סיכום אלגוריתמים

14 ביולי 2021

1 שבוע 1:

1.1 הרצאה 2:

- **שיבוץ משימות:** אנו מקבלים קלט של n קטעים סגורים, והמטרה היא להרכיב את הקטע המקסימלי שבו הקטעים הקטנים לא חופפים, (המטרה היא לבחור את מספר הקטעים המקסימלי ולא את הקטע הארוך ביותר).
האלגוריתם יפעל כך: נמיין את הקטעים בסדר עולה לפי נקודת הסיום. אחכ נעבור על הרשימה הממויינת לפי הסדר, ונוסיף קטע רק אם הוא לא חופף לקטעים הקודמים שכבר בחרנו. לבסוף נחזיר את כל הקטעים שבחרנו ונצרפם לקטע ארוך.
סיבוכיות: זמן - $O(n \cdot log(n))$. מקום - $O(1)$ מעבר למיקום של הפלט והקלט.

- **מזעור האיחור EDD:** אנו מקבלים כקלט קבוצה של n משימות. כל משימה נתונה ע"י זוג (p_j, d_j) כאשר p_jמסמן את משך המשימה, ו d_j מסמן את מועד הסיום הנדרש. בנוסף נסמן ב L את האיחור.
כלומר כדי לסיים את המשימה בזמן אנו צריכים להתחיל אותו לפחות בזמן $d_j - p_j$.
הפלט הוא שיבוץ תקין של המשימות בציר הזמן. $t : \{1, ...n\} \Rightarrow [0, \infty)$ כך שאין חפיפיה בין שתי משימות.
המטרה: לחשב שיבוץ תקין עבורו האיחור המירבי הוא מינימלי.
האלגוריתם יפעל כך: נמיין את המשימות בסדר עולה לפי זמני הסיום, ונשבץ אותן החל מזמן 0 ללא רווחים לפי הסדר.
סיבוכיות: $O(n \cdot log(n))$.

2 שבוע 2:

2.1 הרצאה 3:

- **בעיית הדפדוף:** בעיית ניהול משאבים ב OS, למחשב יש זיכרון מהיר בעל k דפים, וזיכרון איטי. המטרה היא להעביר את הדפים מהזיכרון האיטי למהיר ביעילות הגבוהה ביותר.
אנו מקבלים כקלט: את מספר הדפים שאנו רוצים לגשת אליהם לפי סדר הגישה, נסמנם ב $\nabla_1, \nabla_2\nabla_3...\nabla_m$.

סהכ יש $n > k$ דפים ולכל t מתקיים: $\nabla_t \in \{1, ..., n\}$ (הדפים יכולים לחזור על עצמם).
הפלט הוא: לכל זמן $t = k+1....m$ שבו אנו מבקשים דף שאינו נמצא בזיכרון המהיר, אנו צריכים לציין איזה דף אנחנו זורקים בכדי לפנות מקום לדף הנדרש.

האלגוריתם של בלאדי: נסמן ב c_t את קבוצת הדפים הנמצאים בזיכרון המהיר בזמן t (כאשר מתקבל ∇_t).
אם $\nabla_t \notin c_t$אז נחליף אותו בדף $\nabla \in c_t$, עבורו הפעם הבאה שמבקשים את ∇ היא המאוחרת ביותר.

3 שבוע 3:

3.1 הרצאה 4:

- **עצים פורשים מינמלים:** אנו מקבלים קקלט גרף $G = (V, E)$ ופונקציית משקל חיובית $w : E \Rightarrow N$.
  המטרה: למצוא תת גרף קשיר מינמלי המכיל את כל קודקודי הגרף.

- **הגדרה - חתך:** בגרף לא מכוון, נבצע חלוקה של צמתי הגרף לשתי קבוצות לא ריקות, חתך הוא קבוצת הקשתות המחברות בין קודקודים בשני הצדדים.

- **טענה:** גרף קשיר וחסר מעגלים בעל n קודקודים מכיל $n-1$ קשתות (זהו עץ).

- **טענה:** אם מוסיפים עץ -T קשת e המחברת בין שני קודקודיו ואינה מוכלת בעץ. אזי בגרף שנוצר יהיה מעגל פשוט העובר דרך הקשת e.
  מסקנה: מחיקת צלע $e' \neq e$ מהמעגל, עדיין תשאיר לנו עץ פורש.

- **למת החתך:** יהי $f \subseteq E$ חתך ב G, ותהי $e \in f$ קשת בעלת משקל מינימלי ב f . אזי בהכרח קיים עץ פורש מינימלי המכיל את e.

- **אבחנה:** יהי T עץ פורש כלשהו של גרף G, ותהי $e \in T$. אם נסיר את e מ T נקבל גרף בעל שני רכיבי קשירות שאינם ריקים.
  אותה החלוקה ב G מגדירה חתך שנסמנו ב $F_{T,E}$. בהכרח מתקיים $e \in F_{T,E}$ וזו הקשת היחידה של T הנמצאת בחתך $F_{T,E}$.
  אם T הוא עץ פורש מינמלי אזי בהכרח e הינה קשת בעל משקל מינימלי ב $F_{T,E}$.

3.2 הרצאה 5:

- **משפט:** יהי C מעגל פשוט ב G, ותהי $e \in C$ קשת במעגל שמשקלה מקסימלי. אזי **קיים** עץ פורש מינימום שלא מכיל את e.

- **שיטה למציאת** MST: נצבע את קשתות הגרף בשני צבעים - כחול ואדום‏ץ בכל שלב קשת יכולה להיות בלשה מצבים: לא צבועה, כחולה או אדומה. בהתחלה כל הקשתות אינן צבועות, והמטרה היא לצבוע את כל קשתות הגרף. לבסוף, קשת כחולה - נמצאת ב MST ואדומה לא.
  נצבע לפי הכללים הבאים:

הכלל הכחול - נבחר חתך F שאין בו אף צלע כחולה., וצובעים $e \in F$ בעלת משקל מינימלי - בכחול.
הכלל האדום - בוחרים מעגל פשוט C שאין בו אף קשת אדומה, וצובעים $e \in C$ בעלת משקל מקסימלי - באדום.
הערה: האלגוריתמים של קרוסקל ופרים למציאת עפ"מ משתמשים בכללים אלו.

- **משפט - עבור כלל כחול אדום:** נפעיל את הכלל הכחול והאדום בסדר כלשהו, כל עוד ניתן להפעיל אותם. אזי בסיום הצביעה תקיימו שני דברים:
  1: כל הקשתות יהיו צבועות. **2:** תת הגרף הנפרש ע"י הקשתות הכחולות הוא MST.

3.3 תרגול 2:

- **אלגוריתם חמדן - הגדרה:** זהו אלגוריתם שבוחר את הדרך הזולה ביותר ללא התחשבות בהשלכות העתידיות.

- **סכמה כללית להוכחת אלגוריתם חמדן:**
  1: מוכיחים חוקיות של הפתרון החמדני - b.
  2: טענה באינדוקציה - שלכל $k < n$ קיים פתרון אופטימלי c שמסכים עם b על k הצעדים הראשונים.

- **בעית תא הדלק הקטן:** מכונית נוסעת ממקור ליעד ואנו רוצים למזער את עצירות המכונית בתחנות הדלק.
  קלט: n מס הק"מ שניתן לנסוע עם תא דלק מלא. $a_1....a_n$ מיקום תחנות הדלק במסלול.
  הנחה: קיים פתרון (נישם לב כי הפתרון האופטימלי הוא לא יחיד).
  פלט: באילו תחנות אנו צריכים לעצור לתדלק.
כיצד האלגוריתם יעבוד: בכל פעם נבדוק האם יש לנו מספיק דלק כדי להגיע לתחנה הבאה. אם כן - נמשיך. אם לא - נעצור לתדלק בתחנה הנוכחית.
  הוכחה: ראשית נראה כי $a_1 = b_1$ ו $a_n = b_n$ ואחכ נוכיח את חוקיות האלגוריתם. לבסוף נוכיח אופטימליות.

- **עצים פורשים מינימלים - הגדרות וטענות:**
  טענה: בין כל שני קודקודים בעץ יש בדיוק מסלול אחד.
  עץ פורש מינימלי MST: זהו עץ שמורכב מגרף, ומכיל את כל קודקודי הגרף עם מסלול מינימלי בניהם.
  פונקצית משקל w: היא פונקציה הנותנת משקל עבור כל צלע בגרף. משקל העץ הוא סכום כל צלעותיו
הגדרה - חתך: יהי G גרף לא מכוון , ויהיו A, B שתי קבוצו קודקודים המקיימות $A \cap B = \emptyset$ ו $A \cup B = B$נגדיר את החתך של B להיות קבוצת הצלעות המקיימת $(A, B) = \{(u, v) \in e | a \in A, b \in B\}$.

- **משפט - עבור כלל כחול אדום :** יהי G גרף קשיר ולא מכוון שחלק מצלעותיו צבועות באדום וחלק בכחול באופן שרירותי. אם לא כל הקשתות צבועות, קיימת צלע לא צבועה שניתן לצבוע בעזרת הכלל הכחול או האדום.

- **אלגוריתם חמדן למציאת** MST:
  1: נאתחל E_b E_r כשתי קבוצות ריקות.
2: כל עוד יש צלעות לא צבועות בגרף - נבחר בין: לבחור את הכלל הכחול ולהוסיף את הצלע הנבחרת ל E_b. לבין לבחור את הכלל האדום ולהוסיף את הצלע לקבוצה E_r.
  טענה: כל אלגוריתם שמתקבל מהדרך הנ"ל מחזיר MST.
  הערה: האלגוריתמים של $prim$ ו $kruskal$ פועלים באופן הנ"ל.

- **האלגוריתם של** $prim$**:** נוסיף את כל הקודקודים לערימת מינימום. ובכל פעם נבחר את הקודקוד המינימלי, אם הוא לא סוגר צלע נוסיף אותו לעץ. אם הוא סוגר צלע - נעבור לקודקוד הבא. למעשה זה כמו להפעיל את הכלל הכחול.
זמן ריצה: פתרון נאיבי - $O(V \cdot E)$. עם ערימת מינימום - $O(E \cdot log(V))$.

- **האלגוריתם של** $kruskal$**:** נאתחל את E_t, V_t להיות שתי קבוצות ריקות. אחכ נמיין את הצלעות בסדר עולה לפי המשקל שלהן ונעבור על הצלעות לפי הסדר. אם הקשת e_i סוגרת מעגל עם הצלעות שנמצאות ב E_t- נפעיל על המעגל את הכלל האדום. ונצבע את e_iבאדום.
אחרת - e_iהיא הצלע הקלה ביותר בחתך לכן נצבע אותה בכחול ונוסיף אותה ל E_tואת v נוסיף ל V_t.
זמן ריצה: אם נממש בעזרת איחוד קבוצות $O(E \cdot log(E))$

4 שבוע 4:

4.1 הרצאה 6 - מערכות של קבוצות - מטרואידים:

- **הגדרה - מערכות של קבוצות:** נתונה קבוצת בסיס (לדוגמה - קבוצת הקשתות של גרף א מכוון). בנוסף נתונה רשימה של תת קבוצות של קבוצת הבסיס (לדוגמה - כל המעגלים הפשוטים).

- **הגדרה - מטרואיד:** הוא זוג סדור $M = (E, I)$ ובו - E: קבוצת בסיס סופית. $I \subseteq 2^E$: אוסף של תת קבוצות של E.
המטרואיד מקיים שתי תכונות:
1 ירושה: אם $A \in I$ ו $B \subseteq A$ אז $B \in I$. כלומר - אם קבוצה נמצאת ב I, אזי כל הת"ק שלה גם נמצאות ב I. **2 הרחבה:** אם $A, B \in I$ ו $|A| < |B|$ אז קיים $e \in B \backslash A$ עבורו $A \cup \{e\} \in I$. כלומר - ניתן להרחיב קבוצות, ועדיין הן ישארו במטרואיד.

- **הגדרה - ב"ת:** הקבוצות שנמצאות ב I נקראות ב"ת.

- **הגדרה - בסיס:** קבצה ב"ת מקסימלית ביחס להכלה נקראת בסיס.

- **טענה:** כל הבסיסים במטרואיד הם שווי גודל.

- **טענה - תכונת ההחלפה:** אם $A, B \in I$ הם שני בסיסים. אזי לכל $a \in A \backslash B$ קיים $b \in B \backslash a$ עבורו $A \backslash \{a\} \cup \{b\}$ גם הוא בסיס של המטרואיד.

- **החלפה סימטרית:** באותם התנאים, לכל $a \in A \backslash B$ קיים $b \in B \backslash a$ עבורם גם $A \backslash \{a\} \cup \{b\}$ הוא בסיס , וגם $B \backslash \{b\} \cup \{a\}$ הוא בסיס.

- **החלפה חח"ע:** באותם תנאים. קיימת העתקה חח"ע $f : A \backslash B \Rightarrow B \backslash A$ עבורה לכל $a \in A \backslash B$ הקבוצה $A \backslash \{a\} \cup \{f(a)\}$ היא בסיס של המטאוריד.

- **הערה - הגדרה חלופית למטרואיד:** אפשר להחליף את תכונת ההרחבה בתכונת ההחלפה. למעשה אם יש מערכת קבוצות לא ריקה המקיימות את תכונת ההחלפה, אזי אם נוסיף למערכת זו את כל תתי הקבוצות של הקבוצות שוות הגודל בה - נקבל מטרואיד שבו המערכת המקורית היא הבסיסים.

- **דוגמאות למטרואידים:**

1 - המטרואיד הטרוויאלי : אוסף כל תת הקבוצות של E הוא מטריאיד ובסיסו היא E.

2 - מטרואיד המעגלים של G, (מטרואיד גרפי): בהינתן גרף סופי לא מכוון G, אז האוסף של כל קבוצות הקשתות $F \subseteq E$ עבורן F חסרת מעגלים , מהווה מטרואיד מעל E. אם G קשיר - הבסיסים הם כל העצים הפורשים של G.

3 - המטרואיד הווקטורי : בהינתן מטריצה A, אוסף קבוצת העמודות של A שהן בת"ל מהווה מטרואיד מעל קבוצת העמודות של A.

4 - המטרואיד הביסירקולרי: בהינתן גרף סופי לא מכוון G, אוסף קבוצות הקשתות $F \subseteq E$ עבורן כל רכיב קשיר של F מכיל לכל היותר מעגל 1, הוא מטרואיד מעל E.

- **אלגוריתם למטרואיד:** נתון מטרואיד (E, I). אלגוריתמית - I לא נתונה באופן מפורש, אלא נתון אלגוריתם יעיל שבהינתן $F \subseteq E$ בודק אם $F \in I$. ונתונה פונקציית משקל חיובית $w : E \Rightarrow N$.

המטרה: למצוא בסיס שמשקלו **מקסימלי**.

האלגוריתם החמדן למטרואידים:

קלט: מטרואיד $M = (E, I)$, ופונקציית משקל $w : E \longrightarrow R^+$.

פלט: תת קבוצה $A \in I$ כך שהגודל של A מקסימלי, ומשקלה מקסימלי.

האלגוריתם הגנרי: נמיין את אברי E בסדר יורד לפי משקלם. לאחר מכן נעבור עליהם וננסה להוסיף אותם לקבוצה $A = \emptyset$ רק אם $A \cup \{e_i\} \in I$. לבסוף נחזיר את A.

זמן הריצה: תחילה אנו ממיינים בזמן של $O(n \cdot log(n))$. אחכ אנו בודקים אם האיבר נמצא במטרואיד וזה תלוי במטרואיד, לכן זמן הריצה הוא $f(n)$. זמן הריצה הכולל הוא $O(n \cdot log(n) + n \cdot f(n))$.

טענה: הקבוצה A היא בסיס.

טענה: הקבוצה A הינה אופטימלית - אין בסיס שמשקלו גדול.

משפט $Rado + Edmonds$**:** תהי (E, I) מערכת קבוצות המקיימת את תכונת הירושה (אם קבוצה היא ב"ת אזי גם ת"ק שלה היא ב"ת). אם האלגוריתם החמדן מוצא פתרון אופטימלי עבור **כל** $w > 0$. אזי מערכת הקבוצות הזו היא מטרואיד בהכרח.

4.2 תרגול 3:

- **מטרואידים - רעיון כללי:** מבנה מתמטי, שהרעיון שלו הוא להגדיר באופן חח"ע בעיות שאפשר לפתור באופן חמדני. כלומר - בעיה היא מטרואיד **אמ"מ** ניתן לפתור אותה בעזרת אלגוריתם חמדני.

- **טענה:** בכל מטרואיד מתקיים $\emptyset \in I$.

- **האלגוריתם החמדן למטרואידים - הערות:**

1: ניתן למיין בסדר עולה במקום בסדר יורד. ואנו נקבל קבוצה מגודל מקסימלי, אך עם משקל מינימלי מבין כל הקבוצות מגודל מקסימלי.

2: בבעיות אופטימיזציה (אנו מתעניינים בפתרון חוקי ואופטימלי) במטרואידים: נאמר שפתרון A הוא חוקי - אם $A \in I$. ואופטימלי - אם המשקל הוא הכי גדול\קטן ב I.

- **מטרואיד השידוכים:**

הגדרה - גרף דו"צ: $G = (L, R, E_G)$ הוא גרף המוגדר על שתי קבוצות קודקודים זרות. וצלעותיו מחברות בין שני צידי הגרף בלבד.

הגדרה - התאמה: התאמה היא ת"ק $A \in E_G$ כך שהיא נוגעת לכל היותר בכל קודקוד פעם אחת.

הגדרה - שידוך מושלם: זאת התאמה שנוגעת בכל הקודקודים, (אפשרי רק אם $|L| = |R|$) .

