אלגוריתמים 2021ב' (יובל רבני) - סיכום מתומצת - ניצן ברזילי

9 ביולי 2021

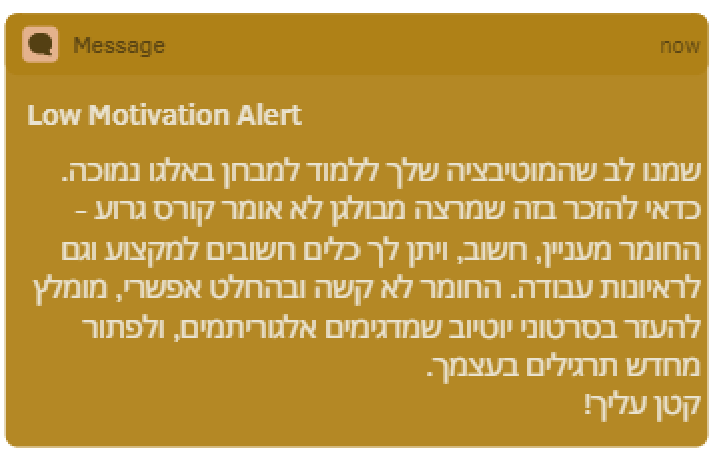
מה זה הסיכום הזה?

- אני הרגשתי (ואני בטוחה שהרבה חולקים איתי את התחושה הזו), שההרצאות של יובל הועברו בצורה מאוד מבולגנת שמאוד הקשתה לעקוב. בפרט, הרבה פעמים הסימונים היו מאוד מורכבים או מאוד קשים למעקב, ומעבר לזה, הרבה פעמים ההרצאה כולה הועברה בסימונים בלי לתרגם אותה ל"שפה של בני אדם". **המטרה שלי בסיכום הזה היא לרכז רק את האלגוריתמים, השיטות והסכמות שחשוב להכיר למבחן** (ללא ההוכחות שלהן, שכן הובהר שבשונה מסמסטרים בהם אלכס מעביר את הקורס, לא מצופה מאיתנו להכיר למבחן את ההרצאות מהכיתה).

- הדגש שלי היה א' ליצור **סיכום מתומצת**, וב' ליצור **סיכום בשפה אנושית**, כלומר **לתרגם למילים את הסימונים שיובל השתמש בהם.** אני יודעת שיש אנשים שיותר קל להם לקרוא פסודו־קוד, אז (איפה שהיה לי כוח, ואיפה שיובל כתב פסודו קוד בעצמו) **השתדלתי להוסיף גם פסודו קוד לחלק מהאלגוריתמים**.

- **הסיכום מחולק לפי נושאים** (כי הרגשתי שהרבה פעמים שיובל קופץ בין נושאים או שהנושאים של התרגולים לא קרו במקביל להרצאות, ולפעמים היה קשה מאוד להבין לאיזה נושא כל אלגוריתם שייך או מה המטרה שלו), אבל **רשמתי ליד כל נושא / אלגוריתם מאיזה הרצאה או תרגול הוא לקוח** למקרה שתרצו להעמיק עוד.

- מקווה שהסיכום יעזור לכם ויזכיר לכם שבסופו של דבר זה קורס לא מאוד קשה ועם חומר מעניין וחשוב. תודה לקרן (שהשתמשתי בסיכום הזה בלא מעט המחשות מצוינות שהיא ציירה), יחיאל ואור שהתבססתי על הסיכומים המצוינים שלהם! אם הסיכום ממש עזר לכם ואתם רוצים לפרגן לי בקפה, אפשר לעשות את זה בקישור $ko - fi.com/sikumim$ ♡ **בהצלחה בלמידה!** ☺



כללי

דברים שחשוב לשים לב אליהם

- <u>הגדרות שרלוונטיות להוכחות וסטודנטים נוטים לטעות בהן:</u>

  – **טענה טריויאלית:** נאמר שטענה היא טריויאלית אם היא נובעת באופן ישיר מההגדרה בלבד. לדוגמא, הטענה "אם גרף הוא עץ אז אין בו מעגלים" היא טריויאלית, אבל הטענה "עץ בעל n קודקודים מכיל $n-1$ צלעות" אינה טריויאלית (כלומר אינה נובעת ישירות מההגדרה, ולכן מצריכה הוכחה).

  – **טענה נכונה באופן ריק:** נאמר שטענה מתקיימת באופן ריק אם היא טענה מהסוג "**לכל** איבר בקבוצה S מתקיים כי ___", ו־S היא קבוצה ריקה. לדוגמא, הטענה "**כל** הסטודנטיות שלומדות תואר תלת חוגי במדמח, וטרינריה ועיצוב אופנה הן ג'ינג'יות" <u>נכונה באופן ריק</u> כיוון שקבוצת הסטודנטיות שעושות תואר שכזה היא ריקה (נראה לי, אם לא, תכירו לי את הסטודנטית שלומדת את זה כי היא נשמעת מגניבה). לעומת זאת, הטענה "בקבוצת הסטודנטיות שלומדות תואר תלת חוגי שכזה **קיימת** סטודנטית ג'ינג'ית" <u>אינה</u> נכונה באופן ריק.

- <u>זמני ריצה:</u>

  – **הגדרת פעולות אלמנטריות כתלות בבעיה:** כאשר אנחנו מנתחים זמן ריצה, אנחנו למעשה סופרים כמות של <u>פעולות אלמנטריות בסיסיות</u> (פעולות בזמן ריצה קבוע - $O(1)$). הפעולה האלמנטרית **תקבע לפי סוג הבעיה:**

    * **אם הבעיה עוסקת <u>במספרים</u> עצמם (כלומר האלגוריתם מקבל מספר/ים ומחזיר מספר):** נניח שאנחנו רוצים לנתח זמן ריצה של אלגוריתם שכופל שני מספרים. אם פשוט נתייחס לפעולת כפל כפעולה אלמנטרית, נחזיר ניתוח מאוד שטחי שאי אפשר להשתמש בו כדי להשוות בין אלגוריתמים שונים שעושים את הפעולה הזו. לכן במקרה כזה נרצה להשתמש ברזולוציה יותר גבוהה: לכן הפרמטר שיקבע את **גודל הקלט** n **הוא מספר הספרות בייצוג הבינארי של המספר** (כשעבור כל מספר n, מספר הספרות בייצוג הבינארי שלו יהיה $O(logn)$). הפעולות האלמנטריות יהיו **פעולות על ביטים** (חיבור ביטים, כפל ביטים וכו'). שימו לב שזה מקרה ממש ספציפי, כלומר נוגע רק לאלגוריתמים שממטרתם לבצע פעולות על מספרים.

    * **בכל מקרה אחר (כלומר בכל מקרה בו האלגוריתם לא פועל על מספרים):** לא נרצה להכנס לרזולוציה של ביטים, לכן במקרה זה (שתקף ב־99% מהאלגוריתמים שנתעסק בהם בקורס!) נתייחס לפעולות אריתמטיות על מספרים (כפל, חיבור, חיסור וכו') בתור פעולות אלמנטריות. דוגמאות נוספות לפעולות אלמנטריות במקרים כאלו יכולות להיות השמה, שמירה בזכרון, השוואה וכו'.

  – **סוגי זמני ריצה:**

    * **זמן ריצה פולינומיאלי:** נאמר על אלגוריתם שהוא רץ בזמן ריצה **פולינומיאלי באורך הקלט** (או **אלגוריתם פולינומי**) אם קיים $m\in\mathbb{N}$ שעבורו מספר הפעולות המקסימליות שהאלגוריתם מבצע הוא $O(n^m)$.

    * **זמן ריצה פסודו־פולינומיאלי / פולינומיאלי במובן החלש:** <u>רלוונטי רק לאלגוריתם שמקבל מספר כקלט.</u> נאמר על אלגוריתם שהוא רץ בזמן ריצה **פסודו־פולינומיאלי** הוא אלגוריתם שרץ בסיבוכיות פולינומית בגודל <u>האבסולוטי</u> של הקלט (כלומר **בגודל של המספר** שהוא מקבל כקלט) ולא <u>באורך</u> של הקלט.

      · <u>דוגמא שממחישה מה זה אלגוריתם פסודו־פולינומיאלי</u> - אלגוריתם שמקבל מספר שלם וחיובי n ובודק אם הוא ראשוני. הוא עושה זאת כך: לכל $i\in[2,\sqrt{n}]$, האלגוריתם בודק אם n מתחלק ב־i ללא שארית, ואם כן מחזיר $False$. אם האלגוריתם מסיים לעבור על כל המספרים (כלומר n לא מתחלק באף i שכזה), הוא יחזיר $True$.

      · היינו חושבים שהאלגוריתם פועל ב־$O(\sqrt{n})=O(n^{\frac{1}{2}})$, ולכן הוא פולינומיאלי עם $m=\frac{1}{2}$. אבל, נשים לב שזהו אלגוריתם שמקבל מספר כקלט, ומה שמשפיע על סיבוכיות הריצה שלו הוא <u>הגודל</u> של אותו המספר (או במילים אחרות, כמות הספרות בייצוג הבינארי שלו). כיוון שלכל מספר n, מספר הספרות בייצוג הבינארי שלו יהיה $O(logn)$, נתייחס לזה בתור גודל הקלט.

      · כעת נחשב את זמן הריצה של האלגוריתם ברזולוציה הזו: מחוקי log וחוקי חזקות, נקבל כי $O(\sqrt{n})=O(\sqrt{2^{log_2 n}})=O(2^{log_2\sqrt{n}})=O(2^{log_2 n})$. כלומר, זמן הריצה של האלגוריתם הוא לא פולינומיאלי $O(\sqrt{n})$ כמו שחשבנו (כשהתייחסנו לאלגוריתם ברזולוציה לא מספיק גבוהה), אלא (ברזולוציה שמתייחסת לגודל הקלט בתור מספר הספרות בייצוג הבינארי) בעצם אקספוננציאלי $O(2^{log_2 n})$.

  – **אלגוריתם יעיל:** לצורך קורס זה, אלגוריתם יעיל הוא אלגוריתם בזמן ריצה פולינומיאלי (ולא אקספוננציאלי או פסודו־פולינומיאלי). נזכור כי יש הבדל משמעותי ביעילות גם בין אלגוריתמים פולינומיאלים (אלגוריתם ב־$O(n^2)$ הוא משמעותית יותר יעיל מאלגוריתם ב־$O(n^5)$), ולכן נעדיף אלגוריתם עם m קטן ככל הניתן.

  – חשוב להציג את זמני הריצה באותם מונחים שבהם הקלט מוגדר (נניח אם הקלט מוגדר באמצעות n קודקודים ו־m צלעות ויש לנו אלגוריתם שעובד על כל גרף בסיבוכיות $O(|V|\cdot|E|^2)$, צריך להציג את הסיבוכיות בתור $O(n\cdot m^2)$.

אלגוריתמים חמדניים ולמת ההחלפה

למת ההחלפה - סכמה כללית להוכחת אופטימליות של אלגוריתמים חמדניים

- אלגוריתם חמדן: אלגוריתם יקרא חמדן אם בכל שלב הוא בוחר באפשרות המשתלמת ביותר באותו הרגע מבלי לקחת בחשבון השלכות עתידיות.
- סכמה כללית להוכחת אופטימליות של אלגוריתם חמדן:
  - הוכחת חוקיות (נכונות) של האלגוריתם החמדן B המחזיר פתרון באורך m.
  - הוכחת האופטימליות באינדוקציה:
    * בסיס האינדוקציה (אם $k=0$ אין מה להוכיח).
    * טענת האינדוקציה: שלכל $k \in [0, m]$ ישנו אלגוריתם אופטימלי C שמסכים עם B על k האיברים הראשונים.
    * צעד האינדוקציה - למת ההחלפה (עבור $k+1 \leq m$): מהנחת האינדוקציה, קיים פתרון אופטימלי שמכיל את k האיברים הראשונים של B. נבנה מ-C אלגוריתם אחר C' שמכיל את $k+1$ האיברים הראשונים של B. כעת יש להוכיח כי C' חוקי ואופטימלי - עושים זאת ע"י שימוש באופטימליות של C ובאופן הפעולה החמדנית של B.
  - הסקה מההוכחה באינדוקציה כי בפרט עבור $k=m$ קיים פתרון אופטימלי C המכיל את כל m האיברים של B, ולכן B אופטימלי.

אלגוריתם שיבוץ משימות / שיבוץ קטעים לא חופפים (הרצאות 1,2)

- קלט: קבוצה של n משימות, כל משימה נתונה ע"י קטע על הישר הממשי (נניח שקצוות הקטעים הם מספרים שלמים), כל קטע נתון ע"י סוג סדור של נק' התחלה ונק' סיום, בצורה הבאה: $(s_1, t_1), (s_2, t_2) ..., (s_n, t_n)$. הקטעים יכולים להיות חופפים ומדובר בקטעים סגורים.
- פלט: קבוצה בגודל מירבי של קטעים לא חופפים.
- אלגוריתם:
  - נמיין את הקטעים בסדר עולה לפי **זמן הסיום** (נסמן בה"כ $t_1 \leq t_2 \leq ... \leq t_n$).
  - נעבור על הרשימה הממוינת לפי הסדר ונוסיף לפלט כל קטע שלא חופף לקטעים הקודמים שכבר בחרנו.
- סיבוכיות זמן: $O(nlogn)$ (כיוון שמתקיים מיון ב-$nlogn$ ואחריו מעבר על הרשימה הממוינת ב-n).
- סיבוכיות מקום: $O(1)$
- הערות: דוגמאות לאלגוריתמים לפתרון אותה בעיה שלא עובדים: מיון לפי נקודת התחלה, מיון לפי אורך הקטע.

אלגוריתם EED למזעור איחור מירבי בהגשת משימות (הרצאות 2,3)

- קלט: קבוצה של n משימות, כל משימה נתונה ע"י זוג מספרים טבעיים (p_j, d_j) כאשר p_j הוא משך הזמן שהמשימה דורשת, ו-d_j הוא מועד הסיום הנדרש (הדדליין).
- פלט: שיבוץ תקין של המשימות על ציר הזמן, כלומר פונקציה $t : \{1, ..., n\} \to \mathbb{R}_+$ שקובעת מתי נתחיל כל משימה. t מקיימת לכל $i, j \in [1, n]$ השונים זה מזה: $(t(i), t(i)+p_i) \cap (t(j), t(j)+p_j) = \emptyset$ (כלומר אין חפיפה בין הזמן שבו מבצעים את המשימה ה-i לזמן בו מבצעים את המשימה ה-j), כך שהאיחור המירבי הוא מינימלי, כלומר מתקיים $L_{max}(t) = \underset{j=1,...,n}{max} \{max\{0, t(j)+p_j-d_j\}\}$.
  - מהו L_{max} במילים - לכל משימה נמצא את האיחור שלה (הזמן שבו הגשנו את המשימה פחות הדדליין שלה - אם זה יוצא שלילי, כלומר הגשנו לפני הדדליין, נבחר 0), ואז L_{max} הוא האיחור המירבי מבין האיחורים של כל המשימות.
- אלגוריתם EED - ממזער את L_{max}:
  - נמיין את המשימות בסדר עולה של זמני ההגשה (נסמן בה"כ $d_1 \leq d_2 \leq ... \leq d_n$).

– נשבץ את המשימות לפי הסדר הנ"ל החל מזמן 0 וללא רווחים בין המשימות.

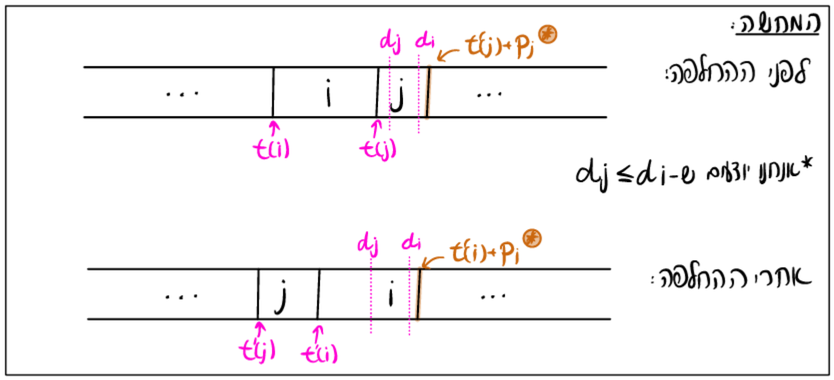
אלגוריתם 1 **EDD - Earliest Due Date($(p_1, d_1), .., (p_n, d_n)$):**

```
t(1)=0
for j← 1 to n:
    t(j+1)=t(j)+p_j
```

– סיבוכיות זמן ריצה: $O(nlogn)$ (כיוון שמתקיים מיון ב־$nlogn$ ואחריו מעבר על הרשימה הממוינת ב־n).

• טענות ומשפטים:

– טענה: אם בשיבוץ כלשהו של משימות הממזער את L_{max} (שאינו זהה לסדר שהחזיר EED) קיימות זוג משימות עוקבות i, j שהסדר ביניהן הפוך לסדר שלהן לפי EED, אם נחליף את סדר הביצוע שלהן (ככה שהוא יהיה כמו ב־EED), לא נשנה את ערכו של L_{max}.



– טענה: אלגוריתם EED ממזער את L_{max} (זה נובע מהטענה הקודמת - נקח סידור שממזער את L_{max} ונחליף בו כל זוג עוקב שלא באותו סדר כמו ב־EED עד שנגיע ל־EED עצמו, מבלי שהגדלנו את L_{max}).

אלגוריתם Belady לפתרון בעית הדפדוף (הרצאה 3)

• הגדרת הבעיה: זכרון של מחשב מחולק לדפים - המעבד יכול לגשת רק לזכרון המהיר (שיכול להכיל k דפים) ולא לזכרון האיטי (שמכיל את הדפים שאין להם מקום בזכרון המהיר), לכן יש לדפדף דפים בין הזכרון המהיר לאיטי לפי הצורך.

• קלט: סדרת הדפים שרוצים לגשת אליהם $(\sigma_1, ..., \sigma_m)$, המסודרת לפי הסדר בו נרצה לגשת לדפים. בסה"כ קיימים $n > k$ דפים (כלומר אין מקום בזכרון המהיר לכל הדפים במקביל), ומתקיים לכל t כי $\sigma_t \in [1, n]$ (כלומר כל אחד מהדפים בסדרה נלקח מתוך n הדפים הקיימים). הדפים יכולים לחזור על עצמם בסדרה (כלומר ניתן לבקש דף מסוים יותר מפעם אחת, ומספר הדפים בסדרה, m, יכול להיות גדול מ־n).

• פלט: לכל זמן $t \in [k+1, m]$ (שחורג מהמקום בזכרון המהיר) שבו מבקשים דף בזכרון המהיר, צריך לציין איזה דף יש להעביר מהזכרון המהיר לאיטי. ה־t־ים הרלוונטיים והדפים הנזרקים תלויים בהחלטות הקודמות.

• האלגוריתם של Belady - משיג מספר דפדופים מינימלי:

– לכל $t \in [k+1, m]$:

∗ נסמן ב־C_t את קבוצת הדפים בזכרון המהיר בזמן t (כאשר הגענו לאיבר σ_t מתוך רשימת הדפים), כך ש־$C_{k+1} = \{\sigma_1, ..., \sigma_k\}$ (כלומר באופן ספציפי, כאשר נגיע לאיבר σ_{k+1}, בהכרח קבוצת האיברים שנמצאת בזכרון המהיר תכיל את k האיברים הראשונים בסדרה).

– אם $\sigma_t \notin C_t$, נחליף אותו בדף $\sigma \in C_t$ שעבורו $min_{\sigma \in C_t}\{t' > t \mid \sigma_{t'} = \sigma\}$ מירבי (כלומר, מבין הדפים שכרגע נמצאים בזכרון המהיר, נבחר את הדף σ שהפעם הבאה שמבקשים אותו היא המאוחרת ביותר - ערך ה־t' המינימלי שלו, כלומר האינדקס הבא שלו ברשימה, הוא המקסימלי מבין כל שאר הדפים שנותרו ברשימה. לא אמרנו את זה מפורשות אבל אני מניחה שבמקרה שיש דף שלא מופיע יותר ברשימה, זה נחשב שהאינדקס t' שלו הוא המקסימלי, ואז נבחר אותו).

אלגוריתם 2 **Belady's Algorithm($(\sigma_1,...,\sigma_m),k$):**

for t$\leftarrow k+1$ to m:
    C_t =current group of pages in fast memory
    if $\sigma_t \notin C_t$:
        $\sigma_{remove} \leftarrow \sigma \in C_t$ such that the next time σ appears in $(\sigma_1,...,\sigma_m)$ is the latest
        $C_t \leftarrow (C_t \backslash \sigma_{remove}) \cap \sigma_t$

אלגוריתם לפתרון בעיית תא הדלק הקטן (תרגול 2)

- הבעיה: ישנה מכונית הנוסעת במסלול מסוים (קו ישר), מתחנת המקור a_1 לתחנת היעד a_n. לאורך המסלול פרושות תחנות דלק, ואנו מעוניינים למזער את מספר עצירות התדלוק שהמכונית מבצעת.
- קלט:
  - $N \in \mathbb{N}$ ־ מספר הקילומטרים שניתן לנסוע עם מיכל מלא.
  - $a_1,...,a_n$ ־ המיקום של תחנות הדלק לאורך הדרך ($a_1=0$), כך שלכל $i \in [1,n]$ מתקיים $a_i = a_{i-1} \le N$ (כלומר אין מצב שיש תחנת דלק a_{i-1} שכדי להגיע ממנה לתחנה הבאה a_i צריך יותר ממיכל מלא, או ־ אם יש לנו מיכל מלא, נוכל בוודאות להגיע לתחנה הבאה).
- פלט: תת סדרה $(b_1,...,b_m)$ באורך מינימלי (כלומר כוללת כמה שפחות עצירות / תדלוקים) המקיימת:
  - $b_1=a_1$ ו־$b_m=a_n$ (כלומר ־ הסדרה מתחילה ומסתיימת באותן תחנות כמו בקלט).
  - לכל $i \in [1,m]$ מתקיים $b_i - b_{i-1} \le N$ (כלומר, שסדרת התחנות תהיה חוקית ־ שאפשר לעבור בין כל תחנה לתחנה הבאה בסדרה באמצעות לכל היותר N ליטרים של דלק).
- אלגוריתם: בכל פעם נסע עד התחנה הרחוקה ביותר שנוכל להגיע אליה, ונמשיך כך עד שנגיע אל היעד. זה מחזיר פתרון אופטימלי כלשהו (כלומר יתכן שהפתרון האופטימלי אינו יחיד). מוכיחים את אופטימליות האלגוריתם באמצעות הסכמה הכללית של אלגוריתמים חמדניים (למת ההחלפה).

