densest Subgraph].
הקלט: גרף לא־מכוון $G = (V, E)$.
הפלט: תת־גרף של G.
המטרה: להביא למקסימום את הצפיפות של תת־הגרף.

עבור $X \subseteq V$ וקבוצת קשתות E על V נסמן ב־$E[X]$ את קבוצת הקשתות ב־E עם שני הקצוות ב־X. בסימונים אלה, הבעיה שלנו היא למצוא $X \subseteq V$ כך שהיחס $|E[X]|/|X|$ יהיה גדול ככל האפשר.

תרגיל 7.17 (תרגיל רשות ברמת קושי גבוה)

1. נניח כי לכל מספר רציונאלי q אנו יכולים לחשב בזמן פולינומיאלי את הפונקציה

$$h(q) = \max_{\emptyset \neq X \subseteq V} \left(|E[X]| - q \cdot |X| \right)$$

ואת קבוצת הצמתים עבורה מתקבל המקסימום. הראו כי אז יש גם אלגוריתם פולינומיאלי לבעיית תת־הגרף הצפוף ביותר.

2. יהי $H = (V \cup E, F)$ *גרף הסמיכויות* של הקשתות והצמתים של G, כלומר, H הוא גרף דו־צדדי שקבוצת הצמתים שלו היא $V_H = V \cup E$, ויש קשת בין $v \in V$ ל־$e \in E$ ב־F אם v הוא קצה של e (ב־E). (הדרך הפשוטה לקבל את H מ־G היא פשוט על־ידי "השתלה" של צומת בכל קשת של G). תהי J רשת זרימה המתקבלת מ־H באופן הזה:
   - מכוונים את כל הקשתות של H מ־V ל־E, ומקצים קיבול 1 לכל קשת.
   - מוסיפים מקור s וקשת (s, v) בקיבול q לכל $v \in V$.
   - מוסיפים בור t וקשת (e, t) בקיבול 1 לכל $e \in E$.

עבור $X \subseteq V$ תהי $A(X) = \{s\} \cup (V \setminus X) \cup (E - E[X])$. הראו כי $A(X)$ הוא חתך s-t ב־J שקיבולו $c(A(X)) = |E| - (|E[X]| - q \cdot |X|)$.

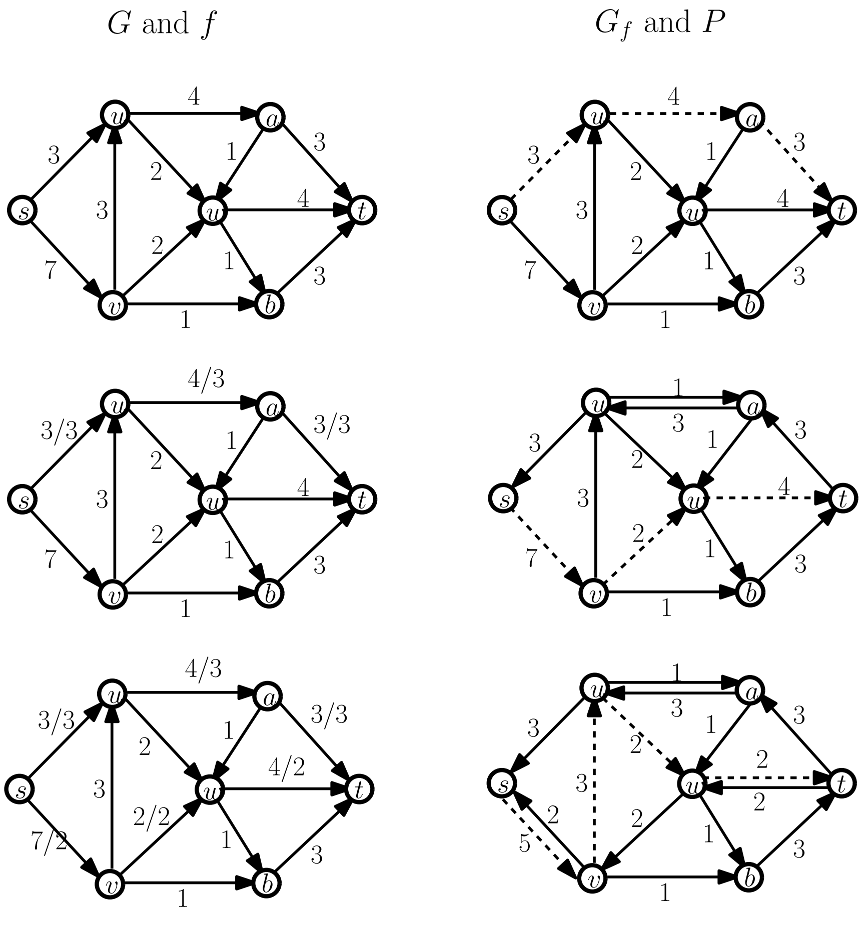
3. מבין כל חתכי s-t המינימליים ב־J, יהי A חתך מינימלי ב־J כך ש־$|A|$ הוא הגדול ביותר. הראו כי אם $v \in A$ עבור $v \in V$, אזי $e \in A$ לכל קשת e הסמוכה ל־v ב־G. כמו כן תארו אלגוריתם פולינומיאלי למציאת A כנ"ל.

4. נסמן $X_q = V \setminus A$. הסיקו מן הסעיף השלישי כי $A = A(X_q)$ ולכן $h(q) = |E[X_q]| - q \cdot |X_q|$. הסיקו מכך כי לכל q אכן ניתן לחשב בזמן פולינומיאלי את $h(q) = \max_{\emptyset \neq X \subseteq V} \left(|E[X]| - q \cdot |X| \right)$.