הגדרה - מטרואיד שידוכים: יהי גרף דו"צ $G = (L, R, E_G)$. נגדיר מטרואיד $M = (E, I)$. $E = L$ ו $I = \{L' \subseteq L | \exists R' \subseteq R : \exists A : L' \mapsto R' :\longmapsto A = perfect\ mach\}$ כלומר - הקבוצה I היא הקבוצה של תתי קבוצות של קודקודים שיוצרות התאמה מושלמת בין קודקודים משתי הקבוצות.

טענה: מטרואיד השידוכים הינו מטרואיד.

4.3 הרצאה 7:

- **בעיית שיבוץ קטעים:** את בעיית שיבוץ קטעים ממושקלים **לא ניתן** לפתור בעזרת אלגוריתם חמדן, משום שזה לא מטרואיד. (אך יכול להיות שקיימת פונקציית משקל עבורה כן אפשר לפתור את הבעיה בעזרת אלגוריתם חמדן שיחזיר לנו פתרון אופטימלי).

- **מטרואידים ואלגוריתמים חמדנים:**

המטרואיד הגרפי: אמרנו כבר שהוא מטרואיד והעצים הפורשים של הגרף הם בסיסיו. ואכן ניתן לפתור את בעיית עפ"מ בעזרת אלגוריתם חמדן. (פשוט חיפשנו בסיס שמשקלו מינימלי במקום בסיס שמשקלו מקסימלי).

האלגוריתם של קרוסקל: הוא אלגוריתם חמדן על מטרואיד המעגלים (גרפי) של גרף הקלט.

- **בעיית התרמיל:** יש לנו תרמיל ומספר חפצים בנפחים שונים ואנו רוצים למלא את התרמיל ביעילות מקסימלית.

באופן פורמלי - נתונה קבוצה E ולכל $e \in E$ יש נפח $v(e)$, ומשקל $w(e)$. נתון גם נפח התרמיל V. אנו רוצים למצוא $F \subseteq E$ עבורה $\sum_{e\in F} v(e) \leq V$, ו $\sum_{e\in F} v(e)$ מקסימלי מבין האריזות המותרות.

אוסף האריזות החוקיות מקיים את תכונת הירושה (כי אם קבוצה נכנסת לתרמיל אזי גם ת"ק שלה נכנסת). אך תכונת ההרחבה לא בהכרח מתקיימת (לדוגמה - אם יש לנו חפץ שמשקלו V ומלא חפצים שגדלם שווה 1, אזי גודל הקבוצות לא שווה ולא ניתן להרחיב). אם כך זה **לא מטרואיד** ואלגוריתם חמדן (אריזה לפי משקל מהכבד לקל) לא יעבוד ולעיתים אף יחזיר פתרון גרוע.

- **אלגוריתם חמדן (גרוע) לפתרון בעיית התרמיל:** נמיין את החפצים בסדר יורד לפי המשקל ליחידת נפח $\frac{w(e)}{v(e)}$ ואז נשתמש באריזה חמדנית לפי הסדר הזה. (האלגוריתם ייכשל כאשר יהיה לנו חפץ גדול עם משקל ליחידת נפח קטן, וחפץ קטן עם משקל ליחידת נפח גבוה ואז האלגוריתם יבחר לנו פתרון לא אופטימלי - את החפץ הקטן).

- **אלגוריתם חמדן משולב לפתרון בעיית התרמיל:** אם נשלב את שני האלגוריתמים למעלה, וכל פעם נבחר את הטוב ביותר - אנו נשיג לפחות חצי מהמשקל האופטימלי.

5 שבוע 5:

5.1 הרצאה 8:

- **בעיית התרמיל:** נגדיר פתרון רקורסיבי לבעיה.

נסמן ב $P(i,V')$ אריזה אופטימלית של חפצים בטווח $\{1,...i\}$ בתרמיל שנפחו V', ואת המשקל נסמן ב $W(P(i,V'))$.
ואת הפריט הבא נבחר באופן הבא:

$$W(P(n,V)) = max\{w_n + W(P(n-1,V-v_n)), W(P(n-1,V))\}$$

מכיוון שבכל שלב $P(i,\cdot)$ (כאשר $\cdot$ מייצגת את פרמטר הנפח) אנו יודעים את השלב הקודם, לכן לא נצטרך לבצע חישובים מיותרים, אלא נשמור את השלב הקודם ונשתמש בו.
כלומר - בהינתן כל הערכים $P(i-1,\cdot)$ (סהכ $V+1$ ערכים) אפשר לחשב את כל הערכים $P(i,\cdot)$, כל ערך דורש $O(1)$ פעולות אריתמטיות. סה"כ יש $V+1$ ערכים שצריך לחשב.
אתחול: לכל $v' \in \{0,...V\}$

$$P(1,V'), W(P(1,V')) = \begin{cases} \{v_1\}, w_1 & v_1 \le V' \\ \emptyset, 0 & else \end{cases}$$

העדכון:

$$P(i,V'), W(P(i,v')) = max \begin{cases} \{i\} \cup P(i-1,V'-v_i), W_i + W(P(i-1,V'-v_i)) & v_i \le V' \wedge (W_i + W(P(i-1,V' - \\ P(i-1,V'), W(P(i-1,V')) & else \end{cases}$$

העדכון יתבצע לפי המשקל הגבוה מבין שניהם.
הפלט הוא: $P(n,V)$
זמן הריצה הוא: V_n.

5.2 הרצאה 9:

- **בעיית התרמיל:** הפתרון שמצאנו בסוף הרצאה קודמת הוא פתרון בעזרת תכנון דינאמי - רקורסיה.

- **טענה:** בסיום ריצת האלגוריתם הרקורסיבי לפתרון בעיה התרמיל $P(n,V)$, מחזיק פתרון אופטימלי - קבוצת חפצים הניתנת לאריזה בתרמיל שמשקלה מקסימלי.

- **סיבוכיות האלגוריתם:**

ריצה: יש לנו $n \cdot (V+1)$ צעדי עדכון, כל פעולת עדכון עולה מספר קבוע של פעולות אריתמטיות: $O(1) = O(n \cdot V)$.
מיקום: האלגוריתם צריך לשמור בכל פעם שתי שורות - זאת שסיימנו לחשב, וזאת שאנו מחשבים עכשיו (אחרי נחשב שורה חדשה נמחק את הקודמת). לכן יש לנו $2 \cdot V = O(V)$ איברים של הטבלה שאנו צריכים לשמור. בכל איבר כזה אנו צריכים לשמור את הארידה ואת משקלה, סה"כ $O(n \cdot V)$ תאי זכרון שמחזיקים כל אחד ערך מספרי או מצביע.
ייצוג הקלט: אם נייצג את V בייצוג אונארי (חד ספרתי) - האלגוריתם יהיה פולינומי בגודל הקלט. אך אם נייצג את

V בייצוג בינארי - האלגוריתם לא בהכרח יהיה פולינומי, וזה תלוי בגודל של V ביחס ל n, (כי את V עצמו ניתן לייצג ע"י $\lceil log(V) \rceil + 1$ ביטים).

- **אלגוריתם פסואודו - פולינומי:** אלגוריתם שהוא פולינומי אמ"מ המספרים בקלט מיוצגים בייצוג אונארי, נקרא פסואודו - פולינומי.

- **אלגוריתם פולינומי לבעיית התרמיל:**
נגדיר:

$$W_{max} = max\{w_i : i \ s.t \ v_i \leq V\}$$

אנו יודעים כי המשקל המירבי של אריזה כלשהי הוא $\leq n \cdot W_{max}$. אם כך המשקל של אריזה נמצא בקבוצה $\{0, ...n \cdot W_{max}\}$.
נגדיר אריזה של חפצים מ $\{1, ..., i\}$ שמשקלה בדיוק W ונפחה מינימלי מבין האריזות הללו, באופן הבא: $Q(i, W)$.
אם אין אריזה של חפצים מ $\{1, ..., i\}$ שמשיגה משקל ששווה בדיוק ל W, נסמן זאת ב $none$.
פתרון בעיית התרמיל מצוי באיבר $Q(n, W)$ עבור W המקסימלי שבו האיבר הזה אינו $none$.
כך נבנה את הטבלה:

$$\forall W \in \{0, ..., n \cdot W_{max}\},$$

אתחול:

$$Q(1, W) = \begin{cases} \{1\} & W = w_1 \\ none & else \end{cases}$$

עדכון:
נניח שנתונה השורה ה $i-1$ של הטבלה Q, נרצה לחשב את $Q(i, W)$ לכל $W \in \{0, ..., n \cdot W_{max}\}$. נחלק למקרים:
אין פתרון:

$$: Q(i-1, W) = none \wedge ((W - W_i < 0) \vee (Q(i-1, W - w_i) = none)) \Rightarrow Q(i, W) = none$$

מקרה שני - נקח את האיבר ה $i-1$:

$$v(Q(i-1), W) < v(Q(i-1, W - w_i)) + v_1 \Rightarrow Q(i, W) = Q(i-1, W)$$

מקרה שלישי - נקח את האיבר ה i:

$$else \Rightarrow Q(i, W) = Q(i-1, W - w_i \cup \{i\})$$

סיבוכיות זמן: עדכון של ערך בטבלה הוא מספר קבוע של פעולות, ולכן $O(1)$, מספר העדכונים הוא גודל הטבלה, סה"כ: $O(n^2 \cdot W_{max})$.
זה אלגוריתם פולינומי אם $W_{max} = poly(n)$, (הדרישה באלגוריתם הזה היא על המשקלים ולא על הנפחים).

שיפור האלגוריתם: נקבע פרמטר k שערכו ינתן בהמשך. כעת, נעגל כלפי מטה את המשקלים כדי שיהיו כפולות של k (כדי שנוכל לחלקם ב k). נקבל לכל היותר $\lfloor\frac{W_{max}}{k}\rfloor+1$. כלומר - נגדיר $\{\lfloor\frac{W_{max}}{k}\rfloor ... ,0\} \in \lfloor\frac{w_i}{k}\rfloor = w'_1$. כעת נריץ את האלגוריתם על הערכים הללו עם w'.

סיבוכיות הזמן של הנוסחה החדשה (v, w', v): היא $O(n^2 \cdot \lfloor\frac{W_{max}}{k}\rfloor)$.

נסמן את פתרון האלגוריתם ב $P \in \{1, ..., n\}$, ופתרון אופטימלי לבעיה המקורית $P_{opt} \in \{1, ..., n\}$.

$$\sum_{i\in P} w_i \geq k \cdot \sum_{i\in P}\lfloor\frac{w_i}{k}\rfloor \geq k \cdot \sum_{i\in P_{opt}}\lfloor\frac{w_i}{k}\rfloor > \sum_{i\in P_{opt}} w_i - n\cdot k$$

כעת, עבור k מאד גדול - הסיבוכיות תהיה טובה, אך אנו נתרחק מהאופטימום. לכן אנו צריכים לבחור k כך שיתן לנו תוצאה טובה עבור שני המבחנים. עבור $\varepsilon > 0$ נגדיר $k = \frac{\varepsilon\cdot W_{max}}{n}$.

סיבוכיות ריצה: $O(\frac{n^3}{\varepsilon})$. **אופטימליות הפתרון:** $\sum_{i\in P} w_i \geq \sum_{i\in P_{opt}} w_i - \varepsilon\cdot W_{max} \geq (1-\varepsilon)\cdot\sum_{i\in P_{opt}} w_i$

כעת יש לנו אלגוריתם שמקבל כקלט את (v, w, V, ε) מחשב פתרון P שערכו לפחות $1-\varepsilon$ כפול הערך האופטימלי, ורץ בזמן פולינומי ב n וב $\frac{1}{\varepsilon}$.

אלגוריתם כזה נקרא - סכימת קירוב פולינומית לחלוטין $FPTAS$.

- **בעיית השיבוץ של קטעים ממושקלים:** נתונים קטעים ממושקלים על הישר הממשי, ואנו רוצים למצוא קבוצה בגודל מקסימלי (שמשקלה מקסימלי) של קטעים לא חופפים.

רעיון: נניח שהקטעים ממויינים לפי נקודת הסיום. לכל נקודת סיום b_j, נתבונן באריזה של קטעים עד לנקודה זו, בהכרח זו אריזה של קטעים מהקבוצה $\{I_1, ...I_j\}$.

נסמן שיבוץ של קטעים מ $\{I_1, ...I_j\}$ שמשקלם מקסימלי ב S_j. המטרה שלנו היא לחשב את S_n.

נסמן $ji = arg\ max\{b_j : b_j \leq a_i\}$ זהו הקטע האחרון שאפשר לארוז, שמסתיים לפני שמתחיל הקטע ה i.

נוסחת הרקורסיה:

$$S_1 = \{I_1\}$$

$$S_i = \begin{cases} \{I_i\}\cup S_{ji} & w(S_{ji}) + w_i \geq S_{i-1} \\ S_{i-1} & else \end{cases}$$

5.3 תרגול 4 - תכנון דינאמי:

- **תכנון דינאמי:** הרעיון הוא פתרון רקורסיבי.

לוקחים בעיה רקורסיבית, ואז משתמשים בזכרון המחשב כדי להימנע מחישובים של תתי בעיות, ולחסוך כך בזמן ריצה. בד"כ נעבור מזמן פולינומיאלי לזמן לינארי.

- **מציאת האיבר ה n בסדרת פיבונאצ'י:** נוכל לשמור את כל הפתרונות במערך, ואח"כ כשאנו ניגשים לפתור נסתכל על המערך ונוציא משם את התשובות. כך אנו מצמצמים את זמן הריצה להיות $O(n)$.

- **שלבים כלליים לפתרון בעיות דינמיות:**

1 - הגדרת תתי הבעיות: נגדיר מה אנו רוצים לשמור בטבלה.

2 - כתיבת נוסחת הרקורסיה: נכתוב תיאור מתמטי שיסביר לנו כיצד ניתן לעבור מבעיה א' ל - ב', בתוך התנאי הרקורסיבי שלנו בהסתמך על תתי הבעות קודמות.

3 - הגדרת טבלה וסדר המילוי שלה: נגדיר גודל לטבלה - תלוי במספר תתי הבעיות שנפתור, ונגדיר באיזה סדר אנו רוצים להכניס את הבעיות לטבלה בכדי שנוכל להשתמש בה כראוי.

4 -חילוץ הפתרון: נגדיר כיצד בהינתן הטבלה, אנו יכולים לגשת לפתרון הבעיה הנוכחית.

5 - זמן ריצה: לרוב, זמן מילוי כל תא כפול גודל הטבלה.

6 - הוכחה נכונות: נוכיח שטבלת ההסטוריה מכילה פתרונות חוקיים לתתי הבעיות. בנוסף נוכיח שחילוץ הפתרון החזיר פתרון נכון.

בד"כ ההוכחה תהיה באנדוקציה על סדר מילוי הטבלה: נניח שכל מה שמילאנו נכון, ונוכיח שהמילוי החדש גם נכון תוך שימוש שנוסחת הרקורסיה.

- **הגדרה - גרף שכבות:** זהו גרף שהקודקודים בו מסודרים בשורות, והצלעות עוברות רק מהשורה ה $i-1$ לשורה ה i.

- **אלגוריתם בלמן שכבות:** נתון גרף מכוון $G=(V,E)$ בעל $k+2$ שכבות בעלות גודל שווה M, ופונקציית משקל w. כאשר בשכבה הראשונה יש קודקוד בודד s ובשכבה האחרונה קודקוד בודד t. אנו רוצים למצוא מסלול קצר ביותר מקודקוד s ל t.

נפתור בעזרת תכנון דינאמי:

הבחנה רקורסיבית: כל קדקוד v שנמצא במסלול הכי קצר מ $s \Rightarrow t$, המסלול מגדיר עבורו מסלול הכי קצר מ $s \Rightarrow v$.

תתי הבעיות: עבור כל קודקוד $v \in V$ נמצא את המסלול הכי קצר מ $s \Rightarrow v$.

נוסחת הרקורסיה: נקח את השכן המינימלי, ונוסיף את המסלול מהשכן אל v.

$$D(v)=\begin{cases}0 & v=s\\ min\{D(u)+w(u,v)\}:u\in neighbors(v) & else\end{cases}$$

טבלה: נגדיר טבלה בגודל $M\cdot(k+2)$, כאשר העמודות מסמנות את השכבות, והשורות מסמנות את הקודקודים בכל אחת מהשכבות. ובכל תא נשמור את המרחק הכי קצר מהקודקוד s לקודקוד v_i בשכבה ה k.

נמלא מקודקוד 1 ל m כל שכבה בנפרד, ואח"כ נמלא משמאל לימין - את השכבות בסדר עולה.

חילוץ הפתרון: בזמן המילוי נשמור עבור כל תא מהיכן הגענו אליו, ולבסוף נחזיר את המסלול.

זמן ריצה: יש לנו טבלה בגודל $m\cdot(k+2)$ לכן $O(V)$. הזמן הנדרש למלא כל תא במקרה הגרוע הוא $O(m)$ (כי צריך לעבור על כל קודקודי השכבה הקודמת). אך למעשה בכל שכבה אנו עוברים על כל הצלעות של השכבה הקודמת פעם אחת בלבד וזהו. לכן זמן הריצה הכולל הוא $O(V+E)$.

הוכחת נכונות: טענה - לכל $0\le k\le k+1$ ולכל $0\le m\le M$ התא ה (k,m) מכיל את המרחק הכי קצר לקודקוד v_m^k. נוכיח באנדוקציה על סדר מילוי הטבלה:

בסיס: התא $(0,0)$ מכיל את המרחק הכי קצת מ s אל עצמו.