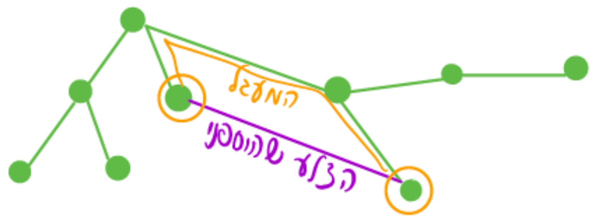
אלגוריתם 3 **Small Gas Tank Algorithm($(a_1,...,a_n),N$):**

$b_1 \leftarrow a_1$
prev$\leftarrow 1$ # holds the index of the last station we stopped at
for$i \rightarrow 2$ to n-1:
    if $a_{i+1} - b_{prev}$>N: #$a_{i+1}$is too far - a_i is the last station we can get to without filling gas
        $b_{prev+1} \leftarrow a_i$ # stop at a_i
        prev++
$b_{prev+1} \leftarrow a_n$
return $(b_1,...,b_{prev+1})$

הגדרות וטענות בנושא גרפים ועצים פורשים (הרצאות 4,5)

- הגדרות רלוונטיות:
  - עץ: גרף קשיר וחסר מעגלים. בין כל שני קודקודים בעץ יש בדיוק מסלול אחד.
  - עץ פורש: עבור גרף G, עץ פורש T הוא תת גרף של G המכיל את כל קודקודי G.
  - משקל של עץ פורש: תהי w פונקצית משקל, אז המשקל של העץ הפורש הוא $w(T) := \sum_{e \in T} w(e)$.
  - עץ פורש מינימלי MST: עץ פורש בעל משקל מינימלי, כלומר עץ פורש T שעבור כל עץ פורש אחר T' מקיים $w(T) \le w(T')$.
  - חלוקה זרה של קודקודים: יהי $G=\{V,E\}$ גרף לא מכוון, נאמר ש־A,B הן חלוקות זרות של V אם מתקיים $A \cup B = V$ וגם $A \cap B = \emptyset$.
  - חתך: חתך בין החלוקות הזרות A,B הוא קבוצת הצלעות שמקיימת $\{\{v,u\} \in E \mid v \in A, u \in B\}$, כלומר כל הצלעות שמחברות בין קודקוד מ־A לקודקוד מ־B.
- משפטים רלוונטיים:

- <u>טענה:</u> גרף קשיר וחסר מעגלים עם n קודקודים מכיל בדיוק $n-1$ צלעות.
- <u>טענה:</u> אם מוסיפים לעץ T צלע חדשה (כלומר צלע שלא קיימת ב־E אך מחברת שני קודקודים מ־V), אז בגרף שנוצר יש מעגל פשוט שעובר דרך הצלע החדשה שהוספנו.



- <u>מסקנה:</u> יהי G גרף, ויהי $T\subseteq E$ עץ פורש שלו. תהי $e\in E$ כך ש־$e\notin T$ (צלע מהגרף המקורי שלא נמצאת בעץ), ותהי $f\neq e$ צלע כלשהי במעגל הפשוט שנוצר מהוספת e ל־T. אזי גם $T\backslash\{f\}\cup\{e\}$ הוא עץ פורש של G.
  - $*$ במילים אחרות ־ אם נוסיף לעץ פורש T של גרף G צלע חדשה e (מתוך הגרף המקורי, אך שלא נמצאת בעץ), נקבל מעגל. אם נסיר מהמעגל הזה צלע אחרת f (שאינה הצלע e שהוספנו), נקבל עץ פורש חדש של G.

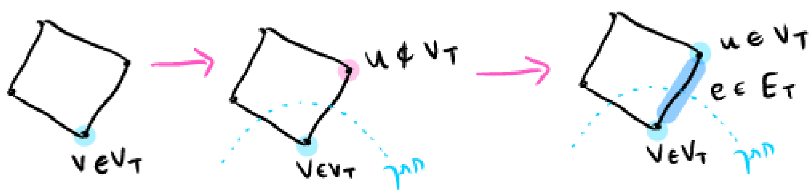
- <u>משפט:</u> יהי $F\subseteq E$ חתך ב־G, ותהי $e\in F$ צלע בעלת משקל מינימלי ב־F (כלומר לכל $f\in F$, מתקיים $w(e)\leq w(f)$). אזי קיים עפ"מ שמכיל את e.
  - $*$ במילים אחרות ־ לכל חתך שניקח, הצלע הכי קלה בו היא חלק מעפ"מ כלשהו (שימו לב ־ זה לא אומר שהיא חלק <u>מכל</u> עפ"מ ־ זה היה נכון אם כל המשקלים בגרף היו שונים).
- <u>אבחנה:</u> יהי T עפ"מ כלשהו של G, תהי $e\in T$. אם נסיר את e מ־T, נקבל גרף ובו שני רכיבי קשירות שאינם ריקים. אותה החלוקה ב־G מגדירה חתך שנסמנו $F_{T,e}$. בהכרח מתקיים כי $e\in F_{T,e}$, וזו הצלע היחידה של T שנמצאת בחתך $F_{T,e}$. אם T הוא עפ"מ, אז בהכרח e צלע בעלת משקל מינימלי בחתך זה.
  - $*$ במילים אחרות ־ לכל עפ"מ שניקח, אם נבחר צלע כלשהי, היא מגדירה חתך שהיא הצלע היחידה בו, ולכן היא גם הצלע המינימלית בחתך הזה.
- <u>משפט:</u> יהי C מעגל פשוט בגרף G, ותהי $e\in C$ צלע במעגל שמשקלה מירבי (כלומר לכל $f\in C$ מתקיים $w(e)\geq w(f)$). אזי קיים עפ"מ שאינו מכיל את e.
  - $*$ במילים אחרות ־ הפוך מהמשפט הקודם ־ לכל מעגל פשוט שניקח, יש עפ"מ שלא מכיל את הצלע הכי כבדה בו (שימו לב ־ זה לא אומר שהצלע הזאת היא לא חלק <u>משום</u> עפ"מ. יכול להיות שהיא זהה במשקל לצלע אחרת באותו מעגל, ולכן יתכן שיש עפ"מ שכן יכיל אותה ולא יכיל את הצלע האחרת).
- <u>טענה:</u> יהי G גרף, אם נוסיף לו צלע חדשה e, יתקיימו אחד משני תרחישים: או שהוספת e הורידה ב־1 את מספר רכיבי הקשירות של G, או ש־e סוגרת מעגל ב־G.

אלגוריתמים לחישוב עץ פורש מינימום (הרצאות 4,5, תרגול 2)

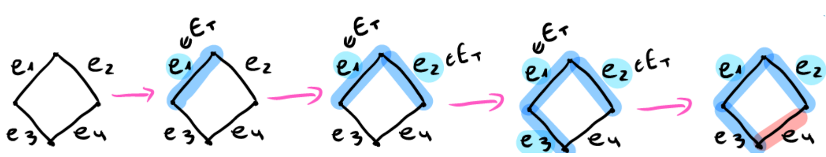
- <u>קלט:</u> גרף סופי, לא מכוון, קשיר $G=\{V,E\}$, ופונקצית משקל חיובית על הצלעות $w:E\rightarrow\mathbb{N}$.
- <u>פלט:</u> גרף קשיר שפורש את כל הקודקודים בעץ ומשקלו מינימלי (אם כל המשקלים חיוביים, בהכרח תת גרף כזה הוא עץ פורש).
- <u>כללי הצביעה:</u> בהנתן גרף לא מכוון $G=(V,E)$ עם פונקצית משקל w, נגדיר את כללי הצביעה הבאים:
  - הכלל האדום (צלעות כבדות): נבחר ב־G מעגל C שאין בו צלעות אדומות. ניקח את הצלע הכי כבדה שאינה צבועה ב־C, ונצבע אותה באדום.
  - הכלל הכחול (צלעות קלות): נבחר ב־G חתך D שאין בו צלעות כחולות. ניקח את הצלע הכי קלה שאינה צבועה ב־D, ונצבע אותה בכחול.
  - <u>טענה:</u> יהי G גרף לא מכוון כך שחלק מהצלעות שלו צבועות באדום וחלק בכחול באופן שרירותי. כל עוד קיימות צלעות שאינן צבועות, קיימת צלע לא צבועה אשר ניתן לצבוע אותה לפי הכלל האדום או לפי הכלל הכחול.

- <u>שיטה לפתרון הבעיה:</u> הוכחנו כי כל אלגוריתם המממש את השיטה הבאה מחזיר עפ"מ לגרף $G=(V,E)$:
  - נאתחל $E_B=\emptyset$ (קבוצת הצלעות הכחולות), $E_R=\emptyset$ (קבוצת הצלעות האדומות).
  - כל עוד יש צלעות לא צבועות בגרף (או כל עוד $|E_B|\neq|v|-1$), נבחר בין להפעיל את הכלל הכחול (ולהוסיף את הצלע שנצבעה ל־E_B), או להפעיל את הכלל האדום (ולהוסיף את הצלע שנצבעה ל־E_R).
  - נחזיר את הגרף $G'=(V,E_B)$.
    * במילים אחרות, בשיטה הזו מתחילים מגרף לא צבוע, ובכל שלב מסתכלים על צלע וצובעים אותה בכחול (כלומר מחליטים <u>שהיא בעפ"מ</u>), או באדום (כלומר מחליטים שהיא <u>לא בעפ"מ</u>). בסוף מחזירים את כל הצלעות הכחולות, שהן עפ"מ. זה לא אלגוריתם בפני עצמו אלא שיטה כללית - מה שצריך כדי לממש את השיטה ולהפוך את זה לאלגוריתם, הוא להחליט באיזה סדר עוברים על הצלעות הלא צבועות.

- <u>אלגוריתמים המממשים את השיטה (נלמדו גם בדאסט):</u>
  - <u>האלגוריתם של פרים Prim (אם נשתמש בתור קדימויות כדי למצוא בכל שלב את הצלע הקלה, נשיג סיבוכיות $O(ElogV)$:</u>
    * נאתחל $E_T=\emptyset$ (קבוצת הצלעות בפתרון שנחזיר - זה שקול ל־E_B בניסוח המקורי של השיטה), $V_T=\emptyset$ (קבוצת הקודקודים בפתרון שנחזיר).
    * נבחר קודקוד באקראי ונוסיף אותו ל־V_T.
    * נפעיל את הכלל הכחול על החתך שמפריד את V_Tמשאר הגרף, ונצבע את הקשר $e=\{v,u\}$ בכחול, כאשר $v\in V_T$ ו־$u\notin V_T$.
    * נוסיף את u ל־V_T ואת e ל־E_T.
    * נחזור לשלב השלישי, עד ש־V_T מכיל את כל הקודקודים בגרף.



  - <u>האלגוריתם של קרוסקל Kruskal (סיבוכיות $O(ElogE)$:</u>
    * נאתחל $E_T=\emptyset$ (קבוצת הצלעות בפתרון שנחזיר - זה שקול ל־E_B בניסוח המקורי של השיטה), $V_T=\emptyset$ (קבוצת הקודקודים בפתרון שנחזיר).
    * נמיין את הצלעות בסדר עולה - $w(e_1)\leq...\leq w(e_{|E|})$.
    * נעבור על הצלעות לפי הסדר:
      · אם הצלע $e_i=\{u,v\}$ סוגרת מעגל עם הצלעות ב־E_T: נפעיל על מעגל זה את הכלל האדום ונצבע את e_i באדום.
      · אחרת, הצלע $e_i=\{u,v\}$ היא הצלע הכי קלה בחתך המחבר את רכיב הקשירות ש־u נמצאת בו ב־V_T עם שאר הגרף: נצבע את e_iבכחול, נוסיף את e_i ל־E_T, נוסיף את u,v ל־V_T.
    * נחזור לשלב השלישי עד ש־V_T יכיל את כל הקודקודים בגרף.



מטרואידים (הרצאות 5־6, תרגול 3)

- **מטרואיד (מבנה מתמטי המבחין בין בעיות חמדניות לבעיות אחרות):** מטרואיד הוא זוג סדור (E,I) כאשר E היא קבוצת בסיס, ו־$I\neq\emptyset$ היא רשימה כלשהי של תתי־קבוצות של הקבוצה E. כל מטרואיד מקיים את שתי התכונות הבאות:
  - <u>ירושה:</u> אם $A\in I$ ו־$B\subseteq A$ אז $B\in I$. (במילים - אם קבוצה נמצאת ב־I, גם כל תתי הקבוצות שלה, כולל הקבוצה הריקה, נמצאות ב־I).
    * <u>הערה:</u> כל מטרואיד חוקי מכיל את הקבוצה הריקה. טיפ שנובע מזה - לפעמים אפשר להפריך את תכונת הירושה ע"י בחירה של $B=\emptyset$.

- הרחבה: אם $A, B \in I$ ו־$|A| > |B|$, אז קיים $e \in A \backslash B$ המקיים $B \cup \{e\} \in I$. (במילים ־ אם נקח שתי קבוצות שנמצאות ב־I שאחת מהן A גדולה מהשניה B, אז יש איבר ב־A שאם נוסיף אותו ל־B נקבל קבוצה שגם נמצאת ב־I).

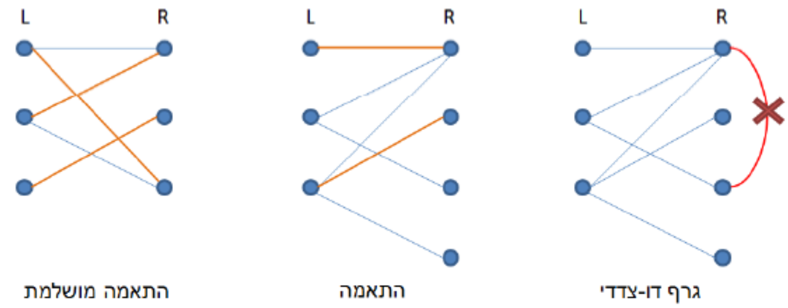
- **קבוצות בלתי תלויות:** הקבוצות ב־I נקראות קבוצות בלתי תלויות.

- **בסיס:** קבוצה בלתי תלויה בעלת גודל מקסימלי ב־I. (במילים אחרות ־ הקבוצה הכי גדולה ב־I).
  - טענה: כל הבסיסים במטרואיד הם שווי גודל.

- **טענות ומשפטים:**
  - טענה (תכונת ההחלפה): ניתן להחליף את תכונת ההרחבה בתכונת ההחלפה (אם $A, B \in I$ בסיסים, אז לכל $a \in A \backslash B$ קיים $b \in B \backslash A$ המקיים כי $A \backslash \{a\} \cup \{b\}$ הוא בסיס ־ כלומר לכל איבר $a \in A \backslash B$ אפשר למצוא איבר שונה ממנו ב־B כך שהחלפה ביניהם לא תפגע בהיותם בסיסים). כלומר, אם (E, I) מקיימת את תכונת הירושה ותכונת ההחלפה (במקום תכונת ההרחבה במקור), אז היא מטרואיד.
  - בכל מטרואיד מתקיימות תכונות ההחלפה סימטרית וההחלפה החח"ע:
    * טענה (תכונת ההחלפה הסימטרית): אם $A, B \in I$ בסיסים, לכל $a \in A \backslash B$ קיים $b \in B \backslash A$ עבורם $A \backslash \{a\} \cup \{b\}$ הוא בסיס וגם $B \backslash \{b\} \cup \{a\}$ הוא בסיס.
    * טענה (תכונת ההחלפה החח"ע): אם $A, B \in I$ בסיסים, קיימת העתקה חח"ע $f : A \backslash B \to B \backslash A$ עבורה לכל $a \in A \backslash B$ הקבוצה $A \backslash \{a\} \cup \{f(a)\}$ היא בסיס. במילים אחרות, קיימת העתקה $f(a) = b$ שלוקחת איבר a ב־$A \backslash B$ ומחזירה איבר b ב־$B \backslash A$, כך שאם נחליף את a ב־b, זה עדיין יהיה בסיס.
  - הערה: מטרואיד הוא כאמור מבנה מתמטי שמבחין בין בעיות חמדניות לבעיות שאינן חמדניות. בבעיות אופטימיזציה אנו מעוניינים בפתרון חוקי ואופטימלי ־ במטרואידים, נגיד שפתרון A הוא חוקי אם מתקיים $A \in I$, ואופטימלי אם הוא בעל המשקל הכי גדול/קטן ב־I (כתלות בהגדרת הבעיה).

- **דוגמאות שאינן מטרורידים:**
  - אי קיום תכונת הירושה: לדוגמא כל קבוצה שלא מכילה את הקבוצה הריקה ־ נגיד $E = \{1, 2\}$, $I = \{1, 2, \{1, 2\}\}$. ניתן להפוך אותה למטרואיד ע"י להוסיף לה את הקבוצה הריקה.
  - אי קיום תכונת ההרחבה: לדוגמא אם $E = \{1, 2, 3, 4\}$ ו־$I = \{\emptyset, 1, 2, 3, 4, \{1, 2\}, \{3, 4\}\}$, כי אם נקח $A = \{1, 2\}$ ו־$B = \{3\}$, הקבוצות $\{1, 3\}$ ו־$\{2, 3\}$ שתיהן לא נמצאות ב־I. ניתן להפוך אותה למטרואיד ע"י לוודא שלכל מספר ב־E ולכל זוג מספרים אפשרי, יהיה ב־I זוג שמשלב את המספר ואת אחד המספרים בזוג (לדוגמא להוסיף ל־I את $\{1, 4\}$ ואת $\{2, 3\}$).

- **דוגמאות למטרואידים:**
  - המטרואיד הטריויאלי / המטרואיד המלא: בהנתן קבוצה E, אוסף כל תת הקבוצות של E הוא מטרואיד (שיש לו בסיס אחד שהוא הקבוצה E עצמה).
  - מטרואיד המעגלים / המטרואיד הגרפי: בהנתן גרף סופי לא מכוון $G = (V, E)$, האוסף של כל קבוצות הצלעות $F \subseteq E$ עבורן F חסרת מעגלים (כלומר ־ כל היערות שאפשר ליצור מהצלעות ב־E) היא מטרואיד. אם הגרף G קשיר, הבסיסים הם כל העצים הפורשים של G.
  - המטרואיד הבי־סירקולרי: הנתן גרף סופי לא מכוון $G = (V, E)$, האוסף של כל קבוצות הצלעות $F \subseteq E$ עבורן כל רכיב קשיר של F מכיל לכל היותר מעגל אחד (כלומר כל ה"יערות" כך שיתכן מעגל אחד בכל עץ), הוא מטרואיד מעל E.
  - המטרואיד הוקטורי: בהנתן קבוצה A של וקטורים מאותו מרחב וקטורי, כל אוסף קבוצות של וקטורים בת"ל הוא מטרואיד מעל A. אפשר לחשוב גם על A בתור מטריצה, ואז קבוצות העמודות של A שהן בת"ל מהווה מטרואיד מעל קבוצת העמודות של A. הבסיסים שלו יהיו בסיסים (במובן של אלגברה לינארית) למרחב הוקטורי שהוקטורים לקוחים ממנו / למרחב העמודות של המטריצה A.
  - מטרואיד השידוכים: עבור $G = \langle L, R, E_G \rangle$ גרף דו צדדי (גרף המוגדר על 2 קבוצות זרות L, R של קודקודים, כך שכל צלע ב־E_G מחברת בין קודקוד ב־L לקודקוד ב־R), נגדיר את המטרואיד (L, I) עם I שהיא אוסף כל תתי הקבוצות של L כך שקיימת עבורם תת קבוצה של R שיש בינן התאמה מושלמת (התאמה חח"ע בין L ו־R ־ תת קבוצה של E_G שאין בה שתי צלעות שנוגעות באותו קודקוד, והיא נוגעת בכל הקודקודים ־ אפשרית אם"ם $|L| = |R|$).



- הנחה: בהנתן מטרואיד (E,I), נניח כי I לא נתונה באופן מפורש, אלא נתון אלגוריתם יעיל שבהנתן $F \subseteq E$ בודק אם $F \in I$.

האלגוריתם החמדן של המטרואיד (הרצאה 6, תרגול 3)

- רעיון כללי: אם יש לנו מטרואיד (כלומר בעיה שאפשר לפתור באופן חמדני) ובעיית אופטימיזציה (כלומר אנחנו רוצים את הקבוצה הכי "קלה" או הכי "כבדה" מבין הקבוצות ב־I לפי פונקציית משקל כלשהו), נוכל להשתמד באלגוריתם החמדן של המטרואיד כדי לפתור את בעית האופטימיזציה. כיוון שהוכחנו את הנכונות שלו, אפשר פשוט להשתמש בו (בלי להוכיח) כדי לפתור בעיות אופטימיזציה שכאלו לכל מטרואיד.
- קלט: מטרואיד (E,I) ופונקציית משקל חיובית $w: E \to \mathbb{N}$.
- פלט: תת קבוצה $A \in I$ שגודלה מקסימלי (כלומר לכל $B \in I$ מתקיים $|B| \geq |A|$, וגם משקלה מקסימלי (כלומר לכל $B \in I$ מתקיים $(w(A) = \sum_{a\in A} w(a) \geq \sum_{b\in B} w(b) = w(B)$.
  - נשים לב שמציאת קבוצה בגודל מקסימלי עם משקל מינימלי היא בעיה שקולה (לוקחים את מינוס האיברים, או ממיינים בסדר עולה).
- אלגוריתם:
  - נאתחל $A = \emptyset$.
  - נמיין את איברי E בסדר יורד חלש לפי w: $E = e_1, \dots, e_n$ כך ש־$w(e_i) \geq w(e_{i+1})$.
  - נעבור על כל איברי E לפי הסדר, ולכל i: אם אפשר להוסיף את e_i ל־A מבלי לחרוג מ־I, כלומר אם $A \cup \{e_i\} \in I$, נוסיף את e_i ל־A.
    - ישנם מקרים בהם נוכל לסיים את ריצת האלגוריתם גם מבלי לעבור על כל איברי E (לדוגמא אם בונים עץ וכבר הוספנו $|V|-1$ צלעות).
  - נחזיר את A (הוכחנו את נכונות האלגוריתם, לכן A מגודל ומשקל מקסימלי).
- סיבוכיות ריצה: נסמן $|E| = n$, אז סיבוכיות הריצה היא $O(nlogn + n \cdot f(n))$ (מיון + לולאה באורך n שמבצעת בדיקה האם $A \cup \{e_i\} \in I$, שתלויה בבעיה הספציפית שהאלגוריתם פותר, שנסמן את הסיבוכיות שלה $f(n)$).
- משפט Rado & Edmonds: תהי (E,I) מערכת קבוצות שמקיימת את תכונת הירושה. אם האלגוריתם הנ"ל מוצא פתרון אופטימלי (כלומר בסיס שמשקלו מקסימלי) עבור כל פונקציית משקל חיובית, אז מערכת הקבוצות היא מטרואיד.
- מקרים פרטיים של האלגוריתם החמדן של המטרואיד:
  - האלגוריתם של קרוסקל: הפעלה של האלגוריתם (עם מיון בסדר עולה במקום יורד) על המטרואיד הגרפי, מקבלים אלגוריתם שמוצא יער בגודל מקסימלי ומשקלו מינימלי (כלומר עץ פורש מינימלי). הפעלה שכזו תהיה זהה בפועל לאלגוריתם של קרוסקל - הוא יוצר יער ומגדיל אותו באמצעות צלע מינימלית באופן חמדני.
  - מציאת שידוך מקסימלי למטרואיד השידוכים: הבדיקה $A \cup \{e_i\} \in I$ (עם A ההתאמה הנוכחית) שקולה לבדיקה האם קיים שידוך מושלם עבור $A \cup \{e_i\}$, שזו (הוכחנו בדיסקרטית) בדיקה בסיבוכיות ריצה מאוד גבוהה, אך נראה בהמשך הקורס דרך יעילה יותר לבצע אותה.