פתרון בעמוד 173

7.11 פתרונות לתרגילים

פתרון תרגיל 7.1 מעמוד 143



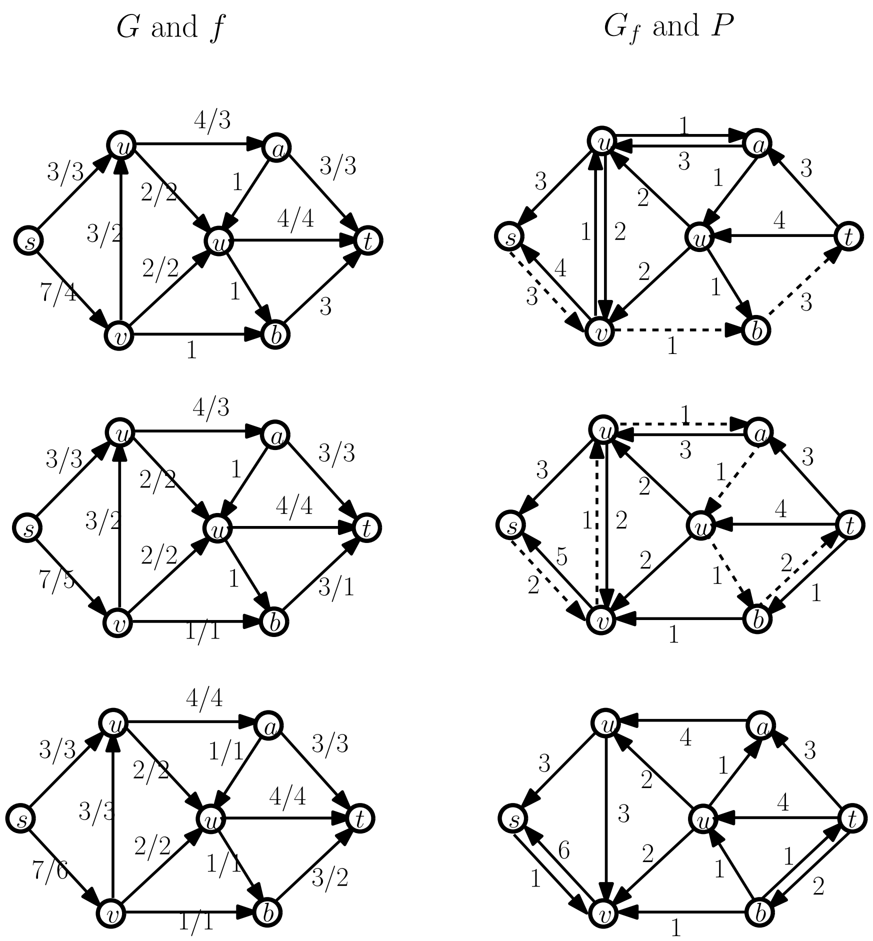
איור 7.10: הרצת פורד־פולקרסון על הרשת שהוצגה באיור 7.4 – חלק א.

איור 7.10 ואיור 7.11 מדגימים את כל השלבים באלגוריתם. יש כמובן גם אפשרויות אחרות. ערך הזרימה שמייצר האלגוריתם הוא 9. נציין כי בדוגמה זו לא נעשה שימוש בקשתות אחורה. ♣

פתרון תרגיל 7.2 מעמוד 146

1. נניח תחילה שב־G_f אין קשת היוצאת מ־A. מכאן שלכל קשת $e = (u, v)$ היוצאת מ־A מתקיים $f(e) = c_e$, בדומה לכך, לכל קשת הנכנסת ל־A מתקיים $f(e) = 0$. מכאן:

$$\nu(f) = \sum_{e \text{ leaves } A} f(e) - \sum_{e \text{ enters } A} f(e) = \sum_{e \text{ leaves } A} c_e - 0 = c(A).$$



איור 7.11: הרצת פורד־פולקרסון על הרשת שהוצגה באיור 7.4 (חלק ב).

כלומר, A הוא חתך s-t בקיבול $\nu(f)$, ומטענה 7.5 הוא חתך מינימלי.

בכיוון השני, נניח ש־A הוא חתך s–t מינימלי. תהי B קבוצת הצמתים הנגישם מ־s ב־G_f. כפי שראינו בהוכחת 2. ⇐ 3. , B הוא חתך מינימלי וקיבולו הוא $c(B) = \nu(f)$, מכאן, ש־$c(A) = c(B) = \nu(f)$, כלומר

$$\sum_{e \text{ leaves } A} f(e) - \sum_{e \text{ enters } A} f(e) = \nu(f) = c(A) = \sum_{e \text{ leaves } A} c_e.$$

כיוון ש־$0 \le f(e) \le c(e)$ לכל $e \in E$, השוויון לעיל יכול לקרות רק כאשר $f(e) = c(e)$ לכל קשת e היוצאת מ־A, ו־$f(e) = 0$ לכל קשת הנכנסת ל־A. ומכאן שב־G_f אין קשתות היוצאות מ־A.

הערה: בחלק השני נמנענו מלהפעיל את משפט זרימה־מקסימלית וחתך מינימלי, כיוון שלא הוכחנו את החלק הנדרש ממנו במדריך עד עתה (בעצם, חלק זה הוכח בתרגיל הזה), אך כמובן שאין מניעה עקרונית מלהשתמש במשפט זה ובכך לקצר במקצת את ההוכחה.

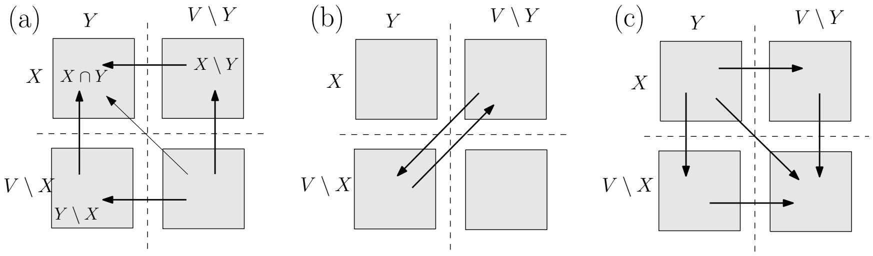
2. נתבונן ברשת השיורית האחרונה. קל לוודא כי כל אחד מהחתכים $A_1 = \{s,u,v\}$, $A_2 = \{s,u,v,a\}$, $A_3 = \{s,u,v,a,w\}$ הוא חתך מינימלי ברשת בקיבול 9. ♣

פתרון תרגיל 7.3 מעמוד 147

נסמן ב־$c(A,B)$ את הקיבול של הקשתות שהולכות מ־A ל־B. נראה כי:

$$\begin{aligned}&c(X)+c(Y)\\&\quad= c(X\cap Y)+c(X\cup Y)+c(X\backslash Y, Y\backslash X)+c(Y\backslash X, X\backslash Y)\end{aligned} \tag{7.5}$$

וכיוון שכל הקיבולים הם אי־שליליים, זה גורר כי פונקציית הקיבול היא תת־מודולרית. כדי להראות את (7.5), נתבונן בתרומות של כל סוגי הקשתות האפשריים לשני האגפים של (7.5). אנו טוענים כי התרומה של כל קשת לאגפי (7.5) היא זהה. באיור 7.12 הקשתות מחולקות לשלושה סוגים.