הנחה: כל התאים שמולאו קודם, מכילים א המרחק הכי קצת מ s אליהם.

צעד: נסתכל על התא ה (k,m), רשום בו - $min_{u\in V_{k-1}}(D(u)+w(u,v_m^k))$, נוכיח כי קיים מסלול באורך כזה:

נסמן ב u_1את הקודקוד שהחזיר לנו ערך מינימלי = $(D(u_1)+w(u_1,v_m^k))$ מהנחת האנדוקציה יש מסלול $s \Rightarrow u_1$באורך $D(u_1)$. נעבור במסלול זה ונוסיף לו את הצלע (u_1,v_m^k) ונקבל את המסלול.

נוכיח כי המסלול מינימלי: נניח בשלילה שהוא לא מינימלי ואז מעקרון ההחלפה היינו צריכים לבחור את המסלול המינימלי, בסתירה.

- **הערות:** האלגוריתם הנ"ל יפעל על כל DAG, נעשה מיון טופולוגי ונעבור לפי סדר הקודקודים. זמן הריצה יהיה $O(V+E)$.

- **תת מחרוזת משותפת הכי גדולה:** אנו מקבלים כקלט שתי מחרוזות $Y=y_1,....y_m$ $X=x_1,...,x_n$ ואנו מחפשים את תת המחרוזת המשותפת הכי ארוכה, לא בהכרח בסדר עוקב.

הבחנה רקורסיבית: נסתבל על התו האחרון x_n,y_m, אם $x_n=y_m$ זה גם בפתרון האופטימלי, אחרת - $x_n \vee y_m$ לא בפתרון האופטימלי.

הגדרת תתי הבעיות: לכל $0\le i\le n$ ולכל $0\le j\le m$ נחשב את תת המחרוזת הכי ארוכה בין $X^i=x_1,...x_i$ ל $Y^j=y_1,...y_j$.

נוסחת הרקורסיה:

$$f(i,j)=\begin{cases}0 & i=0\vee j=o\\ f(i-1,j-1)+1 & x_i=y_j\\ max\left\{f(i-1,j),f(i,j-1)\right\} & x_i\neq y_j\end{cases}$$

טבלה: נגדיר טבלה דו מימדית בגודל $m\cdot n$, נמלא אותה בסדר עולה על השורות - m, ולאחר מכן בסדר עולה על העמודות - n. (כשנבוא למלא את התא (i,j) נצטרך א תהתאים שנמצאים משמאלו ומתחתיו, לכן נמלא בסדר הזה).

בכל תא (i,j) יהיה הפתרון של תת המחזורת ה x_i,y_j.

שחזור הפתרון: נשמור פויינטרים לתא שממנו הגענו, ונחזיר את המסלול בפתרון.

זמן ריצה: יש לנו $O(m\cdot n)$ תאים, ומילוי כל תא זה עלות של $O(1)$ לכן סה"כ $O(m\cdot n)$.

הוכחה נכונות: נוכיח באנדוקציה על גודל הטבלה.

בסיס: עבור מחרוזות ריקות האורך הוא תמיד 0.

הנחה: כל מה שמילאנו עד שלב זה הוא נכון.

צעד: נסתכל על המילוי של התא $M[i,j]$. נסמן את הפתרון האופטימלי ב $S=s_1,...s_r$. ונחלק למקרים:

$x_i=y_j$: אנו יודעים $s_r=x_i=y_j$, כי אחרת היינו יכולים להוסיף אותו למחרוזת ולהאריך אותה. בנוסף אנו יודעים כי $s_1,...s_{r-1}$זאת תת המחרוזת הכי ארוכה ל X^{i-1} ו Y^{j-1} וזה אכן מה שרשום בטבלה.

$x_i\neq y_j$: אנו יודעים ש $s_r\neq y_j \vee s_r\neq x_i$ (או שניהם). זאת אומרת ש s הוא תת מחזורת הכי ארוכה ל $X^{i-1}\wedge Y^j$ או $Y^{j-1}\wedge X^i$ (או שניהם), זאת אומרת ש $max\left\{f(i-1,j),f(i,j-1)\right\}$ וזה מה שכתוב בטבלה.

6 שבוע 6:

6.1 הרצאה 10:

- **מרחק עריכה:** אנו מקבלים כקלט שתי מחרוזות X, Y. ואנו רוצים למזער את פעולות העריכה (הוספת\מחיקת\החלפת אות) של מחרוזת X בכדי שתשתווה למחרוזת Y.

הצגת הבעיה: נייצג את הבעיה בתור גרף מכוון ־ שריג ריבועי מכוון. עמודות השריג זו המחרוזת X, והשורות זו המחרוזת Y. בנוסף בכל ריבוע בשריג יש קשתות אלכסוניות.

קודקודים: קודקוד ההתחלה מייצג התאמה בין שתי מחרוזות ריקות, וכל קודקוד i, j באמצע מייצג את פעולות העריכה שנבצע על המחרוזת X_i בכדי שתשתווה למחרוזת Y_j.

קשתות: קשת ימינה ־ מייצגת מחיקת האות הרשומה מעליה. קשת למטה ־ מייצגת הוספת אות. קשת אלכסונית ־ מייצגת החלפת שתי אותיות במקום ה X_i ו Y_j.

6.2 הרצאה 11:

- **מרחק עריכה:** נסמן $|X| = m, |Y| = n$. אנו נסתכל על מספר הפעולות בכדי להפוך את מחרוזת X ל Y, (למרות שהפעולה הינה סימטרית ע"י היפוך פעולות).

עלות כל החלפה\הוספה\מחיקה שווה ל 1. אלא אם כן ההחלפה היא ריקה ־ מחליפים את אותה האות בעצמה.

- **מציאת מסלול סופי לבעיית מרחק עריכה:** ניתן לפתור את הבעיה למציאת מסלול בגרף הפלט לחישוב מרחק עריכה בעזרת אלגוריתם גנרי לחישוב מסלולים קלים ביותר בגרף מכוון, לדוגמה האלגוריתם של דייקסטרה. מספר הצמתים בגרף שווה ל $(m+1)\cdot(n+1) = O(m \cdot n)$ ומספר הקשתות הוא בסדר גודל של מספר הצמתים $O(n \cdot m)$. לכן זמן הריצה של דייקסטרה יהיה $O(m \cdot n \cdot log(m \cdot n))$.

- **שיפור בעיית מרחק עריכה בעזרת תכנון דינמי:** הרעיון לשיפור מסתמך על כך שאנו יכולים לחסוך חישוב מרחקים של צמתים מסויימים ובכך לצמצם את זמן הריצה. לדוגמה ־ המרחקים לשורה והעמודה הראשונות הם טרוויאלים. כלומר ־ אם נעבור בסדר מסויים אנו נחסוך זמן כי נשתמש במידע קודם.

כיצד נבצע: נחזיק טבלה דו מימדית D, והערכים הרשומים בה הם המרחקים המינימלים מהנקודה $(0,0)$ לקודקוד הנוכחי, כלומר ־ $D(i,j)$ שווה למסלול המינימלי מ $(0,0)$ ל (i,j).

נאתחל תחילה את העמודה הראשונה $D(0,i) = i$ והשורה הראשונה $D(0,j) = j$. כדי להמשיך ־ נסתכל על התאים מלמעלה $D(i-1,j)$, משמאל $D(i,j-1)$ ועל התא האלכסון משמאל למעלה $D(i-1,j-1)$, נקח את המינימלי, ונוסיף את המרחק הנוכחי.

האיבר האחרון בטבלה הוא מרחק העריכה המינימלי בין X ל Y. בנוסף נשמור טבלה נוספת P עם המסלולים וכיצד הגענו לכל צומת.

סיבוכיות הזמן של האלגוריתם הינה $O(m \cdot n)$.

סיבוכיות המקום: בטבלה D ־ $O(min\{m,n\})$ מכיוון שאין טעם לשמור את כל הטבלה.

אם אנו רוצים לשחזר בסוף את המסלול מהטבלה P, סיבוכויות המקום בטבלה P הינה $O(n \cdot m)$. (ניתן לשפר ל

$O(max\{m,n\})$ על ידי שמירה של נקודה שבה עובר המסלול הקל ביותר, ושימוש ברקורסיה בכדי לפסול מקומות שאין טעם לשמור בטבלה, אך זה יפגע בסיבוכיות הזמן).

6.2.1 שידוך מושלם בגרף דו צדדי:

- **הגדרה ־ גרף דו"צ:** נתון גרף לא מכוון דו"צ $G(U,V,E)$, קבוצת הצמתים היא איחוד זר של U,V, וקבוצת הקשתות מקיימת לכל $e \in E \Rightarrow |e \cap U| = 1$.

- **סימונים והגדרות:**

  1: עבור $u \in U$, $N(u)$ היא קבוצת שכני u, כן $N(u) \in V$, ולהיפך.

  2: אם $U' \subseteq U$ אז $N(U') = \cup_{u \in U'} N(u)$. כלומר ־ קבוצת כל השכנים ל הצמתים ב U'

  3: הגדרה ־ שידוך: היא קבוצה של קשתות $M \subseteq E$ כך שבכל צומת של G נוגעת לכל היותר קשת אחת מ M.

  4: הגדרה ־ שידך מושלם: אמ"מ $|M| = |U|$.

  5: הגדרה ־ מסלול מתחלף: בהינתן שידוך כלשהו M, מסלול פשוט שבו הקשתות לסירוגין מ $E \backslash M$ ו M נקרא מסלול מתחלף.

- **משפט הול** $Hall$: בגרף G יש שידוך מושלם, אמ"מ לכל $U' \subseteq U$ מתקיים $|N(U')| \geq |U'|$.

6.3 תרגול 5:

- **בעיית מסילת הרכבת:** אנו מקבלים כקלט n חלקים שונים, מהם אנו צריכים להקים מסילת רכבת. לכל חלק יש חיבור התחלתי ־ s_i, חיבור סופי ־ e_i. אורך ־ l_i ומחיר ־ p_i. K קבוצת סוגי הסיומות.

  אנו צריכים לבנות מסלול באורך L במחיר הטוב ביותר.

  נפתור בעזרת תכנון דינמי:

  אבחנה: אם מורידים חלק אחד מהפתרון האופטימלי i , נקבל פתרון אופטימלי מאורך $L-i$ שנגמר בחיבור s_i.

  תתי הבעיות: עבור $l \in [L]$ ועבור $k \in K$, נמצא את מחיר המסילה הכי זול למסילה באורך L שנגמרת ב k.

  הנוסחה הרקורסיבית:

  נגדיר $min(\emptyset) = \infty$

$$f(l,k) = \begin{cases} 0 & l = 0 \\ min_{i \in N \wedge e_i = k \wedge l_i \geq l} \{f(l - l_i, s_i) + p_i\} & else \end{cases}$$

טבלה: נגדיר טבלה בגודל $k \times (l+1)$ השורות ייצגו את ה־ k. והעמודות ייצגו את האורכים מ 0 עד l. אנו נמלא את הטבלה החל מהעמודה הראשונה, ובכל פעם נסתכל על הפתרון שבעמודות הקודמות.

חילוץ הפתרון: המחיר המינימלי יופיע בעמוד האחרונה, לכן נבחר את המינימום מהעמודה האחרונה.

זמני ריצה: גודל הטבלה הוא $O(l \cdot k)$ ומילוי של כל תא בעלות של $O(n)$, לכן בסהכ $O(l \cdot k \cdot n)$. ־ פסואודו פולינומיאלי.

הוכחה: נוכיח באנדוקציה על סדר מילוי הטבלה.

בסיס: $l = 0$ טריוויאלי.

הנחה: נניח כי כל תא מולא כנדרש.

צעד: נסתכל על התא l,k ונוכיח כי יש מסילה באורך זה. אם רשום בתא lk אינסוף,אז זה מינימום ל קבוצה ריקה ולכן אין מסילה (הגדרה).

המינימום אינו אינסוף, לכן קיים חלק i' שעבורו כתוב $f(l-l_i,s_i)+p_i$, מהנחת האנדוקציה קיימת מסילה באורך $f(l-l_i,s_i)$ שנגמרת ב s_iנוסיף לה את i ונקבל את הדרוש.

אופטימליות: נניח בשלילה כי קיימת מסילה קצרה יותר. נסמן את החלק האחרון שלה ב i'' . אורך המסילה הוא $f(l-l''_i,e_i)+p_i$ ואז החלק הנוכחי הוא הנמוך מעיקרון ההחלפה.

- **אלגוריתם פלוייד וורשל - כל המסלולים הקצרים:** אנו מקבלים כקלט גרף $G=(V,E)$ מכוון, ופונקציית משקל w. בנוסף נניח כי הגרף חסר מעגלים שליליים. אנו רוצים למצוא את המרחק הקצר ביותר בין כל קודקוד לכל קודקוד. הפלט הינו מטריצה עם המרחקים הקצרים.

אבחנה: תתי מסלולים של מסלול אופטימלי הינם אופטימלים.

תתי הבעיות: לכל $m,i,j\in[n]$ כאשר m מייצג את מספר הצלעות, נמצא את המסלול האופטימלי באורך m.

נוסחת הרקורסיה:

$$f(i,j,m)=\begin{cases} if\ i\neq j\Rightarrow\infty, if\ i=j\Rightarrow 0 & m=0\\ min_{v_k\in V}\left\{f(i,k,m-1)+w(v_k,v_j)_{v_k,v_j\in E}\right\} & else\end{cases}$$

טבלה: נבנה טבלה תלת מימדית, נוכל לפרק למספר טבלאות עבור $m=0$ עד n. נגדיר טבלה בגודל $n\times n\times n$ ונמלא את הטבלאות החל מ $m=0$. (סדר המילוי בתוך הטבלה לא משנה).

זמן ריצה: גודל הטבלה $O(V^3)$ ומילוי כל תא $O(V)$ סהכ $O(V^3)$.

אם נבחר להגדיר ש v_k הוא שכן של הקודקוד הנוכחי, אזי נעבור על כל הצלעות רק פעם אחת ולכן זמן הריצה יהיה $O(V^2\cdot E)$.

אבחנת פלוייד וורשל: ניתן להסתכל על קודקודים במקום על צלעות. בכל פעם נבדוק האם יש לנו מסלול v_i-v_j שלא כולל בתוכו את הקודקוד v_k.

תתי בעיות: נמספר את הקודקודים מ $v_1,....,v_n$. לכל $k,i,j\in[n]$ נמצא את המסלול הכי קל מ v_i ל v_j ונשתמש לכל היותר בקודקודים $v_1,...,v_k$.

נוסחת רקורסיה:

$$f(i,j,0)=\begin{cases} w(v_i,v_j) & (i,j)\in E\wedge i\neq j\\ \infty & (i,j)\notin E\wedge i\neq j\\ 0 & i=j\end{cases}$$

$$f(i,j,k)=min\begin{cases} f(i,j,k-1) & v_k\ not\ in\ path\\ f(i,j,k-1)+f(i,j,k+1) & v_k\ in\ path\end{cases}$$

טבלה: נבנה טבלה תלת מימדית, נוכל לפרק למספר טבלאות עבור $k=0$ עד n. נגדיר טבלה בגודל $n\times n\times n$ ונמלא את הטבלאות החל מ $k=0$. (סדר המילוי בתוך הטבלה לא משנה).

זמן ריצה: גודל הטבלה $O(V^3)$ מילוי כל תא $O(1)$ סהכ $O(V^3)$

הוכחה: נוכיח באנדוקציה על סדר מילוי הטבלה.

בסיס: $k=0$ טרוויאלי.

הנחה: כל התאים מולאו כנדרש.

צעד: נסתכל על התא ה i,j,k נסמן את המסלול הכי קצר מ v_i ל v_j שעובר לכל היותר ב $v_1,....,v_k$. $v_i \Rightarrow v_j$.
אם v_k במסלול הזה: אפשר לחלק אותו ל 2. $v_i \Rightarrow v_k$ ו $v_k \Rightarrow v_j$. לכן $f(i,j,k)=f(i,k,k-1)+f(k,j,k-1)$
אם v_k לא נמצא במסלול הזה: $f(i,j,k)=f(i,j,k-1)$

7 שבוע 7:

7.1 הרצאה 12:

- **אלגוריתם למציאת שידוך מושלם בעזרת משפט הול:** האלגוריתם מקבל כקלט גרף דו"צ, ומחזיר - שידוך מושלם אם יש, או קבוצה $U' \in U$ כך ש $|N(U')|<|U'|$ שמפירה את תנאי הול.

האלגוריתם יעבוד כך: נתחיל משידוך $M=\emptyset$ ובכל איטרציה נגדיל את M ב 1 ונבדוק האם זה שידוך מושלם, ואז - כשנגיע לסוף נקבל את הפלט.

אם אין $u \in U$ שאינו משודך - אזי M שידוך מושלם. אם יש u כזה - נמצא u',v' ע"י פרוצדורה כמו BFS, נמצא מסלולים שהם לסירוגין בשידוך ולא בשידוך ע"י הפרדה בין צעדים זוגיים לא"ז.

u נמצא בשכבה 0, אם מגיעים לצומת y שנמצא בשכבה אי זוגית ואין המשך לשכבה הבאה - מצאנו מסלול מתחלף אי זוגי מקסימלי $u \Rightarrow y$ וזה מאפשר את הגדלת M ב 1.

אם אין y כזה: הצמתים בשכבות האי זוגיות הם כל השכנים של הצמתים בשכבות הזוגיות, ולכן מספר הצמתים בשכבות הזוגיות גדול ממספר הצמתים בשכבות האי זוגיות. נחזיר את הצמתים בשכבות הזוגיות שהם הקבוצה U' שמפרה את התנאי של משפט הול.