תכנון דינאמי

תכנון דינאמי - כללי

- הרעיון בתכנון דינאמי הוא לגשת לפתרון בעיות רקורסיביות באופן שמשתמש ביותר זכרון כדי להפחית את זמן הריצה (מאקספוננציאלי לפולינומיאלי). הסיבה היא שבבעיות רקורסיביות אנחנו הרבה פעמים עושים את אותם החישובים שוב ושוב באופן שמנפח את זמן הריצה, כשלמעשה אפשר לשמור את כל חישובי הביניים ופשוט לשלוף אותם כשצריך במקום לחשב אותם שוב.

- **איך יראה פסודו קוד של בעית תכנון דינאמי:**
  - ניצור טבלה שבה נשמור את ההיסטוריה
  - נעשה לולאה על תתי הבעיות, ונשמור בכל שלב את תת הבעיה במקום המתאים בטבלה
  - נחזיר את הפתרון לבעיה המקורית, שאותו ניתן להסיק מטבלת ההיסטוריה שיצרנו

- **שלבים לפתרון בעיה באמצעות תכנון דינמי:**
  - <u>הגדרת תתי הבעיות:</u> בשלב זה נתאר איזה תתי בעיות נרצה לשמור במערך ההיסטוריה שיצרנו.
  - <u>כתיבת נוסחת רקורסיה:</u> בשלב זה נתאר מתמטית כיצד ניתן להגיע לפתרון כל תת בעיה בעזרת שימוש בהיסטוריה שכבר שמורה לנו.
  - <u>הגדרת טבלה + סדר המילוי שלה:</u> נגדיר מה הגודל של טבלת ההיסטוריה שאנחנו צריכים ליצור (כמה תתי בעיות אנחנו מתכוונים לחשב?). לאחר שהגדרנו את הגודל של הטבלה, נצטרך להסביר באיזה סדר נוכל למלא אותה בתתי בעיות - נצטרך לבחור סדר בו כל פעם שנבוא למלא תת בעיה כלשהי, כל ההיסטוריה הדרושה לנו כבר תהיה זמינה בטבלה (כלומר חושבה בשלבים קודמים).
  - <u>אופן חילוץ הפתרון האופטימלי:</u> נסביר כיצד ניתן להשתמש בטבלה, לאחר שמילאנו בה את תתי הבעיות, כדי לחלץ את הפתרון לבעיה המקורית שרצינו לפתור.
  - <u>ניתוח זמן ריצה:</u> ננתח את זמן הריצה של האלגוריתם. מכיוון שהאלגוריתם הוא לולאה על טבלה בגודל ידוע, לרוב זמן הריצה יהיה מספר התאים בטבלה $\times$הזמן שלוקח למלא כל תא.
  - <u>הוכחת נכונות:</u> נוכיח את נכונות האלגוריתם שכתבנו. נוכיח כי כל התאים בטבלת ההיסטוריה אכן מלאים בתתי הבעיות בהגדרנו (בד"כ נעשה זאת באינדוקציה על סדר מילוי הטבלה, בה נראה כי נוסחת הרקורסיה שהגדרנו אכן פותרת נכונה את תתי הבעיות). לאחר מכן, נוכיח כי ניתן להשתמש בטבלה המלאה על מנת לקבל את הפתרון לבעיה.

- **הערות וטיפים:**
  - גישה נפוצה לעיצוב אלגוריתם דינאמי היא להסתכל על הצעד האחרון בפתאון אופטימלי כלשהו, ולנסות להבין האם כשמורידים צעד זה נשארים עם פתרון אופטימלי עבור משימה <u>כלשהי</u>. אם כן, המשימה הזו מגדירה לנו את תתי הבעיות. הדרך שבה מחליטים מהו הצעד האחרון על סמך אוסף הפתרונות של תתי הבעיות, הוא שיגדיר לנו את נוסחת הרקורסיה.

בעיית התרמיל השלם - הקדמה (הרצאות 7-8)

- <u>קלט:</u> תרמיל בנפח שלם V, וקבוצה של n חפצים, כאשר לחפץ ה-$i \in [1, n]$ יש נפח (המסומן v_i) ומשקל שלם (המסומן $w_i \in \mathbb{N}$).

- <u>פלט:</u> קבוצה חוקית של חפצים (שלמים, כלומר לא ניתן לקחת חלקי חפצים) שסכום המשקלים שלהם מקסימלי, והנפח שלהם לא גדול מ-V (נפח התרמיל). כלומר, $argmax_{P\subseteq\{1,..,n\}}\left(\sum\limits_{i\in P} w_i \ s.t. \ \sum\limits_{i\in P} v_i \leq V\right)$.

- <u>אלגוריתמים לא טובים:</u>
  - <u>אלגוריתם גרוע 1 - שימוש באלגוריתם החמדן של המטרואיד:</u> לא אפשרי פשוט להשתמש באלגוריתם החמדן לפי משקל מהכבד לקל, כיוון שזהו לא מטרואיד - לא מתקיימת תכונת ההרחבה (לדוגמא, אם יש חפץ אחד בנפח V והרבה חפצים בנפח 1). מעבר לכך, זה היה עובד גרוע - לדוגמא, אם יש איבר אחד בנפח V ומשקל 1, וכל $n-1$ האיברים האחרים הם בנפח זהה של $\frac{V}{n-1}$ ומשקל $1-\varepsilon$ (עם ε קטן מאוד, כלומר המשקל שלהם הוא ממש כמעט 1). אז האלגוריתם החמדן ישיג משקל 1 (ע"י אריזה של האיבר הראשון), אבל הפתרון האופטימלי הוא במשקל $(1-\varepsilon)(n-1)$.
  - <u>אלגוריתם גרוע 2 - שימוש במשקל סגולי:</u> נמיין את האיברים בסדר לא עולה לפי משקל סגולי (משקל ליחידת נפח - $\frac{w_i}{v_i}$), ואז נשתמש באלגוריתם החמדן של המטרואיד.

- תהליך המחשבה בדרך ליצירת אלגוריתם טוב:
  - הקדמה: נראה איך נשתמש באלגוריתם 2 כדי ליצור אלגוריתם רקורסיבי מאוד לא יעיל (כי הוא מבצע כמות עצומה של חישובים, שהרבה מהם חוזרים על עצמם), ואז ע"י ביצוע של כמות סופית של חישובים מראש ושמירה של התוצאות שלהם בטבלה, נחסוך את כמות החישובים שהאלגוריתם הזה צריך לבצע. זה בדיוק העקרון של תכנון דינאמי - להשתמש ביותר מקום כדי למנוע מעצמנו לחשב את אותו החישוב יותר מפעם אחת.
  - נזהה כי אם נארוז לפי אלגוריתם 2 ונקבל קבוצה P, יתקיים $\sum_{i\in P} w_i + w_{max} \geq opt$ (עם w_{max} המשקל המירבי של חפץ בקלט). אנחנו יודעים ששימוש באלגוריתם 2 נותן לנו את המשקל הכי טוב שאפשר עבור יחידת נפח (ככה עובד משקל סגולי). עכשיו נסתכל על שני מקרים שונים:
    - אם החפץ ה-n נכלל בפתרון P הנ"ל: אזי הפתרון P הוא אריזה אופטימלית של $\{1,...,n-1\}$ האיברים בנפח לכל היותר $V - v_n$, שהוסיפו אליו את האיבר ה-n.
    - אחרת: אזי הפתרון P הוא אריזה אופטימלית של חפצים $\{1,..,n-1\}$ בנפח V.
  - נסמן ב-$P(i,V')$ אריזה אופטימלית של חפצים מבין החפצים $\{1,...i\}$ בתרמיל בנפח V', וב-$w\left(P(i,V')\right)$ את המשקל של הפתרון הזה.
    - כעת פתרון אופטימלי לבעיה יהיה במקרה בו $i=n$ ו-$V'=V$, כלומר: $w\left(P(n,V)\right)$, שניתן לחשב אותו ע"י הנוסחא $max\left(\overbrace{w_n + w(P(n-1,V-v_n)}^{n\ item\ \boldsymbol{in}\ solution},\ \overbrace{w\left(P(n-1,V\right)}^{n\ item\ \boldsymbol{not\ in}\ solution}\right)$, כלומר לבחור את האפשרות שנותנת את המשקל הכבד יותר, מבין שתי האפשרויות - האפשרות בה ניתן להוסיף את האיבר ה-n (ואז ניקח את הפתרון של הנוסחא עבור $n-1$ האיברים הראשונים, עם הנפח V פחות הנפח של האיבר ה-n), והאפשרות בה לא ניתן לעשות זאת (ואז נקח את הפתרון של הנוסחא עבור $n-1$ האיברים הראשונים, עם הנפח V כולו).
  - כלומר, קיבלנו נוסחא רקורסיבית שניתן לפתור באמצעותה את הבעיה, אך היא מאוד לא יעילה (כדי לפתור אותה צריך לבדוק 2^n אפשרויות).
    - נזהה שאפשר לחסוך בחישובים אם לכל $V' \in [0,V]$ ולכל $i \in [n]$, נחשב מראש את $P(i,V')$ ואת $w\left(P(i,V')\right)$.
    - כלומר, ניצור טבלה שהשורות שלה הן ערכי i, העמודות שלה הן ערכי V', וכל תא מכיל את $P(i,V')$ ואת $w\left(P(i,V')\right)$ ונמלא אותה בסדר חכם (ככה שכשנבוא למלא תא כלשהו, נוכל להתבסס על החישובים שביצענו בשלבים קודמים של מילוי הטבלה), ואז האלגוריתם הרקורסיבי לא יצטרך לרדת בכל פעם לעומק הרקורסיה ולבצע בדרך חישובים מיותרים שהוא כבר חישב בעבר, אלא רק לשלוף מהטבלה את הפתרון $P(i,V')$ הרצוי (שחישבנו מראש).

אלגוריתם לפתרון בעיית התרמיל השלם (הרצאות 7-9)

$i\ /\ V'$	0	$\cdots$	V'	$\cdots$	V
1	$\emptyset$	$\cdots$	$\{1\}$	$\cdots$	$\{1\}$
$\vdots$					
i			$P(i,V')\ ,\ w\left(P(i,V')\right)$		
$\vdots$					
n					$\boldsymbol{P(n,V)\ ,\ w\left(P(n,V)\right)}$

- אלגוריתם:
  - שלב ראשון - מילוי הטבלה: ניצור טבלה שהשורות שלה הן ערכי i והעמודות שלה הן ערכי V'. נמלא כל תא i,V' ב-$P(i,V')$ ו-$w\left(P(i,V')\right)$. המילוי יתבצע שורה-שורה, כאשר נתחיל מהשורה הראשונה $(i=1)$ ונמלא כל שורה משמאל לימין (כלומר נעבור על ערכי V' מהקטן לגדול). המילוי יתבצע באופן הבא:
    - אתחול - נמלא את השורה הראשונה (של i=1): לכל $V' \in [0,V]$ מהקטן לגדול, נחשב: $P(1,V') = \begin{cases}\{1\} & v_1 \leq V \\ \emptyset & else\end{cases}$
    וכן $w\left(P(1,V')\right) = \begin{cases}w_1 & v_1 \leq V \\ 0 & else\end{cases}$.
    - מילוי הטבלה (לכל $i \in [2,n]$ בסדר עולה): לכל $V' \in [0,V]$ מהקטן לגדול, נחשב את שתי האפשרויות הבאות ונבחר את זו מביניהן שנותנת משקל מקסימלי:

- אפשרות ראשונה: אם ניתן להוסיף את האיבר ה־i מבלי לחרוג מהנפח V', נעשה זאת:
  $\begin{array}{l} P(i,V') = \{i\} \cup P(i-1, V'-v_i) \\ w\left(P(i,V')\right) = w_i + wP(i-1, V'-v_i) \end{array}$, כלומר ־ אם החלטנו להוסיף את האיבר ה־i לתרמיל, נוסיף (הן עבור הנפח והן עבור המשקל) את האיבר ה־i לפתרון שחישבנו (בטבלה) עבור $i-1$ איברים והנפח V' פחות הנפח של האיבר ה־i (כדי שיהיה מקום להוסיף את האיבר ה־i).
- אפשרות שניה: אם **לא** ניתן להוסיף את האיבר ה־i מבלי לחרוג מהנפח V', נקח פתרון שלא כולל אותו:
  $\begin{array}{l} P(i,V') = P(i-1, V') \\ w\left(P(i,V')\right) = wP(i-1, V') \end{array}$, כלומר ־ ניקח את הפתרון שחישבנו (בטבלה) עבור $i-1$ איברים והנפח V' כולו.

– שלב שני: חישוב הפתרון מתוך הטבלה: הפתרון הוא התא ה־n,V בטבלה (כי הוא מכיל, מההגדרה, אריזה אופטימלית של חפצים מבין החפצים $\{1,...,n\}$, בתרמיל בנפח V).
  * אפשר להוכיח את נכונות האלגוריתם באינדוקציה לפי סדר מילוי התאים (כלומר להראות שכל תא i,V' מכיל באמת אריזה אופטימלית של חפצים מבין החפצים $\{1,...,i\}$, בתרמיל בנפח V').

- סיבוכיות:
  – סיבוכיות זמן ריצה: הסיבוכיות היא $O(nV)$:
    * מילוי הטבלה: בטבלה יש $V(n+1)$ תאים, שמילוי של כל אחד מהם לוקח $O(1)$ (ביצוע פעולות אריתמטיות על תוצאות של חישובים משלבים קודמים של מילוי הטבלה).
    * שליפת התוצאה: $O(1)$ (שליפה מהטבלה לפי אינדקס).
  – סיבוכיות מקום:
    * להחזיק בזכרון את הטבלה כולה דורש $V(n+1) = O(Vn)$ תאים.
    * נזהה שכדי להיות מסוגלים למלא תא מסוים, אנחנו צריכים גישה רק לשורה הקודמת. לכן, אפשר בכל שלב להחזיק בזכרון רק את השורה הנוכחית והשורה הקודמת (ולמחוק את השורה שלפניה), כלומר צריך לשמור רק $2(V+1) = O(V)$ תאים.

אלגוריתם נוסף לפתרון בעיית התרמיל השלם (הרצאה 9)

- מטרה: האלגוריתם שהצענו הוא בעל סיבוכיות שתלויה מאוד בנפח התרמיל V. נרצה לפתח אלגוריתם שאין לו תלות גבוהה בערך של V. מצטערת מראש שהחלק הזה לא מאוד ברור, היה מאוד קשה לפענח מה הוא רוצה שנפיק מזה.
  – נגדיר $w_{max} = max\{w_i \mid v_i \le V\}$, כלומר המשקל המקסימלי מבין המשקלים של האיברים שניתן להכניס לתרמיל (כלומר שהנפח שלהם קטן מ־V). נזהה שהמשקל הכי גדול שהתרמיל כולו יכול לקבל הוא $n \cdot w_{max}$.
  – נגדיר את $Q(i,w)$ להיות אריזה של חפצים מבין $\{1,...,i\}$, שמשקלה בדיוק w ונפחה מינימלי.

i / w	**0**	$\cdots$	w	$\cdots$	$n \cdot w_{max}$
1	$none$	$\cdots$	$\{1\}$	$\cdots$	$\{1\}$
$\vdots$					
i			$P(i,V')\ ,\ w\left(P(i,V')\right)$		
$\vdots$					
n	$none$	$\cdots$	$Q(n,w)$	$\cdots$	$Q(n, n \cdot w_{max})$

- אלגוריתם:
  – שלב ראשון ־ מילוי הטבלה: ניצור טבלה שהשורות שלה הן ערכי i, והעמודות שלה הן ערכי $w \in [0, n \cdot w_{max}]$. נמלא בכל תא i,w את $Q(w,i)$ ־ נתחיל מהשורה הראשונה ונמלא כל שורה משמאל לימין:
    * אתחול ־ נמלא את השורה הראשונה (של i=1): לכל $V' \in [0,V]$ מהקטן לגדול, נחשב: $Q(1,w) = \begin{cases} \{1\} & v_1 \le V \\ none & else \end{cases}$.
    * מילוי הטבלה (לכל $i \in [2,n]$ בסדר עולה): לכל $V' \in [0,V]$ מהקטן לגדול, נחשב את $Q(i,w)$ לפי המקרה המתאים:
      - מקרה ראשון ־ אין פתרון: כלומר, לא קיימת אריזה שמאפשרת לארוז את $\{1,..i\}$ במשקל של בדיוק w. במקרה זה נמלא $Q(i,w) = none$.
      - מקרה שני ־ ניתן להשתמש ישירות בפתרון עבור $Q(i-1,w)$: כלומר, ניתן לארוז את $\{1,..i-1\}$ במשקל של בדיוק w. במקרה זה נמלא $Q(i,w) = Q(i-1,w)$ (שמילאנו בשלב מוקדם יותר).

· <u>מקרה שלישי ־ ניתן להוסיף את האיבר ה־i לפתרון $Q(i-1, w-w_i)$:</u> כלומר, ניתן לארוז את $\{1, .., i-1\}$ האיברים במשקל של בדיוק $w - w_i$. במקרה זה נמלא $Q(i, w) = Q(i-1, w-w_i) \cup \{i\}$.

* <u>שליפת התוצאה מהטבלה:</u> נעבור על השורה האחרונה, ונבחר את הפתרון שמשקלו מקסימלי (כלומר העמודה הכי ימנית שערכה לא $none$).

- <u>סיבוכיות:</u> באלגוריתם הנ"ל, יש בטבלה $n \cdot nw_{max}$ תאים, שלמלא כל אחד מהם לוקח $O(1)$. אם w_{max} הוא פולינומי ב־n, קיבלנו בסה"כ אלגוריתם פולינומי.

- <u>הצעה לשינוי האלגוריתם למקרה ש־w_{max} אינו פולינומי ב־n:</u>

  – נסמן פרמטר k, ונעגל כלפי מטה את כל המשקלים לכפולות של k, כך שנקבל לכל היותר $1 + \lfloor \frac{w_{max}}{k} \rfloor$ ערכים שונים למשקלים אפשריים ־ מה ששקול למספר העמודות בטבלה שלנו. כעת, כמות התאים בטבלה היא $n \cdot n \left(1 + \lfloor \frac{w_{max}}{k} \rfloor\right) = O\left(n^2 \lfloor \frac{w_{max}}{k} \rfloor\right)$.

  – נזהה שככל ש־k גדול יותר, אנחנו מתרחקים מהפתרון האופטימלי. כדי לבחור k טוב, נבחר $\varepsilon > 0$ כלשהו, ונגדיר את $k = \frac{\varepsilon \cdot w_{max}}{n}$, וכעת הסיבוכיות היא $O\left(\frac{n^3}{\varepsilon}\right)$.

  – כלומר, קיבלנו פתרון שמקבל כקלט את V, w, ε ומוצא פתרון $(1-\varepsilon)$־מקרב, בסיבוכיות פולינומית ב־n וב־$\frac{1}{\varepsilon}$.

בלמן שכבות (תרגול 4)

- <u>קלט:</u> גרף מכוון $G = (V, E)$, המכיל $K+2$ שכבות המסומנות $V_0, ..., V_{K+1}$ ופונקצית משקל $w : E \to \mathbb{R}$ כך שמתקיים:

  – $V_0 = \{s\}$ הוא המקור, $V_{K+1} = \{t\}$ הוא המטרה.

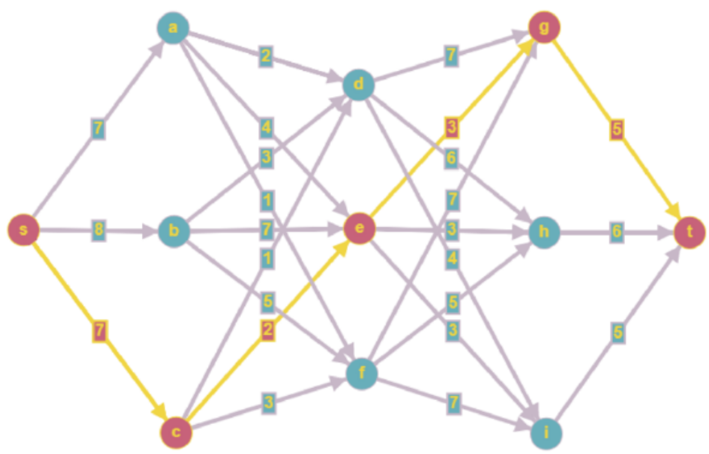
  – הגודל של כל השכבות זהה ומסומן M

  – האיחוד של כל השכבות $\bigcup_{k=0}^{K+1} v_k$ הוא V והחיתוך של כל שתי שכבות שונות הוא ריק, כלומר השכבות הן איחוד זר של V.

  – לכל צלע $e = (u, v)$ קיים $k \in [0, K]$ כך ש־u נמצא בשכבה ה־k ו־v נמצא בשכבה ה־$k+1$, כלומר כל צלע בגרף מחברת בין קודקוד משכבה אחת לקודקוד בשכבה הבאה.

  – כלומר, ניתן לחשוב על G בתור גרף שמתחיל ב־s ונגמר ב־t, כשבין הקודקודים האלו יש K שכבות בגודל M, כך שהצלעות בגרף מחברות בין שכבות עוקבות.

  – <u>דוגמא:</u> גרף עם $K = 3$ שכבות שכל אחת מהן בגודל $M = 3$:



- <u>פלט:</u> המסלול בעל המשקל הנמוך ביותר המחבר בין s ו־t.

- <u>אלגוריתם תכנון דינאמי:</u>

  – <u>אוסף תתי הבעיות:</u> לכל קודקוד $v \in V$, נמצא את המסלול הקל ביותר מ־s ל־v (כלומר, הפתרון האופטימלי יתקיים עבור $t \in V$).

  – <u>נוסחת הרקורסיה:</u>

  * נסמן ב־$D[v]$ את משקל המסלול הקל ביותר בין s ו־v (אם זה מסלול מ־s לעצמו, נגדיר $D[s] = 0$), ונסמן ב־V_kאת השכבה ש־v שייך אליה.

* נשים לב שמשקל המסלול הקל ביותר מ־s ל־v שקול לסכום משקל המסלול הקל ביותר מ־s לקודקוד שלפני v + משקל הצלע שמחברת ביניהם. לכן מספיק להסתכל על השכנים של v (שמהגדרת הגרף נמצאים בשכבה שלפני השכבה ש־v נמצא בה), כלומר, הנוסחא הרקורסיבית תראה כך: $D[v] = \begin{cases} 0 & k=0 \\ min_{u\in N(v)}(D[u]+w(u,v)) & else \end{cases}$.