איור 7.12: הקבוצות X, $V\setminus X$ הן "השורות" והקבוצות Y, $V\setminus Y$ הן "העמודות".

קל לוודא כי תרומת הקשתות בחלק (a) של איור 7.12 לכל אחד מאגפי (7.5) היא 0. נתבונן בתרומת הקשתות בחלק (b) של איור 7.12. הקשתות שהולכות מלמעלה למטה תורמות את קיבולן ל־$c(X)$ באגף שמאל של (7.5), ולאיבר $c(X\setminus Y, Y\setminus X)$ באגף ימין. הקשתות שהולכות מלמטה למעלה תורמות את קיבולן ל־$c(Y)$ באגף שמאל של (7.5), ולאיבר $c(Y\setminus X, X\setminus Y)$ באגף ימין. התרומה של הקשתות האלה לאיבר $c(X\cap Y)+c(X\cup Y)$ היא 0.

לבסוף, נתבונן בתרומת הקשתות בחלק (c) של איור 7.12. תרומת הקשתות שאינן אלכסוניות ל־$c(X)+c(Y)$ זהה לתרומתן ל־$c(X\cap Y)+c(X\cup Y)$, בעוד שתרומתן לשאר האיברים של (7.5) היא 0. לדוגמה, הקשתות שהולכות מ־$X\cap Y$ ל־$Y\setminus X$ תורמות את קיבולן רק ל־$c(X)$ ול־$c(X\cap Y)$. הקשתות האלכסוניות שהולכות מ־$X\cap Y$ ל־$V\setminus(X\cup Y)$ תורמות לשני האגפים של (7.5) תרומה בגודל כפול מן הקיבול שלהן.

סעיף 2 נובע מסעיף 1. יהיו X,Y חתכי s-t מינימליים, ונסמן את קיבולם כך: $\nu = c(X) = c(Y)$. כיוון ש־X,Y הם חתכי s-t, מתקיים $s\in X, t\in V\setminus X$ וגם $s\in Y, t\in V\setminus Y$. מכאן נובע כי $s\in X\cap Y, t\in V\setminus(X\cap Y)$ וגם $s\in X\cup Y, t\in V\setminus(X\cup Y)$ (הבהירו לעצמכם שאכן זה כך), ולכן גם $X\cap Y$ ו־$X\cup Y$ הם חתכי s-t. בפרט, $c(X\cup Y)\geq\nu$ ו־$c(X\cap Y)\geq\nu$. מכאן, שאפשר

להשתמש בתת־מודולריות של פונקציית הקיבול כדי לקבל:

$$\nu + \nu = c(X) + c(Y) \geqslant c(X \cap Y) + c(X \cup Y) \geqslant \nu + \nu.$$

ולכן, כל אי־השוויונים מתקיימים כשוויונים, ובפרט $c(X \cap Y) = c(X \cup Y) = \nu$.
♣

פתרון תרגיל 7.4 מעמוד 147
נבנה רשת זרימה (G', c', s, t') באופן הזה: הגרף G' מתקבל מ־G על ידי הוספת צומת בור חדש t', והוספת קשת (t, t') בקיבול $b(t)$ לכל $t \in T$. קל לראות כי יש זרימה שעונה על הדרישות אם ורק אם ערך הזרימה המקסימלית ברשת המתקבלת הוא $B = \sum_{t \in T} b(t)$. אם קיימת ב־G' זרימה f' שערכה B, אזי הזרימה המתאימה f ב־G היא הצמצום של f' לקשתות G. סיבוכיות האלגוריתם זהה לסיבוכיות אלגוריתם מציאת הזרימה המקסימלית. ♣

פתרון תרגיל 7.5 מעמוד 148
נציג שני אלגוריתמים: אחד עם סיבוכיות $O(m \log C)$ והשני עם סיבוכיות $O(m \log m)$.

אלגוריתם עם סיבוכיות $O(m \log C)$. כאן בא לעזרתנו רעיון פשוט מאוד, שאתם בוודאי מכירים בשם "חיפוש בינארי", או בשם "חיפוש האריה במדבר". אין צורך לעבור על כל ערכי Δ – אפשר להפעיל חיפוש בינארי באופן הזה: באיטרציה הראשונה בודקים אם קיים מסלול שיפור עם $\varepsilon(P, f) \geq \Delta$ עבור $\Delta = \lceil C/2 \rceil$. אם כן, אנו מסיקים שצריך לחפש את Δ המקסימלית בתחום שבין $\lceil C/2 \rceil$ ל־C; אחרת, תחום החיפוש הוא בין 1 ל־$\lceil C/2 \rceil - 1$. ובאופן דומה ממשיכים בשאר האיטרציות, על־ידי חציית תחום החיפוש ובדיקה באיזה תחום נמצא הערך המבוקש. מספר האיטרציות הדרושות כדי למצוא את הערך המבוקש הוא $O(\log C)$, וכל איטרציה ניתן ליישם בזמן $O(m)$. הסיבוכיות הכוללת היא $O(m \log C)$, כלומר פולינומיאלית בגודל הקלט.

אלגוריתם עם סיבוכיות $O(m \log m)$. גם כאן נחפש בעזרת חיפוש בינארי. תחילה נמיין את הקשתות לפי הקיבול השיורי שלהן. בכל שלב נעדכן את התחום $[\ell, h)$ של ערך קיבול־הסף שבכל הקשתות מעליו יש מסלול מ־s ל־t. בהתחלה $\ell = 0$, $h = C + 1$. נסמן ב־$\#[\ell, h)$ את מספר הקשתות שקיבולן השיורי הוא בטווח $[\ell, h)$. בסעיף הקודם בחרנו לבדוק בכל שלב את הערך $\lceil (\ell+h)/2 \rceil$. בסעיף הזה נמצא את הערך p כך ש־$\#[\ell, p)$ ו־$\#[p, h)$ הם (בערך) מחצית $\#[\ell, h)$. זאת ניתן לעשות על־ידי מציאת החציון, ולכן בזמן לינארי. כמו בסעיף הקודם, נבדוק בשלב הזה אם יש מסלול מ־s ל־t בקשתות שמשקלן לפחות p. אם כן, נעדכן $\ell \leftarrow p$; אחרת נשים $h \leftarrow p$, ונמשיך כך עד אשר $h - \ell \leq 1$. בצורה זו נזדקק ל־$O(\log m)$ איטרציות. שימו לב, הזמן הדרוש לכל איטרציה הוא רק $O(m)$, ובזאת סיימנו. ♣

פתרון תרגיל 7.6 מעמוד 148
1. נתבונן ברשת השיורית G' בשלב i (לפני השיפור), כאשר ערך הזרימה המקסימלית היה עדיין ν_i. נסמן ב־c'_e את קיבול הקשת e ברשת שיורית זו. יהי P מסלול שיפור רחב ביותר שמוצאים בשלב i, כלומר, שיפור הזרימה לאורך P מקטין

את הזרימה המקסימלית ברשת השיורית מ־ν_i ל־ν_{i+1}. תהי e קשת צוואר־הבקבוק ב־P. ההוכחה נובעת משתי האבחנות האלה:

א. $\nu_{i+1} \le \nu_i - c'_e$;

ב. $c'_e \ge \nu_i/m$.

כדי להוכיח את א. נתבונן באיזשהו חתך מינימלי A ב־G'. על־פי המשפט זרימה־מקסימלית וחתך־מינימלי, $\nu_i = c'(A)$. כאשר אנו משפרים את הזרימה לאורך P ב־c'_e, אנו מורידים ברשת השיורית G' את קיבולי הקשתות ששייכות ל־P ב־c'_e. עקב כך, קיבול החתך A קטן לפחות ב־c'_e, ועל־ידי שימוש חוזר במשפט זרימה־מקסימלית וחתך־מינימלי נקבל $\nu_{i+1} \le c'(A) - c'_e = \nu_i - c'_e$, כנדרש.

כדי להוכיח את ב., תהי A קבוצת הצמתים ב־G' שאפשר להגיע אליהם מ־s על־גבי קשתות שקיבולן השיורי גדול ממש מ־c'_e. לפיכך $s \in A, t \in V \setminus A$ (הבהירו לעצמכם כי אכן זה כך), ולכן A הוא חתך s-t. קיבול כל קשת היוצאת מ־A ברשת השיורית G' הוא לכל היותר c'_e, לכן $c'(A) \le mc'_e$. כעת נפעיל את המשפט זרימה־מקסימלית וחתך־מינימלי ונקבל כי $\nu_i \le c'(A) \le mc'_e$, כנדרש.

2. על־פי סעיף 1, $\nu_i \le \nu(1-1/m)^i$. לכן מספיק להוכיח כי $\nu(1-1/m)^{m\ln\nu} <$ 1. לשם כך נשתמש באי־שוויון ידוע $(1-1/m)^m < e^{-1}$ (כאן e אינה קשת, אלא בסיס הלוגריתם הטבעי). לפיכך נקבל

$$\nu\left(1-1/m\right)^{m\ln\nu} = \nu\left(\left(1-1/m\right)^m\right)^{\ln\nu} < \nu e^{-\ln\nu} = 1.$$

3. על־פי סעיף 2, ערך הזרימה המקסימלית ברשת השיורית אחרי $i = \lceil m\ln\nu \rceil$ שלבים, יהיה קטן ממש מ־1, כלומר יהיה 0, שהרי כל הקיבולים הם שלמים. כיוון שערך הזרימה המקסימלית ברשת השיורית הוא 0, הזרימה ברשת המקורית היא מקסימלית. ♣

פתרון תרגיל 7.7 מעמוד 149

הסעיפים 1 ו־2 הם מיידים. כיוון שערך הזרימה עולה בכל שלב, האלגוריתם נעצר ומחשב את הזרימה המקסימלית. בקורס מבני נתונים בוודאי נוכחתם כבר כי מספר הערכים ש־Δ מקבל במהלך האלגוריתם הוא $O(\log C)$.

משום־כך נוכיח כאן בפירוט רק את סעיף 3. נסמן ב־q את מספר הפעמים שמחשבים עבור Δ את מסלול השיפור באיטרציה. עבור $i \ge 1$ נסמן ב־Δ_i את הערך של Δ באיטרציה ה־i. נסמן $\nu_0 = 0$ ונסמן ב־ν_i את ערך הזרימה בסוף השלב ה־i בדיוק לפני שמבצעים את העדכון $\Delta_{i+1} \leftarrow \lceil \Delta_i/2 \rceil$ (כלומר כאשר כבר אין מסלול מ־s ל־t ב־$G_f(\Delta)$). נסמן ב־ν את הערך של הזרימה המקסימלית. כיוון ש $\Delta_{i-1} \le 2\Delta_i$, אנו מקבלים:

- $\nu \ge \nu_i = \nu_{i-1} + q\Delta_i$, כאשר q הוא מספר הפעמים שהעלינו את הזרימה באיטרציה ה־i.
- $\nu \le \nu_{i-1} + m\Delta_{i-1} \le \nu_{i-1} + 2m\Delta_i$, על פי הטענה לעיל.

לכן $\nu_{i-1} + q\Delta_i \leqslant \nu \leqslant \nu_{i-1} + 2m\Delta_i$, ובפרט $q \le 2m$. ♣

פתרון תרגיל 7.8 מעמוד 154

1. תחילה נוכיח כי $t\left(G\right) \leqslant |V| - \nu\left(G\right)$. בהינתן זיווג M, נתאר אלגוריתם

המחשב כיסוי־בקשתות F בגודל $|F| = |V| - |M|$. מספר הצמתים שאינם קצוות של קשת ב־M הוא $|V| - 2|M|$. עבור כל צומת שאינו קצה של קשת ב־M נוסיף קשת כלשהי שסמוכה אליו, ונקבל כיסוי־בקשתות שגודלו

$$|M| + (|V| - 2\,|M|) = |V| - |M|\ .$$

בפרט, אם M הוא זיווג מקסימלי, נקבל כיסוי־בקשתות F בגודל $|V| - \nu(G)$. לכן $t(G) \leq |V| - \nu(G)$.

כעת נוכיח כי $t(G) \geq |V| - \nu(G)$. יהי F כיסוי־בקשתות בעל גודל מינימלי, ויהי M זיווג בעל גודל מקסימלי בגרף (V, F). לפיכך נקבל:

$$t\,(G) = |F| = |M| + (|V| - 2\,|M|) = |V| - |M| \geqslant |V| - \nu\,(G)\,.$$

2. תיארנו בסעיף 1. של תרגיל זה אלגוריתם שבהינתן זיווג מקסימלי M מחשב כיסוי־בקשתות F בגודל

$$|F| = |V| - |M| = |V| - \nu\,(G)\,.$$

על פי סעיף 1, זהו כיסוי־בקשתות קטן ביותר. קל לראות כי אפשר ליישם את כל שלבי האלגוריתם בזמן $O(m)$. לכן הזמן הדומיננטי מושקע בחישוב הזיווג המקסימלי, חישוב שאפשר לבצע בזמן $O(mn)$. ♣

פתרון תרגיל 7.9 מעמוד 154

מהעובדה שדרושים לפחות $|M|$ צמתים כדי "לכסות" את קשתות M בלבד, נובע כי לפחות קצה אחד של כל קשת של M צריך להיות ב־U. ♣

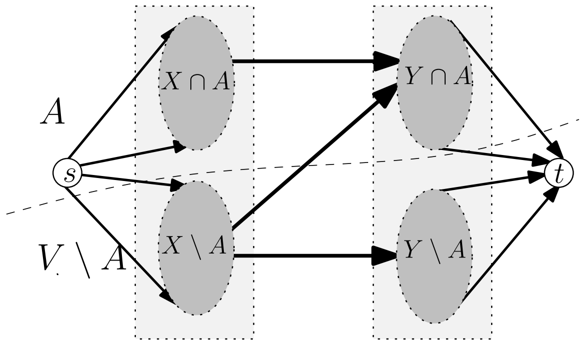
פתרון תרגיל 7.10 מעמוד 155

הנה אבחנה פשוטה: גם אם ניתן לקשתות ב־E קיבולים "אינסופיים", זה לא ישנה את ערך הזרימה המקסימלית, כי בכל קשת $(x, y) \in E$ עדיין תוכל לעבור לכל היותר רק יחידת זרימה אחת. בנוסף נציין כי כמות הזרימה שיכולה להיכנס לצומת $x \in X$ או לצאת מצומת $y \in Y$ גם היא לכל היותר 1. הסיבה למגבלה הזו היא: לכל צומת $x \in X$ נכנסת בדיוק קשת אחת (הקשת (s, x)) וקיבולה של קשת זו הוא 1, ומכל צומת $y \in Y$ יוצאת בדיוק קשת אחת (הקשת (y, t)) וקיבולה של קשת זו הוא 1, לכן, אם f היא זרימה מקסימלית עם ערכים שלמים ב־G', אזי $M = \{e \in E :\ f(e) = 1\}$ הוא עדיין זיווג מקסימלי בגרף הדו־צדדי המקורי G.

כעת נתבונן בחתך s-t מינימלי, A, ב־G'.

נוכיח את סעיף 1. קיבול החתך A הוא לכל היותר m, כי ב־G' יש חתך שקיבולו m, למשל החתך $\{s\}$ או החתך $V \setminus \{t\}$. כיוון שהקיבול של כל קשת ב־E הוא $m + 1$, ואילו הקיבול של A הוא m לכל היותר, לכן אין קשת של E היוצאת מ־A. אבל הקשתות של E שיוצאות מ־A הן בדיוק הקשתות מ־$X \cap A$ ל־$Y \setminus A$, ולכן אין קשתות כאלה – ראו איור 7.13, ושימו לב כי הקשתות מ־$X \setminus A$ ל־$Y \cap A$ נכנסות ל־A, ולא יוצאות מ־A. זה מוכיח את סעיף 1.

מכאן נובעת בקלות גם ההוכחה של סעיף 2: כל קשת ב־G סמוכה לצומת בקבוצה $U = (X \setminus A) \cup (Y \cap A)$, כי קצה אחד של כל קשת שאינה סמוכה לצומת ב־U נמצא ב־$X \cap A$ וקצה שני נמצא ב־$Y \setminus A$ – אבל הראינו שאין קשתות כאלה.



איור 7.13:

נוכיח כעת את סעיף 3. הקשתות שיוצאות מ־A הן בדיוק הקשתות שמובילות מ־s ל־$X \setminus A$, או מ־$Y \cap A$ ל־t (ראו איור 7.13). מספר הקשתות האלה הוא בדיוק $|U| = |Y \cap A| + |X \backslash A|$.

כעת נוכל להסיק את משפט קניג. יהי M זיווג מקסימלי ב־G. על־פי תרגיל 7.9, מספיק להראות כי ב־G יש כיסוי־בצמתים בגודל $|M|$. יהי A חתך מינימלי ב־G', ותהי $U = (X \backslash A) \cup (Y \cap A)$. לפי הסעיפים הקודמים, U היא כיסוי־בצמתים ב־G בגודל $|U| = c(A)$. כפי שראינו, $|M|$ שווה לערך הזרימה המקסימלית ב־G', ולכן, על־פי המשפט זרימה־מקסימלית וחתך־מינימלי, $|M| = c(A)$. קיבלנו כי ב־G יש כיסוי־בצמתים U שגודלו כגודל הזיווג המקסימלי $|M|$ – כנדרש. ♣

פתרון תרגיל 7.11 מעמוד 155

באמצעות רדוקציה של הבעיה שהוצגה בשאלה, נוכל להמירה לבעיית הזיווג המקסימלי בגרף דו־צדדי. תהי X קבוצת הגברים, Y קבוצת הנשים. תחילה נבנה גרף דו־צדדי $G = (X \cup Y, E)$, שבו קבוצת הקשתות של הגרף היא רשימת הזוגות E של (גבר, אישה) מאנשי האי שמוכנים להתחתן עם זולתם. כדי לבטא את העובדה שלכל גבר מותר להתחתן עם 3 נשים לכל היותר, נבנה מהגרף G גרף דו־צדדי $G' = (X' \cup Y, E')$ באופן הזה: נחליף כל צומת $x \in X$ ב־3 צמתים x_1, x_2, x_3 וכל קשת xy נחליף ב־3 קשתות $(x_1, y), (x_2, y), (x_3, y)$. כעת נמצא זיווג מקסימלי M בגרף G'. עבור $1 \leq i \leq 3$, הקשת $(x_i, y) \in M$, מייצגת את הזוג (x, y). כיוון שלכל צומת $x \in X$ מתאימים ב־G' 3 צמתים, כל גבר יופיע ב־3 זוגות לכל היותר. מספר הנשים המזווגות הוא מקסימלי, כי M הוא זיווג מקסימלי. סיבוכיות האלגוריתם היא כשל בעיית מציאת זיווג מקסימלי בגרף דו־צדדי. ♣

פתרון תרגיל 7.12 מעמוד 158

נוכיח כל כיוון בנפרד.

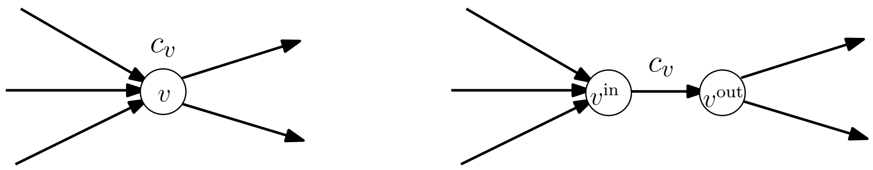
הכיוון הראשון: יש k מסלולים זרים בקשתות מ־s ל־t ⇐ קיימת זרימה בגודל k מ־s ל־t.

נגדיר $f(e) = 1$ לכל קשת השייכת לאחד המסלולים, ו־$f(e) = 0$ לכל קשת אחרת. ברור שתנאי הקיבול מתקיימים. וגם תנאי שימור הזרימה מתקיימים – כי אם P הוא מסלול פשוט מ־s ל־t, ואם לכל צומת $v \in V \setminus \{s,t\}$ יש קשת של P שנכנסת ל־v, אזי יש גם קשת של P שיוצאת מ־v. כמו־כן, ערך הזרימה הוא k מפני שהמסלולים הם זרים בקשתות.

הכיוון השני: קיום זרימה בגודל k מ־s ל־t ⇐ יש k מסלולים זרים בקשתות מ־s ל־t. זוהי מסקנה כמעט מיידית ממשפט פירוק הזרימה. כיוון שכל הקיבולים הם 1, אזי על־פי משפט פירוק הזרימה, כל זרימה f ניתנת לפירוק ל־$\nu(f)$ מסלולים במשקל 1 כל אחד, ואולי יש מעגלים נוספים. אם נתעלם מהמעגלים בפירוק, נקבל כי לכל זרימה בגודל k אפשר להתאים k מסלולים זרים בקשתות מ־s ל־t. ♣

פתרון תרגיל 7.13 מעמוד 160

הרדוקציה הפשוטה הבאה ממירה קיבולי צמתים לקיבולי קשתות. נניח שנתונה לנו רשת זרימה $G=(V,E),c$ שבה $c:V\setminus\{s,t\}\to[0,\infty)$ הם קיבולי הצמתים. כל צומת $v\in V\setminus\{s,t\}$ מוחלף בשני צמתים $v^{\text{in}}, v^{\text{out}}$ המחוברים ביניהם בקשת $(v^{\text{in}},v^{\text{out}})$ בקיבול c_v; כמו־כן, מחליפים את הראש v של כל קשת (u,v) שנכנסת ל־v בצומת v^{in}, ומחליפים את הזנב u של כל קשת (u,v) שיוצאת מ־u בצומת u^{out} (לקשתות אלה לא יהיו אילוצי קיבול). ראו איור 7.14 להמחשה.



איור 7.14: המרת קיבולי צמתים לקיבולי קשתות

ברשת הזרימה $G'=(V',E'),c'$ המתקבלת (עם קיבולים על הקשתות) מחשבים את הזרימה המקסימלית f'. לכל קשת $e=(u,v)\in E$ מתאימה הקשת $e'=(u^{\text{out}},v^{\text{in}})\in E'$. כלומר, יש התאמה חד־ערכית בין קשתות E לקשתות E' שצורתן אינה $(v^{\text{in}},v^{\text{out}})$. לכל צומת v מתאימה הקשת (v^{in},v^{out}) בעלת אותו קיבול כמו v, ולכן אילוצי הקיבול לא מופרים. לכן, הזרימה המתאימה ברשת G,c (עם קיבולים על הצמתים) לזרימה f' היא $f(u,v)=f'(u^{\text{out}},v^{\text{in}})$ לכל $e\in E$. אפשר גם לראות את f כמתקבלת מ־f' על־ידי כיווץ כל קשת $(v^{\text{in}},v^{\text{out}})$ של E' לצומת אחד. את הבדיקה כי f מקיימת את חוק שימור הזרימה ואת אילוצי הקיבול אנו משאירים לקורא. ♣

פתרון תרגיל 7.14 מעמוד 160

תחילה נוכיח טענה כללית יותר, ואחר כך נראה כיצד להסיק ממנה את גרסת הצמתים של משפט מנגר. נסמן ב־$\kappa_G(s,t)$ את המספר המקסימלי של מסלולים מ־s ל־t בגרף G, כך שכל שניים מהם זרים בצמתים. הטענה הכללית יותר שנוכיח היא:

טענה 7.22 יהי $G=(V,E)$ גרף מכוון ויהיו $s,t\in V$. אז $\kappa_G(s,t)$ שווה למספר המינימלי של צמתים ב־$V\setminus\{s,t\}$ ושל קשתות ב־E שלאחר השמטתן מהגרף לא יהיה מסלול מ־s ל־t; כלומר

$$\kappa_G(s,t)=\min\{|F| : F\subseteq(V\setminus\{s,t\})\cup E,\ G\setminus F \text{ has no } (s,t)-\text{path}\}.$$

שימו לב, ביטלנו כאן את האיסור על חיבור קשת מ־s ל־t, ואנו מרשים להשמיט גם קשתות ולא רק צמתים. הנה הוכחת טענה 7.22. נפעיל את הרדוקציה ששימשה אותנו בתרגיל 7.13 (כאשר קיבולי כל הקשתות ב־G' הם 1). ממשפט מנגר (גרסת הקשתות) נקבל כי ב־G' המספר המקסימלי של מסלולים מ־s ל־t כך שכל שניים מהם זרים בקשתות, שווה לגודל המינימלי של קבוצת קשתות F' כך שבגרף $G' \setminus F'$ אין מסלול מ־s ל־t. טענה 7.22 נובעת משתי האבחנות הפשוטות הבאות.

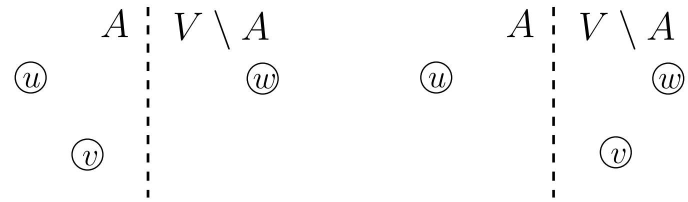
האבחנה הראשונה היא שמסלולים זרים בקשתות בגרף G' ממופים למסלולים זרים בצמתים של G. ואכן, לשני מסלולים הזרים בקשתות ב־G' אין קשת משותפת שצורתה $(v^{\text{in}}, v^{\text{out}})$, וזה מבטיח כי אין צומת המשותף למסלולים הזרים ולמסלולים המתאימים להם ב־G.

האבחנה השנייה היא כי כל קבוצת קשתות F' ב־G' ממופה לקבוצה של צמתים וקשתות F ב־G, כאשר לכל קשת שצורתה $(v^{\text{in}}, v^{\text{out}})$ מתאים הצומת $v \in F$, ולכל קשת $(u^{\text{out}}, v^{\text{in}}) \in F'$ מתאימה באופן טבעי הקשת $(u,v) \in F$.

כעת נראה כיצד להסיק מהטענה לעיל את גרסת הצמתים של משפט מנגר. תהי $F \subseteq (V \setminus \{s,t\}) \cup E$ בעלת גודל מינימלי, כך שאין מסלול מ־s ל־t בגרף $G \setminus F$. על־פי הטענה לעיל $|F| = \kappa_G(s,t)$. לכל קשת $(u,v) \in F$ מתקיים: $u \neq s$ או $v \neq t$, כלומר, לפחות קצה אחד של הקשת הוא לא s ולא t; כאן אנו משתמשים בהנחה כי $(s,t) \notin E$, ולכן גם $(s,t) \notin F$. כעת נחליף כל קשת ב־F בקצה שלה שאינו שייך ל־$\{s,t\}$, ונקבל קבוצת צמתים U שלאחר השמטתם מהגרף לא יהיה מסלול מ־s ל־t. ברור כי $|U| \leq |F|$ וכי $|U| \geq \kappa_G(s,t)$. מכאן נקבל $\kappa_G(s,t) \leq |U| \leq |F| = \kappa_G(s,t)$. לכן $|U| = \kappa_G(s,t)$, כנדרש. ♣

פתרון תרגיל 7.15 מעמוד 160

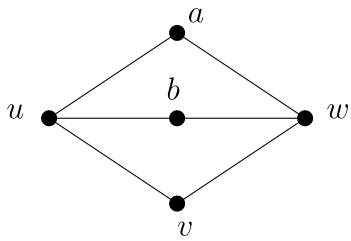
1. על־פי משפט מנגר לגרפים לא־מכוונים, $\lambda(s,t)$ שווה למספר המינימלי של קשתות בחתך המפריד בין s ל־t. נסמן $k = \lambda(u,w)$ ויהי A חתך עם k קשתות המפריד בין u ל־w, נניח $u \in A,\ w \in V \setminus A$. כעת יש שני מקרים: $v \in A$ או $v \in V \setminus A$, ראו איור 7.15.



איור 7.15: שני מקרים למיקומו של v: $v \in V \setminus A$ או $v \in A$

אם $v \in A$, אז החתך מפריד בין v ל־w, ולכן $\lambda(v,w) \leq k$. אם $v \in V \setminus A$, אז החתך A מפריד בין u ל־v, ולכן $\lambda(u,v) \leq k$. ובכל מקרה, $\min\{\lambda(u,v), \lambda(v,w)\} \leqslant k = \lambda(u,w)$, כנדרש.

2. נתבונן באיור 7.16. דרגתו של הצומת v היא 2, לכן בינו לבין כל צומת אחר יכולים להיות לכל היותר 2 מסלולים זרים בקשתות. לכן $\lambda(u,v), \lambda(v,w) \leq 2$. לעומת זאת, בין u ל־w קיימים שלושה מסלולים זרים בקשתות. לכן $\lambda(u,w) \geq 3$.



איור 7.16: $\lambda(u,w)=3$ ו־$\lambda(u,v)=\lambda(v,w)=2$,

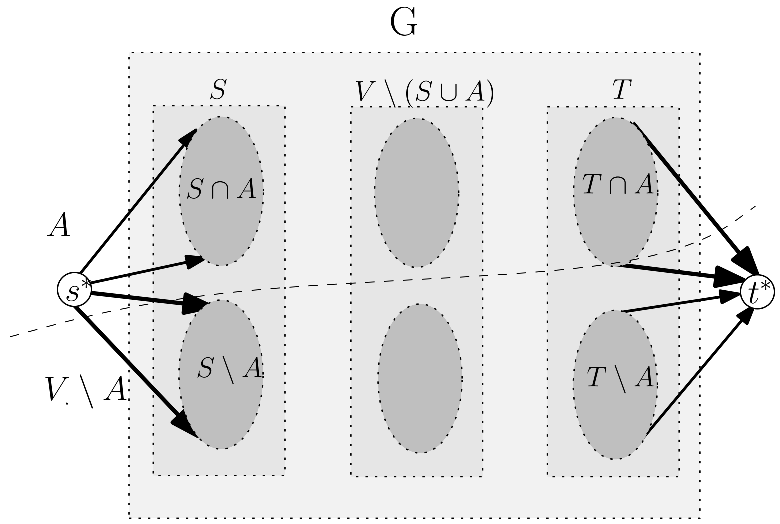
3. הטרנזיטיביות נובעת מיידית מסעיף 1. אם $\lambda(u,v)\geq k$, ו־$\lambda(v,w)\geq k$, אז

$$\lambda(u,w)\geq \min\{\lambda(u,v),\lambda(v,w)\}\geq k.$$

♣

פתרון תרגיל 7.16 מעמוד 161

נתבונן ברשת G' שנבנית ברדוקציה של בעיית ההפצה לבעיית הזרימה המקסימלית, ראו איור 7.17.



איור 7.17: רדוקציה של בעיית ההפצה לבעיית הזרימה המקסימלית.

אנו יודעים כי פתרון אפשרי לבעיית ההפצה קיים אם ורק אם יש ב־G' זרימה בערך $\sum_{s\in S}d_s=-\sum_{t\in T}d_t$. בהנחה כי $\sum_{s\in S}d_s=-\sum_{t\in T}d_t=k$ תהיה זרימה בגודל כזה אם ורק אם, הקיבול של כל חתך s^*-t^* ברשת G' הוא לפחות k. כלומר, לכל $A\subseteq V$ צריך להתקיים $c'\left(A\cup\{s\}\right)\geqslant k$ כאשר c' היא פונקציית

הקיבול של קשתות G'. באיור 7.17 רואים כי

$$c'(A \cup \{s\} = c(A) + \sum_{s \in S \setminus A} d_s - \sum_{t \in T \cap A} d_t \ .$$

לכן התנאי לקיום פתרון אפשרי לבעיית ההפצה הוא:

$$c\left(A\right) + \sum_{s \in S \setminus A} d_s - \sum_{t \in T \cap A} d_t \geq \sum_{s \in S} d_s .$$

על־ידי העברת אגפים נקבל:

$$c\left(A\right) \geq \sum_{s \in S} d_s - \sum_{s \in S \setminus A} d_s + \sum_{t \in T \cap A} d_t = \sum_{s \in S \cap A} d_s + \sum_{t \in T \cap A} d_t ,$$

כנדרש. ♣

פתרון תרגיל 7.17 מעמוד 162

.1 נשים לב כי

$$h\left(q\right) = \max_{\emptyset \neq X \subseteq V} \left(\left|E\left(X\right)\right| - q\left|X\right|\right) = \max_{\emptyset \neq X \subseteq V} \left|X\right| \left(\left|E\left(X\right)\right| / \left|X\right| - q\right) .$$

ברור שהפונקציה $h(q)$ היא מונוטונית יורדת. תהי

$$q* = \max_{\emptyset \neq X \subseteq V} \left|E\left(X\right)\right| / \left|X\right|$$

הצפיפות המקסימלית של תת־גרף של G. מתקיים $h(q) = 0$ אם ורק אם $q = q^*$. מכאן נקבל (על־סמך היותה של $h(q)$ פונקציה מונוטונית יורדת), שאפשר לחשב את q^*, כמו גם את קבוצת הצמתים המתאימה X^*, בזמן פולינומיאלי באמצעות חיפוש בינארי.

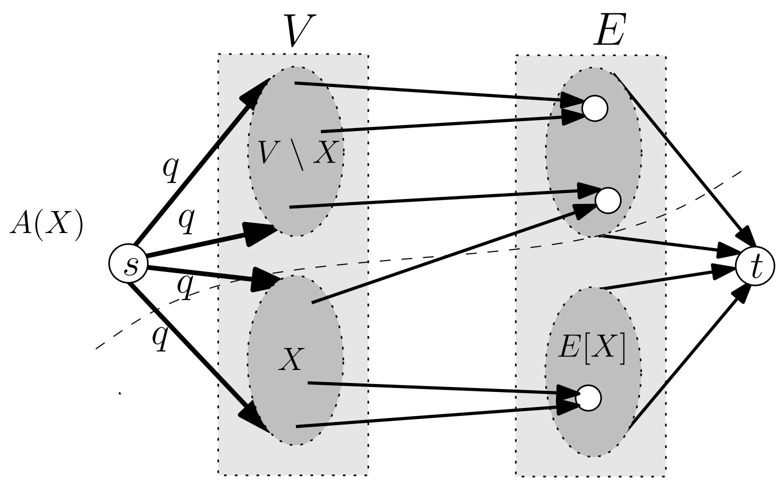
.2 $A(X)$ הוא חתך s-t ב־J כי $s \in A(X),\ t \notin A(X)$. כדי לראות מהו קיבולו של החתך $A(X)$ נתבונן באיור 7.18.

כאן האבחנה המרכזית היא כי ברשת J אין קשת מ־$V \setminus X$ ל־$E[X]$, כי בגרף G, לא יכולה להיות ב־$E[X]$ קשת שסמוכה לצומת ב־$V \setminus X$. לכן, הקשתות שיוצאות מ־$A(X)$ הן משני סוגים בלבד:

- הקשתות מ־s ל־X – יש $|X|$ קשתות כאלה, ולכל אחת מהן קיבול של q.
- הקשתות מ־$E \setminus E[X]$ ל־t – יש $|E| - |E[X]|$ קשתות כאלה, ולכל אחת מהן קיבול של 1 .

לכן קיבול החתך $A(X)$ הוא:

$$c\left(A\left(X\right)\right) = q \cdot \left|X\right| + \left|E\right| - \left|E\left[X\right]\right| = \left|E\right| - \left(\left|E\left[X\right]\right| - q \cdot \left|X\right|\right)$$



איור 7.18: החתך $A(X)$ ברשת J

3. ברשת J, לכל צומת e (כלומר, הצומת שמתאים לקשת e) מתקיים:

- מהצומת יוצאת בדיוק קשת אחת, הקשת (e,t) שקיבולה הוא 1.
- הקשתות שנכנסות לצומת הן $(u,e),(v,e)$; u ו־v הם הקצוות של e ב־G, והקיבול של כל קשת הוא 1.

לכן, אם A הוא חתך s-t ב־J, ואם $v \in A$ עבור $v \in V$ אבל $e \notin A$ עבור קשת e הסמוכה ל־v ברשת G, אז $A \cup \{e\}$ הוא חתך s-t ב־J שקיבולו אינו עולה על הקיבול של A. הטענה הזו נובעת מהבחירה של החתך A בשאלה.

אלגוריתם פולינומיאלי למציאת A: לפי תרגיל 7.2, A היא קבוצת הצמתים מהם לא ניתן להגיע ל־t ברשת השיורית האחרונה באלגוריתם דיניץ-אדמונדס־קרפ.

4. לפי הפתרון שהוצג בסעיף 2 נוכל להסיק כי $e \in E \setminus A$ אם ורק אם הזנבות של שתי הקשתות שנכנסות ל־e הם ב־$X_q = V \setminus A$. זה שקול לכך שמתקיים $E \setminus A = E[X_q]$, כלומר: $A = \{s\} \cup (V \setminus X_q) \cup (E - E[X_q])$. כעת נשים לב כי

$$\begin{aligned} h(q) &= \max_{\emptyset \neq X \subseteq V} (|E(X)| - q|X|) \\ &= |E| - \min_{\emptyset \neq X \subseteq V} (|E| - (|E(X)| - q|X|)) = |E| - \min_{\emptyset \neq X \subseteq V} c(A(X)) . \end{aligned}$$

לכן, המקסימום של $h(q) = \max_{\emptyset \neq X \subseteq V} (|E(X)| - q|X|)$ והמינימום של $\min_{\emptyset \neq X \subseteq V} c(A(X))$ מתקבל באותה קבוצת צמתים X. כיוון שהחתך $A = A(X_q)$ הוא חתך מינימלי, המינימום של $\min_{\emptyset \neq X \subseteq V} c(A(X))$ מתקבל עבור $X = X_q$. ולכן $h(q) = |E[X_q]| - q \cdot |X_q|$. ♣