סיבוכיות זמן: נסמן ב n את מבפר הצמתים, וב m את מספר הקשתות. אנו מבצעים לכל היותר $|U|$ איטרציות של הגדלת M. ואנו יודעים ש $n \leq |U|$, כל איטרציה זו הרצה של פרוצדורה דמויית BFS שעולה $O(m+n)$ לכן סה"כ $O(mn+n^2)$.

סיבוכיות מקום: $O(m+n)$ תאי זיכרון (בנוסף לקלט).

7.2 תרגול 6 - רשתות זרימה:

- **רשת זרימה:** זהו מבנה שבנוי בעזרת גרף מכוון, עם קדקוד התחלה s, וסיום t. בנוסף לכל צלע יש ערך קיבול, וכאשר נעבר זרימה דרך צלע מסויימת, נוריד מהקיבול שלה את הערך שהעברנו.

- **הגדרות:**

רשת זרימה היא רביעייה $N=(G,c,s,t)$.

G: גרף סופי מכוון $G=(V,E)$ מבנה הרשת.

c: פונקציית קיבול $c:E \Rightarrow N$ מציינת כמה זרימה במקסימום יכולה לעבור בכל צינור ברשת.

s: קודקוד המקור $s \in V$.

t: קודקוד המטרה\בור $t \in V$.

- **הנחה:** ברשתות זרימה אנו מניחים שאין קשתות נכנסות ל s, ואין קשתות שיוצאות מ t.

- **זרימה חוקית:** זרימה חוקית ברשת N היא פונקציה $f : E \Rightarrow R^+$, המקיימת שני תנאים:

1: אילוץ הקיבול: $\forall e \in E : f(e) \leq c(e)$, כלומר - לא ניתן להזרים בצלע יותר זרימה מהקיבולת של הצלע.

2: שימור הזרימה: $\forall v \in V \backslash \{s,t\} : \sum_{u:(u,v)E} f(u,v) = \sum_{u':(v,u')E} f(v,u')$, כלומר - למעט קודקוד המקור וקודקוד המטרה, עבור כל קודקוד הזרימה שנכנסת אליו שווה לזרימה שיוצאת ממנו.

- **הגדרה - ערך של זרימה f:** מספר יחידות הזרימה שעוברות שיוצאות מקודקוד המקור. $|f| = \sum_{u:(s,u)E} f(s,u)$.

טענה: הערך של f שווה גם לסכום הזרימה שנכנסת ל t.

- **טענות:**

1: תמיד קיימת זרימה חוקית $f = 0$ זרימת האפס, והיא מקיימת $|f| = 0$.

2: כל הזרימות החוקיות חסומות מלמעלה על ידי הצלעות היוצאות מ s. $\sum_{u:(s,u)E} c(s,u)$.

- **בעיית הזרימה:** אנו מקבלים כקלט רשת זרימה $N = (G,c,s,t)$. הפלט הינו זרימה חוקית f ומקסימלית.

אלגוריתם פורד פלקסון FF: אלגוריתם פסאודו־פולינומיאלי לפתרון בעיית הזרימה בזמן של $O(|E| \cdot |f*|)$ כאשר $|f*|$ = ערך זרימה מקסימלי.

אלגוריתם אדמונד קארפ EK: אלגוריתם פולינומיאלי לפתרון בעיית הזרימה בזמן של $O(|E|^2 \cdot |V|)$.

תכונה שימושית של האלגורימים הנ"ל: אם c היא בשלמים, אזי קיימת זרימה מקסימלית f בשלמים, והאלגוריתמים הנ"ל ימצאו אותה.

- **שימושים לרשתות זרימה:**

מציאת שידוך מקסימלי בגרף דו צדדי: נקבל קלט גרף דו"צ $G = (L,R,E)$ ואנו רוצים למצוא התאמה מקסימלית ב G. - קבוצת צלעות מקסימלית שנוגעת בכל קודקוד לכל היותר פעם אחת.

כיצד נממש:

1: נגדיר רשת זרימה $N = (G',c,s,t)$ על ידי יצירת גרף חדש - נוסיף לגרף המקורי קודקוד מקור ויעד, ואחכ נכוון את הצלעות מהמקור ליעד, בנוסף נחבר את המקור לכל הקודקודים, ואת כל הקודקודים נחבר ליעד.

2: ניצור גרף G' באופן הבא: **קודקודי הגרף -** $V' = \{s,t\} \cup V$. נסמן ב $E^{\Rightarrow}$ את הקשתות של הגרף המקורי לאחר כיוונן מ L ל R: $E^{\Rightarrow} = \{(u,v) \in E | u \in L, v \in R\}$. נגדיר $E_L = \{(s,v) \in E | u \in L\}, E_R = \{(u,t) \in E | u \in R\}$. **קשתות הגרף -** $E' = E^{\Rightarrow} \cup E_L \cup E_R$.

3: נגדיר את פונקציית הקיבול $\forall e \in E' : c(e) = 1$.

4: נריץ FF למציאת זרימה מקסימלית - f.

5: לכל $e \in E$ מתקיים $f(e) \in \{0,1\}$. נחזיר את $M = \{e \in E^{\Rightarrow} : f(e) = 1\}$

הוכחת נכונות: נוכיח התאמה חוקית ואופטימלית.

נשים לב כי מנכונות FF נובע ש f חוקית ומקסימלית.

חוקיות: נניח בשלילה כי M לא חוקית - כלומר קיים קודקוד $v \in R \cup L$ כך ש M נוגעת בו פעמיים. אם $v \in L$ -

יצאו ממנו לפחות שתי יחידות זרימה, ומבניית הגרף נכנסו לכל היותר 1. בסתירה לתכונת שימור הזרימה של f. אם $v \in R$ ־ נכנסו אליו לפחות שתי יחידות זרימה, ומבניית הגרף יצאו לכל היותר 1. בסתירה לתכונת שימור הזרימה של f.

אופטימליות: נניח בשלילה כי קיימת $|M'| > |M|$ וחוקית.
ראשית נוכיח כי $|M| = |f|$:

$$|f| \overset{1}{=} \sum_{u \in L} f(s,u) \overset{2}{=} \sum_{(s,u)\in E^{\Rightarrow}} f(u,v) \overset{4}{=} \sum_{(s,u)\in E^{\Rightarrow} \wedge f(u,v)=1} f(u,v) \overset{4}{=} \sum_{(s,u)\in E^{\Rightarrow} \wedge f(u,v)=1} 1 \overset{5}{=} |M|$$

מעברים 1,4,5: מההגדרה. 2: שימור הזרימה. 3: הורדת סכימה של איברים ששווים ל 0.
כעת נוכיח כי קיימת f' חוקית כך ש: $|f'| = |M'|$.
נסמן $L' \subseteq L, R' \subseteq R$ הקודקודים שההתאמה נוגעת בהם. נגדיר

$$f'(e) = \begin{cases} 1 & e \in M' \cup \{(s,u) | u \in L'\} \cup \{(u,t) | u \in R'\} \\ 0 & else \end{cases}$$

נוכיח כי f' חוקית: כל זרימה קטנה או שווה ל 1 ולכן אילוץ הקיבול מתקיים. מכיוון ש M' חוקית, לכל קודקוד ב L' הוצאת זרימה אחת בדיוק, ולכל קודקוד ב R' הכנסנו זרימה אחת בדיוק. וזה גם מה שהכנסנו\ הוצאנו מהם, ושאר הקודקודים שווים ל 0, לכן שימור הזרימה מתקיימת גם כן ן f' חוקית.

$$|f'| = \sum_{u \in L} f'(s,u) = \sum_{u \in L'} 1 = |L'| = |M'|$$

אם כן קיבלנו ש $|f| = |M| < |M'| = |f'|$ בסתירה לאופטימליות של f.

7.3 הרצאה 13 ־ רשתות זרימה:

- **הגדרה ־ חתך:** קבוצת קשתות $B \subseteq E$ תיקרא חתך $s-t$ אמ"מ קיימת קבוצת צמתים $S \cup \{s\} \subseteq V \backslash \{t\}$ עבורה B היא קבוצת הקשתות שיוצאות מ S (קשת שהזנב שלה ב s, והראש שלה לא ב s).
נסמן ב B' את קבוצת הקשתות שנכנסות ל S.
טענה: לכל חתך $s-t$ B מתקיים: $|f| = \sum_{a \in B} f(a) - \sum_{a \in B'} f(a)$.
- **טענה:** לכל חתך $s-t$ B ולכל זרימה f ברשת, מתקיים: $|f| \le c(B)$.
- **רשת שיורית:** נתונה רשת זרימה $N = (G,c,s,t)$ וזרימה f, אנו רוצים להגדיר רשת חדשה $N_f = (G_f, c_f, s, t)$שמתארת את "המקום" שנותר ברשת להעביר עוד זרימה.
כיצד נבצע: נקח את הרשת N ונהפוך את הכיוון של כל הקשתות שהעברנו בהן זרימה, וכך תהיה לנו אופציה להחזיר זרימה ולבטל קשתות ולעבור דרך קשתות חדשות.
באופן פורמלי: $V(G_f) = V(G)$. $E(G_f) = \{(u,v) \in E : f(u,v) < c(u,v)\} \cup \{(v,u) : f(v,u) > 0)\}$ $c_f(u,v) =$ $c(u,v) - f(u,v) > 0, \quad c_f(v,u) = f(v,u)$.
הגדרה: אם f זרימה ב N, ו f' זרימה ב N_f. אז $f+f'$ היא זרימה ב N שמוגדרת ע"י $(f+f')(u,v) =$

$f(u,v)+f'(u,v)-f'(v,u)$, אזי $f+f'$ זרימה חוקית ברשת המקורית N.
טענה: $|f+f'|=|f|+|f'|$.

- **זרימה מקסימלית - שיטת** FF: אנו מחפשים את f_{max}שהיא הזרימה f המקיימת $|f|$ מקסימלית.
  1: נתחיל מזרימה f שערכה 0 על כל קשת.
  2: אחכ נפעיל את האיטרציה הבאה: נחשב את N_f נמצא ב N_f מסלול מכוון מ s ל t (מסלול p כזה נקרא **מסלול משפר**).
  3: נגדיר זרימה ברשת השיורית:

$$f'(u,v)=\begin{cases} min\ c_f(a)_{a\in p} & (u,v)\in p \\ 0 & else \end{cases}$$

  4: נחליף את f ב $f+f'$.
  5: תנאי הסיום הוא: כאשר אין ב N_f מסלול מכוון מ s ל t.
  משפט: בסיום של כל האיטרציות, f היא זרימת מקסימום ברשת N.

8 שבוע 8:

8.1 הרצאה 14:

- **מסקנות מהוכחת אלגוריתם** FF:
  1: האלגוריתם מוצא גם חתך B מינימלי.
  2: זרימת מקסימום שווה לקיבולת של החתך המינימלי.

- **הגדרה - זרימה בשלמים:** זרימה f נקראת זרימה בשלמים, אם הערך שלה על כל קשת הוא מספר שלם.

- **זמן ריצה של** FF: נניח כי הקיבולים ברשת הקלט N הם מספרים שלמים (או שברים רציונלים).
  טענה: אם הקיבולים שלמים, אז בכל שלב בריצת FF הזרימה היא בשלמים.
  מסקנה: בכל איטרציה הזרימה גדלה ב 1 לפחות.
  לכן אם נסמן את הזרימה המקסימלית ב f_{max} אזי מספר האיטרציות חסום על ידי $|f_{max}|$.

8.2 הרצאה 15:

- **אלגוריתם אדמונדס קארפ** EK: זהו אלגוריתם שמשפר את האלגוריתם של FF, והוא יעבוד באופן הבא:
ברשת השיורית נמצא מסלול משפר שמספר הקשתות בו מינימלי. (ניתן לעשות זאת עי BFS) **סיבוכיות:** אם m מספר הקשתות ו n מספר הצמתים אזי $O(n+m)$ (ברשת השיורית יש לכל היותר $2m$ קשתות).
משפט: נדרשות $O(mn)$ איטרציות.
מסקנה: האלגוריתם EK מוצא זרימה מקסימום בזמן של $O(m^2n)$ פעולות אריתמטיות (בהנחה ש m הוא לפחות $n-1$, וזה נובע מההנחה שהרשת קשירה).

- **מסקנה מהוכחת EK:** אם הקשת (u,v) הופיעה באיזו רשת שיורית במהלך ריצת האלגוריתם ואז נעלמה, ואחכ חזרה והופיעה שוב, אזי המרחק $d(s,v)$ גדל ב 2 לפחות. כלומר ־ בין שתי הופעות עוקבות של הקשת (u,v) כשבראשונה היא נעלמה, $d(s,v)$ גדל ב 2 לפחות.

חסימת מספר האיטרציות: אנו יודעים שמתקיים $d(s,v) \leq n-1$, כלומר הקשת יכולה להיעלם ולהופיע לכל היותר $\frac{n-2}{2}$ פעמים. לכן מספר הפעמים שהקשת (u,v) יכולה להיות צוואר בקבוק לשיפור הזרימה היא לכל היותר $\frac{n}{2}$. בכל איטרציה יש לכל היותר קשת 1 שהיא צוואר בקבוק לשיפור הזרימה, סה"כ מספר האיטרציות הוא לכל היותר $2m \cdot \frac{n}{2} = mn$.

מסקנה: האלגוריתם רץ בזמן פולינומי, $O(m^2n)$) פעולות אריתמטיות) בגודל הקלט.

סיבוכיות המקום: $O(m+n)$ בנוסף לקלט.

8.3 תרגול 7:

- **הגדרות:**

1 שטף של זרימה: מוגדר להיות סכום הזרימה שיוצאת מהמקור $|f| = \sum_{u:(s,u)\in E} f(s,u)$

2 חתך: ברשת זרימה, הוא חלוקה זרה של הקודקודים $V = S \uplus T$ כך ש $s \in S \land t \in T$.

3 קיבול של חתך: מוגדר להיות $c(S,T) = \sum_{(u,v):u\in S, s\in T} c(u,v)$.

- **טענה:** לכל חתך $s-t$ ולכל זרימה חוקית f מתקיים: $|f| \leq c(s,t)$. כלומר ־ קבוצת כל החתכים חוסמת מלמעלה את קבוצת השטפים של כל הזרימות החוקיות.

- **משפט השטף והחתך:** זרימת מקסימום שווה לקיבולת של החתך המינימלי (יש נקודת מפגש בין הקבוצות).

מסקנה: אם מתקיים $|f| = c(s,t)$ אזי f מקסימלית ו $s-t$ חתך מינימלי.

- **שימושים לחתך מינימלי ברשת זרימה:**

1 ־ מציאת דרגת קשירות של גרף: יהי G גרף קשיר לא מכוון, נסמן ב $c(G)$ את המספר המינימלי של צלעות אשר הסרתן מהגרף הופכת אותו ללא קשיר. אנו רוצים למצוא את $c(G)$ עבור גרף G.

אלגוריתם:

א: נגדיר קבוצת קודקודים V , את הקשתות בגרף המקורי נשכפל לשני הכיוונים ונסמן ב $E^{\Rightarrow}$, נגדיר קיבול $c(e) = 1$ לכל $e \in E^{\Rightarrow}$. נגדיר מקור $s \in V$ קודקוד שרירותי כלשהו.

ב: לכל $v \neq s$ נמצא חתך מינימלי ברשת בעזרת זרימה מקסימלית על הרשת $N = (V, E^{\Rightarrow}, c, s, v)$ (הבור הוא כל פעם קודקוד אחר). נונסמן את הקיבול של החתך ב $c(v)$.

ג: נחזיר את $min_{v\neq s}\{c(v)\}$ ששווה ל $c(G)$.

זמן ריצה: בנינו $O(V)$ רשתות זרימה כאשר בכל אחת אותו מספר קודקודים כמו בגרף המקורי ו $2|E|$ צלעות. בניית כל רשת $O(E)$. מציאת החתך באמצעות EK בעלות של $O(E \cdot V^2)$ לכן זמן הריצה בסה"כ: $O(V(E+V\cdot E^2)) =$ $O(V^2 \cdot E^2)$.

2 ־ בעיית המשקיעים והשחקנים: אנו מקבלים כקלט קבוצת שחקנים $A = \{a_1, ..., a_n\}$, ומשכורות של שחקנים $S = \{s_1, ..., s_n\}$, וקבוצת משקיעים $B = \{b_1, ..., b_k\}$, וקבוצת שחקנים שהמשקיעים אוהבים $B^{\sim} = \{A_1, ..., A_k\}$

(כלומר ־ $A_i \subseteq A$ שהמשקיע b_i אוהב), בנוסף נתון סכום השקעה של כל משקיע $D = \{d_1, ..., d_k\}$.
ניתן לחשוב על הבעיה בצורה של גרף דו צדדי כאשר בצד אחד המשקיעים ובצד השני השחקנים, וקיימת קשת בין שחקן למשקיע אם המש'יע רוה שהשחקן הזה ישחק בסרט.
הפלט הוא: תתי קבוצות $A' \subseteq A$ ו $B' \subseteq B$ המקיימות: **חוקיות ־** לכל $b_i \in B'$ מתקיים $A_i \in A'$. **אופטימליות ־** הרווח של A', B' מקסימלי, כאשר הרווח מוגדר להיות $P(A', B') = \sum_{i:b_i \in B'} d_i - \sum_{i:a_i \in A'} s_i$
אלגוריתם:
א: נבנה רשת זרימה $N = \{V, E, c, s, t\}$ כך: **קודקודים:** $V = \{s, t\} \cup A \cup B$. **צלעות:** $E = \{(s, b_i) | b_i \in B\} \cup \{(b_j, a_i) | a_i \in A_j\} \cup \{(a_i, t) | a_i \in A\}$. **פונקציית קיבול:**

$$c(u,v) = \begin{cases} d_i & u = s, v = b_i \\ \infty & u = b_i, v = a_j \\ s_i & u = a_i, v = t \end{cases}$$

ב: כעת, נמצא חתך מינימלי ברשת ונסמנו $S - T$.
ג: נגדיר $A' = A \cap S$, $B' = B \cap S$ ונחזיר את A', B'.
זמן ריצה: בנינו רשת עם $O(n+k)$ קודקודים ו $O(nk)$ קשתות. ולכן בניית הרשת תקח $O(nk)$. מציאת חתך מינימלי $O((n+k) \cdot (nk)^2)$. הגדרת הקבוצות A', B' בעלות של $O(n+k)$. לכן זמן הריצה הכולל הוא $O((n+k) \cdot (nk)^2)$.
הערה: חשוב לנתח את זמן הריצה במונחי הקלט, גם אם בנינו רשת זרימה במהלך האלגוריתם צריך לחזור ולתרגם את $|V|, |E|$ למונחי קלט. (במקרה שלנו n, k).