– <u>הגדרת הטבלה:</u> נגדיר טבלה בגודל $K \times M$. נמלא אותה לפי נוסחת הרקורסיה, עמודה אחת בכל פעם, משמאל לימין (קודם עבור השכבה $k=1$, ולאחר מכן את השכבות הבאות בסדר עולה). בזמן מילוי הטבלה נשמור בכל תא גם מצביע לקודקוד שממנו הגענו במרחק הכי קצר לקודקוד הנוכחי.

– <u>חילוץ הפתרון האופטימלי מהטבלה:</u> נלך מהסוף להתחלה ־ נבחר את התא (u,t) שהערך השמור בו + משקל הצלע בינו ל־t הוא מינימלי. כעת נשתמש במצביעים ששמרנו בכל תא כדי לשחזר אחורנית את המסלול עד s.

– <u>זמן ריצה:</u> גודל הטבלה הוא $KM=O(V)$, ומילוי כל תא יכול לקחת עד $O(M)$ (כי בודקים את כל השכבה הקודמת), אך נבדוק כל צלע לכל היותר פעם אחת, לכן כל פעולות המילוי בסה"כ יקחו $O(E)$. כלומר, בסה"כ נקבל שזמן הריצה $O(V+E)$.

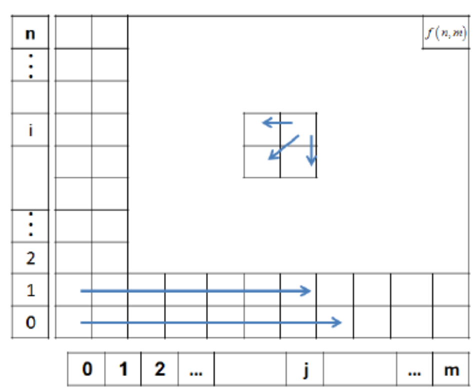
תת מחרוזת משותפת מקסימלית (תרגול 4)

- <u>קלט:</u> שתי מחרוזות: מחרוזת $X=x_1x_2..x_n$ ומחרוזת $Y=y_1y_2..y_n$.
- <u>פלט:</u> תת מחרוזת משותפת (לא בהכרח רציפה, אבל בהכרח באותו סדר) ארוכה ביותר.
- <u>פתרון נאיבי:</u> לכל תת מחרוזת של X נעבור על כל תתי המחרוזות של Y ונבדוק התאמה. זה פתרון שמצריך מעבר על 2^n תתי מחרוזות ולכן הוא מאוד לא יעיל.
- <u>פתרון דינמי:</u> נסמן ב־X^i את התחילית באורך i של X , כלומר את $x_1x_2..x_i$, וב־Y^j את התחילית באורך j של Y.

– <u>אוסף תתי הבעיות:</u> לכל $i\in[n]$ ו־$j\in[m]$, מציאת תת מחרוזת משותפת ל־X^iו־Y^j.

– <u>נוסחת הרקורסיה:</u> נסמן את האורך של תת המחרוזת המשותפת הארוכה של X^i ו־Y^j ב־$f(i,j)$, כעת נוסחת הרקורסיה היא: $f(i,j) = \begin{cases} 0 & i=0 \lor j=0 \\ f(i-1,j-1)+1 & x_i=y_j \\ max\{f(i-1,j),f(i,j-1)\} & x_i\neq x_j \end{cases}$, כלומר אם אחד מהם הוא 0 נמלא 0, אם הם שווים ניקח את הפתרון הקודם עבור שניהם ונוסיף לו 1, אם הם שונים ניקח את הגדול מבין הפתרון הקודם של i עם j והפתרון הקודם של j עם i.

– <u>מילוי טבלה:</u> נבנה טבלה בגודל $(m+1)(n+1)$, נמלא אותה לפי נוסחת הרקורסיה ־ נתחיל ממקרי הבסיס (ש־i או j הם אפס), ואז נמלא שורה־שורה (נתחיל מהשורה העליונה, ונמלא כל שורה משמאל לימין). בזמן מילוי הטבלה נשמור בכל תא גם מצביע לתא באמצעותו חושב הפתרון.



– <u>שחזור הפתרון האופטימלי מתוך הטבלה:</u> נעבור על הטבלה מלמעלה למטה ־ נתחיל מהתא (n,m) ונעקוב אחרי המצביעים ששמרנו. בכל פעם שהתא (i,j) מצביע לתא $(i-1,j-1)$, נוסיף את האות הנוכחית למחרוזת משמאל. נחזיר את המחרוזת שהתקבלה.

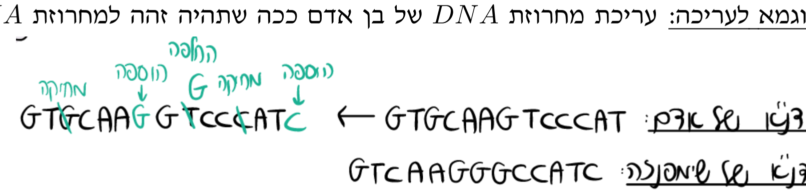
– <u>זמן ריצה:</u> גודל הטבלה הוא $(m+1)(n+1)$, ומילוי כל תא לוקח $O(1)$ (כי בודקים רק 3 תאים סמוכים), לכן בסה"כ נקבל $O(mn)$.

שיבוץ קטעים ממושקלים (הרצאות 9-10)

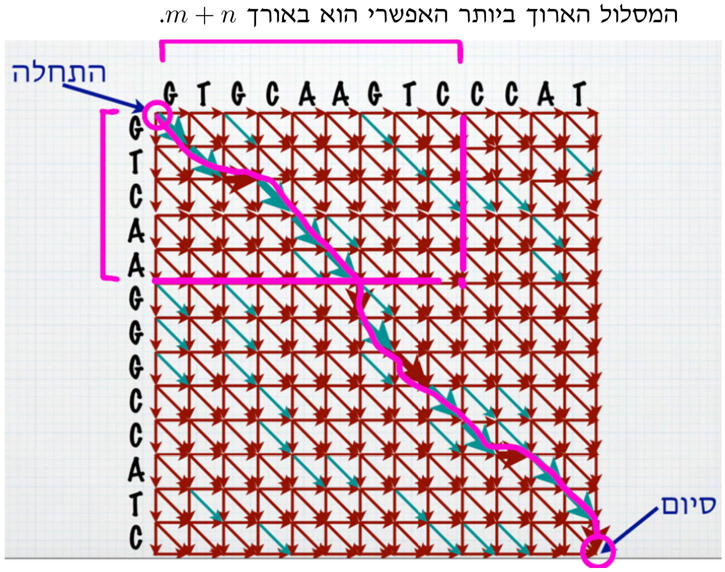
- קלט: n קטעים על הישר הממשי, כאשר לכל קטע $\{I_1,...,I_n\} \ni I_j$ מסומן $I_j=(a_j,b_j)$ ויש לו משקל w_j.
- פלט: קבוצת קטעים לא חופפים שמשקלה מקסימלי.
- אלגוריתם:
  - נניח שהקטעים ממוינים לפי זמן הסיום $b_1 \leq b_2 \leq ... \leq b_n$, וניצור מערך באורך n.
  - שלב ראשון - מילוי המערך: לכל b_i, נתבונן באריזה של קטעים שכוללת קטעים מהקבוצה $\{I_1,.,.,I_i\}$. נגדיר את S_i להיות שיבוץ של קטעים מתוך $\{I_1,.,.,I_i\}$שמשקלו מקסימלי, ונשמור את S_i בתא ה-i במערך. נחשב את S_i באופן הבא:
    - אתחול - עבור i=1: נמלא $S_1=\{I_1\}$.
    - מילוי שאר המערך - לכל $j\in[2,n]$ מהקטן לגדול: נסמן $j_i=argmax\{b_j \mid b_j\leq a_i\}$ (כלומר נקודת הסיום הכי גדולה שלא חופפת עם הקטע I_i). כעת נמלא $S_i=\begin{cases}\{I_i\}\cup S_{j_i} & w(S_{j_i})+w(i)\geq w(S_{j_{i-1}})\\ S_{i-1} & else\end{cases}$, כלומר - אם המשקל של S_{i_j} יחד עם האיבר ה-i הוא מיטבי (כלומר יותר טוב מהמשקלים עם כל הקטעים האחרים שנבדקו עד כה), נבחר אותו, אחרת - נבחר את המשקל של האיבר הקודם במערך.

מרחק עריכה של מחרוזות (הרצאה 10)

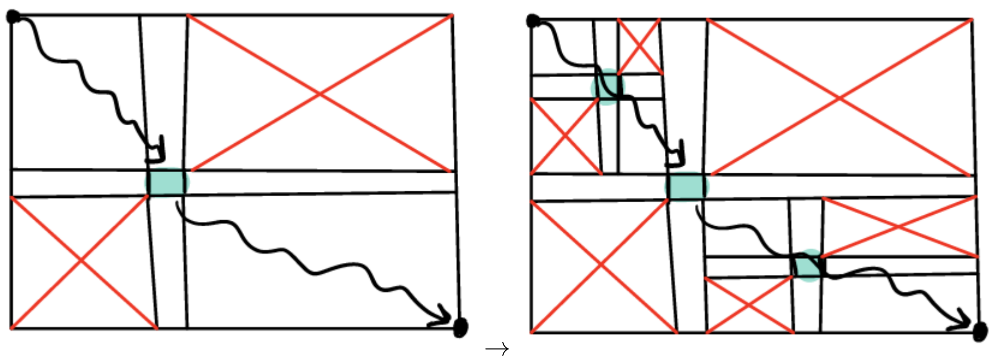
- קלט: שתי מחרוזות x ו-y שלקוחות מתוך אלפבית המסומן $\sum$.
- פלט: המספר המינימלי של פעולות עריכה (הוספת אות, מחיקת אות, החלפת אות באות אחרת) הנדרשות על מנת להפוך את x ל-y.
  - דוגמא לעריכה: עריכת מחרוזת DNA של בן אדם ככה שתהיה זהה למחרוזת DNA של שימפנזה:



  - הצגת העריכה הנ"ל באמצעות גרף:
    - נצייר גרף שנראה כמו טבלה, שהשורות בה הן האותיות מחרוזת היעד באורך m (DNA של שימפנזה), והעמודות בה הן באותיות במחרוזת ממנה התחלנו באורך n (DNA של בן אדם). נתחיל בפינה השמאלית עליונה של הגרף, ונתקדם לכיוון הפינה הימנית תחתונה, כאשר לכל צעד יש עלות של 1 (מסומן באדום) או 0 (מסומן בכחול).
    - צלעות ימינה ייצגו מחיקה (בעלות של 1), צלעות למטה ייצגו הוספה (בעלות של 1), וצלעות אלסוניות ייצגו החלפה בין שתי אותיות (בעלות של 1 אם האותיות שונות, ובעלות של 0 אם הן זהות).
    - כעת ניתן לחשוב על הבעיה המקורית בתור מציאת המסלול הזול ביותר מנקודת ההתחלה לנקודת הסיום בגרף. המסלול הארוך ביותר האפשרי הוא באורך $m+n$.



- <u>שימוש באלגוריתם של דייקסטרה:</u> ניתן להשתמש באלגוריתם של דייקסטרה למציאת המסלול הקל ביותר מנקודת ההתחלה לנקודת הסיום. כיוון שיש בגרף $(n+1)(m+1)$ קודקודים, ומספר הצלעות הוא באותו סדר גודל (לכל היותר פי 3 ממספר הקודקודים), נקבל כי סיבוכיות הריצה של דייקסטרה במקרה הזה היא $O(mn \cdot log(mn))$.
- <u>אלגוריתם אחר:</u>
  - <u>שלב ראשון - בניית ומילוי הטבלה:</u> נבנה טבלה D בגודל $(m+1)(n+1)$ שהערכים בה מתאימים לקודקודים בגרף הנ"ל, וכל תא $D(i,j)$ מכיל את המחיר של המסלול הקל ביותר מנקודת ההתחלה לקודקוד ה־(i,j).
    - <u>אתחול העמודה הראשונה:</u> לכל תא נוסיף את התא הקודם (זה שקול להמרה של מחרוזת ריקה למחרוזת X).
    - <u>אתחול השורה הראשונה:</u> לכל תא נסיר את התא הקודם (זה שקול להמרה של מחרוזת למחרוזת ריקה).
    - <u>אתחול שאר הטבלה:</u> לכל i ולכל j, $D(i,j) = min\left\{ \overset{right}{D(i-1)+1}, \overset{down}{D(j-1)+1}, \overset{diagonal}{D(i-1,j-1)} + \overset{indicator}{[x_i \neq y_i]} \right\}$.
  - <u>שלב שני - שליפת הפתרון מהטבלה:</u> נשלוף את הפתרון מהתא האחרון בטבלה $D(n,n)$.
- <u>סיבוכיות זמן ומקום:</u> מילוי כל תא לוקח $O(1)$, והטבלה מכילה $O(mn)$ תאים.
  - ניתן לשפר את סיבוכיות המקום (מבלי להשפיע על סיבוכיות הזמן) ע"י לחשב את הטבלה רק עד השורה האמצעית, למצוא את הנקודה בשורה האמצעית שהמסלול בטוח עובר בה, וככה לפסול 2 בלוקים מתוך הטבלה שאנחנו יודעים שלא צריך לחשב. נחזור על התהליך מספר פעמים עד שנצטמצם למסלול עצמו:



בעיית מסילת הרכבת (תרגול 5)

- <u>הקדמה:</u> בבעיה זו אנחנו מרכיבים מסילת רכבת באורך L מתוך מלאי חלקים שברשותנו. לכל חלק נתון מחיר, אורך וסוגי חיבור מימין ושמאל אשר קובעים אילו חלקים יכולים להופיע לפניו ואחריו.
- <u>קלט:</u>
  - $L \in \mathbb{N}$ אורך המסילה
  - $\{1,..,K\}$ רשימת סוגי חיבורים אפשריים
  - N חלקים, כשלכל חלק i נתון אורך d_i, מחיר p_i, ושני סוגי חיבורים - חיבור בהתחלת החלק (משמאל) s_i, וחיבור בסוף החלק (מימין) e_i.
- <u>פלט:</u> המחיר המינימלי עבור מסילה חוקית(שבה חלק i מופיע מימין לחלק j אם"ם $e_j = s_i$) באורך L.
- <u>אלגוריתם:</u>
  - <u>אוסף תתי הבעיות:</u> לכל $l \in [L]$ ולכל $k \in [K]$, נרצה למצוא את המחיר המינימלי למסילה מאורך l שנגמרת בחיבור מסוג k.
  - <u>נוסחת הרקורסיה:</u> נסמן ב־$f(l,k)$ את המחיר המינימלי למסילה באורך l שנגמרת בחיבור מסוג k, כעת נוסחת הרקורסיה היא: $f(l,k) = \begin{cases} 0 & l=0 \\ \underset{i \ \mid \ e_i=k \wedge d_i \geq 0}{min} \{p_i + f(l-d_i, s_i)\} & l>0 \end{cases}$. כלומר, נמצא את החלק i העומד בתנאים (עם אורך תקין ושנגמר בחיבור מסוג k) שעבורו המחיר (העלות שלו + העלות של המסילה שמגיעה עד אליו) מינימלי.
  - <u>בניית טבלה:</u> נבנה טבלה בגודל $(L+1)K$, ונמלא אותה לפי נוסחת הרקורסיה. נתחיל בשורה התחתונה (מקרה הבסיס ברקורסיה - $l=0$), ונמלא שורה־שורה (לא משנה באיזה סדר ממלאים כל שורה).
  - <u>שליפת הפתרון האופטימלי מהטבלה:</u> נעבור על השורה האחרונה ונחזיר את האיבר המינימלי מתוכה.
- <u>סיבוכיות ריצה:</u> הטבלה היא בגודל $(L+1)K$ ומילוי כל תא לוקח $O(N)$, לכן זמן הריצה הוא $O(NLK)$.

בעיית All Pairs Shortest Paths (תרגול 5)

- קלט: גרף מכוון $G=(V,E)$ שאינו מכיל מעגלים שליליים, ופונקצית משקל על הצלעות $w:E\rightarrow\mathbb{R}$ (לאו דווקא חיובית).
- פלט: מטריצה בגודל $|V|\times|V|$ כך שהתא ה־i,j שלה מכיל משקל של המסלול הקל ביותר מ־v_i ל־v_j.
- אלגוריתם תכנון דינמי נאיבי:
  - הגדרת תתי הבעיות: לכל $i,j\in[n]$ ולכל $m\in[n-1]$ נמצא את משקל המסלול הקל ביותר בין v_i ל־v_j שמשתמש בלכל היותר m צלעות, נסמנו $f(i,j,m)$.
  - נוסחת הרקורסיה: $f(i,j,m)=\begin{cases}0 & m=0,i=j\\ \infty & m=0,i\neq j\\ min_{v_x\in V}\left\{f(i,x,m-1)+w(v_x,v_j)\right\} & else\end{cases}$ - כלומר במקרה הכללי (שאינו מקרה הבסיס $m=0$), נעבור על הקודקודים $v_x\in V$, וניקח את המשקל המינימלי שנוצר מ"המשקל המינימלי מ־v_i ל־v_x + משקל הצלע (v_x,v_i)".
  - זמן ריצה: $O(|V|^4)$ בכלל שהטבלה היא בגודל $|V|^3$ ומילוי כל תא לוקח $O(|V|)$.
- אלגוריתם פלויד ורשל:
  - אוסף תתי הבעיות: לכל i,j,k נמצא את המחיר של מסלול קצר ביותר בין v_i ל־v_j שמשתמש רק בקודקודים $\{v_1,..,v_k\}$ כקודקודי ביניים.
  - נוסחת הרקורסיה: נסמן את המחיר של מסלול קצר ביותר בין v_i ו־v_j שמשתמש רק בקודקודים מתוך $\{v_1,..,v_k\}$ בתור $f(i,j,k)$. כעת נוסחת הרקורסיה היא:
    * במקרה הבסיס - $f(i,j,0)=\begin{cases}0 & i=j\\ w\left(v_i,v_j\right) & (v_i,v_j)\in E\\ \infty & (v_i,v_j)\notin E\end{cases}$ (כלומר אם יש צלע בין v_i ו־v_j ניקח את המשקל שלה, אחרת ניקח ∞ (או 0 אם הם אותו קודקוד).
    * בשאר המקרים - $f(i,j,k)=\min\{\underbrace{f(i,j,k-1)}_{v_k\text{ is unused}},\underbrace{f(i,k,k-1)+f(k,j,k-1)}_{v_k\text{ is used}}\}$ כלומר ניקח את המינימלי בין המסלול שלא כולל את v_k (כלומר מגיע מ־i ל־j תוך שימוש ב־$k-1$ הקודקודים הראשונים) לבין המסלול שכולל את v_k (המסלול מ־i ל־k ואז מ־k ל־j).
  - הגדרת ומילוי הטבלה: נבנה טבלה עם $|V|$ תאים (כל תא מייצג ערך k אחר), כשכל תא מכיל מטריצה $|V|\times|V|$. נמלא את הטבלה לפי נוסחת הרקורסיה - נתחיל במטריצה התחתונה כלפי מעלה.
  - חילוץ הפתרון האופטימלי מהטבלה: נחזיר את המטריצה העליונה (i,j,n).
  - זמן ריצה: הטבלה היא בגודל $|V|^3$ וכל תא ממולא ב־$O(1)$ לכן בסה"כ קיבלנו $O(|V|^3)$, משמעותית יותר טוב מהאלגוריתם הנאיבי.

רשתות זרימה

רשתות זרימה - הגדרות וטענות (הרצאה 13)

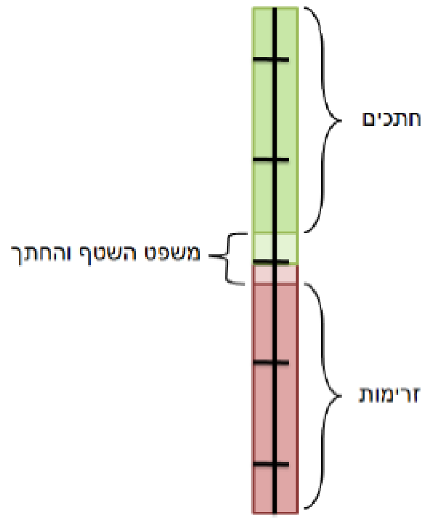
- הגדרות:
  - רשת זרימה: רשת זרימה היא רביעיה (G,c,s,t) המשמשת להעברת זרימה מקודקוד מקור לקודקוד מטרה דרך צלעות בגרף:
    * $G=(V,A)$ הוא גרף מכוון
    * $c:A\rightarrow\mathbb{N}$ היא פונקצית קיבול (שמגדירה כמה זרימה אפשר להעביר בצלע מסוימת)
      · נגדיר את הקיבול של קבוצת צלעות B ע"י $c(B)=\sum\limits_{a\in B}c(a)$.
    * $s\in V$ הוא קודקוד המקור
    * $t\in V$ הוא קודקוד המטרה / בור
  - זרימה f: זרימה ברשת (G,c,s,t) היא פונקציה $f:A\rightarrow\mathbb{R}$ שמקיימת את שתי התכונות הבאות:

* שימור הזרימה: בכל קודקוד $v \in V$ (שאינו s או t) מתקיים $\sum_{a=(v,j)\in A} f(a) = \sum_{a=(u,v)\in A} f(a)$ - כלומר, לכל קודקוד, כמות הזרימה הנכנסת אליו זהה לכמות הזרימה היוצאת ממנו (קודקוד לא "בולע" זרימה).
* אילוץ הקיבול: לכל $a \in A$ מתקיים $0 \le f(a) \le c(a)$, כלומר זרימה היא אי שלילית, ולא ניתן להזרים בצלע יותר מהקיבול שלה.

- שטף / ערך של זרימה $|f|$: בהנתן זרימה f ברשת זרימה (G,c,s,t), הערך של f הוא $|f| = \sum_{a=(s,v)\in A} f(a) - \sum_{a=(v,s)\in A} f(a)$, כלומר הזרימה שיוצאת מקודקוד המקור s פחות הזרימה הזרימה שנכנסת אליו.
  * מסקנה: זה שקול ל-$|f| = \sum_{a=(v,t)\in A} f(a) - \sum_{a=(t,v)\in A} f(a)$, כלומר הזרימה שנכנסת לקודקוד המטרה t פחות הזרימה שיוצאת ממנו.
- חתך (מכונה גם חתך s-t): קבוצת צלעות $B \subseteq A$ תקרא חתדאם"ם קיימת קבוצת קודקודים $S \subseteq V \setminus \{t\}$ כך ש-$s \in S$ עבורה B היא קבוצת הצלעות שיוצאות מ-S. (במילים אחרות - אם מחלקים את הקודקודים לשתי קבוצות זרות ש-s נמצאת באחת ו-t נמצאת בשניה, ו-B היא החתך שמחבר בין שתי הקבוצות האלו).
  * נסמן ב-B' את קבוצת הצלעות שנכנסות ל-S.
- קיבול של חתך: קיבול של חתך מוגדר להיות $c(S,T) = \sum_{(u,v)\in S\times T} c(u,v)$, כלומר הסכום של הקיבולים של כל הצלעות בחתך. אם צלע לא נמצאת בחתך, נתייחס לזה כאילו הקיבול שלה הוא 0.
- רשת שיורית: עבור רשת $N = (G,c,s,t)$ וזרימה f, נגדיר את הרשת השיורית $N_f = (G_f, c_f, s, t)$ בתור הרשת שמתארת את "המקום הפנוי" (שאפשר להעביר בו זרימה) שנותר ברשת. רשת שיורית מקיימת:
  * יש בה את אותם הקודקודים כמו ברשת המקורית
  * הצלעות שבה מורכבות מאיחוד של קבוצות הצלעות הבאות:
    · צלעות מהגרף המקורי שהזרימה בהן קטנה מהקיבול שלהן (כלומר שעוד אפשר להעביר בהן זרימה). הקיבול שלהן מוגדר ע"י $c_f(u,v) = c(u,v) - f(u,v) > 0$, כלומר כמות הזרימה שאפשר היה להזרים בה בכיוון המקורי.
    · צלעות שמהוות שהכיוון שלהן הפוך מכיוון של צלע כלשהי מהקבוצה הראשונה, והזרימה בהן חיובית. הקיבול שלהן מוגדר ע"י $c_f(v,u) = f(u,v)$, כלומר, בכל פעם שנעבור דרך צלע (u,v) עם זרימה חיובית, נוסיף לרשת השיורית צלע הפוכה (v,u) עם אותה כמות זרימה כמו שהזרמנו בכיוון המקורי.
- זרימה בשלמים: זרימה f נקראת זרימה בשלמים אם הערך שלה על כל צלע הוא מספר שלם.