9 שבוע 9:

9.1 הרצאה 16:

- **הגדרה ־ קבוצה בלתי תלויה:** קבוצת צמתים I היא ב"ת אם לכל $e \in E$ מתקיים $|e \cap I| \leq 1$.

- **הגדרה ־ כיסוי בצמתים:** אנו מקבלים כקלט גרף סופי לא מכוון $G = (V, E)$. קבוצה $X \subseteq V$ נקראת כיסוי בצמתים של G, אם לכל $e \in E$ מתקיים $e \cap X \neq \emptyset$.
  מסקנה: $V \backslash X$ היא קבוצה בלתי תלויה ב G.
הערה: הבעיה של מציאת כיסוי בצמתים בעל גודל מינימלי, שקולה לבעיה של מציאת קבוצה ב"ת בעלת גודל מקסימלי.

- **בעיית כיסוי בצמתים:** נניח כי $G = (L, R, e)$ הוא **גרף דו"צ**, ו לכל $e \in E$ מתקיים: $|e \cap L| = |e \cap R| = 1$
האלגוריתם של *konig*: אנו נסתכל על שידוך מקסימום $M \subseteq E$ בגרף G. נסמן ב Z את קבוצת הצמתים הבאה ־ היא מכילה את כל הצמתים ב L שאינם משודכים, בנוסף היא מכילה את כל הצמתים שניתן להגיע אליהן מצומת לא משודך ב L דרך מסלול מתחלף.
  בנוסף נגדיר S קבוצה באופן הבא: $S = (L \backslash Z) \cup (R \cap Z)$.
  טענה: הקבוצה S היא כיסוי בצמתים שגודלו מינימלי.

מסקנה מההוכחה: בגרף דו"צ מתקיים כי הגודל של כיסוי בצמתים מינמלי שווה לגודל של שידוך מקסימום.
כיצד האלגוריתם יעבוד: **1:** תחילה נחשב שידוך מקסימום M, (יש לנו אלגוריתם כזה שרץ בזמן של $O(E \cdot V)$). **2:** אח"כ נמצא את הקבוצה Z בעזרת פרוצדורה דמויית BFS בזמן של $O(E+V)$. **3:** חישוב של S - מעבר על כל הצמתים $O(V)$.
סיבוכיות המקום: $O(E+V)$.

- **בעיית כיסוי בצמתים עבר גרף כללי:** אנו נמצא שידוך שלא ניתן להרחבה (אין לנו עוד לאן להתרחב) באופן הבא: נבחר קשת ונסיר את כל הקשתות שיש להן קודקוד משותף איתה. נמשיך כך עד שלא ישארו לנו קשתות.
טענה: יהי M שידוך שלא ניתן להרחבה, אזי הקבוצה $S = \{x \in V : \exists e \in M : x \in e\}$ היא כיסוי בצמתים.
אנו יודעים כי $|S| = 2|M| \leq 2|M_{max}|$ כלומר S היא כיסוי בצמתים שגודלו לכל היותר פי 2 מהאופטימום.
סיבוכיות: $O(E+V)$ פעולות ומיקום.

- **בעיית כיסוי בצמתים עבר גרף ממושקל:** נתון גרף סופי לא מכוון $G=(V,E)$, ונתונה פונקציית משקל חיובית על הצמתים $w : V \Rightarrow N$. אנו רוצים למצוא כיסוי בצמתים קל ביותר.
אלגוריתם לחישוב כיסוי בצמתים קל: עבור כל קשת $e = \{u,v\} \in E$ נחזיק משתנה עזר שנסמנו ב $y_{u,v}$ (משתנה זה דומה ל "חלוקת כסף" לכל קשת).
איתחול: לכל קשת $(u,v) \in E$ נקבע $y_{u,v} = 0$, אחכ נעבור על כל הקשתות בסדר כלשהו - עבר קשת $(u,v) \in E$ נעדכן: $y_{u,v} = min\left\{w(u) - \sum_{s:(u,s)\in E} y_{u,s}, w(v) - \sum_{s:(s,v)\in E} y_{s,v}\right\}$.
בסיום עדכון המשתנה y נגדיר: $S = \left\{u : w(u) = \sum_{s:(u,v)\in E} y_{u,v}\right\}$ האלגוריתם יחזיר כפלט את S.
סיבוכיות זמן ומיקום: $O(E+V)$.

9.2 תרגול 8 - תכנון לינארי:

- **מוטיבציה:** אנו רוצים למצוא אופטימום (מינימום או מקסימום) של פונקציה לינארית על המשתנים, בתחום שמוגדר ע"י מערכת אי שוויונים לינארים.

- **הגדרה - בעיית תכנון לינארי:** בעיית אופטימיזציה תיקרא בעיית תכנון לינארי (LP) אם ניתן לכתוב אותה בצורה הבאה:

$$max_{X\in R^n}\left\{C^T X\right\} s.t\ AX \leq b \wedge x \geq 0$$

כאשר X הוא ווקטור של n משתנים ממשיים, ו $C \in R^n, b \in R^m, A \in R^{m \cdot n}$ (A הינה מטריצה ו b, c הם ווקטורים).
הערות:
1: הצורה הנ"ל נקראת הצורה הסטנדרטית לבעיית LP, ישנן צורות שונות השקולות לצורה הסטנדרטית.
2: אי השוויונים $AX \leq b \wedge x \geq 0$ מוגדרים כאי שוויון לכל קאורדינטה בנפרד.
3: זוהי בעיית אופטימיזציה קלאסית, יש סט פתרונות חוקיים, (כל ווקטור X המקיים את אי השוויונים). ואנו מחפשים בתוך קבוצה זו את הווקטור האופטימלי שממקסם את $C^T X$.
4: ניתן להגדיר את בעיית התכנון הלינארי גם כבעיית מינימיזציה. על ידי שינוי הסימן ל A, b, c כך:

$$min\left\{C^T X\right\} s.t\ AX \geq b \wedge x \geq 0$$

- **הגדרה - על מישור:** יהי ווקטור $a \in R^n$ וסקלאר $b \in R$, הקבוצה $\{x \in R^n : a^T X = B\}$ נקראת על מישור.

- **הגדרה - חצי מרחב:** הקבוצה $\{x \in R^n : a^T X \leq B\}$ נקראת חצי מרחב.

- **הגדרה - פוליהדרון:** קבוצת נקודות ב R^n שניתן לתאר בצורה $\{x \in R^n : aX \leq B\}$ כאשר $b \in R^m, A \in R^{m \cdot n}$ נקראת פוליהדרון.

  נשים לב: שקבוצת פתרונות לבעיית תכנון לינארי היא פוליהדרון, הבנוי מחיתוך של $m+n$ חצאי מרחבים.

- **פתרון לבעיית LP:** ידועים כמה אלגוריתמים לפתרון בעיות LP בזמן פולינומי בגודל הקלט ($m+n$). אם זאת בקורס לא נראה אלגוריתם כזה, אך נניח שניתן לפתור בזמן פולינומי ב $m+n$.

- **התרמיל השיברי כבעיית LP:** נתונים לנו n פריטים שלכל אחד משקל w_i וערך v_i. אנו מחפשים את ווקטור $X \in R^n$ כך ש $x_i \leq 1$ וגם $\sum w_i x_i \leq W$ הממקסמת את $\sum w_i x_i$.

  נגדיר את הווקטורים:

  1: $max_{X \in R^n} C^T X$ כאשר $c = v$.

  2: A תהיה מטריצה שהשורה הראשונה בה תכיל את האיברים w_i, והשורה המתאימה ב b שווה ל W.

  3: חוץ מהשורה הזו: הווקטור b יכיל אחדות. והמטריצה A תהיה מטריצת היחידה.

  נשים לב: שכדי לקבל פתרון לתרמיל השלם היינו צריכים להוסיף אילוצים $x_i \in Z$. לבעיה כזו קוראים תכנון לינארי בשלמים (ILP).

- **הגדרה - בעיית ILP:** בעיה תיקרא ILP אם יתן לכתוב אותה כך: $max \left\{C^T X\right\} s.t\ AX \leq b \wedge x \geq 0 \wedge X \in Z$ חלק גדול מהבעיות שמעניינות אותנו ניתנות להצגה כ ILP, אך ILP היא בעיית NP קשה.

- **שטף מקסימלי כבעיית LP:** בהינתן רשת זרימה N, נשים לב שנו רוצים למקסם את $\sum_{(s,i)\in E} 1 \cdot f_{(s,i)}$. נחפש את $X \in R^{|E|}$ ונגדיר את פונקציית המטרה להיות $max_{x \in R^{|E|}} d^T X$, כאשר $X = [1, f_{(i,j)}, 1]$ ואינדיקטור $d_{(i,j)} = 1_{i=s}$.
  נדרוש שלכל (i,j) יתקיים: $0 \leq f(i,j) \leq c(i,j)$. וגם:

$$\forall i \in V \backslash \{s,t\} : \sum f(j,i) = \sum f(i,j) \iff \sum_{j \in V} f(j,i) - \sum_{j \in V} f(i,j) = 0$$

אך LP דורש אי שוויונים וכאן יש לנו שוויון, לכן:

$$\sum_{j \in V} f(j,i) - \sum_{j \in V} f(i,j) \leq 0 \wedge \sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) \leq 0$$

בניית הצורה הסטנדרטית: אנו מחפשים את $max_{x \in R^{|E|}} d^T X : AX \leq b \wedge x \geq 0$ כאשר $X = [1, f_{(i,j)}, 1]$ ואינדיקטור $d_{(i,j)} = 1_{i=s}$. הווקטור b יכיל $|E|$ שורות של $c(i,j)$ ושאר השורות $2(|V|-2)$ יכילו אפסים. המטריצה A תכיל את מטריצת היחידה $I_{|E|}$ בחלקה העליון ($|E|$ שורות), ובחלקה התחתון היא תכיל מטריצה $M \in R^{|E| \cdot 2(|V|-2)}$ שתייצג

את האילוצים של שימור החומר. נגדיר את המטריצה M באופן הבא:

$$m_{l^1,k}=\begin{cases}+1 & \exists u\in V: k=(u,l)\in E\\ -1 & \exists u\in V: k=(l,u)\in E\\ 0 & else\end{cases}$$

זמן ריצה: נסמן ב $f(m+n)$ את זמן הריצה של אלגוריתם LP. הגדרת d עולה $O(|E|)$. הגדרת b עולה $O(|E|+|V|)$. הגדרת A עולה $O(|E|\cdot|V|)$. פתרון LP עולה $O(f(|E|+|V|))$. לכן סה"כ: $O(|E|\cdot|V|+f(|E|+|V|))$.

- **בעיית הסרת משולשים:** אנו מקבלים כקלט גרף G לא מכוון. והפלט הינו תת קבוצה של צלעות $S\in E$ מינימלי, שהסרתה מ $\langle G\rangle$תשאיר גרף $G'=(V,E\backslash S)$ ללא משולשים. (משולש הוא קליקה מגודל 3).
נגדיר ווקטור משתנים $X\in\{0,1\}^{|E|}$ המציין לכל צלע $e\in E$ האם היא ב S או לא. נשים לב ש $|S|=\sum_{e\in E}X[e]$, לכן נגדיר את פונקציית המטרה להיות: $min_{x\in R^{|E|}}\sum_{e\in E}X[e]$. בנוסף נדרוש שלכל משולש ב G לפחות אחת מהצלעות במשולש מקיימת: $x[e]=1$.
סה"כ נקבל את התכנית הלינארית בשלמים הבאה:

$$min_{x\in R^{|E|}}C^TX\ s.t\ \forall u,v,w\in V:(v,u),(u,w),(v,w)\in E, x(v,u)+x(u,w)+x(v,w)\geq 1\wedge e\in E: x[e]\leq 1\wedge x\geq 0\wedge x\in$$

10 שבוע 10:

10.1 הרצאה 17:

- **המשך בעיית כיסוי בצמתים:**
- **טענה:** אם S היא התוצאה של האלגוריתם, אזי היא כיסוי בצמתים של G.
- **טענה:** בסיום ריצת האלגוריתם עבור פתרון S מתקיים: $\sum_{u\in S}W_u\leq 2\cdot\sum_{(u,v)\in E}y_{u,v}$
- **טענה:** בסיום ריצת האלגוריתם מתקיים: משקל כיסוי בצמתים קל ביותר $\leq\sum_{(u,v)\in E}y_{u,v}$.
מסקנה: האלגוריתם שלנו מחשב 2 - קירוב לבעיית כיסוי בצמתים ממושקל.

10.2 הרצאה 18:

- **בעיית התחזיות והמומחים:** אנו מקבלים כל יום תחזיות של n מומחים שמפרסמים כל יום תחזית בינארית מה יקרה למחרת (לדוגמה - ירד גשם או לא). אנחנו נחליט בכל יום החלטה בהתאם לתחזית, ונרצה לא להיכשל יותר מהמומחה הכי טוב (בהסתכלות בדיעבד).
האלגוריתם יעבוד באופן הבא: נחזיק קבוצה X של מומחים שתכיל את כל המומחים, ונחליט ע"פ דעת הרוב בקבוצה X. אם טעינו - נוריד את המומחים שטעו מ X. אם $X=\emptyset$ נאתחל שוב עם כל המומחים.

משפט: אם המומחה הטוב ביותר שוגה $k \geq 0$ פעמים, אזי האלגוריתם שוגה לכל היותר $(k+1)log_2(n)$ פעמים.

הוכחת המשפט: באנדוקציה על k.

בסיס: $k = 0$ - בכל פעם שהאלגוריתם שוגה - אנו מסירים מ X לפחות חצי מהמומחים ב X. יש מומחה שאינו שוגה כלל ולכן X אף פעם לא ריקה. לכן מספר השגיאות שלנו הוא לכל היותר $log_2(n)$ כי בבסיס $X = n$.

צעד: נניח את נכונות הטענה אם המומחה הטוב ביותר שוגה $l < k$ פעמים, ונוכיח עבור $l = k$. נניח כי המומחה ה i שוגה סה"כ k פעמים. נתבונן בצעדי האלגוריתם עד ש X מתרוקנת בפעם הראשונה - עד לנקודה זו האלגוריתם שגה לכל היותר $log_2(n)$ פעמים, וגם i שגה לפחות פעם אחת. החל מנקודה זו i שוגה לכל היותר $k-1$ פעמים, לכן לפי הנחת האנדוקציה האלגוריתם לא ישגה יותר מ $k \cdot log_2(n)$ פעמים, נוסיף את הפעם האשונה ונקבל את הדרוש.

הצעת ייעול לאלגוריתם: נסמן את התחזיות ב $\{1, -1\}$. נסמן ב c_i^t את התחזית של המומחה ה i בצעד ה t. בנוסף, נחזיק לכל מומחה משקל w_i^t שיאותחל ל 1 בהתחלה. ההחלטה שלנו בצעד ה t תהיה הסימן של: $\sum_i w_i^t c_i^t$. עדכון: לכל מומחה i שטעה - נוריד את משקלו בחצי, ולכל מומחה שצדק לא נבצע שום שינוי במשקל. נסמן ב m_i^t את מספר הטעויות של המומחה ה i ב t הצעדים הראשונים. בנוסף נסמן ב $M(t)$ את מספר הטעויות של האלגוריתם ב t הצעדים הראשונים.

משפט: לכל מומחה i ולכל t מתקיים: $M(t) \leq \frac{1}{log_2\left(\frac{4}{3}\right)} \cdot (m_i^t + log_2(n))$.

הוכחה: נסמן $\Phi(t) = \sum_i w_i^t$. על פי האתחול $\Phi(1) = n$, וגם לכל מומחה i ולכל t: $\Phi(t+1) > w_i^{t+1} = 2^{-m_i^t}$. כאשר האלגוריתם טועה: משקל המומחים ששגו הוא לפחות חצי מהמשקל הכולל, כלומר - אם שוגים בצעד t אז:

$$\sum_{i-false} w_i^t \geq \frac{1}{2} \sum_i w_i^t$$

. לכן:

$$\sum_i w_i^{t+1} = \sum_{i-false} \frac{1}{2} \cdot w_i^t + \sum_{i-true} w_i^t \leq \frac{3}{4} \sum_i w_i^t$$

כלומר -

$$\Phi(t+1) \leq \left(\frac{3}{4}\right)^{M(t)} \cdot \Phi(1) = \left(\frac{3}{4}\right)^{M(t)} \cdot n.$$

נחבר את החסם העליון והתחתון ונקבל:

$$2^{-m_i^t} \leq \left(\frac{3}{4}\right)^{M(t)} \cdot n$$

נוציא log משני הצדדים ונקבל:

$$-m_i^t \leq log_2\left(\frac{3}{4}\right) \cdot M(t) + kog_2(n)$$

נעביר אגפים ובסוף נקבל ש:

$$M(t) \leq \frac{1}{log_2\left(\frac{4}{3}\right)} \cdot \left(m_i^t + log_2(n)\right)$$

כנדרש.

10.3 תרגול 9 - אלגוריתמי קירוב:

- **אלגוריתמי קירוב:** אלגוריתמי אופטימיזציה, שמביאים לנו תוצאה קרובה לתוצאה שאנו מחפשים, ולא את התוצאה המדוייקת. נשתמש בהם כשאין לנו פתרון פולינומיאלי, או שיש לנו פתרון פולינומיאלי אך אנו רוצים לשפר את זמן הריצה.

- **הגדרה - c מקרבת:** יהי x מרחב הפתרונות החוקיים לבעיה $f : x \Rightarrow R^+$. נחלק לבעיות:
מקסימיזציה: המטרה היא למצוא את $x^{opt} = max\left(f(x)\right)$, נאמר שהבעיה היא c מקרבת. אם יש לנו אלגוריתם **שלכל קלט** מחזיר פתרון x שמקיים: $f(x) \geq \frac{1}{c} \cdot f\left(x^{opt}\right)$.
מינימיזציה: המטרה היא למצוא את $x^{opt} = min\left(f(x)\right)$, נאמר שהבעיה היא c מקרבת. אם יש לנו אלגוריתם **שלכל קלט** מחזיר פתרון x שמקיים: $f(x) \leq c \cdot f\left(x^{opt}\right)$.

- **הגדרות:**
1: ליטרל: עבור משתנה בוליאנו x, ליטרל הוא x או השלילה של x.
2: פסוקית - $3\!-\!CNF$: אוסף של 3 ליטרלים המחוברים עם פעולות "או".
3: נוסחת $3\!-\!CNF$: "גימום" של פסוקיות $3-CNF$.

- **בעיית $3-SAT$:** היא בעיית הכרעה שבאה לבדוק האם קיימת לנוסחת $3-CN$ השמה מספקת.
אנו מקבלים כקלט נוסחת $3-CNF$ עם m פסוקיות ו n משתנים: $x_1, ..., x_n$. נניח כי מתקיים $n \leq 3m$. בנוסף, נניח כי בכל פסוקית המשתנים שונים זה מזה. הפלט יהיה השמה המספקת את כל הפסוקיות (בעיית NP קשה).
פלט חלופי - השמה המספקת כמה שיותר פסוקיות.
אלגוריתם קירוב לפתרון הבעיה: נקח את ההשמה $FFF...F$ ונראה כמה פסוקיות היא מספקת, ואחכ נעשה את אותו הדבר עבור ההשמה $TTT...T$, ונחזיר את המקסימלית מבניהם.
הוכחת נכונות: נוכיח כי האלגוריתם הוא 2 - מקרב.
חוקיות: החזרנו אחת משתי השמות חוקיות לכן ערך ההחזרה חוקי.
מקרב: נסתכל על פסוקית c_i, בוודאות היא תסופק ע"י הכל T או הכל F או שניהם. נסתכל על מספר הפסוקיות המסופקות ע"י T וע"י F ונסמנן: $f + t$ אנו יודעים שמתקיים :

$$F + T \geq m \Rightarrow max(f, t) \geq \frac{1}{2} \cdot m \geq \frac{1}{2} \cdot opt \Rightarrow\ -2$$

הערות:
1: האלגוריתם היה פועל עם כל השמה ושלילתה.
2: האלגוריתם היה פועל גם ל $4-CNF$ וכו'...

- **בעיית התרמיל השלם:** נראה כי האלגוריתם החמדן שלוקח כל פעם פריט לפי משקלו הסגולי $p_i = \frac{w_i}{v_i}$, לא משיג 2 - קירוב. ונציע אלגוריתם שמשיג 2 - קירוב לבעיה.
נראה כי האלגוריתם לא c מקרב לאף $c > 0$: בינתן c נבחר $v > c + 1$. ונגדיר את הפריטים:

$element$	w_i	v_i	p_i
1	1	1	1
2	$v-1$	v	$\frac{v-1}{v} = 1 - \frac{1}{v}$

האלגוריתם החמדן יבחר רק את פריט 1 , בעוד האופטימלי יבחר את פריט 2.

$$\frac{v-1}{c}=1\Rightarrow\frac{1}{c}f(x^{opt})\geq f(x)$$

נמצא אלגוריתם שמשיג את הקירוב הדרוש: נמיין את כל הפריטים לפי p_i. נמצא את ה k המינימלי עבורו $\sum_{i=1}^{k}v_i>V$ (האלגוריתם החמדן x_1 יבחר את כל ה $k-1$ פריטים ראשונים). נגדיר אלגוריתם x_2 שיבחר רק את הפריט ה k, ונחזיר $argmax_x(f(x_1),f(x_2))$.

נוכיח חוקיות: x_1 חוקי מחוקיות האלגוריתם החמדן, x_2חוקי כי הוא מכיל רק פריט אחד, ועבור כל פריט v מתקיים $v_i<V$.

נוכיח קירוב: נסתכל על הפתרון של התרמיל השיברי Z^{opt}, הפתרון הוא מהצורה $k-1$ איברים שלמים ועוד פריט שברי $0\leq\alpha\leq1$.

אנו יודעים ש:

$$f(x^{opt})\leq f(Z^{opt})=\sum_{i=1}^{k-1}w_i+\alpha w_k=f(x_1)+\alpha f(x_2)\leq f(x_1)+f(x_2)\leq2\cdot max\left(f(x_1),f(x_2)\right)=2f(x)$$

כנדרש.

זמן ריצה: $O(n\cdot log(n))$.

- **בעיית חתך מקסימום בגרף:** אנו מקבלים כקלט גרף לא מכוון G, ואנו מחפשים את החתך (A,B) מקסימלי.

האלגוריתם יפעל כך: נגדיר $A=\emptyset,B=V$, אח"כ נעבור על כל הקודקודים לפי הסדר $1....n$, ולכל קדקוד נבדוק: אם מספר השכנים שלו בקבוצה שהוא נמצא בה גדול ממספר השכנים בקבוצה השניה - נחליף לו קבוצה.
אח"כ נבדוק אם בשלב הקודם (בריצה על כל הקודקודים) החלפנו איזשהו קודקוד - נחזור שוב על השלב הקודם.
נעצור כשלא נשאר אף קודקוד להחליף.

נוכיח שהאלגוריתם עוצר: בכל איטרציה של האלגוריתם כאשר החלפנו קודקוד בין הקבוצות - העלינו את גודל החתך בלפחות 1 (החתך רק גדל ולא קטן). החתך המקסימלי חסום ע"י $|E|$ מלמעלה, לכן לכל היותר לאחר $|E|$ איטרציות אנו נעצור.

הוכחת חוקיות: התחלנו עם חתך חוקי, ובכל איטרציה רק שינינו לקודקוד יחיד את הקבוצה אליה הוא שייך ולכן A,B עדיין חלוקה זרה, והחתך עדיין חוקי (צריך להוכיח באנדוקציה).

נוכיח קירוב: לכל קודקוד $v\in V$ נסמן את דרגת הקדקוד d, בנוסף נסמן ב v_c את מספר הצלעות הנוגעות בקודקוד v ונוגעות בחתך שהחזרנו.

טענה: $|(A,B)|\geq\frac{1}{2}|opt|$. הוכחה:

$$|(A,B)|=\frac{1}{2}\cdot\sum_{v\in V}v_c\geq\frac{1}{2}\sum_{v\in V}\frac{1}{2}\cdot d_v=\frac{1}{4}\sum_{v\in V}d_v=\frac{1}{4}\cdot2|E|=\frac{1}{2}|E|\geq\frac{1}{2}|opt|$$

11 שבוע 11:

11.1 הרצאה 19:

- **המשך בעיית המומחים והתחזיות:**

- **נניח שנעדכן באופן יותר מתון:** עבור כל מומחה שטועה נוריד את משקלו ב $\varepsilon > 0$. נקבל:

$$M(t) \leq \frac{ln(1+\varepsilon)}{ln\left(\frac{2+2\varepsilon}{2+\varepsilon}\right)} \cdot m_i^t + \frac{ln(n)}{ln\left(\frac{2+2\varepsilon}{2+\varepsilon}\right)}$$

כך שהחלק הראשון קטן מ $1+\varepsilon$ והחלק השני קטן מ $\frac{1}{\varepsilon}$.

נניח שבכל יום מוטלים על המומחים קנסות בקטע $[-1,1]$. נסמן ב $c_i^t \in [-1,1]$ אם הקנס שמוטל על המומחה ה i ביום ה t.

האלגוריתם בוחר בכל יום t התפלגות p^tעל המומחים (לפני שחושפים את הקנסות), כלומר - לכל i: $p_i^t \geq 0$ וגם $\sum_{i=1}^n p_i^t = 1$. ביום ה t האלגוריתם משלם $\sum_{i=1}^n p_i^t \cdot c_i^t$. סה"כ הקנסות שמשלם האלגוריתם עד זמן T יהיה: $\sum_{t=1}^T \sum_{i=1}^n p_i^t \cdot c_i^t$, ואת התשלום הזה נרצה להשוות לתשלום של המומחה הטוב ביותר, כלומר: $min_{i\in[n]} \sum_t c_i^t$.

נחזיק משקלים לכל המומחים, ונסמן ב w_i^t את המשקל של המומחה ה i בתחילת היום ה t. ההתפלגות שהאלגוריתם "משחק" היא: $p_i^t = \frac{w_i^t}{\sum_{j=1}^n w_j^t}$. המשקלים מתעדכנים אחרי חשיפת הקנסות באופן הבא: $w_i^{t+1} = (1-\varepsilon \cdot c_i^t)w_i^t$ עבור $\varepsilon > 0$ שיקבע בהמשך.

- **משפט:** בהינתן ש $0 < \varepsilon \leq \frac{1}{2}$ מתקיים אחרי T צעדים, לכל מומחה i:

$$\sum_{t=1}^T \sum_{i=1}^n p_i^t \cdot c_i^t \leq \sum_{t=1}^T c_i^t + \varepsilon \sum_{t=1}^T |c_i^t| + \frac{1}{\varepsilon} ln(n)$$

11.2 הרצאה 20 :

- **נתבונן בתכנית הלינארית הבאה:** $min \sum w_v x_v \ s.t \ x \geq 0 \ \wedge x_u + x_v \geq 1 \ \forall (u,v) \in E$ (בעיית כיסוי בצמתים). נניח שאנו רוצים לפתור את התכנית הזו במקורב. בכדי לפתור את התכנית במקורב מספיק לפתור את בעיית ההכרעה הבאה: האם קיים $x \geq 0$ עבורו לכל קשת $(u,v) \in E$ מתקיים $x_u + x_v \geq 1$ וגם $\sum_v w_v x_v \leq \beta$ עבור קלט β.

בהינתן פתרון לבעיית ההכרעה הזו אפשר לפתור את התכנית הלינארית על ידי חיפוש בינארי על β. לכן, ניתן להחליף את הבעיה המקורית בבעיה הבאה - נבדוק האם קיים $x \geq 0$ עבור מערכת אי שוויונים לינארים $Ax \geq b$ מתקיימת (עבור מטריצת אילוצים A ווקטור b).

פתרון מקורב לבעיית ההכרעה הזו: נזהה בוודאות שאין x כזה, או נמצא $x \geq 0$ עבורו לכל אי שוויון i יתקיים $A_i x \geq b_i - \delta$ עבור $\delta > 0$ כלשהו.

נסמן את מספר האילוצים ב m ואת מספר המשתנים ב n, כך ש: $A \in R^{m \cdot n}, x \in R^n, b \in R^m$. מפתרון מקורב לבעיית ההכרעה אפשר למצוא פתרון קרוב לאופטימלי לתכנית הלינארית.

הנחה: נצטרך להניח שקיים בידינו אלגוריתם O לבעיה פשוטה יותר - בהינתן התפלגות P על השורות ש A, לבדוק האם קיים $x \geq 0$ עבורו $\sum_{i=1}^{m} P_i A_i x \geq \sum_{i=1}^{m} P_i b_i$.

נניח בנוסף: שאם קיים פתרון x כזה, אנחנו מוצאים x כזה עבורו לכל i : $|A_i x - b_i| \leq \rho$ עבור איזשהו פרמטר ρ שנקרא "הרוחב" של הבעיה.

נשים לב: אם קיים $x \geq 0$ עבורו $Ax \geq b$ אזי ה x הזה מקיים את אי השוויון: $\sum_{i=1}^{m} P_i A_i x \geq \sum_{i=1}^{m} P_i b_i$.

משפט: בהינתן $\delta \geq 0$ ואלגוריתם O עם $\rho \geq \frac{\delta}{2}$, קיים אלגוריתם שקורא ל O לכל היותר $O\left(\frac{\rho^2}{\delta^2} log(m)\right)$ פעמים, ומחזיר $x \geq 0$ עבורו לכל $i \in [m]$: $A_i x \geq b_i - \delta$, או מזהה נכון שאין $x \geq 0$ עבורו $Ax \geq b$.

11.3 תרגול 10 - $Online - learning$:

- **בעיות סיווג -** $clasification$: בבעיות סיווג המטרה היא לחלק קבוצה של פריטים לתתי קבוצות. לדוגמה - לסווג תפוחים לקבוצה של בשלים ובוסר. אנחנו נתמקד בסיווג בינארי (סיווג לשתי קבוצות).

- **הגדרה:** ישנם T סיבובים $t = 1, ...T$, בכל סיבוב מקבלים פריט $x_t \in X$ לכל פריט יש סיווג $y_t \in \{-1, 1\}$ שאותו נרצה למצוא. בנוסף יש לנו קבוצה של n מומחים $f_1, ...f_n$ שכל מומה נותן את דעתו על הסיווג של x_t. באופן פורמלי: $f_i : x \Rightarrow \{-1, 1\}$ אנו לא יודעים מראש כמה המומחים טובים.

- **נשחק משחק:**

1: נאתחל את מספר הטעויות שלנו $mistakes = 0$.

2: עבור כל אחד מהסיבובים $t = 1....T$, מקבלים אוביקט מהסביבה x_t ורשימה של N סיווגים מהמומחים: $f_1^t, ...f_N^t$.
אנו יוצרים סיווג משלנו $\tilde{y_t}$ כפונקציה של סיווגי המומחים. מקבלים מהסביבה את הסיווג האמיתי y_t. אם $\tilde{y_t} \neq y_t$ נעלה ב 1 את הערך של $mistakes$.

המטרה: ליצור סיווגים $\tilde{y_1}, ...\tilde{t_t}$ כך שמספר הטעויות הכולל יהיה נמוך כמה שיותר.

- **אלגוריתם** $Halving$:

לכל סיבוב נשתמש רק במומחים שצדקו בכל הסיבובים הקודמים, מבין הסיווגים אנו נבחר את הסיווג שהרוב מסכימים עליו. פורמלית: בכל סיבוב נגדיר את: $experts_t = \{f_i \mid f_i^d = y_d \; \forall d = i, ...t - 1\}$.

נחזיר את הסיווג: $$\tilde{y_t} = \begin{cases} 1 & |\{f_i \in experts | f_i^t = 1\}| \geq |\{f_i \in experts | f_i^t = -1\}| \\ -1 & else \end{cases}$$

הערה: האלגוריתם מניח שקיים מומחה שלא טועה אף פעם, כי אחרת נוכל להישאר עם קבוצה ריקה.

טענה: אם יש מומחה f_iשצודק תמיד, אז מתקיים: $mistakes \leq log(N)$.

נשים לב שקיבלנו חסם עליון על כמות הטעויות שהאלגוריתם יכול לעשות, כ"כ חסם זה לא תלוי במספר הסיבובים.

נשים לב שחסם זה הוא הדוק. (ניתן להוכיח שזוהי השגיאה המינימלית **לכל** אלגוריתם סיווג תחת ההנחה שיש מומחה אחד מושלם).

- **אלגוריתם רוב ממושקל:**

לכל מומחה f_iניצור משקל w_i שתייצג כמה חשיבות אנו נייחסים לדעתו של המומחה ה i. נאתחל $w_i = 1$, בכל

סבוב נחזיר את הניחוש $y_t^{\sim} = sign\left(\sum_{i=1}^{N} w_i f_i^t\right)$. בכל סיבוב נעדכן את המשקל של מומחה שטעה להיות: $w_i^{t+1} = \frac{1}{2} w_i^t$. אם נסמן ב $mistakes_i$ את מספר הטעויות של המומחה ה i, הוכחנו בכיתה כי $mistakes_i \leq \frac{1}{log\left(\frac{4}{3}\right)} \cdot (mistakes_i + log(N))$. נשים לב שעבור מומחה מושלם מתקיים $mistakes_i \leq \frac{1}{log\left(\frac{4}{3}\right)} \cdot (0 + log(N))$. הקירוב הזה פחות טוב מהאלגוריתם של $halving$ אך אלגוריתם זה עובד גם בלי ההנחה שיש מומחה מושלם.

הערה: נשים לב שמיכיוון ש $y_t^{\sim} \in \{-1, 1\}$ צריך לשנות את הפונקציה $sign$ כך שתחזיר 1 או 1- במקרה של $sign(0)$ לכן במקרה זה נבחר שרירותית.

- **משקלות כפליים:**

נכליל את הבעיה - במקום לספור לכל מומחה כמה טעויות הוא ביצע, נגדיר פונקציה על הסיווג של כל מומחה בכל סיבוב, ונסמנה ב p: $p_i^t \in [-1, 1]$, נשים לב שניתן לתת קנס שלילי וזהו בעצם פרס על תשובה נכונה.