• משפטים וטענות:

- טענה: לכל B חתך $s-t$ מתקיים $|f| = \sum_{a\in B} f(a) - \sum_{a\in B'} f(a)$, כלומר הזרימה שיוצאת מ-S פחות הזרימה שנכנסת ל-S (כאשר S היא קבוצת הקודקודים ש-s נמצא בה).
- טענה: לכל B חתך $s-t$ ולכל זרימה f מתקיים $|f| \le c(B)$.
- טענה: תהי רשת N עם זרימה f ורשת שיורית שלה N_f עם זרימה f'. אז $f + f'$ היא זרימה ב-N (כלומר מקיימת את שימור הזרימה והקיבול) שמוגדרת ע"י $(f+f')(u,v) = f(u,v) + f'(u,v) - f'(v,u)$.
- טענה: לכל חתך (S,T) ולכל זרימה חוקית f מתקיים כי $|f| \le c(S,T)$, כלומר הקיבול של החתך גדול או שווה לזרימה בכל הגרף.
- מסקנה: אם מצאנו זרימה f וחתך (S,T) כך ש-$|f| = c(S,T)$, אז בהכרח f מקסימלית (כלומר בעלת שטף מקסימלי) ו-(S,T) מינימלי.
- משפט השטף והחתך: לכל רשת זרימה, קיימים זרימה f וחתך (S,T) כך ש-$|f| = c(S,T)$.



רשתות זרימה - חישוב זרימת מקסימום (הרצאות 13-15)

- <u>קלט:</u> רשת זרימה $N = (G, c, s, t)$.
- <u>פלט:</u> זרימת מקסימום f_{max} של N, כלומר זרימה שהערך שלה $|f_{max}|$ מקסימלי.
  - <u>הערה:</u> זאת בעית אופטימיזציה, כאשר פתרון הוא "חוקי" אם הוא מהווה זרימה חוקית ברשת, והוא "אופטימלי" אם יש לו ערך מקסימלי.
  - הפתרון האופטימלי הוא לא בהכרח יחיד.
  - בכל רשת יש לפחות זרימה חוקית אחת, שהיא זרימת האפס.
  - הערך של כל זרימה חוקית ברשת חסום מלמטה ע"י 0, ומלמעלה ע"י סך הקיבול היוצא מ-s.
- <u>שיטת Ford-Falkerson:</u>
  - <u>השיטה:</u>
    * נתחיל מזרימה f שערכה 0 על כל צלע.
    * נפעיל את האיטרציה הבאה עד שנגיע למצב בו N_f לא מכילה מסלול מכוון מ-s ל-t:
      · נחשב את N_f.
      · נמצא ב-N_f מסלול מכוון מ-s ל-t, זהו מסלול משופר, נסמנו p, ואת הזרימה שעוברת בו ב-f'.
      · נגדיר $f'(u,v) = \begin{cases} min_{a \in p} c_f(a) & (u,v) \in p \\ 0 & else \end{cases}$, כלומר אם הצלע במסלול, נבחר להזרים בה את הקיבולת המינימלית של כל הצלעות במסלול.
    * נחליף את f ב-$f + f'$.
  - <u>סיבוכיות:</u> $O(|E| \cdot |f_{max}|)$.
  - <u>משפטים וטענות על השיטה:</u>
    * <u>טענה:</u> נניח שהרצה של השיטה מגיעה לסיומה אחרי מספר סופי של איטרציות, ותהי f הזרימה שהשיטה החזירה - אז f היא זרימת מקסימום ב-N.
    * <u>מסקנה:</u> זרימת מקסימום שווה לקיבול של חתך $s - t$ מינימלי.
    * <u>טענה:</u> אם הקיבולים שלמים, אז בכל שלב בריצת השיטה הזרימה היא בשלמים.
    * <u>מסקנה:</u> בכל איטרציה, הזרימה גדלה ב-1 לפחות.
- <u>אלגוריתם Edmonds-Karp:</u> אלגוריתם שממשש את שיטת Ford-Falkerson:
  - <u>האלגוריתם:</u> נממש את השיטה באופן הבא: ברשת השיורית, נמצא מסלול משפר שמספר הצלעות בו מינימלי. נשתמש באלגוריתם BFS החל מ-s, וכשנגיע ל-t זה אומר שמצאנו מסלול משפר.
  - <u>סיבוכיות:</u>
    * <u>סיבוכיות זמן:</u> $O(|V| \cdot |E|^2)$.
    * <u>סיבוכיות מקום:</u> $O(|V| + |E|)$.

מציאת שידוך מושלם בגרף דו צדדי (הרצאות 12,13,15)

- הגדרות:
  - גרף דו צדדי: גרף סופי, לא מכוון, דו צדדי: $G=(U,V,E)$, כך ש־U,V הן שתי קבוצות קודקודים זרות, וכל צלע $e\in E$ מחברת בין קודקודים מ־U לקודקוד מ־V.
  - קבוצת שכנים של קודקוד: עבור קודקוד $u\in U$, נסמן ב־$N(u)$ את קבוצת השכנים שלו.
  - קבוצת שכנים של קבוצת קודקודים: לכל תת קבוצה $U'\subseteq U$ של קודקודים, מתקיים $N(U')=\bigcup_{u\in U'}N(u)$, כלומר קבוצת השכנים של כל U' היא איחוד של קבוצות השכנים של כל הקודקודים ב־U'.
  - שידוך: קבוצת צלעות $M\subseteq E$ כך שבכל קודקוד של G נוגעת לכל היותר צלע אחת מ־M.
    * הערה ־ הקבוצה הריקה נחשבת לשידוך.
  - שידוך מושלם: שידוך M יקרא שידוך מושלם אס"ם $|M|=|U|$, כלומר אם כל הצלעות ב־M מחברות כל קודקוד ב־U לקודקוד אחר ב־V (כלומר ־ מהווה התאמה חח"ע ועל בין U ל־V).
  - מסלול מתחלף: בהנתן שידוך M, מסלול מתחלף הוא מסלול פשוט בו הצלעות הן לסירוגין מ־M ומ־$E\backslash M$ (כלומר מזגזג בין צלע מ־M לצלע שלא מ־M).

- משפטים:
  - משפט Hall: בגרף דו צדדי G ישנו שידוך מושלם אס"ם לכל $U'\subseteq U$, מתקיים $|N(U')|\geq|U'|$. כלומר, אם לכל תת קבוצה שניקח של U, נקבל שלכל קודקוד בה יש לפחות שכן אחד.
  - מסקנות מהוכחת משפט Hall: נבחר קודקוד $u\in U'\subseteq U$, וניקח את אוסף המסלולים המתחלפים שמתחילים בו (נסמן ב־V' את קבוצת הקודקודים ב־V שמשתתפים באוסף המסלולים הזה). אז מתקיים:
    * כל הקודקודים ב־V' משודכים לקודקודים ב־U'.
    * מתקיים $N(U')\subseteq V'$ (כל השכנים של U' הם מ־V).

- קלט: גרף דו צדדי $G=(U,V,E)$.
- פלט: שידוך מושלם ב־G.
- אלגוריתם שמבוסס על משפט Hall: משפט $Hall$ מרמז על אלגוריתם למציאת שידוך מושלם (אם קיים):
  - נאתחל שידוך ריק $M\leftarrow\emptyset$.
  - בכל איטרציה נגדיל את M ב־1.
  - אם כל $u\in U$ משודך, M הוא שידוך מושלם ונחזיר אותו. אחרת, נמצא את U' ו־V' ע"י פרוצדורה דמויית BFS:
    * נחלק את G לשכבות ־ השכבות הזוגיות נמצאות ב־U בשמאל, והאי זוגיות נמצאות ב־V בצד ימין.
    * נתחיל בקודקוד u שנמצא בשכבה 0 ונתחיל להתקדם בכל פעם לשכבה הבאה.
      · אם מגיעים לקודקוד y בשכבה אי זוגית (כלומר שנמצא ב־V) ואין המשך לשכבה הבאה, אז מצאנו מסלול מתחלף מקסימלי מ־u ל־y, ונגדיל את M ב־1.
      · אחרת, נקבל כי הקודקודים בשכבות האי זוגיות (ב־V) הם כל השכנים של הקודקודים בשכבות הזוגיות (ב־U). מכיוון שלכל הקודקודים בשכבות האי זוגיות יש המשך, הם משודכים לקודקודים בשכבות הזוגיות, ולכן מספר הקודקודים בשכבות הזוגיות (כלומר $|U|$) גדול ממספר הקודקודים בשכבות האי זוגיות (כלומר $|V|$), וזה מפר את תנאי משפט $Hall$.
  - סיבוכיות זמן: נסמן ב־n את מספר הקודקודים וב־m את מספר הצלעות, אז סיבוכיות הזמן היא $O(mn+n^2)$: מבצעים לכל היותר $n=|U|$ איטרציות של הגדלת m, כל איטרציה כזו לוקחת $O(m+n)$ (מסיבוכיות BFS).
  - סיבוכיות מקום: $O(m+n)$.

- אלגוריתם שמבוסס על רשתות זרימה:
  - נוסיף קודקוד s שמחובר לכל הקודקודים ב־U, וקודקוד t שכל הקודקודים ב־V מחוברים אליו. נמצא ברשת זרימת מקסימום בשלמים, ואז נתאים לה שידוך מקסימום.

מציאת התאמה מקסימלית בגרף דו צדדי (תרגול 6)

- קלט: גרף דו צדדי $G=(L,R,E)$.
- פלט: התאמה מקסימלית ב־G, כלומר, תת קבוצה של הצלעות $M\subseteq E$ כך שאין שתי צלעות ב־M שנוגעות באותו קודקוד. אם יש שידוך מושלם, הוא ההתאמה המקסימלית.
- אלגוריתם:
  - נגדיר $V'=L\cup R\cup\{s,t\}$ (כלומר נוסיף לקודקודים של G שני קודקודים s,t).
  - נגדיר את $E'=\overrightarrow{E}\cup E_L\cup E_R$ שמורכבת מקבוצות הצלעות הבאות:
    * נגדיר $\overrightarrow{E}=\{(u,v)\in E\mid u\in L, v\in R\}$ כך שתכיל את הצלעות מ־E המקורי כך שהן מהכוונות מ־L ל־R.
    * נגדיר את $E_L=\{(s,u)\in E\mid u\in L\}$, כלומר כל הצלעות מקודקוד המקור s ל־L.
    * נגדיר את $E_R=\{(u,t)\in E\mid u\in R\}$, כלומר כל הצלעות מ־$R$ לקודקוד המטרה.
  - נגדיר את פונקצית הקיבול c ע"י $c(e)=1$ לכל $e\in E$.
  - נגדיר את $G'=(V',E')$. כעת נגדיר את רשת הזרימה (G',c,s,t).
  - נריץ על הרשת הזו את אלגוריתם FF למציאת זרימה מקסימלית, נסמן ב־f את הזרימה שהוא החזיר.
  - נשים לב ש־f היא זרימה בשלמים (כיוון שהקיבולים שלמים), ולכן לכל $e\in E$ מתקיים $f(e)\in\{0,1\}$, נחזיר את ההתאמה $M=\left\{e\in\overrightarrow{E}\mid f(e)=1\right\}$, כלומר רק את הצלעות מהגרף המקורי שהועברה בהן זרימה.
- זמן ריצה: $O(|E||V|)$.

בעיית הסוכנים (תרגול 6)

- הקדמה: n סוכני מוסד נמצאים ב־n ערים שונות באמריקה $d_1,..,d_n$, והמוסד רוצה להעביר אותם ל־n ערים שונות באסיה $c_1,..,c_n$ (לא משנה איזה סוכן מגיע לאיזו עיר, העיקר שכל סוכן יגיע לעיר כלשהי). אין טיסות ישירות בין אמריקה לאסיה, לכן כל סוכן חייב לעבור דרך אחת מ־n הערים השונות באירופה, $e_1,..,e_n$. מטעמי בטיחות המוסד דורש שבכל עיר אירופאית לא יהיו יותר מ־k סוכנים.
- קלט:
  - רשימות הערים באמריקה $d_1,..,d_n$, באסיה $c_1,..,c_n$ ובאירופה $e_1,..,e_n$
  - רשימת קווי הטיסות מאמריקה לאירופה $A=\{(d_i,e_j)\mid 1\le i,j\le n,\ \text{there is a flight from } d_i \text{ to } e_j\}$
  - רשימת קווי הטיסות מאירופה לאסיה $B=\{(e_j,c_k)\mid 1\le j,k\le n,\ \text{there is a flight from } e_j \text{ to } c_k\}$
- פלט: "אמת" אם אפשרי להעביר את הסוכנים כך שלא יהיו יותר מ־k סוכנים שונים באותה עיר באירופה, "שקר" אחרת.
- אלגוריתם:
  - נבנה רשת זרימה באופן הבא:
    * ניצור קודקוד s וקודקוד t.
    * נשכפל את הקודקודים של אירופה $e_1,..,e_n$ כך שנקבל שתי קבוצות קודקודים: $\left\{e_1^1,..,e_n^1\right\}$ ו־$\left\{e_1^2,..,e_n^2\right\}$.
    * לכל $l\in[n]$, נמתח צלע מקודקוד s ל־d_l (באמריקה), ומ־c_l (באסיה) ל־t.
    * לכל טיסה ב־A $(d_i,e_j)\in A$, נמתח צלע בין d_i ל־e_j^1.
    * לכל טיסה ב־B $(e_j,c_k)\in B$, נמתח צלע בין e_j^2 ל־c_k.
    * נגדיר את פונקצית הקיבול c שנותנת לכל צלע קיבולת של 1.
  - נריץ אלגוריתם למציאת זרימה מקסימלית ברשת הנ"ל, ונסמן את הזרימה שהוא מחזיר ב־f.
  - אם $|f|=n$ נחזיר "אמת", אחרת נחזיר "שקר".

מציאת דרגת הקשירות של גרף (תרגול 7)

- קלט: $G=(V,E)$ גרף קשיר לא מכוון.
- פלט: $C(G)$, שהוא המספר המינימלי של צלעות אשר הסרתן מן הגרף הופכת אותו ללא קשיר.
- אלגוריתם:
  - נגדיר:
    * $G'=(V,\overrightarrow{E})$ עם $\overrightarrow{E}$ שהוא שכפול כל צלע בגרף המקורי לשתי צלעות מכוונות.
    * c היא פונקצית הקיבול שנותנת לכל צלע קיבול של 1.
    * נבחר קודקוד שרירותי $s\in V$ להיות קודקוד המקור.
  - לכל קודקוד $t\in V$ שאינו s, נמצא חתך מינימלי ברשת בעזרת מציאת זרימה מקסימלית על הרשת (G',c,s,t), ונסמן את הקיבולת שלו ב־$c(t)$.
  - נחזיר את $c(t)$ המינימלי שמצאנו.
- סיבוכיות ריצה: $O(|V|^2|E|^2)$, כיוון שבנינו $O(|V|)$ רשתות זרימה, ומציאת החתך המינימלי בכל רשת לוקח $O(|V||E|^2)$.

בעיית המשקיעים והשחקנים (תרגול 7)

- קלט:
  - $A=\{a_1,..,a_n\}$ קבוצה של שחקנים, כשלכל שחקן נתונה משכורת s_i.
  - $B=\{b_1,..,b_k\}$ קבוצה של משקיעים. לכל משקיע b_iנתונה תת קבוצה $A_i\subseteq A$ של שחקנים שהוא אוהב, ואם כל השחקנים ב־A_iישחקו בסרט, אז הוא יהיה מוכן להשקיע סכום של d_i.
- פלט: תתי קבוצות $A'\subseteq A$ ו־$B'\subseteq B$ כך ש:
  - לכל $b_i\in B$ מתקיים $A_i\subseteq A'$ (כלומר, לכל משקיע, כל השחקנים שהוא אוהב נכללים ב־A')
  - הרווח של A',B' מקסימלי (כאשר הרווח מוגדר להיות $p(A',B')=\sum\limits_{b_i\in B}d_i-\sum\limits_{a_i\in A'}s_i$, כלומר הסכום שכל המשקיעים ב־B ישלמו פחות המשכורות של כל השחקנים ב־A').
- אלגוריתם:
  - נבנה רשת זרימה כך:
    * $V=\{s,t\}\cup A\cup B$ (כלומר נוסיף ל־A,B קודקוד מקור וקודקוד מטרה).
    * $E=\{(s_i,b_i)\mid b_i\in B\}\cup\{(b_i,a_j)\mid a_j\in A_i\}\cup\{(a_i,t)\mid a_i\in A\}$ (כלומר נחבר את קודקוד המקור s לכל משקיע ב־B, נחבר כל משקיע ב־B לכל השחקנים שהוא אוהב ב־A, ונחבר כל שחקן ב־A לקודקוד המטרה t).
    * פונקצית הקיבול הבאה: $c(v,u)=\begin{cases}d_i & u=s\wedge v\in B\\ \infty & u=b_i\in B\wedge v\in A_i\\ s_i & u\in A\wedge v=t\end{cases}$ (כלומר אם הצלע היא בין s למשקיע, הקיבול הוא הסכום שהמשקיע מוכן לשלם. אם היא מחברת בין משקיע לשחקן שהוא אוהב, הקיבול הוא ∞. אם היא מחברת בין שחקן וקודקוד המטרה, הקיבול הוא המשכורת של השחקן).
  - נמצא חתך מינימלי ברשת, נסמנו (S,T).
  - נגדיר $A'=A\cap S$ ו־$B'=B\cap S$, נחזיר את A',B'.
- סיבוכיות ריצה: היא $O(n^2k^2(n+k)$, כיוון שבנינו רשת עם $O(n+k)$ קודקודים ו־$O(nk)$ צלעות, לכן בניית הרשת עולה $O(nk)$, מציאת חתך מינימלי עולה $O(|E|^2|V|)=O(n^2k^2(n+))$, והגדרת הקבוצות A',B' עולה $O(n+k)$.

כיסוי בצמתים

כיסוי בצמתים - הקדמה (הרצאה 15)

- הגדרות:

- <u>כיסוי בצמתים:</u> עבור גרף לא מכוון $G=(V,E)$, קבוצת קודקודים $X\subseteq V$ נקראת כיסוי בצמתים של G אם"ם לכל צלע $e\in E$ מתקיים $e\cap X=\emptyset$.
- <u>קבוצה בלתי תלויה של קודקודים:</u> קבוצת קודקודים I היא בלתי תלויה אם לכל צלע $e\in E$ מתקיים $|e\cap I|\leq 1$ (כלומר - אין צלע שמחברת בין שני קודקודים ב־I).

- <u>טענות:</u>
  - <u>מסקנה (מההגדרות):</u> אם נסיר את X מ־V נקבל קבוצה בלתי תלויה ב־G.
  - <u>מסקנה (מהאלגוריתם של Konig):</u> בגרף דו צדדי, הגודל של כיסוי בצמתים מינימלי זהה לגודל של שידוך מקסימום (זה לא בהכרח נכון עבור גרף כללי).

כיסוי בצמתים מינימלי בגרף דו צדדי (הרצאה 16)

- <u>קלט:</u> גרף דו צדדי $G=(V_L,V_R,E)$ כך שלכל $e\in E$ מתקיים $|e\cap V_L|=|e\cap V_R|=1$ (כלומר כל הצלעות מחברות בין קודקוד מ־V_Lוקודקוד מ־V_R)
- <u>פלט:</u> כיסוי בצמתים בגרף G בגודל מינימלי.
- <u>האלגוריתם של Konig:</u>
  - יהי $M\subseteq E$ שידוך מקסימום ב־G.
  - נגדיר את קבוצת הקודקודים Z שמכילה את כל הקודקודים ב־V_Lשאינם משודכים, ואת הקודקודים שניתן להגיע אליהם מקודקוד לא משודך ב־V_L דרך מסלול מתחלף.
  - נגדיר את קבוצת הקודקודים S, שמכילה את $V_L\backslash Z$ (כלומר הקודקודים המשודכים ב־V_L) ואת $V_R\cap Z$ (כלומר הקודקודים ב־V_Rשמשתתפים במסלול מתחלף שמתחיל בקודקוד לא משודך ב־V_L). אפשר למצוא את S ע"י BFS.
- <u>סיבוכיות זמן:</u> הסיבוכיות היא $O(|V|+|E|)$ - חישוב שידוך מקסימום M לוקח $O(|V||E|)$, מציאת הקבוצה Z באמצעות BFS לוקח $O(|V|+|E|)$, חישוב S דורש מעבר על כל הקודקודים $O(|V|)$.

כיסוי בצמתים מינימלי בגרף כללי (הרצאה 16)

- <u>קלט:</u> גרף כלשהו $G=(V,E)$.
- <u>פלט:</u> כיסוי בצמתים מינימלי.
- <u>אלגוריתם (2-מקרב):</u> נסתמך על טענה לפיה אם M הוא שידוך שלא ניתן להרחבה, אז קבוצת הקודקודים כך שקיימת בשידוך צלע שיוצאת מהם, היא כיסוי בצמתים. באלגוריתם נבחר צלע, נסיר את כל הצלעות שיש להן קודקוד משותף איתה, ונחזור על התהליך עד שלא נותרות צלעות.
- <u>סיבוכיות:</u> סיבוכיות הזמן וסיבוכיות המקום היא $O(|V|+|E|)$.

כיסוי בצמתים קל ביותר בגרף עם קודקודים ממושקלים (הרצאה 16)

- <u>קלט:</u> גרף $G=(V,E)$ ופונקצית משקל חיובית <u>על הקודקודים</u> $w:V\to\mathbb{N}$.
- <u>פלט:</u> כיסוי בצמתים קל ביותר.
- <u>אלגוריתם:</u> הרעיון הוא שהמשקל של כל קודקוד מייצג את המחיר שעולה לקנות אותו, נעבור ונחלק לכל צלע סכום כסף שמאפשר לה לקנות קודקודים, ובסוף נשתמש בכסף הזה כדי לקנות קודקודים (כלומר לבחור את הקודקודים שישתתפו בכיסוי בצמתים).
  - לכל צלע $(u,v)\in E$ נשמור משתנה עזר $y_{u,v}$ (שמייצג את ה"כסף" שהצלע מחזיקה) שמאותחל עם 0.
  - "נחלק את הכסף" בין הצלעות: נעבור על כל הקשתות בסדר כלשהו, ועבור כל קשת $(u,v)\in E$ נעדכן את $y_{u,v}$ באופן הבא:

$$y_{u,v}=min\left\{\overbrace{w(u)-\sum_{s\ |\ (u,s)\in E}y_{u,s}}^{\text{cost of u - sum of money of edges that touch u}},\overbrace{w(v)-\sum_{s\ |\ (s,v)\in E}y_{s,v}}^{\text{cost of v - sum of money of edges that touch v}}\right\}$$

, כלומר נביט על שני הקודקודים בקצה הצלע - לכל קודקוד נחסר מהמחיר שלו את סכום הכסף של הצלעות שנוגעות בו. נבחר את המינימלי ביניהם וניתן את סכום הכסף הזה לצלע.

– "נקנה קודקודים" - נגדיר $S=\left\{u \mid w(u)=\sum_{v \mid (u,v)\in E} y_{u,v}\right\}$, כלומר כל הקודקודים שהמשקל שלהם שווה לסכום הכסף של הצלעות שנוגעות בהם (כלומר - קודקודים שאפשר "לקנות" אותם באמצעות הכסף של הצלעות שנוגעות בהם).

– נחזיר את S.

- <u>סיבוכיות:</u> סיבוכיות הזמן והמקום היא $O(|V|+|E|)$.

- <u>טענות על האלגוריתם:</u>

– <u>טענה:</u> בסיום ריצת האלגוריתם שהחזירה פתרון S, מתקיים $\sum_{u\in S} w_u \le 2 \sum_{v \mid (u,v)\in E} y_{u,v}$, כלומר המשקל הכולל של כל הקודקודים ב־S קטן או שווה לפעמיים סכום הכסף של כל הצלעות בגרף (זכרו שסכום הכסף של צלע מסוימת יכול לשמש פעמיים - כדי לקנות את שני הקודקודים שיוצרים אותה).

– <u>טענה:</u> בסיום ריצת האלגוריתם מתקיים כי $\sum_{(u,v)\in E} y_{u,v}$ (סכום הכסף של כל הצלעות בגרף) קטן או שווה למשקל כיסוי בצמתים קל ביותר.

– <u>מסקנה:</u> זהו אלגוריתם 2־מקרב לבעיית כיסוי בצמתים קל ביותר.

<u>תכנון לינארי</u>

תכנון לינארי - כללי (תרגול 8, הרצאה 19)

- <u>מוטיבציה כללית:</u> גישה לפתרון בעיות (שהיא באופן מסוים הפוכה לגישה של אלגוריתמים חמדניים ותכנון דינאמי) שזה מתחילים באפיון פורמלי של מחלקת בעיות, ואחר כך מנסים להבין אילו אלגוריתמים יכולים לפתור את בעיות האלו. אפילו לא נראה אלגוריתם שפותר בעיות תכנון לינארי, אלא רק נדע שהוא קיים (בסיבוכיות זמן של $O(m+n)$) ונתייחס אליו כ"קופסא שחורה", ונלמד כיצד להציג בעיות בנוסחא של תכנון לינארי כך שאפשר יהיה להכניס אותן לאלגוריתם פותר ($solver$) שכזה.

- <u>הגדרות:</u>

– <u>בעית תכנון לינארי Linear Programming (LP):</u> בעית אופטימיזציה תקרא בעית תכנון לינארי אם ניתן לכתוב אותב הצורה הבאה:

משקולות של הבעיה (מאפשר למשקל חלקים מעניינים מתוך הבעיה/להתעלם מחלקים בבע"ה)

$\in\mathbb{R}^n$

$$\begin{aligned} \max_{x\in\mathbb{R}^n}\ & c^T\cdot x \\ \text{s.t}\quad & Ax\le b \\ & x\ge 0 \end{aligned}$$

$\in\mathbb{R}^n$

אילוצים של הבעיה

מטריצה $m\times n$

∗ כלומר:

· x הוא מה שאנחנו רוצים למקסם בבעיית האופטימיזציה המקורית אותה אנחנו מנסים לפתור באמצעות תכנון לינארי.

· c הוא וקטור משקולות, שמגדיר את המשקל של כל חלק x_i בבעיה אותה אנחנו רוצים למקסם

· A היא מטריצת אילוצים ו־b הוא וקטור חסם עליון לאילוצים, כלומר - כשנכפול את A ב־x נקבל וקטור, שנרצה שכל קואורדינטה בו תהיה קטנה מהקואורדינטה התואמת ב־b.

∗ שני אי השוויונים מוגדרים קואורדינטה־קואורדינטה / $element-wise$ (כלומר אי השוויון צריך להתקיים לכל קואורדינטה i)

∗ ניתן להגדיר את בעית התכנון הלינארי גם כבעיית מינימיזציה:
$$\begin{aligned} min\quad & c^\top x \\ s.t.\quad & Ax\ge b \\ & x\ge 0 \end{aligned}$$

– <u>על־מישור Hyperplane:</u> יהיו וקטור ממשי $a\in\mathbb{R}^n$ וסקלר $b\in\mathbb{R}$. על מישור הוא הקבוצה $\left\{a\in\mathbb{R}^n \mid a^\top x=b\right\}$ (כלומר כל הנקודות שנמצאות בדיוק במישור b - זאת הכללה של קו ליותר מימדים).

- חצי־מרחב Halfspace: יהיו וקטור ממשי $a \in \mathbb{R}^n$ וסקלר $b \in \mathbb{R}$. חצי מרב הוא הקבוצה $\{a \in \mathbb{R}^n \mid a^\top x \le b\}$ (כלומר כל הנקודות שנמצאות על העל־מישור או מתחתיו).
- פוליהדרון Polyhedron: חיתוך בין חצאי מרחבים ־ קבוצת נקודות שניתן לתאר בצורה $\{x \in \mathbb{R}^n \mid Ax \le b\}$ עבור $A \in \mathbb{R}^{m \times n}$ ו־$b \in \mathbb{R}^m$.

  * אם נסמן את השורה ה־j של A ע"י a_j ואת האיבר ה־j ב־b ע"י b_j, נראה שלמעשה כל חיתוך של m חצאי מרחבים מגדיר פוליהדרון. לכן קבוצת הפתרונות החוקיים של בעית תכנון לינארי $\{x \ge 0 \mid Ax \le b\}$ היא פוליהדרון הבנוי מחיתוך של $m+n$ חצאי מרחבים, לכן תכנון לינארי הוא למעשה מיקסום של פונקציה לינארית על פני פוליהדרון.

בעיית התרמיל השברי כבעיית תכנון לינארי (תרגול 8)

- הקדמה: זאת בעיה דומה מאוד לבעית התרמיל השלם שראינו בפרק של תכנון דינאמי, רק שבניגוד לבעיה המקורית (בה הוספנו לתרמיל חפצים בשלמותם), בבעיה הזו מותר להוסיף חלקי חפצים. גם בבעיה זו, עלינו למלא התרמיל כך שנשיג משקל מקסימלי, מבלי לעבור את נפח התרמיל.

  - הערה: לא ניתן לפתור את בעית התרמיל השלם באמצעות תכנון לינארי, כיוון שכדי לאלץ את הפתרונות להיות שלמים, עלינו להכניס אילוץ נוסף ־ $x \in \{0,1\}^n$ (וקטור באורך n של אפסים ואחדות). לבעיה שבה המשתנים הם וקטורים של שלמים קוראים בעית תכנון לינארי בשלמים (ILP - Integer Linear Programming) ־ לא נכנס אליה בקורס הזה. זוהי בעיה NP־ קשה (כלומר לא ניתן לפתור אותה בזמן פולינומיאלי, ואם מישהו יצליח לפתור אותה בזמן פולינומיאלי הוא יזכה בפרס טיורינג ובמיליון דולר).

- קלט: תרמיל בנפח V, ו־n חפצים, כאשר לכל חפץ יש משקל w_i ונפח v_i.
- פלט: וקטור $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ שלכל $i \in [n]$, הקואורדינטה $x_i \in [0,1]$ מייצגת כמה אחוזים מהפריט ה־i הוספנו לתרמיל, כך שמשקל כל החפצים שהוספנו מקסימלי, ונפחם אינו עולה על V. כלומר, הבעיה היא: $\begin{array}{ll} max & \sum_{i=1}^{n} w_i x_i \\ s.t. & \sum_{i=1}^{n} v_i x_i \le V \\ & x_i \in [0,1] \end{array}$.
- ביטוי הבעיה כבעית תכנון לינארי: כדי לבטא את הבעיה בפורמט של תכנון לינארי, נגדיר את A,b,c באופן הבא:

  - וקטור משקולות $c = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ (כלומר הקואורדינטה ה־i תכיל את המשקל של החפץ ה־i), כעת $max(c^\top x) = max \sum_{i=1}^{n} w_i x_i$, כלומר c באמת מייצג את המשקול שלפיו נקבעת האופטימליות של הפתרון.
  - מטריצת אילוצים $A = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ 1 & 0 & & 0 \\ 0 & 1 & & 0 \\ & & \ddots & \\ 0 & 0 & & 1 \end{pmatrix}$ (כלומר השורה ה־0 תכיל את הנפחים של החפצים ו"תטפל" באילוץ $\sum_{i=1}^{n} v_i x_i \le V$, ולכל $i \in [1,n]$ העמודה ה־i תכיל 1 בשורה ה־i ואפס בשאר השורות ־ זה בעצם I_n).
  - חסם עליון לאילוצים $b = \begin{pmatrix} V \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$ (כלומר הקואורדינטה הראשונה תכיל את נפח התרמיל ־ כדי לוודא שהשורה הראשונה ב־A, כלומר הנפחים של כל החפצים, לא עוברים אותו. שאר הקואורדינטות יכילו 1, כדי לאכוף את האילוץ ש־$x_i \le 1$ לכל i).

מציאת ערך מקסימלי ברשת זרימה כבעיית תכנון לינארי (תרגול 8)

- קלט: רשת זרימה (G, c, s, t) עם זרימה f, כשלכל צלע (i, j) נגדיר את $f_{i,j} \in \mathbb{R}^{|E|}$ בתור הזרימה בצלע הזו.
- פלט: זרימה חוקית מקסימלית ברשת, כלומר - $\max_{f\in\mathbb{R}^{|E|}} \sum_{(s,i)\in E} f_{s,i}$, תחת האילוצים הבאים:
  - אי שליליות: לכל $(i, j) \in E$ מתקיים $f_{i,j} \geq 0$ (לא צריך לטפל בו במטריצת האילוצים שניצור כיוון שה־$solver$ יודע לטפל בו).
  - אילוץ הקיבול: לכל $(i, j) \in E$ מתקיים $f_{i,j} \leq c(i, j)$ (כלומר הזרימה לא עוברת את הקיבול של הצלע)
  - שימור הזרימה: לכל $x \in V \backslash \{s, t\}$ $\sum_{(x,i)\in E} f_{u,i} = \sum_{(i,x)\in E} f_{i,u}$(כלומר הזרימה שנכנסת לקודקוד u זהה לזרימה שיוצאת ממנו).
    - נזהה שאילוץ שימור הזרימה יוצר אילוץ שהוא שוויון, ולא אי שיוויון. נמיר זאת לצורת אי שוויון (שהיא צורת האילוצים בתכנון לינארי) באופן הבא: $\forall i \in V \backslash \{s, t\} : \sum_{(x,i)\in E} f_{x,i} - \sum_{(i,x)\in E} f_{i,x} \leq 0$. כלומר, פשוט המרנו את השוויון לשני אי שוויונות - במקום $X = Y$ כתבנו $X \leq Y$ וגם $X \geq Y$ ואז העברנו אגפים כדי לקבל אי"ש ביחס לאפס.

- ביטוי הבעיה כבעיית תכנון לינארי: (התאמצתי מאוד להבין מה קורה פה אבל זה לא באמת הוסבר לצערי)
  - וקטור המשקולות c כך שכל קואורדינטה בו מוגדרת ע"י $\mathbb{R}^{|E|} \ni c_{i,j} = \begin{cases} 1 & i = s \\ 0 & else \end{cases}$
    - (כיוון אנחנו מתעניינים רק בזרימה של הצלעות שיוצאות מ־s, כעת אם נכפול את הוקטור הזה בוקטור x, נשאיר את כל הקואורדינטות ב־x שמייצגות צלעות שיוצאות מ־s כמו שהן, ונאפס את כל האחרות).
  - מטריצת האילוצים $\in \mathbb{R}^{(|E|+|V|-2)\times|E|}$ $A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \hline & & M & & \end{pmatrix}$ כאשר M היא המטריצה $M_{j,k} = \begin{cases} +1 & i, k = i \\ -1 & j = i \\ 0 & \text{else} \end{cases}$

    (עם i מאיך שהגדרנו את אילוץ הקיבול $\forall i \in V \backslash \{s, t\} : \sum_{(i,x)\in E} f_{i,x} - \sum_{(x,i)\in E} f_{x,i} \leq 0$).
  - חסם עליון לאילוצים $\in \mathbb{R}^{|E|+|V|-2}$ $b = \begin{pmatrix} | \\ c_{ij} \\ | \\ 0 \\ \vdots \\ 0 \end{pmatrix}$

בעיית הסרת משולשים (תרגול 8)

- קלט: גרף לא מכוון $G = (V, E)$
- פלט: תת קבוצה מינימלית של צלעות $S \subseteq E$ שהסרתן תשאיר גרף $G' = (V, E \backslash S)$ ללא משולשים (כאשר משולש מוגדר להיות קליקה בגודל 3 / מעגל עם 3 צלעות).
- נסיון לפתרון:
- הצגת הבעיה כבעיית תכנון לינארי: זוהי בעיה NP־קשה. נציג את בעית הסרת המשולשים כבעיית תכנון לינארי בשלמים (שלא ניתן לפתור אותה בזמן פולינומי, אך ניתן לקרב את הפתרון האופטימלי שלה):
  - נגדיר וקטור משתנים $x \in \{0, 1\}^{|E|}$ שמציין לכל $i \in [|E|]$ אם הצלע e_i הוסרה מהגרף ($x_i = 1$) או לא הוסרה מהגרף ($x_i = 0$).
  - הבעיה:

* $min \sum_{e_i \in E} x_i$ (נזהה שסכום הקואורדינטות של x הוא $|S|$ (כיוון ש־S מכילה את הצלעות שהסרנו, שעבורן הקואורדינטה היא 1, ושאר הקואורדינטות הן 0). לכן, נגדיר את פונקצית המטרה להיות $min \sum_{e_i \in E} x_i$).

- <u>האילוצים:</u>
  * לכל משולש ב־G, לפחות אחת מהצלעות מקיימת $x_i = 1$ (כדי שנסיר אותה ונפרק את המשולש). פורמלית, לכל שלושה קודקודים $u, v, w \in V$ כך שהצלעות $(u,v),(v,w),(w,u) \in E$, יתקיים $x_{(u,v)} + x_{(v,w)} + x_{(w,u)} \geq 1$.
  * לכל צלע $e \in E$ יתקיים $x_e \in \{0,1\}$, ובפרט x_e שלם.
- <u>אלגוריתם מקרב לבעיה:</u>
  * כדי להפוך את הבעיה לבעיית תכנון לינארי רגילה (כלומר לא לבעיה של תכנון בשלמים), נחליף את האילוץ השני באילוץ $x_e \in [0,1]$, כלומר נסכים לקבל ערך שברי ל־x.
  * כעת זו בעיית תכנון לינארי ונוכל להכניס אותה ל־$solver$. איך נתרגם את התוצאה שה־$solver$ יחזיר ככה שהיא תתאים חזרה לבעיה המקורית (לפני שינוי האילוץ שביצענו): לכל קואורדינטה בוקטור שה־$solver$ החזיר, אם היא גדולה מסף מסוים (לדוגמא, $\frac{1}{3}$), נהפוך אותה להיות 1 ונסיר את הצלע, אחרת - נשאיר את הצלע. זה נותן לנו (במקרה שהסף הוא אכן $\frac{1}{3}$) אלגוריתם 3־מקרב לבעיה המקורית.

<u>אלגוריתמי קירוב</u>

אלגוריתמי קירוב - הגדרות (תרגול 9)

- <u>בעיית אופטימיזציה:</u> בעיה חישובית של מציאת פתרון מיטבי (אופטימלי) מבין פתרונות חוקיים לבעיה נתונה.
- <u>אלגוריתם מקרב:</u> יהי X מרחב הפתרונות החוקיים לבעיית אופטימיזציה כלשהי. תהי פונקציה $f : X \to \mathbb{R}^+$, ויהי $c \geq 1$.
  - <u>אלגוריתם c־מקרב לבעיית מינימיזציה:</u> נסמן ב־$x_{opt} \in X$ את הפתרון האופטימלי ביחס ל־f, כלומר $argmin_{x \in X} f(x) = x_{opt}$. נאמר שאלגוריתם הוא c־מקרב עבור הבעיה הנתונה אם לכל קלט הוא מחזיר פתרון חוקי $x \in X$ המקיים $f(x) \leq c \cdot x_{opt}$.
  - <u>אלגוריתם c־מקרב לבעיית מקסימיזציה:</u> נסמן ב־$x_{opt} \in X$ את הפתרון האופטימלי ביחס ל־f, כלומר $argmin_{x \in X} f(x) = x_{opt}$. נאמר שאלגוריתם הוא c־מקרב עבור הבעיה הנתונה אם לכל קלט הוא מחזיר פתרון חוקי $x \in X$ המקיים $f(x) \geq \frac{1}{c} \cdot x_{opt}$.
- <u>טיפים:</u>
  - <u>הפרכת קירוב (כלומר להראות שאלגוריתם הוא לא c־מקרב):</u> נמצא דוגמא נגדית עם פונקציה f (שאותה אנחנו רוצים למקסם) שתלויה ב־c, שאפשר להגיע ממנה לשלילת הפתרון, כלומר ל־$f(x) > c \cdot x_{opt}$ (בבעיית מינימיזציה) או ל־$f(x) < \frac{1}{c} \cdot x_{opt}$ (בבעיית מקסימיזציה).

בעיית חתך גדול ביותר Max Cut (הרצאה 21, תרגול 9)

- <u>קלט:</u> $G = (V, E)$ גרף לא מכוון.
- <u>פלט:</u> חתך (A, B) בו מספר הצלעות מקסימלי.
- <u>אלגוריתם 2- מקרב:</u> כיוון שזו בעיה NP קשה, נמצא לה אלגוריתם מקרב:
  - <u>אלגוריתם:</u>
    * נמספר את הקודקודים $V = \{v_1, .., v_n\}$.
    * נאתחל את קבוצות הקודקודים A, B באופן טריויאלי - A תכיל את כל הקודקודים ב־V, ו־B תהיה ריקה.
    * נעבור על הקודקודים לפי סדר המספור. לכל קודקוד, אם מספר השכנים שלו <u>בקבוצה שלו</u> **גדול** ממספר השכנים שלו בקבוצה השניה, נעביר אותו לקבוצה השניה.
    * נחזור על השלב הקודם עד שנסיים איטרציה שלמה מבלי להעביר קודקוד.
  - <u>זמן ריצה:</u> $O(|E| \cdot (|E| + |V|))$ - נבצע לכל היותר $|E|$ איטרציות שבכל אחת מהן נעבור על כל הקודקודים וכל הצלעות.

בעיית ה־3SAT (תרגול 9)

- הקדמה: זוהי בעית הכרעה, בה נכריע האם קיימת השמה מספקת לנוסחת $3-CNF$ שמוגדרת באופן הבא:
  - משתנים בוליאנים: $x_1,..,x_n$ כאשר לכל $i\in[n]$ מתקיים $x_i\in\{\mathbb{T},\mathbb{F}\}$.
  - ליטרלים: משתנה x, או השלילה שלו $\neg x$.
  - פסוקית $3-CNF$: אוסף של שלושה ליטרלים שמחוברים בדיסיונקציה (יחס "או" $\vee$), למשל $(x_1\vee\neg x_2\vee x_3)$
  - נוסחת $3-CNF$: אוסף של פסוקיות מחוברות בקוניונקציה (יחס "וגם" $\wedge$), למשל $C=(x_1\vee\neg x_2\vee x_3)\wedge(x_2\vee x_3\vee x_5)\wedge(\neg x_1\vee\neg x_4\vee x_6)$.
- קלט: נוסחת $3-CNF$ בעלת m פסוקיות $C_1,..,C_m$ מעל n המשתנים $x_1,..,x_n$, ונניח כי מתקיים $n\leq 3m$ (כלומר שיש יותר פסוקיות ממשתנים) ושכל פסוקית מורכבת משלושה משתנים שונים.
- פלט: השמה ל־$x_1,..,x_n$ שממקסמת את מספק הפסוקיות שמקבלות ערך $\mathbb{T}$.
- אלגוריתם נאיבי 2־מקרב:
  - אלגוריתם:
    - נבדקו כמה פסוקיות מסתפקות ע"י ההשמה $\overrightarrow{x_{\mathbb{F}}}=(\overbrace{\mathbb{F},\mathbb{F},\ldots,\mathbb{F}}^{n\text{ times}})$, ונסמן ערך זה ב־$f$.
    - נבדקו כמה פסוקיות מסתפקות ע"י ההשמה $\overrightarrow{x_{\mathbb{T}}}=(\overbrace{\mathbb{T},\mathbb{T},\ldots,\mathbb{T}}^{n\text{ times}})$, ונסמן ערך זה ב־$t$.
    - אם $f<t$ נחזיר את ההשמה $\overrightarrow{x_{\mathbb{T}}}$, אחרת נחזיר את ההשמה $\overrightarrow{x_{\mathbb{F}}}$.
  - זמן ריצה: אנחנו עושים 2מעברים על הנוסחאות ועושים בהן השמה לכל אחד מ־n המשתנים. מההנחה ש־$n\leq 3m$, נקבל שמספר הפעולות הוא $2n=O(m)$.

אלגוריתם 2־מקרב לבעיית התרמיל השלם (תרגול 9)

- קלט: תרמיל בנפח שלם V, וקבוצה של n חפצים, כאשר לחפץ ה־$i\in[1,n]$ יש נפח (המסומן v_i) ומשקל שלם (המסומן $w_i\in\mathbb{N}$).
- פלט: קבוצה חוקית של חפצים (שלמים, כלומר לא ניתן לקחת חלקי חפצים) שסכום המשקלים שלהם מקסימלי, והנפח שלהם לא גדול מ־V (נפח התרמיל). כלומר, $argmax_{P\subseteq\{1,..,n\}}\left(\sum\limits_{i\in P}w_i\ s.t.\ \sum\limits_{i\in P}v_i\leq V\right)$.
- אלגוריתם:
  - האלגוריתם החמדן אינו 2־מקרב: האלגוריתם החמדן (האלגוריתם הגרוע השני שהצענו כשדיברנו על בעית התרמיל השלם בפרק של תכנון לינארי, שממיין את החפצים לפי משקל סגולי, עובר עליהם לפי הסדר ומוסיף כל חפץ לתרמיל אם יש מקום בשבילו) אינו 2־מקרב לבעיה. אפשר להוכיח את זה באופן הבא:
    - בהנתן c, נבחן נפח V כך ש־$V-1>c$. כעת אם ניקח פריט x_1 עם משקל ונפח 1 (כלומר משקל סגולי 1), ופריט x_2 עם משקל $V-1$ ונפח V (כלומר משקל סגולי $1-\frac{1}{V}$), האלגוריתם החמדן יוסיף רק את x_1 (במשקל 1).
    - בשונה מהפתרון האופטימלי שיוסיף את x_2 (במשקל $V-1$), כלומר $f(x_{opt})=(V-1)=(V-1)\overbrace{f(x_1)}^{=1}$, ולכן $f(x_1)=\frac{1}{V-1}f(x_{opt})<\frac{1}{c}f(x_{opt})$ והאלגוריתם אינו c־מקרב עבור כל c שנבחר.
  - שיפור לאלגוריתם החמדן שהופך אותו ל־2־מקרב:
    - אלגוריתם:
      - נמיין את החפצים לפי המשקל הסגולי שלהם $p_i=\frac{w_i}{v_i}$
      - נמצא את הערך k המינימלי שמקיים $\sum\limits_{i=1}^{k}v_k>V$ (כלומר האיבר הראשון שכשמוסיפים אותו חורגים מהנפח של התרמיל)
      - נחזיר את הפתרון x מבין $x_1=\begin{pmatrix}\overbrace{1,...,1}^{k-1\ times},0,,,0\end{pmatrix}$ ו־$x_2=\begin{pmatrix}\overbrace{0,...,0}^{k-1\ times},1,0,,,0\end{pmatrix}$ שממקסם את הפתרון, כלומר שמקיים $f(x)=max\left\{f(x_1),f(x_2)\right\}$.

בעיות סיווג - Online Learning

בעיות סיווג - כללי (תרגול 10)

- מוטיבציה: בבעיות סיווגת המטרה היא לחלק קבוצה של עצמים לתתי קבוצות (לדוגמא - לסווג חתולים לקבוצות לפי הצבע שלהם). נתמקד בקורס בבעיות סיווג בינארי, כלומר בבעיות בהן צריך לחלק עצמים ל-2 קבוצות, או להכריע לכל עצם האם הוא מקיים תנאי (שייך לקבוצה ראשונה) או לא מקיים אותו (שייך לקבוצה שניה).

בעיית המומחים - אלגוריתם halving, רוב ממושקל, משקול כפלי (הרצאות 17-18, תרגול 10)

- קלט:
  - T עצמים המסומנים $x_1,..,x_T\in\mathbb{X}$, כך שלכל חפץ x_i יש סיווג נכון $y_i\in\{-1,1\}$ (כלומר הסיווג יהיה 1 אם הוא שייך לקבוצה הראשונה, ומינוס 1 אם הוא שייך לקבוצה השניה) אותו נרצה למצוא.
  - N מומחים המסומנים $f_1,..,f_N$ שיודעים לקבל חפץ ולהגיד מה לדעתם הסיווג שלו - $f_i:\mathbb{X}\rightarrow\{-1,1\}$. לא ידוע מראש כמה המומחים טובים (כלומר, תוחלת השגיאה שלהם אינה ידועה).
  - בעיה במסגרתה עורכים T סיבובים של ניסוי מסוים באופן הבא:
    * מאתחלים את מספר הטעויות ל-0.
    * לכל סיבוב $t\in[T]$:
      - מקבלים אובייקט x_t ורשימה של סיווגים שהחזירו N המומחים: $f_1^t,...,f_N^t$.
      - נעזרים (בשלב זה עוד לא הגדרנו איך) בסיווגי המומחים כדי לקבוע את$\hat{y}_t$, שהוא הסיווג המשוער שלנו עבור x_t.
      - משווים את הסיווג שיצרנו לסיווג האמיתי y_t, ואם הם לא זהים מוסיפים 1 למספר הטעויות.
      - נשמור את הסיווג $\hat{y}_t$
- פלט: סיווגים $\hat{y}_1,..,\hat{y}_T$ כך שמספר הטעויות יהיה מינימלי.
- אלגוריתם halving:
  - מאתחלים את מספר הטעויות ל-0, ואת רשימת המומחים $experts_0=f_1,...,f_N$ כך שתכיל את כל המומחים.
  - לכל סיבוב $t\in[T]$:
    * נקבל אובייקט x_t ורשימה של סיווגים $\hat{y}_1,..,\hat{y}_k$ שהחזירו המומחים ברשימת המומחים $experts_{t-1}$.
    * נבחר את$\hat{y}_t$ להיות הסיווג שרוב המומחים בחרו בו, כלומר

$$\hat{y}_t=\begin{cases}1 & |\{f_i\in\text{ experts }_t:f_i^t=1\}|\geq|\{f_i\in\text{ experts }_t:f_i^t=-1\}|\\ -1 & \text{otherwise}\end{cases}.$$

    * משווים את הסיווג שיצרנו לסיווג האמיתי y_t, ואם הם לא זהים מוסיפים 1 למספר הטעויות.
    * נעדכן את רשימת המומחים כך שתכיל רק את המומחים שצדקו בסיבוב הנוכחי - $\text{ experts }_t=\{f_i:f_i^\tau=y_\tau,\forall\tau=1,\ldots,t-1\}$.
- אלגוריתם רוב ממושקל:
  - מאתחלים את מספר הטעויות ל-0. קובעים לכל מומחה f_i את המשקולת w_i, ומאתחלים את כל המשקולות ל-1 $w_i^1=1$.
  - לכל סיבוב $t\in[T]$:
    * נקבל אובייקט x_t ורשימה של סיווגים $\hat{y}_1,..,\hat{y}_N$ שהחזירו המומחים ברשימת המומחים.
    * נבחר את$\hat{y}_t$ להיות ההחלטה הממושקלת של כל המומחים, כלומר - $\hat{y}_t=\operatorname{sign}\left(\sum_i w_i^t f_i^t\right)$ (כיוון שהחיזוי צריך להיות 1 או 1- ופונקצית הסימן יכולה להחזיר 0, נקבע שרירותית שאם היא מחזירה 0, נקבע את החיזוי להיות 1).
    * משווים את הסיווג שיצרנו לסיווג האמיתי y_t, ואם הם לא זהים מוסיפים 1 למספר הטעויות.
    * נעדכן את המשקולות של המומחים כך שמשקלו של כל מומחה שטעה יקטן פי 2 - $w_i^{t+1}=\frac{1}{2}w_i^t$.
- אלגוריתם משקול כפלי:
  - רעיון: מה שמייחד את האלגוריתם הזה הוא שני רעיונות:

* שימוש בקנסות: במקום לספור לכל מומחה כמה פעמים הוא טעה, יהיו טעויות גרועות יותר וגרועות פחות. כלומר, אחרי שקיבלנו את הסיווג מהמומחה ה־i, ניתן לו קנס חיובי אם טעה (בגובה לכל היותר 1, וכתלות בחומרת הטעות) וקנס שלילי / פרס אם צדק (בגובה עד 1־).
* בחירת מומחה בודד בכל סיבוב באופן הסתברותי:
  · ניתן לכל מומחה משקל, שמייצג את הסיכוי לבחור אותו (משקל גדול = סיכוי גבוה להבחר) ומתעדכן בכל סיבוב כתלות בכמה המומחה היה טוב (כלומר, מתחילים כשלכל המומחים יש משקל שווה, וככל שמומחה יצדק ביותר סיבובים, המשקל שלו יעלה).
  · בכל סיבוב, כדי לבחור את החיזוי שלנו, נבחר באופן הסתברותי (התלוי במשקל) מומחה אחד שלו נקשיב.
  · לאחר שבחרנו מומחה, נשתמש בחיזוי שלו ונעדכן את הקנס של עצמנו להיות תוחלת הקנסות של המומחה הזה אם היינו משתמשים רק בו לאורך כל T הסיבובים.
  · בסיום הסיבוב, נעדכן את המשקל של כל המומחים לקראת הסיבוב הבא (כדי שהמשקל שלהם ייצג גם את העובדה שהם צדקו / טעו בסיבוב הנוכחי), באופן שלוקח בחשבון את המשקל הנוכחי שלהם (שכאמור משקלל כמה הם צדקו בכל הסיבובים עד כה), את הקנס שלהם מהסיבוב הנוכחי, ואת המידה בה אנחנו רוצים "להעניש" מומחים על טעויות (כלומר כמה אנחנו סלחנים / נוקשים על טעויות).

– מוטיבציה ומשמעות הרעיונות היחודיים: האלגוריתם משיג חסם עליון יותר הדוק מהאלגוריתמים הקודמים שהצענו. העובדה שהאלגוריתם פועל באופן הסתברותי גורמת לו לעבוד באופן יותר טוב בתוחלת. האופן בו אנחנו נותנים קנסות למומחים הוא יותר מתוחכם מסתם לספור כמה פעמים הם טעו, ומאפשרת לנו להשתמש במידע מוקדם שיש לנו על כל מומחה כדי לבחור באופן חכם יותר את הפתרון בכל שלב ואת המשקל של המומחים שישפיע על השלב הבא.

– אלגוריתם:
* קובעים לכל מומחה f_i את המשקל w_i^1 שמאותחל ל־1 (שיקבע את הסיכוי להגריל את המומחה ־ ככל ש־w_i גדול יותר, הסיכוי להגריל את המומחה גבוה יותר), את הקנס $punishment_i^0$ שמאותחל ל־0.
* לכל סיבוב $t \in [T]$:
  · נקבל אובייקט x_t ורשימה של סיווגים $\hat{y}_1, .., \hat{y}_N$ שהחזירו המומחים ברשימת המומחים.
  · לכל מומחה, נשווה את הסיווג שהוא החזיר לסיווג האמיתי. אם הוא טעה, ניתן לו קנס 1, אחרת ניתן לו קנס 0.
  · נגריל מומחה מתוך קבוצת המומחים בעזרת ההתפלגות הבאה: $p_i = \frac{w_i^t}{\sum\limits_{j \in [N]} w_j^t}$ (כלומר, ההסתברות לבחור את המומחה ה־i היא המשקולת של המומחה ה־i חלקי סכום המשקולות של כל המומחים).
  · נקבע את החיזוי שלנו לסיבוב זה להיות החיזוי שהחזיר המומחה שהגרלנו.
  · כדי לקבוע מה הקנס שלנו (מריצי האלגוריתם) בסיבוב הזה, נדמה מצב בו הקשבנו למומחה שבחרנו לאורך כל T הסיבובים, כלומר נקבע את הקנס שלנו להיות תוחלת העונש שהוא היה מקבל לאורך T הסיבובים: $\sum\limits_{t=1}^{T} \mathbb{E}_p[punishment_i^t]$.
  · ניקח $\varepsilon \in [\frac{1}{2}]$, כאשר ε מייצג כמה אנחנו "סומכים על הקנסות" ־ ככל ש־ε יותר קטן, זה מעיד שאנחנו לא סומכים על הקנסות, ונאפשר למומחים לטעות יותר (זה למעשה פרמטר רגולריזציה, למי שמכיר מ־IML). הסיבה שהוא צריך להיות קטן מחצי קשורה להוכחת החסם העליון לזמן הריצה של האלגוריתם.
  · לכל מומחה נעדכן את המשקל שלו לסיבוב הבא w_i^{t+1} ע"י הנוסחא הבאה: $w_i^{t+1} = (1-\varepsilon \cdot punishment_i^t) \cdot w_i^t$. המשמעות של הקנס היא שאם העונש חיובי $\leftarrow \varepsilon \cdot punishment_i^t$ חיובי $\leftarrow 1-\varepsilon \cdot punishment_i^t \leq 1 \leftarrow$ נקטין את הסיכוי שהמומחה יבחר בסיבוב הבא. באותו אופן, אם הקנס שלילי ("פרס"), נגדיל את הסיכוי שהמומחה יבחר בסיבוב הבא.

• טענות ומשפטים:

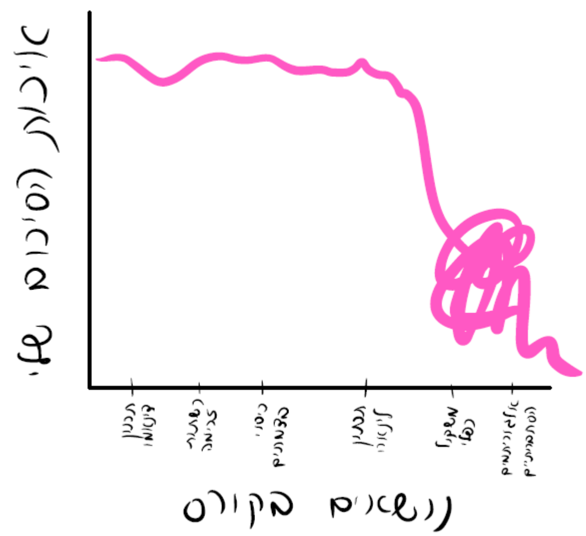
– טענות על אלגוריתם halving:
* טענה: אם יש מומחה מושלם f_i שצודק תמיד, אז מתקיים שמספר הטעויות שיתקבל בשימוש באלגוריתם $halving$ יהיה לכל היותר $logN$.
* טענה: אם המומחה הטוב ביותר שוגה $k \geq 0$ פעמים, אז האלגוריתם שוגה לכל היותר $(k+1)logN$ פעמים.

– טענות על אלגוריתם רוב ממושקל:
* משפט: נסמן ב־$mistakes_i$ את מספר הטעויות הכולל של המומחה f_i, אז לכל i מתקיים כי $mistakes_i \leq \frac{mistakes_i + logN}{log\left(\frac{4}{3}\right)}$.
  · כלומר, מספר הטעויות שנעשה תלויה במספר הטעויות של המומחה הטוב ביותר. נשים לב שאם המומחה הטוב ביותר הוא מומחה מושלם, מספר הטעויות שנעשה הוא בערך $2.41logN$, כלומר אנחנו במצב פחות טוב מאלגוריתם $halving$, אבל הרווחנו אלגוריתם שפועל היטב גם במצב שבו המומחים אינם מושלמים.

– טענות על אלגוריתם משקול כפלי:

* <u>טענה:</u> מספר השגיאות שהאלגוריתם עושה חסום ע"י $\sum_{t=1}^{T} \mathbb{E}_p[punishment_i^t]$, שקטן או שווה ל־ $(1+\varepsilon)\cdot$ $\text{(num of mistakes of best expert)} + \frac{1}{\varepsilon}logN$.

· כלומר, אם המומחה הכי טוב הוא מומחה מושלם ו־$\varepsilon = \frac{1}{2}$, נקבל חסם עליון של $2logN$, שהוא יותר הדוק מהחסם של רוב ממושקל.



(סתם, השתדלתי לסכם טוב גם את סוף הקורס, פשוט כאן רמת ההבנה שלי יורדת משמעותית)

אלגוריתמים הסתברותיים

אלגוריתמים הסתברותיים - כללי (הרצאה 20, תרגול 11)

- <u>מוטיבציה:</u>
  - ישנן בעיות רבות שאיננו יודעים לפתור באופן דטרמיניסטי (כלומר לכתוב להן אלגוריתם שתמיד יחזיר את אותה התשובה עבור אותו קלט). מתוך סל הבעיות האלו, יש מעט שניתן לפתור אותן באמצעות אלגוריתם הסתברותי.
  - אלגוריתמים הסתברותיים יעבדו בצורה טובה כאשר יש אחוז גבוה מספיק של פתרונות נכונים שהוא יכול להחזיר.

- <u>הגדרות:</u>
  - <u>אלגוריתם הסתברותי:</u> אלגוריתם שלכל קלט, בזמן פולינומיאלי, מחזיר בהסתברות גדולה כרצוננו פתרון חוקי ואופטימלי, כאשר יש הסתברות נמוכה שהוא יכשל (כלומר לא ידע להחזיר תשובה).
    * כלומר, אלגוריתם הסתברותי הוא אלגוריתם אשר במהלך ריצתו משתמש ב"הטלות מטבע", כלומר המבצע הגרלות.
  - <u>אלגוריתמי לאס־וגאס:</u> אלגוריתמים שבהם "הטלות המטבע" אינם משפיעות על פלט האלגוריתמיסת אלא על פרמטרים אחרים בריצתו.
    * לדוגמא אם נתון לנו מערך מעורבב שחצי מהערכים בו הם 0 וחצי הם 1 ואנחנו רוצים למצא אינדקס כלשהו שמכיל 1, נוכל להציע אלגוריתם שמגריל אינדקס באקראי (בלי החזרות) עד שהוא מוצא אינדקס מתאים. זמן הריצה במקרה הגרוע הוא $O(n)$ (כי נצטרך לעבור על חצי מערך), אבל זמן הריצה הממוצע (בתוחלת) הוא $O(1)$.
  - <u>אלגוריתמי מונטה קרלו:</u> אלגוריתמים בהם הטלות המטבע משפיעות על הפלט, ובפרט, ריצות שונות על אותו קלט יכולות להסתיים בתוצאה שונה, והאלגוריתם עלול להכשל (כלומר להחזיר תשובה לא נכונה).
    * עבור אותה דוגמא כמו בשורה הקודמת, נציע אלגוריתם שמגריל באקראית אינדקס, אם הוא מכיל 1 יחזיר אותו, ואחרת יחזיר $fail$. כלומר או שהוא צודק בוודאות (בהסתברות כלשהי) או שהוא מחזיר שהוא לא יודע לתת תשובה.
    * אלגוריתמי מונטה קרלו נפוצים במיוחד בבדיקת ראשוניות של מספרים, כיוון שהם מאפשרים לעשות את זה בזמן לוגריתמי.
  - <u>ניפוח:</u> אחד החוזקות של אלגוריתמים הסתברותיים היא שניתן בקלות "לנפח" את ההסתברות לקבלת תשובה נכונה ע"י הרצה חוזרת.

* עבור אותה דוגמא, נציע אלגוריתם שבהנתן $k\in\mathbb{N}$, נריץ את האלגוריתם הקודם $log(e)\cdot k$ פעמים, אם הוא קיבל לפחות פעם אחת אינדקס i, הוא יחזיר אותו, אחרת הוא יחזיר $fail$. הוא טועה בהסתברות לכל היותר $\frac{1}{e^k}$ ורץ בסיבוכיות $O(k)$.

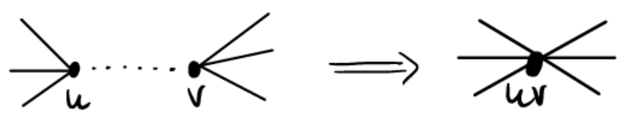
– <u>אלגוריתם קירוב הסתברותי:</u> אלגוריתם שלכל קלט, בזמן פולינומיאלי, מחזיר פלט חוקי ו־c־מקרב בהסתברות גדולה כרצוננו, וקיימת הסתברות נמוכה כי יכשל.

בעיית חתך גדול ביותר Max Cut (תרגול 11)

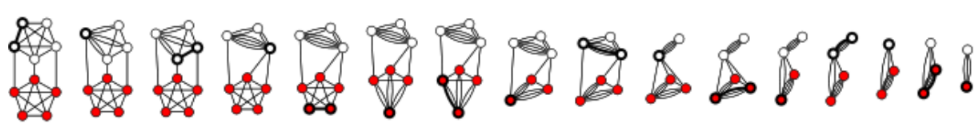
* <u>קלט:</u> גרף לא מכוון $G=(V,E)$
* <u>פלט:</u> חתך $F\subseteq E$ עם מספר צלעות מקסימלי.
* <u>אלגוריתם הסתברותי פשוט לקירוב הבעיה:</u>
  – <u>אלגוריתם:</u>
    * נבחר צלע כלשהי $(u,v)\in E$, ונשים את u ואת v בצדדים שונים של החתך.
    * לכל אחד מהקודקודים האחרים $w\in V\backslash\{u,v\}$: נגריל (באופן בלתי תלוי בהגרלות קודמות, ובהתפלגות אחידה) את אחת משתי קבוצות הקודקודים שיוצרות את החתך.
  – <u>טענות על האלגוריתם:</u>
    * <u>משפט:</u> יהי F החתך שנוצר ע"י האלגוריתם, אזי מתקיים $\mathbb{E}[|F|]>\frac{|E|}{2}$ (כלומר ־ בדרך כלל, כמות הצלעות בפתרון שנחזיר תהיה יותר מחצי מכמות הצלעות בגרף).
    * <u>מסקנה:</u> $\mathbb{E}[|F|]$ הוא קירוב בפקטור 2 לגודל של חתך מקסימום (שמכיל לכל היותר את כל הצלעות).

בעיית חתך מינימום גלובלי Min Cut (הרצאה 22־23)

* <u>קלט:</u> גרף סופי לא מכוון $G=(V,E)$.
* <u>פלט:</u> חתך $F\subseteq E$ שגודלו מינימלי.
* <u>הגדרה ־ כיווץ צלע:</u>
  – תהי צלע $E\ni(u,v)\in e$. כיווץ של e מייצר גרף חדש $G\backslash e=(V',E')$ עם:
    * $V'=V\backslash(\{u,v\}\cup\{uv\}$ כלומר החלפנו את הצלע (u,v) בקודקוד יחיד שנקרא uv.
    * $E'=\{f\in E\quad f\cap\{u,v\}=\emptyset\}\cup\{((uv,u)\in E\mid(U,W)\in E\vee(v,w)\in E)\wedge w\notin\{u,v\}\}$, כלומר ־ הורדנו את הצלעות עם u או v, וחיברנו את השכנים שלהן (כלומר הוספנו צלעות חדשות) עם הקודקוד החדש uv.



* <u>אלגוריתם הסתברותי פשוט ־ האלגוריתם של Karger:</u>
  – <u>אלגוריתם:</u> כל עוד יש ב־G יותר משני קודקודים: נבחר (בהתפלגות אחידה) צלע e ונחליף את G ב־$G\backslash e$ (הכיווץ של e). נחזיר את הקשתות שמחברות בין שני הקודקודים הנותרים.



  – <u>סיבוכיות האלגוריתם:</u> $O(|E|\cdot|V|^2)$.
  – <u>טענות ומשפטים:</u>
    * <u>טענה:</u> אם $F\subseteq E$ הוא חתך מינימום ב־G, אזי $|E|\geq\frac{|V|\cdot|F|}{2}$ או באופן שקול $2|E|\geq|V|\cdot|F|$, כלומר אם נכפול את כמות הצלעות בחתך המינימום בכמות הקודקודים, זה יהיה קטן או שווה לפעמיים כמות הצלעות.
    * <u>מסקנה:</u> לאחר הכיווץ הראשון נקבל כי $P[F\subseteq E(G\backslash e)]\geq1-\frac{2}{|V|}$., כלומר ההסתברות שחתך המינימום מוכל בצלעות שנשארו לאחר הכיווץ הראשון גדולה מ־ $1-\frac{2}{|V|}$.

* <u>משפט:</u> $\prod_{k=3}^{|V|}(1-\frac{2}{k})$ $P[F=F_{min}]\geq$, כלומר ההסתברות ש־F הוא אכן חתך מינימום גדולה או שווה למכפלה $\prod_{k=3}^{|V|}(1-\frac{2}{k})$.
* <u>מסקנה:</u> $P[F=F_{min}]\geq\frac{1}{\binom{n}{2}}$
* <u>מסקנה:</u> בכל גרף G עם n קודקודים, מספר חתכי המינימום הוא לכל היותר $\binom{n}{2}$.
* <u>מסקנה:</u> הסיכוי של האלגוריתם להכשל היא לכל היותר $1-\frac{1}{n^2}$. אם נריץ את האלגוריתם N פעמים וניקח את החתך הקטן ביותר מבין N החתכים שקיבלנו, סיכויי הכשלון הם לכל היותר $(1-\frac{1}{n^2})^N\leq e^{-\frac{N}{n^2}}$.

בעיית Max-3SAT (תרגול 11)

* <u>הקדמה:</u> זוהי בעיית הכרעה, בה נכריע האם קיימת השמה מספקת לנוסחת $3-CNF$ שמוגדרת באופן הבא:
  - <u>משתנים בוליאנים:</u> $x_1,..,x_n$ כאשר לכל $i\in[n]$ מתקיים $x_i\in\{\mathbb{T},\mathbb{F}\}$.
  - <u>ליטרלים:</u> משתנה x, או השלילה שלו $\neg x$.
  - <u>פסוקית $3-CNF$:</u> אוסף של שלושה ליטרלים שמחוברים בדיסיונקציה (יחס "או" $\vee$), למשל $(x_1\vee\neg x_2\vee x_3)$
  - <u>נוסחת $3-CNF$:</u> אוסף של פסוקיות מחוברות בקוניונקציה (יחס "וגם" $\wedge$), למשל $C=(x_1\vee\neg x_2\vee x_3)\wedge(x_2\vee x_3\vee x_5)\wedge(\neg x_1\vee\neg x_4\vee x_6)$.
* <u>קלט:</u> נוסחת $3-CNF$ בעלת m פסוקיות $C_1,..,C_m$ מעל n המשתנים $x_1,..,x_n$, ונניח כי מתקיים $n\leq 3m$ (כלומר שיש יותר פסוקיות ממשתנים) ושכל פסוקית מורכבת משלושה משתנים שונים.
* <u>פלט:</u> השמה ל־$x_1,..,x_n$ שממקסמת את מספק הפסוקיות שמקבלות ערך $\mathbb{T}$.
* <u>אלגוריתם הסתברותי $\frac{8}{7}$־מקרב:</u> (יותר טוב מהאלגוריתם שראינו בפרק של אלגוריתמי קירוב, שהיה 2־מקרב)
  - לכל משתנה x_i :
    * נטיל מטבע הוגן. אם יצא עץ, נשים את המשתנה להיות $\mathbb{T}$.
    * אם יצא פלי, נשים את המשתנה להיות $\mathbb{F}$.
  - אם ההשמה שהגרלנו מספקת לפחות $\frac{7}{8}m$ מהפסוקיות, נחזיר אותה. אחרת, נחזיר $fail$.
* <u>סיבוכיות ריצה:</u> $O(n+m)=O(m)$ ־ n עבור הגרלת המשתנים ו־m עבור בדיקת ההשמה, וידוע כי $n\leq 3m$.

בעיית פתרון מערכת משוואות מודולו 2 ־ Max-Lin2 (הרצאה 22, תרגול 12)

* <u>קלט:</u> אוסף משוואות לינאריות מעל שדה מודולו 2 $\mathbb{F}_2$ (דוגמא למשוואה ־ $X_2\oplus X_3\oplus X_4\oplus X_{30}=0$). אפשר גם לחשוב על זה בתור מטריצה A (שמייצגת את המשוואות) ווקטור תוצאות b כך ש־$Ax=b$.
* <u>פלט:</u> הצבה של ערכי $0,1$ למשתנים עבורה מספר המשוואות שמתקיימות מקסימלי.
  - <u>הערה:</u> זוהי הכללה של $max-cut$:
    * לכל קודקוד $v\in V$ נגדיר משתנה X_v, לכל צלע $(u,v)\in E$ נוסיף משוואה לינארית $X_v\oplus X_u=1$.
    * המשוואה מתקיימת אם"ם $X_u\neq X_v$, לכן פתרון הוא חתך S שמכיל את כל הקודקודים $u\in V$ כך ש־$X_u=0$, ומספר המשוואות שמתקיימות הוא בדיוק מספר הצלעות שנחתכות.
  - <u>הערה:</u> אם נגריל את המשתנים בהתפלגות אחידה ובאופן בלתי תלוי, ההסתברות שמשוואה מתקיימת היא $\frac{1}{2}$ בתוחלת, ובהתאם תוחלת מספר המשוואות המתקיימות הוא $\frac{\text{num of equations}}{2}$.
* <u>אלגוריתם 2־מקרב הסתברותי:</u>
  - <u>אלגוריתם בסיסי:</u> נגריל השמה מקרית $s\in\mathbb{F}_2^n$. אם s מספקת חצי או יותר מהמשוואות, נחזיר את s. אחרת, נחזיר $fail$.
    * <u>הערה:</u> האלגוריתם הבסיסי נכשל בהסתברות קטנה או שווה ל־$\frac{m}{m+1}$.
  - <u>אסטרטגיה לשימוש באלגוריתם:</u>
    * נחשב את תוחלת המשוואות שמסופקות ע"י השמה מקרית.
    * נשתמש באי שוויון מרקוב כדי לקבל חסם מלמעלה על ההסתברות שהאלגוריתם הבסיסי נכשל.

בעיית זיהוי תבניות בטקסט (הרצאה 25)

- קלט:
  - טקסט T (בגודל n) שהוא רצף של אותיות מתוך אלפבית Σ (בגודל d).
  - תבנית p שהיא רצף בגודל m של אותיות מתוך Σ (כאשר $n \gg m$).
- פלט: כל המופעים של p בתוך T.
- אלגוריתם נאיבי:
  - אלגוריתם:
    * נעבור על כל המקומות ב־T שבהם p יכולה להתחיל, כלומר על האינדקסים $i \in [1, n-m+1]$.
    * לכל אינדקס i שכזה, נשווה בין p לבין $T[i, i+m-1]$ שהוא קטע הטקסט שבין האינדקסים i ו־$i+m-1$.
  - סיבוכיות זמן: נבצע $O(nm)$ פעולות השוואת אותיות או גישה לאיברים לפי אינדקס.
  - סיבוכיות מקום: $O(1)$ (שני תאים שמאחסנים את האינדקסים).
- אלגוריתם הסתברותי ־ אלגוריתם Karp-Rabin: הופך את בעיית השוואת המחרוזות לבעיית השוואת מספרים. הרעיון של האלגוריתם דומה ל־$hashing$ ־ אנחנו רוצים להחזיק טבלה שהערכים שלה הם בטווח גדול, אבל להיות מסוגלים לשלוף ממנה m ערכים ביעילות. הערכים אותם אנחנו רוצים לגבב הם החלונות האפשריים שאנחנו מזיזים על גבי הטקסט.
  - הקדמה: ניתן להתייחס ל־p ול־$T[i, i+m-1]$ כאל מספר שרשום בבסיס d: # תהרגו אותי אני לא יודעת מה הוא רוצה
    * $p = d^{m-1} \cdot p[1] + d^{m-2} \cdot p[2] + ... + d^0 p[m]$
    * $t_i = d^{m-1} \cdot T[1] + d^{m-2} T[2] + .. + d^0 T[i+m-1]$
      · מכאן נובע גם ש־$t_i = p$ שקול ל־$T[i, i+i+m-1]$, וגם ש־$t_{i+1} = dt_i - d^m T[i] + T[i+m]$.
      · ההמרה הזאת לא קידמה אותנו מאוד ־ בגלל ש־p ו־t_i הם מספרים גדולים (בגודל עד d^m), הם עלולים לדרוש $logd^m = mlogd$ ביטים כדי לייצג אותם, לכן נצרך $O(nmlogd)$ פעולות כדי לבצע את החיפוש.
  - אלגוריתם:
    * נבחר קבוצה גדולה Q של מספרים ראשוניים, ונגריל מתוכה בהתפלגות אחידה $q \in Q$ ונבצע את כל החישובים מודולו q.
    * נסמן $q_{max} = max(q \in Q)$ ־ אם $logq_{max}$ קטן יחסית לעומת m, החישובים היו יותר יעילים.
  - משפט: מספר המספרים הראשוניים שקטנים מ־X הוא $\frac{X}{ln(X)}(a \pm O(1))$.

בעיית צביעה מקסימלית של גרף ב־3 צבעים (תרגול 12)

- קלט: גרף לא מכוון $G = (V, E)$.
- פלט: צביעה של הקודקודים של G ב־3 צבעים, כך שמספר הצלעות שהקודקודים שהן מחברות צבועים בצבעים שונים, הוא מקסימלי. כלומר, נרצה למצוא פונקציית צביעה $c : E \to \{1,2,3\}$, כך שנמקסם את הגודל של הקבוצה $A = \{(i,j) \in E \mid c(i) \neq c(j)\}$.
  - במקור הבעיה היא להחזיר את מספר הצבעים המינימלי שניתן לצבוע את קודקודיו של G כך שאין זוג קודקודים שכנים החולקים את אותו הצבע ־ אך זו בעיה NP־שלמה ולא ידוע פתרון בזמן פולינומי עבורה, לכן בקורס הזה נתמקד בתת הבעיה שהזכרנו למעלה (בעית אופטימיזציה).
- אלגוריתם בסיסי: הסיכוי שלו להצליח הוא לפחות $\frac{1}{1+|E|}$ (בהנחה שיש לפחות צלע אחת בגרף).
  - לכל קודקוד $v_i \in V$:
    * נגריל צבע בהתפלגות אחידה מבין $c(v_i) = \begin{cases} 1 & w.p\ \frac{1}{3} \\ 2 & w.p\ \frac{1}{3} \\ 3 & w.p\ \frac{1}{3} \end{cases}$.
  - נחשב את A (קבוצת הצלעות שמחברות קודקודים בצבעים שונים) ע"י מעבר על כל הצלעות בגרף.
  - אם $|A| \geq \frac{2}{3}$ נחזיר את c, אחרת $fail$.

- <u>אלגוריתם משופר:</u> מרחיב את האלגוריתם הבסיסי כך שהוא יכשל בהסתברות קטנה כרצוננו, כלומר בהנתן k, נרצה שההסתברות שהאלגוריתם יצליח תהיה לפחות $1-\frac{1}{e^k}$.
  - נחזור על האלגוריתם הבסיסי $k\left(|E|+1\right)$ פעמים.
  - אם האלגוריתם הצליח באחת הריצות, נחזיר את הפונקציה c שהאלגוריתם הבסיסי החזיר בריצה שהצליחה. אחרת, נחזיר $fail$.
- <u>זמן ריצה של האלגוריתם המשותף:</u> צביעת כל הקודקודים לוקחת $O(|V|)$, חישוב A לוקח $O(|E|)$, לכן האלגוריתם הבסיס לוקח $O(|V|+|E|)$. כיוון שאנחנו חוזרים על האלגוריתם הבסיסי $k|E|$ פעמים, נקבל בסה"כ שהסיבוכיות היא $O(k|E|^2+k|E|\cdot|V|)$.

בעיות בדיקת זהות של פולינומים מרובי משתנים (הרצאה 23-24, תרגול 13)

- <u>הגדרות:</u>
  - <u>פולינום ב־n משתנים:</u> פולינום בעל n משתנים מעל שדה $\mathbb{F}$ הוא פונקציה $p:\mathbb{F}^n\to\mathbb{F}$ מהצורה $P\left(x_1,\ldots,x_n\right)=\sum_{i_1=0}^{d_1}\cdots\sum_{i_n=0}^{d_n}a_{i_1,\ldots,i_n}\cdot x_1^{i_1}\cdot\ldots\cdot x_n^{i_n}$ כאשר $d_1,..,d_n\in\mathbb{N}$.
  - <u>מונום:</u> בהמשך להגדרה הקודמת, כל איבר מהצורה $a_{i_1,\ldots,i_n}x_1^{i_1}\cdots x_n^{i_n}$ נקרא מונום. לכל מונום יש <u>מקדם</u> $a_{i_1,\ldots,i_n}$ ו<u>משתנים</u> $x_1^{i_1}\cdots x_n^{i_n}$. ככלל, נתעלם ממונומים עבורם המקדם הוא 0 שכן הם אינם משפיעים על הפולינום.
  - <u>דרגה של מונום (עם מקדם שאינו אפס):</u> סכום חזקות המשתנים בו: $\deg\left(a_{i_1,\ldots,i_n}\cdot x_1^{i_1}\cdot\ldots\cdot x_n^{i_n}\right)=\sum_{j=1}^{n}i_j$
  - <u>דרגה של פולינום:</u> המקסימום של הדרגות של כל המונומים בו: $\deg\left(P\left(x_1,\ldots x_n\right)\right)=\max_{i_1,\ldots i_n}\left\{\sum_{j=1}^{n}i_j\mid a_{i_1\ldots i_n}\neq 0\right\}$.
  - <u>שורש של פולינום:</u> איבבר $x\in\mathbb{F}^n$ כך שמתקיים $p(x)=0$.
  - <u>פולינום האפס:</u> פולינום p הוא פולינום האפס אם"ם בהצגה של p כסכום מונומים, כל המקדמים הם 0 / אם מתקיים לכל $x\in\mathbb{F}^n$ כי $p(x)=0$.
    - <u>טענה:</u> פולינום הוא פולינום האפס אם"ם כל המקדמים בו הם אפס.
- <u>קלט:</u> פולינום רב משתנים p ממעלה d, בייצוג שאפשר חישוב יעיל של ערכו בהנתן הצבה למשתנים.
- <u>פלט:</u> "אמת" אם p הוא פולינום האפס, ו"שקר" אחרת.
- <u>אלגוריתם בסיסי:</u>
  - <u>אלגוריתם:</u>
    - נבצע את הפעולות הבאות M פעמים:
      - נבחר מתוך $\mathbb{F}$ קבוצה A שגודלה $|A|=2d$ (שימו לב שיש כאן דרישה מובלעת - שב־$\mathbb{F}$ יהיו לכל הפחות $2d$ איברים).
      - נגריל הצבה מקרית מ־A ונציב אותה ב־p.
    - אם באחת ההצבות קיבלנו ערך שונה מאפס, נחזיר "שקר", אחרת נחזיר "אמת".
      - (שימו לב שאם מחזירים "שקר" אז האלגוריתם בטוח צדק, כלומר זה בוודאות לא פולינום האפס, אך אם מחזירים "אמת" יש סיכוי שהאלגוריתם טעה - ההסתברות היא לכל היותר 2^{-M}).
  - <u>משפט Schwartz-Zippel:</u> מראה שהאלגוריתם הבסיסי טועה בהסתברות קטנה או שווה ל־$\frac{deg(p)}{|A|}$:
    - יהי פולינום $p\in\mathbb{F}[x_1,..,x_n]$ שאינו פולינום האפס, ותהי A קבוצה סופית של איברי $\mathbb{F}$.
    - נגריל הצבה $x_i=a_i$ לכל $i\in[n]$, כאשר כל ה־a_i־ים תלויים זה בזה, וכל אחד מהם מתפלג אחיד ב־A.
    - אזי $P\left[p(a_1,..,a_n)=0\right]\le\frac{deg(p)}{|A|}$, כלומר ההסתברות ש־$p$ הוא פולינום האפס קטנה או שווה ל־$\frac{deg(p)}{|A|}$.
      - <u>אינטואיציה למשפט:</u> עבור פולינום <u>שאינו</u> פולינום האפס, אם נבחר קבוצה A מספיק גדולה (כלומר גדולה משמעותית מ־$(deg(p))$, ונגריל הצבה מתוכה, הסיכוי שהגרלנו דווקא שורש של הפולינום הוא מאוד קטן. לכן אם האלגוריתם החזיר שהפולינום <u>אינו</u> פולינום האפס, הוא בטוח צודק, אבל אם האלגוריתם יחזיר שהפולינום הוא <u>כן</u> פולינום האפס, יכול להיות שהוא טועה (במקרה הנדיר שבו ההצבה שהגרלנו היא שורש של הפולינום).
  - <u>סיבוכיות:</u> צריך לבצע M הגרלות של $a_1,.,.,a_n$ ולהציב את הערכים האלו ב־p, לכן זמן הריצה הוא M כפול (סיבוכיות ההגרלה + סיבוכיות ההצבה).
- <u>אלגוריתם כללי:</u> נשתמש בניפוח (חזרה על האלגוריתם הבסיסי) כדי להגדיל את ההסתברות להצלחה.
  - <u>אלגוריתם:</u>

* נחזור על האלגוריתם הבסיסי $\frac{k}{ln\left(\frac{|A|}{deg(p)}\right)}$ פעמים.
* אם לפחות פעם אחת קיבלנו קי הפולינום אינו פולינום האפס, נחזיר כי הפולינום אינו פולינום האפס.
* אחרת, נחזיר כי הפולינום הוא פולינום האפס.

– טענה: ההסתברות שהאלגוריתם הכללי נכשל קטנה או שווה ל־$\frac{1}{e^k}$.

שימוש במשפט שוורץ־זיפל לזיהוי שידוך מושלם (הרצאה 24, תרגול 13)

* הגדרות:
  – דטרמיננטה: תהי A מטריצה ריבועית $n\times n$, נסמן ב־S_n את קבוצת כל הפרמוטציות האפשריות על המספרים $1,...,n$. הדטרמיננטה של A מוגדרת ע"י $|A|=\det(A)=\sum_{\sigma\in S_n}\operatorname{sign}(\sigma)A_{1,\sigma(1)}\cdot A_{2,\sigma(2)}\cdot\dots\cdot A_{n,\sigma(n)}$.
  – תכונות של דטרמיננטה:
    * עבור מטריצה כלשהי מגודל $n\times n$, הדטרמיננטה היא פולינום בעל n^2 משתנים מדרגה לכל היותר n.
    * ניתן לחשב את הדטרמיננטה של מטריצה A באמצעות דירוג בזמן $O(n^3)$.
  – מטריצת Edmonds: יהי גרף דו צדדי $G=(L,R,E)$ עבורו $|R|=|L|=n$. מטריצת Edmonds היא מטריצה M בגודל $n\times n$ שהשורות שלה ממוספרות באיברי L והעמודות שלה ממוספרות באיברי R: $M_{i,j}-\begin{cases}X_{i,j} & (i,j)\in E\\ 0 & else\end{cases}$.
    * טענה: מתקיים כי $|M|=0$ (הדטרמיננטה של M היא אפס) אם"ם אין ב־G שידוך מושלם.
    * הדטרמיננטה של M היא פולינום בשני משתנים $\sum\limits_{\sigma:L\to R}(-1)^{\sigma}\cdot\prod\limits_{i\in V}M_i\sigma(i)$ ־ כאשר σ היא פרמוטציה של שידוך.
  – מטריצת Tuttle: עבור גרף לא מכוון $G=(V,E)$ שהקודקודים שלו ממוספרים בסדר כלשהו $v_1,..,v_n$. מטריצת $Tuttle$ של G היא מטריצה $n\times n$ המוגדרת באופן הבא: $M_{ij}=\begin{cases}x_{ij} & \{i,j\}\in E, i<j\\ -x_{ji} & \{i,j\}\in E, i>j\\ 0 & \text{otherwise}\end{cases}$.
    * טענה: מתקיים $|M|=0$ אם"ם אין ב־G התאמה מושלמת.
* קלט: גרף דו צדדי $G=(L,R,E)$ שמקיים $|R|=|L|=n$.
* פלט: "אמת" אם יש בגרף שידוך מושלם, "שקר" אחרת.
* אלגוריתם הסתברותי:
  – נגדיר מטריצת Edmonds המתאימה ל־G ונסמנה M.
  – נבחר קבוצה $S\subseteq\mathbb{R}$ ונגריל ממנה ערכים למטריצה M בהתפלגות אחידה.
  – נחשב את הדטרמיננטה של M. אם יצא 0 נחזיר "שקר" (כלומר שלא קיים שידוך מושלם), אחרת נחזיר "אמת".
* טענה: לכל $k\in\mathbb{N}$ ניתן לבחור $S\geq n\cdot e^k$ כך שהאלגוריתם יצליח בהסתברות גדולה או שווה ל־$1-\frac{1}{e^k}$.
* סיבוכיות ריצה: לחשב את M לוקח $O(|V|+|E|)$, להציב במטריצה לוקח $O\left(|V|^2log(|V|\cdot e^k)\right)=O\left(|V|^2log(|V|)+|V|^2k\right)$, ולחשב את הדטרמיננטה לוקח $O(|V|^3)$. לכן בסה"כ נקבל שהסיבוכיות היא $O(|V|^3+|V^2|k)$.
* הערה: אם G הוא גרף לא מכוון $G=(V,E)$, אפשר להשתמש באותו האלגוריתם בדיוק, רק להחליף את מטריצת $Edmonds$ במטריצת $Tuttle$ ולקבל אותה נכונות ואותו זמן ריצה.

שימוש במשפט שוורץ־זיפל לבדיקת נכונות כפל מטריצות (תרגול 13)

* קלט: מטריצות $A,B,C\in\mathbb{R}^{n\times n}$
* פלט: "אמת" אם $AB=C$, "שקר" אחרת.
* אלגוריתם נאיבי (סיבוכיות $O(n^3)$): נכפול את A,B ונבדוק איבר־איבר אם התוצאה שווה ל־C.
* אלגוריתם הסתברותי:
  – אלגוריתם:

* נגדיר $x=\begin{bmatrix}x_1\\x_2\\\vdots\\x_n\end{bmatrix}$, ונגדיר קבוצה סופית $S\subseteq\mathbb{R}$.
* נחשב את $A(Bx)$ ואת Cx.
* נדגום ערכים באופן אחיד לוקטור x, ונציב ב־$A(Bx)$ וב־Cx.
* אם כל הערכים שווים, נחזיר "אמת", אחרת נחזיר "שקר".

- טענה: לכל $k\in\mathbb{N}$ ניתן לבחור $|S|\geq e^k$ כך שההסתברות שהאלגוריתם הצליח גדולה מ־$1-\frac{1}{e^k}$.
- סיבוכיות ריצה: כדי לחשב את $A(Bx)$ נדרשות $O(n^2)$ פעולות, לדגום וקטור מ־S לוקח $O(nk)$, ולבדוק אם הם שווים לוקח $O(n)$. כלומר, בסה"כ הסיבוכיות היא $O(n^2+nk)$.
  * אם $n\gg k$ ההסתברות יורדת ל־$O(n^2)$ ולכן האלגוריתם יעיל יותר מהאלגוריתם הנאיבי.