נשים לב שזוהי הכללה של הבעיה הקודמת, אם נגדיר $p_i^t = \begin{cases} 1 & f_i^t \neq y_t \\ 0 & else \end{cases}$ נחזור לבעיה המקורית.

במקרה הקודם מדדנו את עצמנו אל מול המומחה שטעה הכי מעט, בבעיה זו נבדוק את עצמנו מול המומחה שקיבל את הקנס הנמוך ביותר. אנו מחפשים את: $min_i \sum_{t=1}^{T} p_i^t = \sum_{t=1}^{T} p_{opt}^t$. נרצה להגדיר כמה קנס אנו נשלם עבור הסיווגים שלנו, כדי לעשות זאת במקום לבחור סיווג לפי רוב המומחים, נעבור להקשיב למומחה יחיד בכל סיבוב. בכל סיבוב נקבל את הקנס של המומחה לו הקשבנו.

האלגוריתם יפעל כך: לכל מומחה i נחזיק משקל $w_i = 1$ באתחול. בבואנו לבחור מומחה להקשיב לו בסיבוב t - נדגום מומחה מקבוצת המומחים לפי ההתפלגות $P_i^t = \frac{w_i^t}{\sum_j w_j^t}$.

מכיוון שבכל סיבוב אנו מגרילים את המומחה שלו נקשיב, אנו יכולים לקבל קנסות שונים כתלות במומחה שאותו באמת דגמנו. לכן, במקום להסתכל על קנס ספציפי שקיבלנו בהרצה מסויימת נתעניין בתוחלת הקנס על פני T הסימונים. כלומר נסמן $op = \sum_{t=1}^{T} E_{Pt}[op^t] = \sum_{t=1}^{T} \sum_{i=1}^{N} P_i^t \left(p_i^t\right)$. נסמן את הקנס בסיבוב ה t של המומחה הטוב ביותר ב p_{opt}^t. אנחנו רוצים ש op יהיה קרוב ככל הניתן ל $\sum_{t=1}^{T} p_{opt}^t$, בכיתה הגדרנו את כלל העדכון עבור המשקולות: $w^{t+1} = (1 - \varepsilon \cdot p_i^t) w_i^t$, עבור $0 < \varepsilon \leq 0.5$. אלגוריתם זה חסום מלעיל עם חסם הדוק יותר מאשר אלגוריתם רוב ממושקל.

12 שבוע 12 - אלגוריתמים הסתברותיים:

12.1 הרצאה 21 :

- **בעיית חתך גדול ביותר:** נתון גרף לא מכוון G, אנו רוצים למצוא חתך $F \subseteq E$ עם מספר מירבי של קשתות. (אנו לא מכירים אלגוריתם פולינומי לפתרון הבעיה הזו).

אלגוריתם הסתברותי לקירוב הבעיה:

1: נבחר קשת כלשהי $(u, v) \in E$ ונשים את u, v בצדדים שונים של החתך.

2: לכל $w \in V \backslash (u, v)$ נגריל צד באופן בלתי תלוי בהגרלות האחרות, ובהתפלגות אחידה. מספר הטלות המטבע של האלגוריתם שווה ל $|V| - 2$.

משפט: יהי F החתך שנוצר ע"י האלגוריתם אזי $\mathbb{E}[|F|] > \frac{|E|}{2}$.

מסקנה: $\mathbb{E}[|F|]$ הוא קירוב בפקטור 2 לגודל של חתך מקסימום.

12.2 הרצאה 22:

- הרעיון של הגרלה באופן ב"ת ובהתפלגות אחידה מאפשר פתרון של בעיות נוספות, לדוגמה:

- **בעיית סיפוק משוואות:** נתון אוסף משוואות לינאריות מעל $\mathbb{Z}_2$(מודולו 2). אנו רוצים למצוא הצבה של ערכי $0,1$ למשתנים, עבורה מספר המשוואות שמתקיימות ־ מקסימלי.

נשים לב כי בעיה זו היא הכללה של $max-cut$:

נגדיר משתנים x_v לכל צומת $v \in V$ ולכל קשת $(u,v) \in E$ נוסיף משוואה לינארית $x_u \oplus x_v = 1$, המשוואה מתקיימת אם"ם $x_u \neq x_v$. לכן פתרון הוא חתך שבו $S = \{u \in V : x_u = 0\}$ ומספר המשוואות שמתקיימות הוא בדיוק מספר הצלעות שנחתכות.

במקרה הכללי: אם נגריל את המשתנים בהתפלגות אחידה באופן ב"ת, ההסתברות שמשוואה מתקיימת היא: $\frac{1}{2}$ משום שלבחירת האיבר האחרון יש הצבה אחת שמקיימת את המשוואה. לכן תוחלת מספר המשוואות המתקיימות הוא חצי מכל המשוואות.

- **בעיית חתך מינימום (גלובלי) ־** $min-cut$: נתון גרף סופי לא מכוון G, אנו רוצים למצוא חתך $F \subseteq E$ שגודלו **מינימלי.**

הערה: אפשר לפתור את הבעיה הזו באמצעות $|V|-1$ חישובים של זרימת מקסימום (בגרף לא מכוון).

אלגוריתם הסתברותי לבעיה: נגדיר פעולה של כיווץ קשת ־ תהיי $e=(u,v)$ כיווץ של e מייצר גרף חדש, נסמנו ב $G \backslash e = (V', E')$ כך:

$$V' = V \backslash (u,v) \cup (uv)$$

כלומר החלפנו את u,v בצומת חדש שנקרא uv.

$$E' = \{f \in E : f \cap (u,v) = \emptyset\} \cup \{(uv,w) : ((u,w) \in E \vee (v,w) \in E) \wedge (w \notin (u,v))\}$$

כלומר ־ כיווץ מוחק את הקשת ומחבר את שני צדדיה.

כיווץ זה יכול ליצור קשתות מקבילות, כך שהגרף לא גרף פשוט.

האלגוריתם של $Karger$: כל עוד יש ב G יותר מ 2 צמתים ־ נבחר קשת בהתפלגות אחידה ונחליף את G ב $G \backslash e$.

בסיום נישאר עם שני צמתים וכל אחת מהן מייצגת קבוצה לא ריקה של צמתים בגרף הקלט, החלוקה של הגרף המקורי בקלט לשתי קבוצות לא ריקות ע"פ התוצאה, משרה חתך. אוסף הקשתות המקבילות בין שני הצמתים בתוצאה הסופית, הן קשתות החתך.

האלגוריתם מצליח אם"ם החתך שקיבלנו הינו חתך מינימום.

טענת עזר: אם $f \subseteq E$ הוא חתך מינימום ב G, אזי $|E(G) \geq \frac{|V|\cdot|f|}{2}|$. כלומר ־ מספר הקשתות ב G יחסי לגודל החתך.

מסקנה: אחרי הכיווץ הראשון מתקיים: $\mathbb{P}\left(f \subseteq E(G \backslash e)\right) \geq 1 - \frac{2}{|V|}$.

12.3 תרגול 11 ־ אלגוריתמים הסתברותיים:

- **הגדרה ־ אלגוריתם הסתברותי:** אלגוריתם הסתברותי הוא אלגוריתם אשר במהלך ריצתו משתמש בהטלות מטבע. כלומר ־ הוא מבצע הגרלות במהלך הריצה.
באופן פורמלי: לכל קלט, בזמן פולינומיאלי מחזיר בהסתברות גדולה כרצוננו פתרון חוקי ואופטימלי. ונכשל בהסתברות נמוכה.

- **סוגי אלגוריתמים הסתברותיים:** נבחין בין שני סוגים עיקריים של אלגוריתמים הסתברותיים
1 אלגוריתמי לאס וגאס: אלגוריתמים שבהם הטלות המטבע אינן משפיעות על פלט האלגוריתם, אלא על פרמטרים אחרים במהלך הריצה.
2 אלגוריתמי מונטה קרלו: אלגוריתמים שבהם הטלות מטבע משפיעות על הפלט. ובפרט ריצות שונות על אותו הקלט יכולות להסתיים בתוצאה שונה. בנוסף האלגוריתם עשוי להיכשל.

- **בעיית חיפוש מספרים במערך:** אנו מקבלים כקלט מערך A כאשר חצי מהערכים בו שווים ל 0 וחצי ל 1. אנו מחפשים אינדקס כלשהו שבו יש את הספרה 1.
הפתרון הנאיבי: נעבור על המערך לפי הסדר עד שנמצא את האינדקס שמקיים את הדרוש. זמן הריצה במקרה הגרוע $O(n)$.
פתרון לאס וגאס: נגריל באקראי אינדקס. אם קיבלנו את הדרוש נחזיר את האינדקס, אחרת ־ נגריל שוב ללא האינדקס הקודם. נשים לב שהאלגוריתם משתמש בהגרלות ולכן הוא הסתברותי, אך הוא תמיד מחזיר את הפלט הנכון.
זמן ריצה לאס וגאס: $O(n)$. אך מכיוון שאנו מגרילים אינדקסים באקראי, כאן לא משנה מה הקלט הנתון. הסיכוי להצליח בכל הגרלה בודדת הוא $\leq \frac{1}{2}$. בתוחלת, נצטרך בסהכ 2 הגרלות לכל היותר, לכן זמן הריצה של האלגוריתם הוא $O(1)$ בתוחלת.
פתרון מונטה קרלו: נגריל באקראי אינדקס. אם קיבלנו את הדרוש נחזיר את האינדקס, אחרת נחזיר $fail$.
זמן ריצה מונטה קרלו: אנו מבצעים הגרלה אחת, לכן זמן הריצה הוא $O(1)$. האלגוריתם רץ בזמן טוב משמעותית מהאלגוריתם הנאיבי, אך הוא לא תמיד מחזיר פתרון נכון. **נתמודד עם זה באופן הבא:** נדרוש תמיד כי אלגוריתמים הסתברותיים יחזירו תשובה נכונה לכל קלט בהסתברות גדולה כרצונינו.

- **הגדרה ־ גדולה כרצונינו:** הדרך המקובלת להגדרה היא ־ האלגוריתם צודק בהסתברות שהיא לפחות $1-\frac{1}{e^k}$ עבור $k \in N$ נתון. באופן זה אנו משאירים למשתמש לבחור כמה חשוב לו לקבל תשובה נכונה. ככל שנדייק, בד"כ נצטרך לשלם בזמן ריצה.

- **הערה:** אלגוריתמי מונטה קרלו נפוצים במיוחד בקרב בדיקות ראשוניות. המאפשרים לבדוק ראשוניות בהסתברות גדולה בזמן לוגריתמי.

- **ניפוח באלגוריתמים הסתברותיים:** אחת החוזקות של אלגוריתמים הסתברותיים היא ־ שניתן בקלות יחסית לנפח את ההסתבורת לקבלת תשובה נכונה ע"י הרצה חוזרת.

- **פתרון עם הסתברות גדולה כרצונינו לבעיית חיפוש במערך:** בהינתן k נריץ את האלגוריתם הקודם $log_2(e)\cdot k$ פעמים. אם קיבלנו לסחות סען אחת אינדקס המקיים את הדרוש ־ נחזיר אותו. אחרת ־ נחזיר $fail$.

טענה: לכל קלט, האלגוריתם טועה לכל היותר בהסתברות $\frac{1}{e^k}$.

הוכחה: במערך יש $\frac{n}{2}$ אפסים. ההסתברות לקבל אינדקס שמחזיק 0 היא $\frac{1}{2}$. מכיוון שההגרלות הן ב"ת, ההסתברות שכל הדגימות יחזירו 0 היא:

$$\left(\frac{1}{2}\right)^{log_2(e)\cdot k} = \frac{1}{2^{log(e)\cdot k}} = \frac{1}{\left(2^{log(E)}\right)^k} = \frac{1}{e^k}$$

זמן ריצה: אנו מריצים $O(k)$ פעמים, אלגוריתם שרץ $O(1)$, לכן זמן הריצה הכולל הוא $O(k)$. נשים לב כי ההסתברות לפתרון לא תלויה ב n.

- **הגדרה ־ אלגוריתם קירוב הסתברותי:** הוא אלגוריתם שלכל קלט, בזמן פולינומיאלי מחזיר פתרון חוקי ו c־מקרב. לפתרון האופטימלי בהסתברות גדולה כרצונינו, ונכשל בהסתברות נמוכה.

- **בעיית** $3-SAT$ **כאלגוריתם קירוב הסתברותי:** אנו מקבלים **כקלט** נוסחת $3-CNF$ בעלת m פסוקיות: $c_1,....,c_m$. מעל n משתנים $x_1,...x_n$ **הפלט:** השמה למשתנים שממקסמת את מספר הפסוקיות שמסתפקות (מקבלות ערך אמת).

  הנחה: המשתנים המשתתפים באותה הפסוקית שונים זה מזה.

  אלגוריתם $\frac{8}{7}$ מקרב: בהינתן k נציג אלגוריתם שיצליח בהסתברות $1-\frac{1}{e^k}$, בנוסף זמן הריצה יהיה פולינומי ב k.

  אלגוריתם בסיסי: לכל משתנה x_i נטיל מטבע הוגן, ונגדיר: $H=true, T=false$. אם ההשמה שהגרלנו מספקת לפחות $\frac{7}{8}\cdot m$ פסוקיות ־ נחזיר אותה, אחרת נחזיר $fail$.

  זמן ריצה: עלינו להגריל n משתנים באופן הסתברותי, ואחרי ההגרלות נלבדוק כמה פסוקיות ענו על הדרוש, סה"כ $O(m+n)=O(m)$ משום שהנחנו $n\leq 3m$.

  אלגוריתם אמיתי: נריץ את האלגוריתם הבסיסי $k(m+1)$ פעמים באופן בלתי תלוי, אם באחת ההרצות קיבלנו את הדרוש ־ נחזיר אותה. אחרת נחזיר $fail$.

  זמן ריצה: $O(k\cdot m^2)$.

  הוכחת נכונות: צריך להראות שהאלגוריתם הבסיסי מצליח בהסתברות גבוהה וגם $\frac{8}{7}$ מקרב.

  קירוב: אם האלגוריתם הצליח הוא מחזיר השמה מספקת של $\frac{7}{8}\cdot m$ פסוקיות. ולכן ענינו על הדרוש.

  הסתברות:

  טענה: סיכויי ההצלחה של האלגוריתם הבסיסי $\leq \frac{1}{m+1}$.

13 שבוע 13:

13.1 הרצאה 23:

- נמשיך עם האלגוריתם למציאת חתך מינימלי:

- **משפט:** תהי f תוצאת האלגוריתם, אזי: $\mathbb{P}(f=f_{min})\geq\prod_{k=3}^{|V(G)|}\left(1-\frac{2}{k}\right)$. הוכחה באנדוקציה על מספר הצמתים.

- **מסקנה:** $\mathbb{P}(f=f_{min})\geq\frac{1}{\binom{n}{2}}$.

- **מסקנה:** בכל גרף G על n קודקודים, מספר חתכי המינימום הוא לכל היותר $\binom{n}{2}$.

- **מסקנה:** הסיכוי שהאלגוריתם יכשל, הוא לכל היותר $1-\frac{1}{n^2}$. אם נריץ את האלגוריתם באופן ב"ת N פעמים, ונקח את החתך הקטן ביותר מבין n החתכים שחושבו, סיכויי הכשלון הם לכל היותר $\left(1-\frac{1}{n^2}\right)^N \leq e^{\frac{N}{n^2}}$, כלומר - אחרי n^2 נסיונות, מצליחים בהסתברות של לפחות $1-e^{-1}$.

- **סיבוכיות האלגוריתם:** $O(m \cdot n^2)$ כאשר m הוא מספר הקשתות ב G.

- **בעיית בדיקת זהות פולינומים מרובי משתנים:** נתון שדה F ופולינום P מעל השדה. נתחיל עם פולינום P במשתנה 1. $P=\sum_{i=0}^{d} a_i \cdot x_i$ כאשר $a_0,....a_d \in F$.
השאלה היא האם P הוא זהותית 0.
אם הפולינום מיוצג באופן המפורש לעיל - התשובה טרויאלית, נבדוק אם כל המקדמים הם 0.
אם הפולינום מיוצג באופן הבא - הצגה לא מפורשת: $P=(a_1x+b_1)...(a_dx+b_d)$ הבעיה קשה יותר אך עדיין טרוויאלית, משום שהפולינום הוא עם משתנה 1.
הבעיה הופכת קשה יותר, כאשר $P=F[x_1,...x_n]$, אך P לא רשום מפורשות כצורה של מונומים, כלומר - לא רשום כ $\sum_{\alpha_1...\alpha_n} a_{\alpha_i...\alpha_n} x_1^{\alpha_1} x_2^{\alpha_2}\cdots x_n^{\alpha_n}$.
נניח כי הפולינום מיוצג בצורה קשה יותר, לדוגמה - מכפלה של פונקציות לינאריות:
$$\prod_i \left(c_1^i x_1 + c_2^i x_2 + \cdots + c_n^i x_a + c_0^i\right)$$
המשך בהרצאה ההבאה.

13.2 הרצאה 24 - בדיקת זהות פולינומים:

- **משפט שוורץ - זיפל:** יהי $P \in F[x_1,...,x_n]$ פולינום מרובה משתנים שאינו זהותית 0, ותהי A קבוצה סופית של אברי F. נגריל הצבה $x_i=a_i$ לכל $i\in[n]$ כאשר ב a_i-ים כולם ב"ת הדדית, וכ"א מתפלג אחיד ב A. אזי: $\mathbb{P}(P(a_1,...a_n)=0) \leq \frac{deg(P)}{|A|}$.
כלומר אם נבחר קבוצה A מספיק גדולה ביחס לדרגת הפולינום, אזי ההסתברות שהפולינום זהותית 0 הינה קטנה מאד.

- **מסקנה מהמשפט:** אם F שדה מספיק גדול, אזי קיים אלגוריתם הסתברותי יעיל עבור הבעיה הבאה: נתון פולנום רב משתנים P ממעלה מירבית d, בייצוג שמאפשר חישוב יעיל של ערכו בהינתן הצבה למשתנים. **הפלט:** זיהוי אם P זהותית 0.
הוכחה: נבחר מ F קבוצה A כך ש $|A| \geq 2d$. נגריל הצבה מקרית מ A ונציב ב P, נחזור על הפעולה M פעמים. אם באחת הפעמים קיבלנו ערך שונה מ 0 נחזיר $false$, אחרת נחזיר $true$.
ניתוח: אם הודענו ש P זהותית 0 - אזי הסיכוי שכל M הנסיונות יחזירו ערך 0 בטעות, שווה ל 2^{-M}.
סיבוכיות: צריך להגריל M פעמים ערכים a_i ולהציג את הערכים האלה ב P. לכן זמן הריצה שווה ל: (סיבוכיות ההגרלה+סיבוכיות ההצבה)$\cdot M$.

- **דוגמה:** נתון גרף דוצ $G=(L,R,E)$ ומתקיים $|R|=|L|$ ואנו רוצים לבדוק האם יש שידוך מושלם ב G.
נגדיר מטריצה ריבועית סימבולית M, השורות ממוספרות באברי L, והעמודות באברי R.

$$M_{i,j}=\begin{cases} x_{i,j} & (i,j)\in E \\ 0 & else \end{cases}$$

הדטרמיננטה $det(M)$ היא פולינום במשתנים $\{x_{i,j}:(i,j)\in E\}$.
אם קיים שידוך מושלם ־ אנו נקבל כי המקדם של הפולינום הוא ± 1, ולכן הפולינום אינו זהותית 0, ואם אין שידוך מושלם ־ אזי הפולינום זהותית 0.

13.3 תרגול 12 ־ אלגוריתמים הסתברותיים:

- **אלגוריתמי קירוב והסתברות:** באלגוריתם מקרב אנו מעדיפים את החוקיות אבל מוותרים על האופטימליות. מנגד באלגוריתם הסתברותיים אנו נעדיף אופטימליות ונוותר על החוקיות.

- **אלגוריתמי קירוב הסתברותיים:** נרשה לאלגוריתם להיכשל בהסתברות קטנה כרצוננו, וכשהוא לא נכשל, אנו נסתפק שהתוצאה שתחזור תהיה בקירוב c.

- **דוגמה ־ מערכת משוואות מודולו 2:** אנו מקבלים כקלט מטריצה $A\in F_2^{m\cdot n}$ (יש בשדה רק שני איברים), בנוסף אנו מקבלים וקטור $b\in F^m$. הפלט תהיה השמה ל $x_1,...x_n$ כך שכמה שיותר משוואות מהצורה $Ax=b$ יסופקו.
כלומר ־ נצטרך לספק m משוואות ב n נעלמים. בנוסף, נניח כי אין שורה שכולה אפסים ב A.
אלגוריתם 2־מקרב הסתברותי לפתרון הבעיה:
אסטרטגיה: אנו נמצא אלגוריתם שמצליח לפתור את הבעיה בהסתברות נמוכה, ונריץ אותו מספיק פעמים עד שההסתברות תגדל כרצונינו.
אלגוריתם בסיסי: נגריל לכל משתנה x_i מספר 1 או 0 כך שההסתברות היא $\frac{1}{2}$. אח"כ נבדוק אם מה שהגרלנו מספק לפחות חצי מהמשוואות, אם כן ־ נחזיר את זה. אחרת ־ נחזיר כשלון. **זמן הריצה:** ההגרלות יהיו $O(n)$, הבדיקה שההגרלה מספקת תקח לנו $O(m\cdot n)$.
נוכיח כי האלגוריתם מצליח בהסתברות גדולה מ 0: הטענה ־ השמה מקרית בתוחלת תספק חצי מהמשוואות.
הוכחה: נסמן X מ"מ הסופר כמה משוואות ההשמה מספקת. בנוסף נסמן $x_i,...x_m$ מ"מ האם ההשמה סיפקה את המשוואה ה i.

$$P(x_i=1)=P\left(\sum_{j=1}^{n}A_{ij}x_j=b_i\right)=P\left(\sum_{j=1}^{d}A_{ij}x_j=b_i\right)$$

כאשר d הוא האינדקס המקסימלי עבורו $A_i\cdot d=1$.כעת נשתמש בנוסחת ההסתברות השלמה:

$$P\left(\sum_{j=1}^{d-1}A_{ij}x_j+A_i\cdot d\cdot x_d=b_i\right)=P(x_d=1)\cdot P\left(\sum_{j=1}^{d-1}A_{ij}x_j+x_d=b_i|x_d=1\right)+P(x_d=0)\cdot P\left(\sum_{j=1}^{d-1}A_{ij}x_j+x_d=b_i|x$$

$$=\frac{1}{2}\left(P\left(\sum_{j=1}^{d-1}A_{ij}x_j=b_i-1\right)+P\left(\sum_{j=1}^{d-1}A_{ij}x_j=b_i\right)\right)=\frac{1}{2}\cdot 1=\frac{1}{2}$$

נשים לב כי שני חלקי המשוואה משלימים ולכן שווים ל 1.

$$E(x_i)=\frac{1}{2}P(x_i=1)+\frac{1}{2}P(x_i=0)=\frac{1}{4}+\frac{1}{4}=\frac{1}{2}$$

כעת נחשב את התוחלת של X:

$$E(X)=E\left(\sum_{i=1}^{m}x_i\right)=\sum_{i=1}^{m}E\left(x_i\right)=\sum_{i=1}^{m}\frac{1}{2}=\frac{m}{2}$$

נגדיר מ"מ Y להיות מספר השמוואות שההשמה לא מספקת, כלומר $Y=m-X$ ומתקיים $E(Y)=m-E(X)=\frac{m}{2}$.
נחשב את ההסתברות לכישלון האלגוריתם, ונשתמש באי שוויון מרקוב בכדי לחסום את ההסתברות:

$$P\left(algorithem\ fail\right)=P\left(Y>\frac{m}{2}\right)=P(2Y\geq M+1)=P\left(Y\geq\frac{m}{2}+\frac{1}{2}\right)\leq$$

$$\leq\frac{E(Y)}{\frac{m}{2}+\frac{1}{2}}=\frac{\frac{1}{2}m}{\frac{m}{2}+\frac{1}{2}}=\frac{m}{m+1}\Rightarrow P(algorithem\ success)\geq 1-\frac{m}{m+1}=\frac{1}{m+1}$$

כעת, הוכחנו שהאלגוריתם לא נכשל בהסתברות חיובית. ננפח את ההסתברות כדי להשיג את הקירוב הרצוי.
אלגוריתם כללי: בהינת k נריץ את האלגוריתם הבסיסי $k(m+1)$ פעמים, אם נכשלנו בהכל נחזיר $fail$. אחרת - נחזיר את המשוואות המסופקות.
טענה: האלגוריתם הכללי מצליח בהסתברות $\leq 1-\frac{1}{e^k}$.

$$P(general\ algorithem\ success)=1-P(general\ algorithem\ fail)=1-(P(basic\ algorithem\ fail))^{k(m+1)}$$

$$\geq 1-\left(\frac{m}{m+1}\right)^{k(m+1)}==1-\left(\left(1-\frac{1}{m+1}\right)^{m+1}\right)^{k}>1-\frac{1}{e^k}.$$

זמן ריצה: $O(m^2nk)$.

- **תבנית לאלגוריתם הסתברותי מקרב:**

1: נמצא אלגוריתם בסיסי שיפתור לנו את הבעיה בהסתברות גדולה מ 0.

2: נוכיח שהאלגוריתם פותר את הבעיה בהסתברות חיובית ע"י שימוש במ"מ ובתוחלת.

3: אחרי שנחשב את התוחלת, נשתמש במ"מ משלים אם נצטרך ונחסום את הסיכוי לכשלון ע"י א"ש מרקוב.

4: נשים לב כי א"ש מרקוב עובד עם $\leq$, לכן אם קיבלנו א"ש חזק - נעבור למספרים שלמים ונוסיף 1 לאגף ימין. אח"כ נמשיך כרגיל.

5: נפעיל את האלגוריתם מספר רב של פעמים עד שנגיע קירוב הדרוש.

6: נוכיח כי האלגוריתם בכללי אכן משיג את הקירוב הדרוש.

14 שבוע 14

14.1 הרצאה 25 ־ זיהוי תבניות:

- **זיהוי תבניות:** בעיית זיהוי התבניות, נתון טקסט T־ רצף של אותיות מתוך א"ב $\sum$, ומתקיים $|\sum| = d, |T| = n$,
נתונה תבנית P שהיא רצף של אותיות מתוך $\sum$, ומתקיים $|P| = m$ ונניח ש $m << n$.
אנו רוצים למצוא את כל המופעים של P בתוך T.

- **האלגוריתם הנאיבי לפתרון הבעיה:** נעבור על כל המקומות ב T שבהם P יכולה להתחיל ־ $i = 1, ..., n-m+1$,
במקון ה i נשווה בין P לבין $T[i, ..., i+m-1]$.
סיבוכיות זמן: $O(n \cdot m)$ פעולות השואה, וגישה לאיברים של P, T על פי אינדקס וקידום של אינדקס.
סיבוכיות מקום: מלבד הקלט והפלט, צריך $O(1)$ תאי זיכרון שמחזיקים כל אחד אינדקס.

- **טביעת אצבע ־ אלגוריתם הסתברותי לזיהוי תבניות:** אפשר להתייחס ל P ול $T[i, ..., i+m-1]$ כאל מספר שרשום בבסיס d: $p = d^{m-1} \cdot P[1] + ... + d^0 \cdot P[m]$ וגם $t_i = d^{m-1} \cdot T[1] + ... + d^0 \cdot T[i+m-1]$. ואז מתקיים :

$$T[i, ..., i+m-1] = P \iff t_i = p$$

נשים לב כי : $t_{i+1} = dt_i - d^m T[i] + T[i+m]$. כלומר ־ העדכון נעשה בקלות.
בינתיים לא שיפרנו שום דבר, משום שאנו עובדים על מספרים גדולים, מספרים אלו עלולים לדרוש $m \cdot log_2 d$ ביטים ע"מ לייצג אותם. סה"כ נצטרך $O(n \cdot m \cdot log_2 d)$ פעולות על ביטים כדי לבצע את החיפוש.
האלגוריתם: נבחר קבוצה גדולה Q של מספרים ראשוניים, נבחר $q \in Q$ באופן מקרי בהתפלגות אחידה, ונבצע את כל החישובים $mod(q)$. ואז ־ אם $log_2(q_{max})$ קטן יחסית לעומת m, החישובים האריתמטים יעילים.
כמובן, אם $t_i \neq pmod(q)$ אז $T[i, ..., i+m-1] \neq P$. אך אם $t_i \equiv pmod(q)$ לא מובטח לנו שהתבנית P תואמת לטקסט $T[i, ..., i+m-1]$, אז צריך לבדוק. בפרט אם P מופיעה ב T s פעמים נשלם לכל הפחות $O(sm)$ פעולות.
עבור מקומות שבהן התבנית שונה מהטקסט ־ אם $t_i \neq p$ נקבל ש $t_i \equiv pmod(q)$ רק אם q הוא גורם ראשוני של $|t_i - p|$, יש לנו לכל היותר $m \cdot log_2(d)$ גורמים ראשוניים שונים כאלו. לכן:

$$\mathbb{P}\ \left[t_i \equiv p \bmod q \mid t_i \neq p\right] < \frac{m \log_2 d}{|Q|}$$

כלומר ־ תוחלת מספר המקומות i שנבדוק סתם תהיה קטנה מ $\frac{n \cdot m \cdot \log_2 d}{|Q|}$, כלומר תוחלת מספר הפעולות שנבצע היא: $O\left(m + n + m\left(s + \frac{n \cdot m \cdot \log_2 d}{|Q|}\right)\right)$.
נרצה $|Q| >> n \cdot m \cdot \log_2 d$, כדי לבחור את Q אפשר להשתמש בצפיפות של המספרים הראשוניים:
משפט המספרים הראשוניים: מספר המספרים הראשוניים שקטנים מ x הוא: $\frac{x}{\ln x}(1 \pm o(1))$.
לכן נקח את כל הראשוניים עד $n \cdot m \cdot \log_2 d$, ונבצע הגרלה עד שנגיע למספר ראשוני.

14.2 הרצאה 26 ־ שיעור חזרה:

- ׳

14.3 תרגול 13 - חזרה על תכנון לינארי ומשקול כפלי:

- מבחן: 4 שאלות, אין בחירה. שלש שעות.

- **תכנון לינארי:** זאת דרך להציג בעיות, בדרך זו אנו יכולים לפתור את הבעיה בזמן פולינומיאלי.

הצורה הסטנדרטית - נקח מינימום על פונקציה לינארית, של כל הווקטורים ב R^m, בנוסף תהיה לנו מטריצת אילוצים - A.

אחרי הצגת הבעיה בשיטה זו, נוכל לפתור את הבעיה עם אלגוריתמים מתאימים. לדוגמה - אלגוריתם סימפלקס שרץ בזמן אקספוננציאלי ויעיל לרוב הקלטים.

לכן - נמצא אלגוריתם יעיל שמשיג את המינימום בפקטור מסויים שנבחר להיות קטן כרצונינו. כלומר נמצא אלגוריתם מקרב, שיפתור לנו את הבעיה בעזרת מישקול כפלי.

כיצד נפתור:

1: נבדוק אם הווקטור b הוא ווקטור של אחדות (1), נחסום את הפתרון האופטימלי בתוך קטע כלשהו, אח"כ נבצע חיפוש בינארי בתוך הקטע הזה בכדי למצוא x שעונה על הדרוש.

2: נסתכל על בעיית הכרעה - נצטרך להגדיר יחס סדר, בכדי לדעת באיזה חלק של הקטע לחפש. עבור כל מיקום בקטע החסום נצטרך לענות האם הוא פתרון לבעיה, גדול ממנה או קטן ממנה, וכך נזוז בהתאם ימינה או שאלה בחיפוש הבינארי. כלומר - נגדיר אילוץ נוסף, עבור β נבדוק האם $c^T x \leq \beta$, זהו למעשה האילוץ שמגדיר את יחס הסדר.

3: נוסיף את האילוץ החדש למטריצת האילוצים באופן הבא - נמצא בעיה שקולה לבעיה הקודמת. כלומר - נתאים את אי השוויונים אחד לשני, ונוסיף את האילוץ החדש למטריצה הקיימת בשורה חדשה, בנוסף נוסיף את האילוץ β לווקטור b.

4: כעת נפתור את בעיית אי השוויונים. הקלט שלנו הוא מטריצת האילוצים A', ווקטור b, אנו נחפש קלט של x המקיים את המשוואות, אם אין כזה נחזיר שאין פתרון.

5: נפתור את בעיית אי השוויונים המקורבת. בנוסף לקלט ל שלב 4, נקבל גם סקלאר $\delta > 0$ שיהיה פקטור הקירוב שלנו. הפלט יהיה - האם קיים x המקיים את הדרוש באופן הבא - $A'x \geq b' - \delta$, כלומר אנו מאפשרים טווח טעות של δ.

6: נחלק באגף ימין של המשוואה, ונקבל למעשה ווקטור x חדש שאותו אנו צריכים למצוא. כעת עם הקירוב הנוכחי נוכל להריץ את החיפוש על הבעיה הדשה ולהשתמש בה לבעיה המקורית.

7: בכדי לפתור את הבעיה החדשה נשתמש במשקלות כפליים. נרצה לפתור את הבעיה הבאה - בעיית א"ש עם התפלגות: אנו מקבלים כקלט את המטריצה A', את הווקטור b' ובנוסף נקבל ווקטור הסתברות עם אילוצים - p, שנותן לנו התפלגות בין 0 ל1 על האילוצים (הווקטור p אמור לקיים את כל האילוצים של הסתברות). כפלט אנו נחזיר ווקטור x המקיים את כל האילוצים שדרשנו: $\sum_{i=1}^{m+1} p_i A'_i x \geqslant \sum_{i=1}^{m+1} p_i b'$

8: כעת, אם אין x שמקיים את הבעיה החדשה - אזי אין x שפותר את הבעיה המקורית. בכדי לפתור את הבעיה הנוכחית נשתמש במשקלים כפליים.

9: נמצא אלגוריתם שפותר את בעיית א"ש עם התפלגות. ומחזיר לנו x המקיים את הדרוש אם קיים כזה.

נחזיר את הבעיה לצורתה המקורית - מטריצה ואילוץ נפרד על β. לאחר מכן נחלק בגורם המתאים, נוציא את ה

x־ים מהסכום ונבודד את האקסים בצד השמאלי של המשוואה. לאחר מכן נציב את ה x המקסימלי ונראה אם הוא עונה על הדרוש ומקיים את המשוואה.

- **משפט:** בהינתן בעיה הנתונה בצורה של תכנון לינארי. כאשר הווקטור b הוא ווקטור אחדות (1) והפתרון האופטימלי חסום בקטע. אם קיים אלגוריתם הפותר את בעיית אי השוויונות עם התפלגות, אזי ניתן להשתמש במשקול כפלי בכדי לפתור את הבעיה המקורית (ללא התפלגות).
זמן הריצה: $O\left(\frac{p^2}{\delta^2}log(m)\right)$.

14.4 הרצאה 27 - שיעור חזרה: