אלגוריתמים 1 ־ סיכום הרצאות למבחן

4 בדצמבר 2025

הסיכום נכתב תוך כדי הרצאות סמס א' תשפ"ו (2026) ולכן ייתכן שנפלו טעויות במהלך כתיבת הסיכום, ככה שעל אחריותכם.
גיא יער־און.

תוכן עניינים

1 אלגוריתם קרטסובה

הבעיה: נתונות שני מחרוזות בעלות n ביטים כל אחת. נרצה לכפול את המחרוזות. מה סיבוכיות הפעולה?

פתרון: מכפילים ביטים. יש n^2 מכפלות כאלו. נחשוב על הפתרון באמצעות רקורסיה. כלומר, במקום להכפיל מחרוזת באורך n נרצה לצמצם את הכפל למשהו כמו $\frac{n}{2}$. בהינתן מס' x נרצה לרשום אותו אחרת. נחלקו לשני חלקים ונקבל כי $X = x_1 * 2^{\frac{n}{2}} + x_2$. כאשר נכפיל ב- $2^{\frac{n}{2}}$ זה לא באמת עולה לי כי אנחנו רק מזיזים את הביטים. (כלומר - בהינתן 1234. נוכל לרשום כי $1234 = 12 * 10^2 + 34$ - כלומר הכפל אכן לא באמת עולה)

כעת יש מס' נוסף - $Y = y_1 * 2^{\frac{n}{2}} + y_2$.

נראה כי אם נכפול נקבל -

$$xy = (x_1 * 2^{\frac{n}{2}} + x_2)(y_1 * 2^{\frac{n}{2}} + y_2) = x_1y_12^n + x_1y_22^{\frac{n}{2}} + x_2y_12^{\frac{n}{2}} + x_2y_2$$

מה קיבלנו כאן? נראה כי כל הפעולות של 2 בחזקת הן פעולות הזזת ביטים ולמעשה כאשר נכפיל שני מחרוזות צמצמנו את הבעיה שלנו ל־4 תתי בעיות בגודל $\frac{n}{2}$!
מכאן נקבל כי נוסחת הנסיגה היא ־

$$T(n) = 4T(\frac{n}{2}) + O(n) = \Theta(n^2)$$

זה לא טוב ־ מדוע? זה בדיוק כמו הפתרון הנאיבי. בואו ננסה לצמצם את מס' הקריאות הרקורסיביות:
נגדיר ־

$$A = x_1y_1, B = x_2y_2$$

נזכיר שנרצה לחשב את

$$C = (x_1 + x_2)(y_1 + y_2) = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2$$

כעת נראה כי

$$xy = x_1y_12^n+x_1y_22^{\frac{n}{2}}+x_2y_12^{\frac{n}{2}}+x_2y_2 = x_1y_12^n+2^{\frac{n}{2}}(x_1y_2+x_2y_1)+x_2y_2 = x_1y_12^n+2^{\frac{n}{2}}(C-B-A)+x_2y_2$$

מכאן שקיבלנו נוסחה עם 3 איברים כעת (יש איזשהו קבוע באיבר האמצעי)! ולא 4 כמו קודם, נוכל לרשום ־

$$T(n) = 3T(\frac{n}{2}) + \Theta(n)$$

לפתור לפי מאסטר ולקבל כי $T(n) = \Theta(n^{log3}) \approx \Theta(n^{1.58})$.

2 הרצאה 1 ־ FFT

2.1 פעולות של פולינומים

פולינום ניתן לכתיבה כך ־ $P(x) = a_0 + a_1x + a_2x^2 + .. + a_{n-1}x^{n-1} = \sum_{i=0}^{n-1} a_ix^i$.
פולינום זה הוא מדרגה $n-1$, והוא מדרגה חסומה k לכל $k \geq n$. (כלומר $P(x)$ חסום n, $n+1$, $n+2$, $2n+17$ וכו').

1. **חיבור/חיסור פולינומים:** יהיו $A(x) = \sum_{i=0}^{n-1} a_ix^i, B(x) = \sum_{i=0}^{n-1} b_ix^i$ פולינומים. נגדיר את החיבור/חיסור של $A(x)$ ו־$B(x)$ כך ־

$$C(x) = A(x) \pm B(x) = \sum_{i=0}^{n-1}(a_i \pm b_i)x^i$$

נשים לב כי דרגת הפולינום $C(x)$ נשארה זהה לדרגה של $A(x), B(x)$.

2. **כפל פולינומים:** יהיו $A(x) = \sum_{i=0}^{n-1} a_i x^i, B(x) = \sum_{i=0}^{n-1} b_i x^i$ פולינומים. נגדיר את הכפל של $A(x)$ ו$B(x)$ כך -

$$C(x) = A(x) \times B(x) = \sum_{i=0}^{2n-2} c_i x^i$$

באשר $c_i = \sum_{j=0}^{i} a_j b_{i-j}$. לפעולה זו קוראים **קונבולוציה.**
נשים לב כי דרגת הפולינום C תהיה $2(n-1) = 2n-2$.

3. **חישוב ערך:** יהי פולינום $A(x) = \sum_{i=0}^{n-1} a_i x^i$, בהינתן ערך x_0, נרצה לחשב את $A(x_0)$.

2.2 ייצוג של פולינומים

יהי פולינום $A(x) = \sum_{i=0}^{n-1} a_i x^i$. נרצה להראות מספר דרכים לייצג את הפולינום:

2.2.1 ייצוג ע"י מקדמים

נרצה לשמור את המקדמים בלבד של הפולינום. נשתמש במערך ARR בגודל n, ונשמור בתוכו את המקדמים:

$$ARR = (a_0, a_1, ..., a_n)$$

ייצוג זה באמצעות מקדמים נחשב לטוב, קיימת פונקציה חח"ע ועל בין עולם ה"ייצוגים" ל"עולם הפולינומים" - מה הכוונה? לא ייתכן שנקבל ייצוג זהה עבור $A_1(x) \neq A_2(x)$ ולא ייתכן ייצוג שונה עבור $A_1(x) = A_2(x)$.
חיבור/חיסור: יהיו פולינומים $A(x), B(x)$ חסומים מדרגה $n-1$, אזי נרצה לחשב את $C(x) = A(x) \pm B(x)$.

$$\forall_{0 \leq i \leq n-1} : c_i = a_i \pm b_i$$

נשים לב כי בהינתן שיטת המקדמים, לחשב פולינום $C(x)$ הנ"ל יעלה $O(n)$ ע"י חיבור/חיסור זוג הערכים בתאים $A[i], B[i]$ לתוך אחד המערכים. למעשה גם לא נצרך שימוש במקום נוסף. **סה"כ - חיבור/חיסור** $O(n)$.
כפל: לפי הנוסחה שתוארה לעיל:

$$\forall_{0 \leq i \leq n-1} c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

נראה כי זמן לחישוב מקדם c_i בודד יעלה $O(n)$ זמן, ולכן חישוב כלל המקדמים, כלומר **חישוב הכפל יעלה $O(n^2)$ זמן**.

חישוב ערך: בהינתן x_0 נרצה לחשב את $A(x_0)$. לא יהיה כאן רעיון מתוחכם - נחשב את $x_0^0, x_0^1, ..., x_0^{n-1}$ ולאחר מכן נחשב את החיבור $a_0x_0^0 + a_1x_0^1 + ...a_{n-1}x_0^{n-1}$ וסה"כ **חישוב ערך יעלה $O(n)$**.

2.2.2 ייצוג ע"י נקודות

בהינתן ישר מקביל לאחד הצירים, באמצעות נקודה אחת ניתן לדעת לתאר את הישר. בהינתן שידוע כי הישר הוא קו לינארי ישר $y = mx + b$ אזי בהינתן שתי נקודות ניתן לדעת באופן מדויק את משוואת הישר, בהינתן פרבולה $y = ax^2 + bx + c$ בהינתן שלוש נקודות ניתן לדעת באופן מדויק את משוואת הישר וכן הלאה. באופן כללי, יהא פולינום $A(x) = \sum_{i=0}^{n-1} a_i x^i$ אזי באמצעות n נקודות $(x_0, x_1, ..., x_{n-1})$ ניתן לדעת באופן מדויק ולתאר את הפולינום, **בתנאי שהנקודות שונות אחת מהשנייה.**

סה"כ נייצג את הפולינום A באמצעות הנקודות:

$$(x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1})$$

באשר $\forall_{0 \leq i \leq n-1} : y_i = A(x_i)$.

האם הייצוג הזה טוב? האמת שכן, כיצד נראה זאת? צריך להראות שהייצוג הוא חח"ע ועל. כלומר, שקיימת פונקציה חח"ע ועל בין הייצוג לפולינומים.

נשים לב כי $y_i = A(x_i) = a_0 + a_1x_i + a_2x_i^2 + a_3x_i^3 + ... + a_{n-1}x_i^{n-1}$ לכל $0 \leq i \leq n-1$.
נראה כי ניתן לקבל בסה"כ n משוואות, ולפתור מערכת של n-משוואות, ולמצוא כך את כל ערכי $a_0, a_1, a_2, ..., a_n$. ולכן סה"כ הייצוג הוא חח"ע ועל. באופן פורמלי יותר - נשים לב כי מערכת המשוואות הנ"ל ניתנת לתיאור בצורה מטריציונית:

$$\begin{pmatrix} y_0 \\ y_1 \\ . \\ . \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} x_0^0 & x_0^1 & x_0^2 & .. & x_0^{n-1} \\ x_1^0 & x_1^1 & x_1^2 & .. & x_1^{n-1} \\ x_2^0 & x_2^1 & x_2^2 & .. & x_2^{n-1} \\ .. & .. & .. & .. & .. \\ x_{n-1}^0 & x_{n-1}^1 & x_{n-1}^2 & .. & x_{n-1}^{n-1} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ a_{n-1} \end{pmatrix}$$

מטריצה זו נקראת **מטריצת ונדרמונדה.** נקרא למטריצה V, לוקטור הימני נקרא $\vec{a}$ ולוקטור השמאלי נקרא $\vec{b}$.

טענה: אם במטריצת ונדרמונדה ערכי $(x_0, ..., x_{n-1})$ כולם שונים זה מזה, אזי מטריצת ונדרמונדה הפיכה.

כיוון שV הפיכה אצלנו, נשים לב כי $\vec{a} = V^{-1}\vec{b}$. סה"כ מצאנו דרך לעבור בין ערכי הנקודות ולקבל את המקדמים של הפולינום, ולכן ייצוג זה הוא חח"ע ועל.

כיצד עוברים בין ייצוג ע"י מקדמים לייצוג ע"י נקודות? ע"י n פעולות של חישוב ערך. נבחר $x_0 \neq x_1 \neq ... \neq x_{n-1}$ ונחשב $A(x_0), ..., A(x_{n-1})$. כמה יעלה מעבר זה? כל חישוב עולה $O(n)$ ולכן סה"כ n חישובים יעלו לנו $O(n^2)$.
כיצד עוברים בין ייצוג ע"י נקודות לייצוג ע"י מקדמים? לפעולה זו יש שם - אינטרפולציה. משתמשים בנוסחת לגראנז':

$$A(x) = \sum_{i=0}^{n-1} y_i \cdot \frac{\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i}(x_i-x_j)}$$

נשים לב כי חישוב זה יעלה $O(n^2)$ זמן, ולכן גם המעבר השני עולה כמו המעבר הראשון.

1. **חיבור:** יהיו שני פולינומים:
   א. נניח כי שני הפולינומים מיוצגים ע"י אותם נקודות $(x_0, x_1, ..., x_{n-1})$

$$A = (x_0, A(x_0)), (x_1, A(x_1)), ..., (x_{n-1}, A(x_{n-1}))$$

$$B = (x_0, B(x_0)), (x_1, B(x_1)), ..., (x_{n-1}, B(x_{n-1}))$$

נשים לב כי במקרה זה נייצג את פולינום החיבור:

$$C = (x_0, C(x_0)), (x_1, C(x_1)), ..., (x_{n-1}, C(x_{n-1}))$$

באשר $\forall_{0\leq i\leq n-1} C(x_i) = A(x_i) + B(x_i)$. נשים לב כי בדרך זו, חיבור פולינומים עולה $O(n)$ זמן.

ב. שני הפולינומים לא בהכרח מיוצגים ע"י אותם נקודות.

אין דרך קסם". מה שנעשה יהיה לבצע אינטרפולציה, באמצעות נוסחת לגראנז'. נעבור לייצוג ע"י מקדמים של שני הפולינומים, זה יעלה עבור כל פולינום $O(n^2)$. אח"כ נחבר את שני הפולינומים בשיטת המקדמים, מה שיעלה עוד $O(n)$, ואח"כ נמיר חזרה את פולינום החיבור משיטת המקדמים חזרה לשיטת הנקודות, מה שיעלה עוד $O(n^2)$. סה"כ - $O(n^2)$ לחיבור פולינומים.

2. **כפל:** יהיו שני פולינומים.
   א. נניח כי שני הפולינומים מיוצגים ע"י אותם נקודות $(x_0, x_1, ..., x_{2n-1})$

$$A = (x_0, A(x_0)), (x_1, A(x_1)), ..., (x_{n-1}, A(x_{2n-1}))$$

$$B = (x_0, B(x_0)), (x_1, B(x_1)), ..., (x_{n-1}, B(x_{2n-1}))$$

נשים לב כי במקרה זה נייצג את פולינום הכפל:

$$C = (x_0, C(x_0)), (x_1, C(x_1)), ..., (x_{n-1}, C(x_{2n-1}))$$

באשר $\forall_{0\leq i\leq 2n-1} C(x_i) = A(x_i) \times B(x_i)$. נשים לב כי בדרך זו, **כפל פולינומים עולה $O(n)$ זמן.**

נשים לב כי - הדרגה של פולינום הכפל C היא $2n-2$. כלומר צריך לייצג אותו באמצעות $2n-2$ ערכים. לכן בניגוד למקרים אחרים, כאן דרשנו מA וB להיות מיוצגים ע"י $2n-2$ נקודות. אחרת, לא נוכל לכפול.

ב. שני הפולינומים לא בהכרח מיוצגים ע"י אותם נקודות.
באופן דומה, אין פתרון קסם. נבצע אינטרפולציה. נעבור לשיטת המקדמים, שם נכפול ב$O(n^2)$. אח"כ נשתמש חזרה באינטרפולציה לעבור חזרה לשיטת הנקודות. סה"כ $O(n^2)$ לכפל פולינומים במקרה זה.

3. **חישוב ערך:**
בכל מקרה, צריך לבצע אינטרפולציה ואז לחשב ולכן $O(n^2)$.
הערה: אם אנחנו במקרה בו ערכי הx של שני הפולינומים זהים, והערך שנקראנו לחשב הוא כבר אחד מהערכים שאיתם קיבלנו את הפולינום, כל שיש לעשות בשביל לחשב את הערך הוא לחפש את ערך האיקס הספציפי. מה שיעלה לנו $O(logn)$ בהנחה שהנקודות מסודרות בסדר עולה ביחס לערך האיקס.

2.2.3 סיכום הפעולות בשיטות השונות

סוג השיטה / פעולות	מקדמים	שיטת הנקודות - אותם ערכי x	שיטת הנקודות – לא בהכרח אותם ערכי x
חיבור/חיסור	O(n)	O(n)	O(n^2)
כפל	O(n^2)	O(n): צריך שיהיו 2n-2 ערכים שונים לכל פולינום בשביל שנוכל לכפול.	O(n^2)
חישוב ערך	O(n)	O(n^2)	O(n^2)

2.3 אלגוריתם לכפל פולינומים מהיר: התמרת פורייה FFT

יהיו A,B פולינומים המיוצגים ע"י מקדמים. נרצה לקבל את $C=A\times B$. ראינו שאפשר ב$O(n^2)$. בקורס מבני נתונים", ראינו דרך מעניינת לכפול מטריצות)דומה לפולינומים(באמצעות הפרד ומשול ב$O(n^{log_2 3})$. כעת נרצה למצוא שיטה ב$O(nlogn)$.

מה ראינו עד כה? תקבל מקדמים. תבצע $2n-2$ חישובי ערך, ותעבור לשיטת הנקודות. שם תחשב את הכפל ב$O(n)$ זמן, אח"כ תבצע אינטרפולציה חזרה שתעלה $O(n^2)$ וסיימת. סה"כ עלה לך $O(n^2)$. כעת נראה שיטה, שתאפשר את המעבר הראשון והאחרון ב$O(nlogn)$ זמן, מה שיהפוך את האלגוריתם לזמן $O(nlogn)$. כיצד? נרצה לבחור ערכי x-ים ספציפיים מאוד.

המטרה: נרצה לחשב את $A(x)$ בn ערכים שונים - $x_0,....,x_n$. מדוע לא $2n-2$? קל להסביר n, ואח"כ קל להכליל את הרעיון וברור שאסימפטוטית זו אותה סיבוכיות.
הנחה: n הוא חזקה של 2. ניתן להתגבר על הנחה זו, באופן שלא יפגע בסיבוכיות, אך בשביל הפשטות המתמטית נניח הנחה זו. **מדוע ניתן להניח זאת? אפשר להוסיף מקדמים של אפס ואז תמיד נקבל חזקה של** 2.

נגדיר את הפולינומים הבאים:

$$A_{even}(x) = a_0 + a_2x + a_4x^2 + ... + a_{n-2}^{\frac{n}{2}-1}$$

$$A_{odd}(x) = a_1 + a_3x + a_5x^2 + ... + a_{n-1}^{\frac{n}{2}-1}$$

טענה:

$$A_{even}(x^2) + xA_{odd}(x^2) = A(x)$$

$$A_{even}(x^2) - xA_{odd}(x^2) = A(-x)$$

נשים לב כי A_{even}, A_{odd} הם מדרגה חסומה $\frac{n}{2}$, שזה חצי הגדרה החסומה של $A(x)$.

נסיון ראשון: נחשב את A_{odd} בn ערכי x שונים: $x_0^2, ..., x_n^2$. נחשב את A_{even} בn ערכי x שונים: $x_0^2, ..., x_n^2$. ואז נשתמש בנוסחה לעיל כאן בטענה, $A_{even}(x^2) + xA_{odd}(x^2) = A(x)$. נשים לב כי החישוב בסוף עולה $O(n)$ שהרי מחשבים n ערכים, ובכל קריאה אנחנו קוראים ל"2 תתי בעיות" ולכאורה רקורסיבית אפשר לקבל את הנוסחה הבאה $T(n) = 2T(\frac{n}{2}) + n$ ולפי משפט האב, $T(n) = O(nlogn)$.

זה לא עובד. למה?

1. הובטח כי $x_0, ..., x_{n-1}$ שונים. אך מי אמר ש$x_0^2, ..., x_{n-1}^2$ שונים? יתכן כי $x_2 = -10, x_3 = 10$ אך $x_2^2 = x_3^2 = 100$ ואינם שונים.
2. בשימוש בהפרד ומשול, מובטח לך כי סוג תת הבעיה שתקבל יהיה זהה לבעיה המקורית רק על קלט קטן יותר. בבעיה המקורית, נדרשנו לחשב n ערכים של A, באופן שיעלה $O(n)$. נשים לב כי הפולינום חסום מדרגה n בהתחלה, ואז מחשבים בהתאם n ערכים. אח"כ לאחר שמסתכלים על $\frac{n}{2}$, הפולינום חסום מדרגה $\frac{n}{2}$ אבל גם כאן אנו נדרשים לחשב n ערכים שונים. וכן הלאה - **זו לא תת בעיה**.

2.3.1 תכונת הנגדיות החלשה

יהי k חזקה של 2. לסדרת ערכי האיקס: $(x_0, x_1, ..., x_{k-1})$ יש את תכונת הנגדיות החלשה אם אחד מהתנאים הבאים מתקיימים:

א. $k = 1$

ב.

$$\forall_{0 \leq j \leq \frac{k}{2}-1} : x_{\frac{k}{2}+j} = -x_j$$

דוגמה. הסדרה $5, -1, 3, 2, -5, 1, -3, -2$ היא בעלת תכונת הנגדיות החלשה, נשים לב שהחצי השני הוא הנגדי של החצי הראשון.

נשים לב - כאשר נעלה את איברי הסדרה בריבוע, נקבל כי החצי הראשון של הסדרה שווה לחצי השני.

נסיון שני: נניח כי מטרת העל החדשה שלנו, היא לחשב את A בn ערכי x שמקיימים את תכונת הנגדיות החלשה. מכאן ש: $\forall_{0\leq j\leq \frac{n}{2}-1} : (x_{\frac{n}{2}+j})^2 = (-x_j)^2 = (x_j)^2$. לכן, הסדרה הגדולה שלנו היא מורכבת משתי סדרות זהות שבאות אחת אחרי השניה. לכן אם נחשב את A על החצי הראשון, אין צורך לחשב על החצי השני כי קיבלנו זאת בחינם". האלגוריתם החדש:

א. נחשב את A_{odd} ב$\frac{n}{2}$ ערכי x: $(x_0^2, x_1^2, ..., x_{\frac{n}{2}-1}^2)$

ב. נחשב את A_{even} ב$\frac{n}{2}$ ערכי x: $(x_0^2, x_1^2, ..., x_{\frac{n}{2}-1}^2)$

ג. לכל $0 \leq i \leq \frac{n}{2}-1$ יתקיים: $A(x_i) = A_{even}(x_i^2) + x_i A_{odd}(x_i^2)$

ד. לכל $0 \leq i \leq \frac{n}{2}-1$)החצי השני של הערכים(נשים לב כי לפי הנגדיות החלשה ומעבר שראינו לעיל מתקיים: $A(x_{\frac{n}{2}+i}) = A(-x_i) = A_{even}(x_i^2) - x_i A_{odd}(x_i^2)$, וסה"כ באמצעות א+ב חישבנו גם את החצי הזה של הערכים בלי לבצע פעולות נוספות.

סה"כ חישבנו את n הערכים הפעם, לכאורה ללא הבעיה שהם יהפכו לשונים, לכאורה באופן רקורסיבי ניתן שוב לטעון $T(n) = 2T(\frac{n}{2}) + n = O(nlogn)$. זה שוב לא עובד -

זה שוב לא תת בעיה! **המטרה שלנו הייתה לחשב סדרת ערכים שמקיימים את תכונת הנגדיות החלשה.** לאחר העלאה בריבוע, הם לא מקיימים את תכונת הנגדיות החלשה. ואי אפשר לומר שזו תת בעיה.

2.3.2 תכונת הנגדיות החזקה

יהי k חזקה של 2. לסדרת ערכי האיקס: $(x_0, x_1, ..., x_{k-1})$ יש את תכונת הנגדיות החזקה אם אחד מהתנאים הבאים מתקיים:

א. $k = 1$

ב. לסדרה יש את תכונת הנגדיות החלשה, וגם לסדרה $(x_0^2, x_1^2, ..., x_{\frac{n}{2}-1}^2)$ יש את תכונת הנגדיות החזקה (באופן רקורסיבי).

כעת נשים לב - כי הבעיה שהייתה לנו מקודם נפתרה לגמרי. להלן האלגוריתם:

המטרה: לחשב את A מדרגה חסומה n בn ערכי x שונים שמקיימים את תכונת הנגדיות החזקה.

א. נחשב את A_{odd} שהוא מדרגה חסומה $\frac{n}{2}$, ב$\frac{n}{2}$ ערכי x: $(x_0^2, x_1^2, ..., x_{\frac{n}{2}-1}^2)$ - מההגדרה הרקורסיבית, ערכים אלו מקיימים את תכונת הנגדיות החזקה.

ב. נחשב את A_{even} שהוא מדרגה חסומה $\frac{n}{2}$, ב$\frac{n}{2}$ ערכי x: $(x_0^2, x_1^2, ..., x_{\frac{n}{2}-1}^2)$ - מההגדרה הרקורסיבית, ערכים אלו מקיימים את תכונת הנגדיות החזקה.

ג. לכל $0 \leq i \leq \frac{n}{2}-1$ יתקיים: $A(x_i) = A_{even}(x_i^2) + x_i A_{odd}(x_i^2)$

ד. לכל $0 \leq i \leq \frac{n}{2}-1$)החצי השני של הערכים(נשים לב כי לפי הנגדיות החלשה ומעבר שראינו לעיל מתקיים: $A(x_{\frac{n}{2}+i}) = A(-x_i) = A_{even}(x_i^2) - x A_{odd}(x_i^2)$, וסה"כ באמצעות א+ב חישבנו גם את החצי הזה של הערכים בלי לבצע פעולות נוספות.

סה"כ הבעיות נפתרו - בכל שלב אנחנו מקבלים תת בעיה, וכן הערכים תמיד יקיימו את תכונת הנגדיות החזקה. סה"כ סיבוכיות הזמן של האלגוריתם הינה $T(n) = 2T(\frac{n}{2}) + O(n)$, וממאסטר נקבל $O(nlogn)$ למעבר ממקדמים לייצוג ע"י נקודות.

2.3.3 איזה מספרים מקיימים את תכונת הנגדיות החזקה?

מספרים מרוכבים. נזכר כי מס' מרוכב ניתן לייצג ע"י $z = a + bi = rcis\Theta = re^{i\Theta}$. אנחנו נתמקד במספרים בהם $r = 1$.

מספר ω נקרא שורש היחידה המרוכב מסדר n, אם $\omega^n = 1$. למשל, נראה כי $z = e^{\frac{2\pi i}{8}}$, מתקיים כי $z^8 = 1$.

נגדיר את המספר ω_n להיות: $\omega_n = e^{\frac{2\pi i}{n}}$. נשים לב כי תמיד $\omega_n^n = 1$. (הערה ־ נשים לב כי $\omega_4 = e^{\frac{\pi i}{2}} = i$. מדוע? נראה כי הזווית היא $\frac{\pi}{2}$, כלומר 90 מעלות. אם נזוז 90 מעלות מהכיוון החיובי של הציר הממשי, נגיע בדיוק למספר i. וכך בדיוק מחשבים מספרים אלו).

n שורשי היחידה מסדר n הינם: $\omega_n^0, \omega_n^1, ..., \omega_n^{n-1}$.

טענה: לכל $0 \leq k \leq n-1$ מתקיים כי $(w_n)^k$ הוא שורש יחידה מסדר n.
הוכחה: נרצה להוכיח כי מספר זה בחזקת n שווה לאחד. ובכן ־

$$((w_n)^k)^n = ((e^{\frac{2\pi i}{n}})^k)^n = (e^{\frac{2\pi ink}{n}}) = e^{2\pi ik} = e^0 = 1$$

טענה 2: יהי $n > 1$ חזקה של 2. אזי לכל $0 \leq k \leq \frac{n}{2} - 1$ מתקיים $w_n^{\frac{n}{2}+k} = -w_n^k$.
הוכחה:

$$\omega^{\frac{n}{2}+k} = \omega^{\frac{n}{2}} \times \omega^k = \omega^k \times e^{\frac{2\pi\frac{n}{2}}{n}} = -\omega^k$$

טענה 3: יהי $n > 1$, חזקה של 2. אזי הריבועים של $\omega_n^0, \omega_n^1, ..., \omega_n^{n-1}$ הם בדיוק $\frac{n}{2}$ שורשי היחידה מסדר $\frac{n}{2}$. (כלומר, נעלה אותם בריבוע, נקבל בדיוק את שורשי היחידה מסדר $\frac{n}{2}$, וכל אחד מהם יופיע פעמיים ־ כלומר יהיו כפילויות).

הוכחה: עבור $0 \leq k \leq \frac{n}{2} - 1$ מתקיים

$$(\omega_n^k)^2 = e^{i\frac{2\pi}{n}2k} = (e^{\frac{i2\pi}{\frac{n}{2}}})^k = \omega_{\frac{n}{2}}^k$$

טענה 4: יהי $n \geq 1$ חזקה של 2. סדרת n שורשי היחידה מסדר n: $\omega_n^0, \omega_n^1, \omega_n^2, ..., \omega_n^{n-1}$ מקיימים את תכונת הנגדיות החזקה.

נשים לב ־ n שורשי היחידה מסדר n הם אכן שונים זה מזה.

2.4 האלגוריתם FFT

המטרה: לקחת את המערך עם המקדמים של A, ולחשב את הפולינום A ב־n שורשי היחידה מסדר n (אשר הוכחנו שמקיימים את תכונת הנגדיות החזקה). כלומר לחשב את $A(\omega_n^0), ..., A(\omega_n^{n-1})$.

קלט: $A = (a_0, a_1, ..., a_{n-1})$ (מערך המקדמים של הפולינום A) מדרגה חסומה n

א. אם $n = 1$, החזר a_0.

ב. כעת נגדיר את $A_{even} = (a_0, a_2, ..., a_{n-2})$ מדרגה חסומה $\frac{n}{2}$

ג. כעת נגדיר את $A_{odd} = (a_1, a_3, ..., a_{n-1})$ מדרגה חסומה $\frac{n}{2}$

ד. כעת נתחיל את הרקורסיה: $P_{even} = FFT(A_{even})$, כאשר P_{even} יפעיל את FFT על A_{even} - כלומר מחשבים את A_{even} בn שורשי היחידה מסדר $\frac{n}{2}$.

כלומר, מה שחוזר מהרקורסיה הינו $P_{even} = [A_{even}(\omega_{\frac{n}{2}}^0), A_{even}(\omega_{\frac{n}{2}}^1), ..., A_{even}(\omega_{\frac{n}{2}}^{\frac{n}{2}-1})]$

ה. בדומה - $P_{odd} = FFT(A_{odd})$

ו. החל מ$j = 0$ עד $j = \frac{n}{2} - 1$ בצע: (כעת אנחנו רוצים לחשב את הפלט שלנו - מחזירים לבסוף וקטור $\vec{y}$ עם חישוב הערכים בהתאם לפי הנוסחאות שראינו)

1. $y_j = P_{even}[j] + w_n^j \times P_{odd}[j]$
(נשים לב כי $y_j = A(\omega_n^j)$ ולפי נוסחה שראינו $A(x) = A_{even}(x^2) + xA_{odd}(x^2)$, כמו כן ניתן להמירה בהתאם $A(\omega_n^j) = A_{even}({\omega_n^j}^2) + \omega_n^j A({\omega_n^j}^2)$, וכפי שראינו מתקיים $(\omega_n^j)^2 = w_{\frac{n}{2}}^j$ לפי טענה3 לעיל, ולכן $A(\omega_n^j) = A_{even}(w_{\frac{n}{2}}^j) + \omega_n^j A(w_{\frac{n}{2}}^j)$, כלומר אם נסתכל על מערכי הקלט ששמרנו זה ממש שקול ל$(P_{even}[j] + w_n^j \times P_{odd}[j]$.

2. $y_{\frac{n}{2}+j} = P_{even}[j] - w_n^j \times P_{odd}[j]$

ז. כשהרקורסיה נגמרת - החזר את $(y_0, ..., y_{n-1})$

סיבוכיות זמן הריצה: הנוסחה - $T(n) = 2T(\frac{n}{2}) + O(n) = O(nlogn)$, ולכן סה"כ זמן חישוב האלגוריתם שמקבל פולינום בשיטת המקדמים, וממיר אותם לn נקודות ספציפיות (שורשי היחידה מסדר n) לפי שיטת הנקודות הוא $O(nlogn)$.

2.5 כיצד נעבור כעת משיטת הנקודות חזרה לשיטת המקדמים?

נשים לב כי אנחנו יודעים את ערכי x הנקודות שלנו, הם n שורשי היחידה מסדר n.
נחזור ל**מטריצת ונדרמונדה.** נציב $x_0 = (\omega_n)^0 = 1, x_1 = (\omega_n^1), ..., x_{n-1} = (\omega_n^{n-1})$

$$\begin{pmatrix} y_0 \\ y_1 \\ . \\ . \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & .. & 1 \\ 1 & \omega_n^1 & (\omega_n^1)^2 & .. & (\omega_n^1)^{n-1} \\ 1 & (\omega_n^2)^1 & (\omega_n^2)^2 & .. & (\omega_n^2)^{n-1} \\ .. & .. & .. & .. & .. \\ 1 & (\omega_n^{n-1})^1 & (\omega_n^{n-1})^2 & .. & (\omega_n^{n-1})^{n-1} \end{pmatrix} \times \begin{pmatrix} a_0 \\ a_1 \\ . \\ . \\ a_{n-1} \end{pmatrix}$$

נשים לב כי בידינו קודם לכן היה $\vec{a}$, כפלנו אותו במטריצת הFFT שלנו, V וקיבלנו את $\vec{y}$. באופן כללי - כפל נאיבי של מטריצה בוקטור עולה $O(n^2)$ זמן.

מסקנה חשובה (!!): כפל של מטריצת ונדרמונדה שמוגדרת ע"י n מספרים שמקיימים את תכונת הנגדיות החזקה, בוקטור $\vec{a}$ עולה $O(nlogn)$ (שזה בדיוק אותו תהליך שעשה האלגוריתם).

טענה: המטריצה ההופכית של V, FFT^{-1} הינה המטריצה:

$$V^{-1} = FFT^{-1} = \frac{1}{n}\begin{pmatrix} 1 & 1 & 1 & .. & 1 \\ 1 & \omega_n^{-1} & (\omega_n^{-1})^2 & .. & (\omega_n^{-1})^{n-1} \\ 1 & (\omega_n^{-2})^1 & (\omega_n^{-2})^2 & .. & (\omega_n^{-2})^{n-1} \\ .. & .. & .. & .. & .. \\ 1 & (\omega_n^{-(n-1)})^1 & (\omega_n^{-(n-1)})^2 & .. & (\omega_n^{-(n-1)})^{n-1} \end{pmatrix}$$

אם נסתכל על המטריצה $n \times FFT^{-1}$ (שהרי נרצה להכפיל בn כי נשים לב ל$\frac{1}{n}$ שיצא החוצה מהמטריצה), נראה כי היא מטריצת ונדרמונדה על הערכים:
.$(\omega_n^0, \omega_n^{-1}, \omega_n^{-2}, ..., \omega_n^{n-1})$

מסקנה: על מנת לעבור מוקטור $\vec{y}$ לוקטור המקדמים $\vec{a}$, נראה כי בהכפלה במטריצה ההופכית מקבלים ממש $\vec{a} = FFT^{-1} \times \vec{y}$, זו מכפלה של מטריצת ונדרמונדה על n מספרים שמקיימים את תכונת הנגדיות החזקה (שגם הם, n שורשי היחידה מסדר n) , בוקטור, ראינו בטענה לעיל שמכפלה זו עולה $O(nlogn)$, ולכן **המסקנה שלנו היא שגם המעבר חזרה - משיטת הנקודות חזרה אל שיטת המקדמים, עולה גם הוא $O(nlogn)$.**

סיכום - כפל פולינומים:

א. מקבלים את הפולינומים $A(x), B(x)$ המיוצגים ע"י מקדמים.

ב. בעזרת אלגוריתם FFT, בזמן $O(nlogn)$ מקבלים את $A(x), B(x)$ מיוצגים ע"י n נקודות (שהם שורשי היחידה מסדר n)

ג. מכפילים את שני הפולינומים בזמן $O(n)$ בשיטת הנקודות, כיוון שהם מיוצגים ע"י אותם ערכי x (שורשי היחידה).

ד. משתמשים שוב בFFT, באמצעות הכפלה ע"י המטריצה FFT^{-1} שגם היא מטריצת ונדרמונדה, מחזירים את הפולינומים לשיטת המקדמים, מה שיעלה עוד $O(nlogn)$.

סה"כ - $O(nlogn)$ לכפל פולינומים.

2.6 תרגילים FFT

2.6.1 כפל פולינומים מדרגות חסומות שונות

קלט: $A(x) = \sum_{i=0}^{n-1} a_i x^i, B(x) = \sum_{i=0}^{n-1} b_i x^i$ באשר בה"כ $m < n$.
פלט: $C(x) = A(x) \times B(x)$

פתרון:

1. **אפשרות ראשונה:** נתייחס אל B כפולינום חסום מדרגה n. לפי FFT- כפל שני פולינומים מדרגה זו יעלה $O(nlogn)$.

2. **אפשרות שנייה:** נוכל להשתמש באלגוריתם נאיבי לכפל פולינומים - שעובר עליהם אחד אחד לפי הנוסחה, ולקבל סיבוכיות $O(nm)$

3. **אפשרות שלישית:** להריץ במקביל את אפשרות 1 ואפשרות 2, ונקבל סיבוכיות $O(n \cdot min(m, logn))$

4. **אפשרות רביעית:**
נרצה להגיע לזמן ריצה $O(nlogm)$
הרעיון יהיה לחלק את A לפולינומים קטנים יותר בגודל m, סה"כ $\frac{n}{m}$ פולינומים.
נשים לב כי:

$$A = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_{n-1} x^{n-1} =$$

$$(a_0+a_1x+...+a_{m-1}x^{m-1})+(a_mx^m+....a_{2m-1}x^{2m-1})+........+(a_{n-m}x^{n-m}+.....a_{n-1}x^{n-1})$$

$$(a_0+a_1x+...+a_{m-1}x^{m-1})+x^m(a_m+....a_{2m-1}x^{m-1})+........+x^{n-m}(a_{n-m}+.....a_{n-1}x^{m-1})$$

כעת נשים לב, כל בלוק מהנ"ל יסומן $A_{km} = a_{km} + ... + a_{km+m-1}x^{m-1}$ ונקבל: $A(x) = \sum_{i=0}^{\frac{n}{m}-1} A_i(x) \cdot x^{mi}$
כעת נראה כי:

$$C(x) = A(x) \cdot B(x) = A_0(x)B(x) + x^m B(x)A_1(x) + ... + x^{n-m}B(x)A_{\frac{n}{m}-1}(x)$$

כעת, הכפל של פולינום $C(x)$ מורכב מ$\frac{n}{m}$ כפלים שונים של פולינומים, כל אחת מהפולינומים הינו בדרגה m (פולינום A_{ki} מדרגה m וכן פולינום B מדרגה m)
סה"כ סיבוכיות זמן הריצה תהיה -

$$\frac{n}{m} \times mlogm = O(nlogm)$$

2.6.2 בעיית $3SUM$

פלט: מערך $A = [a_0, ..., a_{n-1}]$ באשר $\forall 0 \leq i \leq n-1 : a_i \in \mathbb{R}$
פלט: האם קיימים a_i, a_j, a_k כך ש $a_k = a_i + a_j$
פתרון ראשון:
האלגוריתם יפעל כך: הרעיון יהיה להגיע לסיבוכיות של $O(n^2)$. האלגוריתם יפעל כך -
א. נמיין את המערך A
ב. עבור $k = 0$ עד $k = n-1$ בצע:
1. הגדר $i = 0, j = n-1$
2. אם $a_k = a_i + a_j$ החזר $true$
3. אם $a_i + a_j > a_k$, אזי שסכום המספרים גדול מדי, לכן צריך להוריד מס' מימין, כלומר: $j = j-1$
4. אם $a_i + a_j < a_k$ אזי שסכום המספרים קטן מדי, לכן צריך להתקדם משמאל לימין. כלומר $i = i+1$

ג. אחרת - החזר $false$

זמן הריצה: $T(n) = \sum_{k=0}^{n-1} n + O(nlogn) = O(n^2)$.
פתרון שני (מספרים מעולם חסום):
כעת נניח כי כל המספרים $a_i \in A$ הינם חסומים בתחום $[1, 10n^{1.5}]$
נרצה לקודד את המערך A לפולינום מדרגה חסומה $10n^{1.5}$ בצורה הבאה:
המקדם של a_i של x^i יהיה שווה 1 אם $i \in A$, 0 אחרת. למשל, עבור $A = [2, 5, 7]$ נקבל $A(x) = x^2 + x^5 + x^7$
כעת: האלגוריתם יבצע את הכפל של הפולינום בעצמו, כלומר $A^2(x)$. לאחר מכן, נחפש מיקום $1 \leq i \leq n$ בפולינום המכפלה $C(x) = A^2(x)$, בו המקדם $c_i \geq 2$, וגם מתקיים כי המקדם a_i במערך A שווה 1. אם האלגוריתם מצא מיקום שכזה, הוא יחזיר 1. אחרת יחזיר שקר.

נכונות: מחוק החזקות $x^{i+j} = x^i x^j$, כאשר נכפול את הפולינום שמייצג את המערך בעצמו, פולינום המכפלה יציג למעשה סכומים של איברי המערך. מכאן, כל חזקה בפולינום שחזקתו ≤ 2 הוא בהכרח מכפלה של שני מספרים אחרים במערך. כעת, כל שנותר לעשות לאלגוריתם הוא לעבור במערך המקורי, ולחפש מס' i שקיים בו. אם מצאנו כזה - קיים מספר ששווה לסכום שניים אחרים.

סיבוכיות זמן הריצה: כפל לפי FFT יעלה $n^{1.5}log(n^{1.5})$, מעבר על פולינום המכפלה ובדיקת המקדמים שלו יעלה $O(n^{1.5})$, וסה"כ סיבוכיות זמן הריצה תהיה $O(n^{1.5}logn)$.

2.6.3 בעיית חישוב מרחק האמינג

בעיית התאמת המחרוזות מוגדרת כך:
קלט: מחרוזת T באורך n - "טקסט", ומחרוזת P באורך $m \leq n$ - "תבנית".
פלט: כל המקומות בT שP מופיע בהם.

מרחק האמינג:
לשתי מחרוזות A, B באורך זהה n , מרחק ההאמינג של A וB הוא מס' האינדקסים בהם A וB שונים. כלומר

$$HD(A,B) = |\{i|A[i[\neq B[i]\}|$$

נגדיר כעת את הבעיה הבאה שמכלילה את בעיית התאמת המחרוזות:

בעיית חישוב מרחקי האמינג בין טקסט לתבנית:
קלט: מחרוזת $T[1,..,n]$ באורך n: "טקסט". ומחרוזת $P[1,...,m]$ באורך $m \leq n$: "תבנית".
פלט: לכל היסט $0 \leq i \leq n-m$ את מרחק ההאמינג $HD(P,T[i+1,...,i+m]$ (נשים לב - הפלט יוחזר במערך בגודל $n-m$, כאשר בתא הראשון יופיע מרחק ההאמינג עבור המחרוזת שהתחילה בתא הראשון והסתרעה על התאים $m-1$, בתא השני יופיע מרחק האמינג עבור המחרוזת שהתחילה בתא השני והסתרעה על התאים $m+1-2$ וכן הלאה. התא האחרון יהיה במיקום $n-m$ שהוא יבדוק את המחרוזת שהתחילה במקום $n-m$ והסתיימה במיקום n)
נחשב בעיה זו כאשר הא"ב הוא בינארי, כלומר מעל $\{0,1\}$.

פתרון נאיבי - נבצע בדיקה של כל היסט באופן נאיבי ע"י בדיקה שתעלה $O(m)$, יש כ$O(n)$ היסטים ולכן הפתרון יעלה $O(nm)$.

פתרון באמצעות FFT:
נסמן את הטקסט באותיות $T = a_0a_1....a_{n-1}$ וכן $P = b_0b_1....b_{m-1}$
FFT מאפשר לכפול פולינומים. ננסה לחשוב על הכפל בצורה מעט יותר מופשטת - כפעולה בינארית הפועלת בין שני אובייקטים ומחזירה מספר.
נראה כי כפל פולינומים, דומה לכפל בין מספרים מרובי ספרות - המתאים לספרה הi (כלומר a_i, המקדם של הבסיס בחזקת i).
ההבדל המשמעותי היחיד - הוא שבכפל בין ספרות יש בסיס, וכאשר המקדם של ספרה גדול מהבסיס הוא עובר כנשא לחזקה הבאה של הבסיס.
כפל בין מספרים ניתן לחשב באמצעות כפל ארוך, ונשים לב שאותו ייצוג יעבוד גם לכפל פולינומים. כמובן, FFT אינו כפל ארוך, אך סוף סוף תוצאת הכפל זהה לא משנה האם נעשתה באמצעות כפל ארוך או באופן מהיר יותר באמצעות אלגוריתם FFT.

נסתכל על כפל ארוך בין שני מספרים/פולינומים המייצגים את הטקסט והתבנית. נכפול את T בP^R (היפוך סדר המקדמים).

דוגמה: אם $n = 5$ ו$m = 3$ נסתכל על הפולינומים $(a_0, a_1, a_2, a_3, a_4)$ (b_2, b_1, b_0) ונכפול אותם בגישת כפל ארוך כדקלמן -

$$
\begin{array}{ccccccc}
 & & a_0 & a_1 & a_2 & a_3 & a_4 \\
 & & & & b_2 & b_1 & b_0 \\
\hline
 & & \color{red}{a_0b_0} & \color{magenta}{a_1b_0} & \color{blue}{a_2b_0} & a_3b_0 & a_4b_0 \\
 & a_0b_1 & \color{red}{a_1b_1} & \color{magenta}{a_2b_1} & \color{blue}{a_3b_1} & a_4b_1 & \\
a_0b_2 & a_1b_2 & \color{red}{a_2b_2} & \color{magenta}{a_3b_2} & \color{blue}{a_4b_2} & &
\end{array}
$$

ניתן לשים לב שבכל אחת משלושת העמודות האמצעיות (שצבועות) נקבל סכום של אותיות מקבילות בתבנית ובטקסט, כאשר שמים את התבנית מול הטקסט בהיסטים שונים. כלומר - קיבלנו את כל אפשרויות ההיסט עבורם בקלט הנוכחי. נרצה שהסכימה הנ"ל תהיה משמעותית. כלומר: נרצה שבשביל לחשב את מרחק ההאמינג, כל שנצטרך יהיה לחבר אצ כל הערכים במסלול האדום ולקבל את מרחק האמינג, בורוד וכן בכחול ופתרנו את הבעיה. נדאג לכך שהסכום יהיה בדיוק מס' המקומות בהם התבנית מתאימה לטקסט.

כלומר, נרצה שכפל בין שני תווים ייצג $ab = 1$ אם $a = b$ וכן $ab = 0$ אחרת. נראה כי פעולה זו ניתנת לייצוג ע"י הטבלה -

$T \backslash P$	0	1
0	1	0
1	0	1

פעולה זו עולה $O(1)$ לחישוב. עם זאת, נראה כי חישוב הכפל הארוך יעלה $O(nm)$. לא יותר טוב מהדרך הנאיבית.

נרצה להשתמש בFFT בשביל לבצע את האלגוריתם בזמן $O(nlogm)$.

עם זאת, FFT מוגדר ככפל פולינומים מעל $\mathbb{C}$, ככפל רגיל, לא כפעולה שתיארנו לעיל. מכאן שנצטרך להתגבר על בעיה זו: נפריד את החישוב לשני חישובים נפרדים.

רצינו לספור את כל המקומות בהם התו בתבנית זהה לתו בטקסט בהיסט המתאים. כל מקום כזה עונה על בדיוק אפשרות אחת מתוך שתיים:

1. $a = b = 1$
2. $a = b = 0$

נספור כל אחד מהמקרים הללו בנפרד, ונסכום.

לספירת המקומות בהם $a = b = 1$ נשתמש בפעולה הבאה:

$T \backslash P$	0	1
0	0	0
1	0	1

זוהי בדיוק פעולת הכפל הרגילה. לכן נוכל להשתמש בFFT באופן מיידי

לספירת המקומות בהם $a = b = 0$ נשתמש בפעולה הבאה:

$T\backslash P$	0	1
0	1	0
1	0	0

כדי לקבל פעולה זו באמצעות כפל רגיל, נהפוך את הביטים בטקסט ואת הביטים בתבנית - כל ביט שהיה 0 יהיה 1 וכל מי שהיה 1 יהיה אפס. ולכן נקבל עבור $a = b = 0$

$\overline{T}\backslash\overline{P}$	1	0
1	1	0
0	0	0

שזו בדיוק פעולת הכפל.

לסיכום: באמצעות שתי פעולות FFT על הקלטים של הפולינומים ופעולות הכפל שהוגדרו לעיל, נדע כמה התאמות יש בין התבנית לטקסט בכל היסט. מה שיוחזר לנו כתוצאה מהפעלת FFT לבדיקת ההתאמות על 1 ייוצג במערך מקדמים A_1, ומה שיוחזר כתוצאה מהפעלת FFT לבדיקת ההתאמות על 2 ייוצג במערך מקדמים A_2, וסה"כ נבצע חיבור למערך חדש $C[i] = A_1[i] + A_2[i]$ לקבלת מס' ההתאמות. לבסוף, נקח את מספר ההתאמות ונחסר אותו מm, ונקבל בדיוק את מספר האי התאמות שזהו בדיוק מרחק האמינג. כלומר, לכל תא i נגדיר $C[i] = m - C[i]$.

כפי שראינו בדוגמה 1: פולינומים מדרגות חסומות, ניתן לכפול שני פולינומים שחסומים בדרגות $n \geq m$ בעלות $O(nlogm)$ וכן זו סיבוכיות האלגוריתם. חיבור הפולינומים יעלה $O(n)$ ולא ישנה את הסיבוכיות האסימפטוטית. וכן, ביצוע פעולת NOT בשביל שנוכל להשתמש בFFT על התאמות של 0 עלה $n + m$. סה"כ סיבוכיות אלגוריתם - $O(nlogm)$.

נשים לב: ניתן להרחיב את הפתרון גם לא"ב שהוא לא רק $\{0, 1\}$. ניתן להרחיב את הפתרון לגרסה טובה יותר מאשר NOT- נניח ונקבל א"ב $\{0, 1, 2\}$. תחילה - נכתוב 1 בכל המיקומים של 2 ובאשר המיקומים נכתוב אפס ונבצע FFT ונדע מהו מס' ההתאמות של 2. לאחר מכן נכתוב 1 בכל המיקומים של 1 ובשאר נכתוב אפס וכן הלאה כנ"ל על אפס. מכאן קיבלנו מסקנה: בהינתן א"ב Σ, נוכל לבצע את אלגוריתם מרחק ההאמינג בעלות כוללת של $|\Sigma| \times O(n + m + nlogm)$ שכן נדרשים לבצע בכל שלב $n + m$ על מנת להפוך את המערך לאחדות ואפסים, וכן עוד $nlogm$ לביצוע הFFT.

להלן האלגוריתם:

א. צור מערך חדש M בגודל $n - m + 1$

ב. לכל $\sigma \in \Sigma$:

ספור לכל היסט אפשרי של התבנית בטקסט את מספר ההתאמות של התו σ ע"י FFT של $T_\sigma \times P_\sigma^R$

ג. החזר את M.

נשים לב כי לכל היותר במצב בו כל התווים שנקבל שונים, יתקיים $|\Sigma| = O(m + n) = O(n)$

3 הרצאה 2 - MST

3.1 עץ פורש מינימום

עץ פורש מינימום, או MST (Minimum spanning tree) הוא מה שנעסוק בו בהרצאה זו.

למה צריך גרפים? למשל, עבור רשתות תקשורת. כל קודקוד בעץ הוא מחשב ברשת, וכל קשת בין הקודקודים מציינת האם יש תקשורת ישירה בין שני המחשבים ברשת. אנחנו חברה, שרוצה למכור רשתות תקשורת, והמטרה שלנו היא שהקשתות שנבחור ישאירו את הגרף קשיר. כלומר - נרצה לבחור קשתות כרצוננו רק נשים לב שבכל שלב נתון נוכל להגיע לכל מחשב ברשת. נשים לב - שבמהלך הבחירה שלנו יתכן ונבחר קשתות מיותרות, בשביל חלילה לא להגיע למצב שהגרף לא קשיר.

עץ פורש: תת גרף של הגרף המקורי, שהוא קשיר וללא מעגלים (עץ). כלומר $V' = V$ וכן $E' \subseteq E$.

ייצוג לגרפים: מטריצת שכנויות, רשימה $(List)$ של שכנויות.

נשים לב כי נרצה למצוא את העץ הפורש הנ"ל שיאפשר לנו למצוא דרך להגיע לכל המחשבים (קודקודים) ברשת (גרף). נשים לב - כי ייתכן ויהיה כמה דרכים לעשות זאת, כמו כן - ייתכן ולכל מעבר ברשת מסוימת יהיה מחיר שונה (כלומר, לכל קשת יהיה מחיר שנשלם שנעלה עליה). ולכן כל קשת תסומן אצלנו בערך מספרי מסויים.

בהינתן x קודקודים בעץ הפורש, נרצה למצוא $x-1$ קשתות שהעלות שלהן יחד היא הנמוכה ביותר.

נשים לב כי כאשר נעבוד עם עצים פורשים מינימום, נניח מראש כי הגרף יהיה קשיר.

יהי $G = (V, E)$ מולטי גרף קשיר ממושקל עם פונקציית משקל על הקשתות $w : E \to \mathbb{R}$.
יהי $T = (V, E_T)$ עץ פורש של G. אזי, נאמר שהמשקל של T הוא

$$w(T) = \sum_{e \in E_T} w(e)$$

עץ פורש מינימום של G הינו עץ פורש שמשקלו הוא הקטן ביותר.

מסקנה: יהי T עפ"מ, אזי לכל עץ פורש $T' \in G$ מתקיים $w(T') \geq w(T)$.

הערה: מולטי גרף הוא גרף בו קבוצת הקשתות הינה *multi-set* , כלומר בין שני קודקודים בגרף, יכולות לעבור מספר קשתות. ומדוע שנרצה להשתמש בו? הרי ברור כי כאשר נחפש עפ"מ, בהינתן שני קודקודים v, u וקשתות e_1, e_2 שמשקלן בהתאמה $1, 2$, נרצה לבחור בקשת שמשקלה 1. בהמשך, נבין מדוע האלגוריתם פועל על מולטי גרף למרות שזה לא נראה אינטואטיבי.

הערה2: גרף לא מכוון הוא גרף בו התנועה דו כיוונית, אם קיימת $e : v \to u$ בהכרח אפשר לנוע גם ב$u \to v$. גרף מכוון הוא זה בו התנועה מוגדרת בכיוון מסויים. כלומר זה שקיימת $e : v \to u$ לא גורר שניתן ללכת בכיוון $u \to v$.

3.2 בעיית מציאת עץ פורש מינימום

קלט: גרף לא מכוון קשיר $G = (V, E)$ ופונקציית משקל $w : E \to \mathbb{R}$

פלט: עץ פורש מינימום של G (ביחס לפונקציית משקל w).

3.3 אלגוריתמים חמדניים $(Greedy)$

מקבלים החלטה לוקאלית וממשיכים רקורסיבית בלי לשנות את ההחלטה ובלי לדעת מה הפתרון ברקורסיה. למשל - חיפוש בינארי.

למת הבחירה החמדנית: הבחירה שהאלגוריתם ביצע באופן חמדני לא מונעת ממנו להגיע לפתרון האופטימלי.

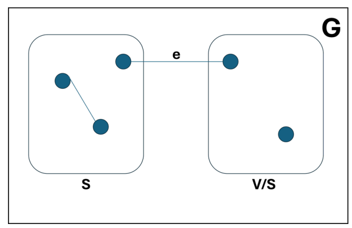
למת תת המבנה האופטימלי: מכלול בחירות חמדניות יביא את התוצאה האופטימלית.

3.4 למת הבחירה החמדנית

חתך: יהי $G=(V,E)$ גרף ויהי $S\subset V$ כך ש$S\neq\emptyset$. החלוקה $(S,V/S)$ נקראת חתך של G והקשתות $\{e=(u,v)|u\in S,v\in V/S\}$ נקראות **קשתות שחוצות את החתך / קשתות בחתך**.

נשים לב - S לעולם לא תהיה שווה לV, ולעולם לא תהיה ריקה.

בתמונה לעיל, e היא קשת שחוצה את החתך.



קשת קלה ביותר בחתך: יהי $G=(V,E)$ גרף לא מכוון עם פונקציית משקל $w:E\to\mathbb{R}$. יהי $(S,V/S)$ חתך של G. קשת $(u,v)\in E$ נקראת קשת קלה ביותר בחתך $(S,V/S)$ אם לכל קשת e' שחוצה את החתך מתקיים $w(e)\leq w(e')$. (נשים לב, יתכן שיש כמה קשתות כאלו באותו משקל שהן הקלות ביותר).

למה 1: יהי $G=(V,E)$ מולטי גרף קשיר עם פונקציית משקל $w:E\to\mathbb{R}$. לכל $\emptyset\neq S\subset V$, לכל קשת e קלה בחתך $(S,V/S)$ קיים עפ"מ שמכיל את e.

הוכחה: יהי $G=(V,E)$ מולטי גרף קשיר עם פונקציית משקל $w:E\to\mathbb{R}$.

יהי $(S,V/S)$ חתך של G, ותהי $e=(u,v)$ קשת קלה בחתך, באשר $u\in S,v\in V/S$.

יהי T עפ"מ של G. (בהכרח קיים כזה, כיוון שG קשיר, לכן יש לו עצים פורשים, ובהכרח אחד מהם מינימלי). אם $e\in T$ אזי סיימנו.

אחרת, $e\notin T$. לפי תכונות של עצים - קיים מסלול פשוט P (כל הקודקודים בו, ופעם אחת בלבד) יחיד בT מu אל v. מסלול פשוט זה, לא מכיל את e כיוון ש$e\notin T$.

בP חייבת להיות קשת שחוצה את החתך. אחרת, כל קשת שנעבור בה תשאיר אותנו בחתך, ואז לא נוכל להגיע אל הצד השני של העץ, מעבר לחתך, שבהכרח יש שם קודקודים כיוון ש$\emptyset\neq S\subset V$.

נסמן ב$e'=(u',v')$ את הקשת הראשונה בP שחוצה את החתך. כלומר, $P:(P_1^{u\rightsquigarrow u'}\to e'\to P_2^{v'\rightsquigarrow v})$

נבנה $T'=(V,E_{T'})$ באשר $E_{T'}=E_T\cup\{e\}\setminus\{e'\}$. כעת נוכיח כי T' הוא MST.

1. נוכיח T' הוא עץ פורש: עלינו להוכיח כי הוא קשיר וכן כי $|E_{T'}|=|V|-1$, ואז בהכרח הוא עץ פורש. נשים לב כי

$$|E_{T'}| = |E_T| + 1 - 1 = |E_T|$$

כיוון שT הוא עפ"מ, הוא בהכרח ע"פ ולכן $|E_T| = |V| - 1$, ומכאן ש$|E_{T'}| = |E_T| = |V| - 1$.

נוכיח קשירות. יהיו $x, y \in V$. נרצה להוכיח כי קיים מסלול בT' בין x לy. T הוא ע"פ, לכן קיים מסלול פשוט יחיד מx לy. נסמן את המסלול בP'.

אם $e' \notin P'$, אזי המסלול P' קיים גם בעץ T' ולכן יש מסלול בין x לy. (כלומר, הצלע שהורדנו לא נמצאת על המסלול בין השניים).

אם $e' \in P'$ (כלומר, הצלע שהורדנו בבניית העץ נמצאת על המסלול הפשוט), נניח בה"כ כי u' מופיע לפני v' בP' ונסמן את המסלול $x \rightsquigarrow u'$ בP'_1 ואת המסלול $y \rightsquigarrow v'$ בP'_2. כלומר,

$$P' : (P_1^{x \rightsquigarrow u'} \to e' \to P_2^{v' \rightsquigarrow y})$$

כעת נבנה מסלול בT' מx לy באופן הבא:

$$P'_{1x \to u'} \rightsquigarrow (P_1^R)_{u' \to u} \rightsquigarrow e_{u \to v} \rightsquigarrow (P_2^R)_{v \to v'} \rightsquigarrow (P'_2)_{v' \to y}$$

הערה. P_1 הוא המסלול שמוביל אותנו מu' ל$u \to$. נרצה לכתוב R (רוורס) כי נרצה ללכת כעת במסלול ההפוך. בדומה עבור P_2^R.

סה"כ, בנינו מסלול מוכל בT' שעובר מx אל y, לכן העץ קשיר, ולכן T' הוא עץ פורש של G.

2. נוכיח כי $w(T') \le w(T)$, כיוון ש$w(T)$ הוא המינימלי, ואם $w(T') \le w(T)$ הוא קטן מהמינימלי ובפרט מכולם. נשים לב כי

$$w(T') = w(E_T \cup \{e\} \setminus \{e'\}) = w(T) + w(e) - w(e')$$

נתון כי e קשת קלה ביותר, ולכן $w(e) \le w(e')$ כלומר $w(e) - w(e') \le 0$ ולכן

$$w(T') = w(T) + w(e) - w(e') \le w(T)$$

כיוון שהורדנו משהו מ$w(T)$. סה"כ, $w(T') \le w(T)$, מצד שני T עפ"מ ולכן המשקל שלו קטן משל כל ע"פ אחר, ולכן המשקל של T' הוא הקטן ביותר.

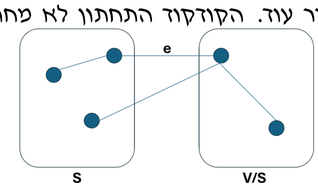
סה"כ T' הוא עץ פורש מינימום, שמכיל את הקשת e.

$\blacksquare$.

3.5 כיווץ קשתות

כרעיון בסיסי מאוד, בהתחשב בלמה שעמלנו קשות להוכיח לפני עמוד, נוכל למצוא את הקשת הקלה ביותר בחתך, לפי הלמה היא נמצאת בMST, ולהפעיל רקורסיה על צד S ורקורסיה על צד V/S וככה באופן רקורסיבי בשיטת הפרד ומשול למצוא את העץ הפורש.
זה לא עובד. למה?

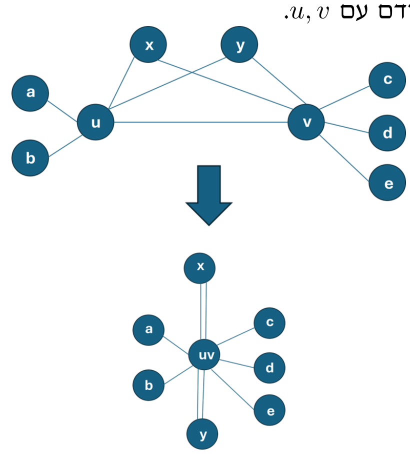
נתבונן בתמונה. מדובר בגרף קשיר. e היא הקשת הקלה ביותר. על פניו - אחלה. נעשה ברקורסיה כפי שאמרנו, כעת, כאשר נפעיל ברקורסיה על החתך (צד S), נקבל שהגרף איננו קשיר עוד. הקודקוד התחתון לא מחובר לקודקודים העליוניים. ולכן - זו איננה תת בעייה.



בעיה נוספת שנצטרך לטפל בה - איך נמנע ממצב בו כל הקשתות בגרף בעלות אותו ערך, 1 נניח, לפי שיטה בו בוחרים קשת קלה ביותר ומתקדמים - במצב זה נתקע כי כל הקשתות בגרף באותו גודל.

כיווץ קשתות:

התהליך יהיה די פשוט: נניח ונרצה להעלים" את הקשת בין $u \to v$, כל שנעשה יהיה כמו בתרשים מטה: נאחד את u, v לקודקוד משותף בשם uv, את השכנים (הלא משותפים שלהם) נחבר באמצעות הוספת קשת לuv בין כל אחד מהשכנים. את השכנים המשותפים שלהם (x, y) נחבר לuv באמצעות הוספת שתי קשתות. כל קשת תקבל את הערך שהיה לה מקודם עם u, v.



פורמלית: יהי מולטי גרף $G = (V, E)$ לא מכוון, פעולת כיווץ על קשת $e = (u, v)$ מייצרת גרף חדש $G/e = (V/e, E/e)$ כך ש:

$$V/e = V \cup \{uv\} \setminus \{u, v\}$$

נגדיר פונקציה: $F : V \to V/e$ כך ש:

$$F(x) := \left\{ \begin{array}{cc} x & x \neq v, u \\ uv & x = u \vee x = v \end{array} \right\}$$

הפונקציה F ממפה את הקודקודים המקוריים, לקודקוד החדש שמייצג אותם.
כעת נגדיר את E/e:

$$E/e := \{(F(x), F(y))|(x, y) \in E\}$$

כך למשל: אם $x = u$ וכן $y \neq u, v$. בהפעלה הבאה של הפונקציה נקבל $(F(x), F(y)) = (uv, y)$ כיוון שכל מה שנשלח ל־u כעת נמצא ב־uv.

נשים לב כי ־ $(F(u), F(v)) = (uv, uv)$ וזו לולאה עצמאית. עם זאת ־ **בגרף לא מכוון אסור לולאות עצמאיות.** לכן, בתהליך הכיווץ נעלמת קשת אחת ־ אין לולאה עצמית, הקשת שהייתה בין u ל־v איננה עוד. כמו כן, אם נתקלנו בגרף בו יש שלוש קשתות בין u ל־v ־ כולן נעלמות בתהליך הכיווץ, לא רק אחת מהן.

3.6 אלגוריתם גנרי של MST

כעת נדון באלגוריתם גנרי, אין לנו עניין בזמן הריצה שלו ואין לנו דרך להריץ אותו ־ כשמו כן הוא. חסרים כאן יותר מדי פרטים טכניים וחשובים לזמן הריצה ולהפעלה שלו. אם כן, הוא חשוב כרעיון כללי עליו נתבסס בהמשך. להלן האלגוריתם ־

MST(G=(V,E),w):
1. $E_T = \emptyset$

2. while $|V| > 1$:
3. let e be the minimum weight edge of **some** cut in G
4. Add e to E_T
5. Contract e

6. Return E_T

נשים לב ־ שהאלגוריתם ממש מתבסס על למת הבחירה החמדנית, שקיים תמיד עפ"מ אם נבחר את הקשת הקטנה ביותר בחתך כלשהו. מה שנצטרך לעשות ־ זה לדבר בתכונת תת המבנה האופטימלי על סדרה של בחירות.

3.7 תכונת תת המבנה האופטימלי

למה2: יהי $G = (V, E)$ מולטי גרף קשיר, עם פונקציית משקל $w : E \to \mathbb{R}$. יהי $(S, V/S)$ חתך ב־G. ויהי $e = (u, v)$ קשת קלה בחתך $(S, V/S)$. יהי $T' = (V/e, E_{T'})$ עפ"מ עבור G/e. יהי $E_T = E_{T'} \cup \{e\}$. אזי, $T = (V, E_T)$ הוא עפ"מ עבור G.

(כלומר, קח את הקשת הקלה ביותר e, תכווץ אותה מהגרף ותקבל עפ"מ חדש של G/e. אם תקח בכל פעם את הקשת הזו ותחבר אותה לקבוצת הקשתות הקודמות שיצרו עפ"מ, אתה תקבל עפ"מ עבור G. כלומר ־ זה בדיוק הצעד ברקורסיה שמתבצע שוב ושוב, מה שקורה באלגוריתם שמופיע כאן מעלה, שיוצר לנו עץ פורש מינימום).

הוכחה:

יהי $G=(V,E)$ מולטי גרף קשיר, עם פונקציית משקל $w:E\to\mathbb{R}$. יהי $(S,V/S)$ חתך בG. ויהי $e=(u,v)$ קשת קלה בחתך $(S,V/S)$. יהי $T'=(V/e,E_{T'})$ עפ"מ עבור G/e. יהי $E_T=E_{T'}\cup\{e\}$.

נרצה להוכיח כי T הוא עץ פורש מינימום:

1. נוכיח T עץ פורש: נוכיח כי T גרף קשיר ללא מעגלים, ושהוא תת גרף של הגרף המקורי (טריוויאלי כי $E_T\subseteq E$). כל שנותר הוא להוכיח קשיר +ללא מעגלים. נוכיח כי הוא קשיר וכן כי $|E_T|=|V|-1$.

ראשית, נראה כי

$$|E_T|=|E_{T'}\cup\{e\}|=|E_{T'}|+1$$

אם כן, T' הוא עפ"מ ולכן מתקיים בו $|E_{T'}|=|V/e|-1=|V|-1-1=|V|-2$, כלומר $|E_{T'}|=|V|-2$.

סה"כ, $|E_T|=|E_{T'}|+1=|V|-2+1=|V|-1$, כנדרש.

כעת, נוכיח קשירות. יהיו $x,y\in V$. אם המסלול הפשוט (היחיד) E מ$F(y)$ $F(x)\to F(y)$ לא משתמש בuv כקודקוד פנימי, אזי אותו מסלול קיים גם בT'.

אחרת, במסלול uv כן קודקוד פנימי. החלפת הקודקוד uv במסלול הפשוט E בקשת $u\to_e v$, מייצרת מסלול פשוט מx לy בT.

מדוע? T' עפ"מ בG/e ובפרט קשיר. לכן שם, קיים המסלול $x\rightsquigarrow uv\rightsquigarrow y$, כאשר נסתכל ב$T$ חזרה, נוכל להסתכל על אותו מסלול בדיוק, בתוספת הקשת $u\to v$. כלומר $x\rightsquigarrow u\to v\rightsquigarrow y$, מסלול זה קיים גם ב$T$ כי לא שינינו דברים פרט לuv, ולכן סה"כ קיים מסלול פשוט בין x לy.

לכן T קשיר.

סה"כ T קשיר וכן $|E_T|=|V|-1$ ולכן T עץ פורש.

2. נוכיח כי T הוא בעל משקל קטן ביותר: נניח בשלילה כי T לא עץ פורש מינימום. אזי קיים $\hat{T}$ עפ"מ עבור G. מלמהו, נניח בה"כ כי $\hat{T}$ מכיל את e (הקשת הקטנה ביותר, אשר קיים עפ"מ שמכיל אותה לפי הלמה. נניח שזה העץ). נכווץ את $\hat{T}$ על e ונקבל את $\hat{T}'$ שמוגדר:

$$\hat{T}'=(V/e,E_{\hat{T}'})$$

באשר $E_{\hat{T}'}=E_{\hat{T}}/\{e\}$.

נשים לב כי $\hat{T}'$ הוא עפ"מ עבור G/e:

א. $|E_{\hat{T}'}|=|E_{\hat{T}}|-1=|V|-1-1=|V/e|-1$

ב. וכן, יש להראות כי $\forall x,y\in V/e$ יש מסלול ב$\hat{T}'$:

אם המסלול $\hat{P}$ מx לy ב$\hat{T}$ משתמש בe אזי הוא נראה כך -

$$x\to u\to_e v\to y$$

לאחר הכיווץ על G/e מסלול זה ב$\hat{T}'$ יראה כך: $x\to uv\to y$, כלומר מסלול זה הוא מסלול מx לy ב$\hat{T}'$.

אחרת, המסלול $\hat{P}$ לא משתמש בe, לאחר הכיווץ הוא ישאר אותו מסלול בדיוק.

סה"כ, בכל מקרה קיים מסלול מx לy ־ לכן הגרף קשיר.
סה"כ $\hat{T}'$ קשיר + $|E_{\hat{T}'}| = |V/e| - 1$ ולכן $\hat{T}'$ הוא עץ פורש.
ג. נראה כי

$$w(\hat{T}') = w(\hat{T}) - w(e) < (*)w(T) - w(e) = (**)w(T')$$

$(*)$ כי מההנחה $\hat{T}$ הוא עפ"מ ולכן $w(T) < w(\hat{T})$. $(**)$ כי לפי ההגדרה, T' הוא העץ שהורידו ממנו את e.
סה"כ, קיבלנו $\hat{T}'$ הוא עפ"מ על G/e, בפרט $w(\hat{T}') < w(T')$, בסתירה! כי T' הוא עפ"מ על G/e (מהנתון), ולכן $w(T') \leq w(\hat{T}')$.
מסקנה ־ T הוא בעל משקל קטן ביותר, וסה"כ T הוא MST. כנדרש.
$\blacksquare$.

3.8 האלגוריתם של פרים $(Prim)$

הרעיון באלגוריתם: להתמודד עם הבעיה עם איזה חתך נתחיל ונבחור בכל שלב?". ברעיון זה בוחרים חתך שבו בצד אחד קודקוד אחד, ובצד השני שאר הקודקודים. כמו באלגוריתם הגנרי, נרצה לבחור את הקשת הקלה ביותר (אם יש כמה באותו גודל ־ בוחרים אחת מהן).
באיטרציה הראשונה ־ בוחרים קודקוד שרירותי. לאחר מכן ממשיכים איתו עד הסוף, נניח שהקודקודים הינם $u_1, ..., u_n$. עם הקשתות $u_1 \to u_2$ וכו'. מכווצים את $u_1 \to u_2$. כעת מקבלים u_1u_2 מצד אחד, ומהצד השני של החתך שאר הקודקודים $u_3, ..., u_n$ וכך ממשיכים את האלגוריתם עד שמקבלים קודקוד יחיד $u_1u_2u_3..u_n$.
האלגוריתם הולך להשתמש בתור קדימויות, באשר יש לו שתי פעולות עיקריות: $Init, ExtractMin$.
האלגוריתם $(G = (V, E), w)$ **:**
1. $E_T = \emptyset$

2. עבור כל $u \in V$ בצע:
א. $u.key = \infty$ (המפתח יגיד את משקל הקשת **הקלה** ביותר בחתך שנוגעת בu ועוברת דרך החתך לצד השני ־ נשים לב: מחליפים את הערך של המפתח רק אם הערך קטן יותר מהערך הקודם שהופיע שם.)
ב. $u.\pi = null$ (הפאי יגיד לנו מי הקודקוד בצד השני של הקשת, שהמשקל שלה הוא $u.key$ ־ ושוב, נשים לב שערך הפאי ישתנה רק אם ערך הkey השתנה, וישתנה למי שמחזיק בערך זה.)

3. בחר קודקוד שרירותי $r \in V$ ואתחל $r.key = 0$
4. $Q.init(V)$ ־ אתחל את תור הקדימויות.

5. כל עוד $|Q| \geq 1$ בצע:
א. $u = Q.extract.Min()$ (שקול לבחירת קשת קלה ביותר" ־ מוציאים אותו מהתור)
ב. אם $u.\pi \neq null$ אזי $Add(u, u.\pi) \to E_T$ (חשוב לשים לב ־ בפעם הראשונה שנכנס לסעיף 5 באלגוריתם, נכנס בהכרח במצב בו $u.\pi = null$, כיוון שבסעיף 2ב' אתחלנו את כולם להיות $null$, ולכן לא נוסיף כלום).
ג. עבור כל $v \in ADJ[u]$: (תעבור על השכנים של u)
אם $v \in Q$ (אם עדיין בתוך התור) וגם $w(u, v) < v.key$ אזי ־

1. $v.key = w(u, v)$
2. $v.\pi = u$

6. החזר E_T.

נכונות האלגוריתם: נובעת מנכונות האלגוריתם הגנרי. בכל שלב מסמלצים" כיווץ על הקשת שבחרנו, ובכל שלב מסתכלים על החתך כקודקוד שבחרנו כעת עם כל מה שמקודם, למול מה שנשאר. זה אלגוריתם שמאוד דומה לאלגוריתם הגנרי, ומשם נכונותו.

זמן הריצה:

ניתן לראות ששלבים $3 - 1$ עולים $O(|V|)$ זמן, שכן עוברים על כל הקודקודים.

שלב 4 - תלוי ב$Queue$.

שלב 5א' - גם כן, תלוי בסוג ה$Queue$. אבל, כמה פעמים נבצע הוצאה? $O(|V|)$ הוצאות, כי כל קודקוד יכול לצאת פעם אחת.

נשים לב, שכל קודקוד יכול לצאת מהתור לכל היותר פעם אחת. כאשר קודקוד יוצא מQ, האלגוריתם עובר על כל השכנים שלו, ובמקרה הגרוע כל שכן גורר עדכון מפתח אחד. לכן, שלב 5ג' יעלה $deg(u)$. אבל - כמה פעמים בכלל הם יכולים להקטין את המפתח שלהם (לשנות את ערך הkey, בהכרח להקטין לפי מה שהסברנו)? מס' הפעמים הוא $O(|E|) = 2|E| = \sum_{u \in V} deg(u)$ שהרי בסכימת הדרגות כל קודקוד נספר פעמיים.

לסיכום - זמן הריצה תלוי בQ:

א. מערך: אתחול יעלה $O(|V|)$, ויתבצע פעם אחת. הוצאת המינימום תעלה $O(|V|)$, ותתבצע $|V|$ פעמים - וסה"כ תעלה $O(|V|^2)$, הקטנת מפתח תעלה $O(1)$ ותתבצע $2|E|$ פעמים. סה"כ סיבוכיות הזמן במערך תהיה $O(|V|^2 + |V| + |E|)$, כמו כן $|E| \leq |V|^2$ ולכן סה"כ סיבוכיות זמן הריצה $O(|V|^2)$.

ב. ערימה בינארית/בינומית: אתחול יעלה $O(V)$ ויתבצע פעם אחת, הוצאת המינימום תעלה $O(log|V|)$ ותתבצע $|V|$ פעמים, וכן הקטנת מפתח תעלה $O(log|V|)$ ותתבצע $2|E|$ פעמים. סה"כ סיבוכיות הזמן בערימה תהיה $O(|V|log|V| + |V| + |E|log|V|)$ וכן, כיוון שהגרף קשיר מתקיים $|E| \geq |V| - 1$ בפרט $|E|log|V| \geq |V|log|V|$ ולכן סה"כ סיבוכיות זמן הריצה הינה $O(|E|log|V|)$.

ג. ערימת פיבונאצ'י: אתחול ב$O(|V|)$, הוצאת מינימום $|V|$ פעמים שתעלה $log|V|$, וכן הפחתת מפתח עולה $O(1)$ לשיעורין שתתבצע $2|E|$ פעמים. לכן סה"כ $|V| + |V|log|V| + E$ וסיבוכיות זמן הריצה $O(|V|log|V| + |E|)$.

לא מומלץ להשתמש במערך. נשאלת השאלה מה עדיף - בערימה בינארית או בערימת פיבונאצ'י? תמיד מתקיים הרי כי $|V| \geq |E|$, ולכן ניתן לראות שעדיף להשתמש בערימת פיבונאצ'י בזמן ריצה של $O(|V|log|V| + |E|)$.

3.9 האלגוריתם של קרוסקל

הרעיון: נבחר בכל פעם את הקשת הקלה ביותר **בכל הגרף**, ונוסיף אותה לעץ פורש המינימום. (בפרט, היא תהיה הכי קלה בחתך כלשהו).

הקושי - לדאוג שאין מעגלים / אין לולאה עצמית בזמן הכיווץ.

להלן האלגוריתם:

MST-KRUSKAL(G=(V,E),w):

1. $E_T = \emptyset$

2. for each $u \in V$:
a. make_set(u)
3. for every edge $e=(u,v) \in E$ in increasing order of weights
4. if find_set(u) $\neq$ find_set(v)
a. Add (u,v) to E_T
b. union (u,v)
5. return E_T

כפי שניתן לראות - האלגוריתם משתמש ביוניון פיינד. בתחילה, לכל קודקוד ניצור קבוצה. לאחר מכן, נתחיל מהקשת הקלה ביותר, ונעלה בסדר עולה של משקלים, כך נעבור על כל הקשתות. בכל שלב, נבדוק האם שני הקודקודים שמרכיבים את הקשת נמצאים באותה קבוצה . אם לא - נוסיף את הקשת בניהם לקבוצת הקשתות, ונאחד בין הקבוצות שלהם (נשים לב שיתכן ונוצר מצב בו הקבוצות כרגע הם $\{u,v\},\{w,x\}$ ואנו נדרשים לבדוק את הקשת uw. אין בניהם כרגע קשת ולכן אנחנו נוסיף אותה לMST וכן נאחד בין הקבוצות לקבוצה גדולה $\{u,v,w,x\}$).

למעשה - מה שהאלגוריתם עושה קורה בשלבים $3-4$. אם שני האיברים זרים זה לזה ולא נמצאים באותה קבוצה, חייבים להוסיף קשת שתחבר בניהם בעץ, בשביל שיהיה עץ פורש ובפרט קשיר. אם הם כבר באותה קבוצה, אין צורך להוסיף קשת שתחבר בניהם. מהיכן מגיע המינימום? מהמעבר על הקשתות לפי הקשת הנמוכה ביותר עד לגדולה ביותר. בכל מקרה, נעבור על כל הקשתות - אך כשנגיע למצב שכל האיברים באותה קבוצה ויש קבוצה אחת - סיימנו.

כיצד האלגוריתם בוודאות לא יבחר מעגל? נניח ובחרנו קודקודים $u_1,...,u_k$. נרצה לעבור על קשת $u_1 \rightarrow u_k$. אם נוסיף אותה, בהכרח יווצר מעגל. בשלב 4 אנחנו בדיוק בודקים את זה - לאחר שלב 4 נקבל תשובה שהקודקודים באותה קבוצה ולכן לא נוסיף קשת זאת, וכך לא יווצר מעגל.

זמן הריצה:

לפי האלגוריתם ניתן לראות כי:

מבצעים $O(|V|)$ פעולות $makeSet$, כלומר $O(1)$ וסה"כ $O(|V|)$, וכן מבצעים $|V|-1$ פעולות $union$, שכל אחת עולה $O(log^*|V|)$ ולכן סה"כ $|V|(\alpha|V|)$. כמו כן מבצעים $|E|$ פעמים $findSet$ שעולה $O(\alpha|V|)$ ולכן סה"כ $O(|E|\alpha|V|)$. כמו כן, עלינו למיין את הקשתות E, לכאורה ניתן למיין ב$|E|log|E|$ אך עם מעט יותר מידע על סוג המשקלים ניתן גם ב$O(|E|)$ זמן. סה"כ סיבוכיות זמן הריצה תהיה -

$$O(|V|+(|E|+|V|)\alpha|V|+sort(E))=O(|E|(\alpha|V|)+sort(E))$$

אם $Sort(E)=|E|$ אזי סיבוכיות זמן הריצה הינה $O(|E|(\alpha|V|)+|E|)$, אחרת $(O(|E|(\alpha|V|)+|E|log|E|)$

בהשוואה בין זמני הריצה של פרים וקרוסקל - בדרך כלל קרוסקל ינצח. אך אם מס' הקשתות גדול יחסית, ממש גדול יחסית - אזי עדיף להשתמש בשל פרים. אחרת, של פרוסקל ינצח.

3.10 תכונת המעגלים הכבדים של MST (תרגול)

למה 1. יהי $G = (V, E)$ גרף לא מכוון עם פונקציית משקל על הקשתות $w : E \to \mathbb{R}$. יהי C מעגל בG כך ש$e \in C$ היא קשת כבדה במעגל. אזי, קיים MST שלא מכיל את e.

הוכחה: נניח בשלילה שקיים MST שמכיל את $e = (x, y)$. יהי T כנ"ל אשר $e \in T$. נביט בגרף שמתקבל מהסרת הקשת e מT. כלומר $T \backslash \{e\}$. מתקבל גרף עם $|V| - 2$ קשתות (קודם לכן היה $|V| - 1$ כי T הינו עץ) ולכן הוא בהכרח אינו קשיר. יתרה מזאת, הקודקודים x וy נמצאים ברכיבי קשירות שונים בגרף זה. נסמן את רכיב הקשירות שמכיל את x בתור S ונסתכל על החתך $(S, V \backslash S)$ בG. אנו יודעים כי המעגל C מכיל מסלול מx לy בG ללא הקשת e, כלומר קיימת קשת $e' = (u, v)$ ב$C \backslash \{e\}$ שחוצה את החתך $(S, V \backslash S)$. בה"כ נניח כי $v \in V \backslash S$ ו$u \in S$. נביט בגרף $T' = (T \backslash \{e\}) \cup \{e'\}$. נרצה לטעון T' עץ. בT' ישנם בדיוק $|V| - 1$ קשתות ולכן נטען כי T' קשיר וזה יהיה מספיק.

יהיו $a, b \in V$ קודקודים. אם שניהם באותו צד של החתך אזי יש ביניהם מסלול כי T קשיר. אחרת נניח שהם בצדדים שונים של החתך, בה"כ $a \in S, b \in V \backslash S$ אזי נוכל לבנות מסלול מa לb כך: $a \rightsquigarrow u \rightsquigarrow v \rightsquigarrow b$ באשר מובטח לנו שיש מסלול מa לu ומv לb כי כל צד של החתך קשיר. אם כן הוכחנו כי T' עץ. נרצה לראות שמשקלו קטן או שווה ממשקלו של T בכדי לקבל סתירה (כי T עפ"מ ובפרט משקלו מינימלי).

$$w(T') = w(T) + w(e') - w(e) \leq w(T)$$

כיוון שe הייתה קשת כבדה במעגל C והקשת e' מקיימת $w(e') \leq w(e)$. סה"כ סתירה לכך שT הוא MST במשקל מינימלי.

מדוע אנחנו זקוקים ללמה זו? מכאן עולה רעיון אלגוריתמי אלטרנטיבי לרעיון שראינו בהרצאה אודות MST. יהי מעגל C בG, אזי אנו יודעים שקיים עפ"מ ללא קשת כבדה ביותר בC, לכן ניתן להוריד את הקשת הזו מהגרף ולהמשיך ברקורסיה על הקשתות שנותרו. קרוסקל, הציע במאמרו המקורי גם את האלגוריתם הבא. שנקרא גם "אלגוריתם מחיקה כפולה".

$reserve - delete\ Algo(G = (V, E), w)$:

א. מיין את E בסדר יורד, יהי הסדר לאחר המיון: $e_1, ..., e_{|E|}$

ב. לכל $i = 1$ עד $i = |E|$

1. מחק את e_i
2. אם הגרף ללא e_i אינו קשיר, החזר את e_i לגרף.

ג. החזר את קבוצת הקשתות שלא נמחקו במהלך ריצת האלגוריתם.

נוכיח את האלגוריתם באמצעות למה 1 ובאמצעות הלמה הבאה:

למה 2. תהי $F \subseteq E$ קבוצת הקשתות שנשארה בגרף בסוף הלולאה של האלגוריתם $reserve - delete$, אזי קיים עפ"מ $T = (V, E_T)$ של G באשר $E_T \subseteq F$.

הוכחה: ההוכחה תהיה באינדוקציה על מס' האיטרציות של הלולאה.

בסיס: קל לראות כי לפני תחילת הלולאה, הטענה נכונה באופן ריק. קבוצת הקשתות $F = E \subseteq E$ ממש ואכן קיים עפ"מ T של G שצלעותיו מוכלות בE, שכן קיים עפ"מ בכל גרף קשיר.

צעד: נניח שהטענה נכונה עבור קבוצת קשתות $F \subseteq E$ בסוף איטרציה מסוימת של הלולאה, כלומר קיים עפ"מ $T = (V, E_T)$ של G באשר $E_T \subseteq F$. נסתכל על קבוצת הקשתות $F' \subseteq E$ שמתקבלת בסיום האיטרציה הבאה של הלולאה ונרצה להוכיח כי קיים עפ"מ $T' = (V, E_{T'})$ של G כך ש$E_{T'} \subseteq F'$. נחלק למקרים:

א. אם לא הוסרה קשת, אזי בהכרח $F' = F$ ולפי הנחת האינדוקציה הטענה מתקיימת.

ב. הוסרה קשת $e \in F$, מהגדרת האלגוריתם הקשת e הייתה חלק ממעגל C בF, נראה כי כל קשת אחרת במעגל לא נבחנה עד כה במהלך ריצת האלגוריתם אחרת האלגוריתם היה מסיר אותה (בהכרח סידרנו את קשתות הגרף לפי גודלן, בוודאות תופיע קודם כל הקשת הכבדה ביותר במעגל),

שכן הגרף שמושרה ישאר קשיר לאחר הסרת קשת יחידה מהמעגל C. מכאן נקבל כי e היא קשת כבדה במעגל. מכאן, לפי למה 1, בהכרח קיים MST של G' שלא מכיל את e, בשילוב עם הנחת האינדוקציה שאומרת שעפ"מ של G' הוא גם עפ"מ של G נוכל להסיק כי קיים עפ"מ $T' = (V, E_{T'})$ של G כך ש-$E_{T'} \subseteq F'$ וכן $e \notin E_{T'}$.

3.11 השפעת סדר מיון הקשתות על הפלט בהרצת האלגוריתם של קרוסקל

טענה. לכל עפ"מ $T = (V, E_T)$ של G קיים סדר של E שנסמנו π_T שהוא מיון של הקשתות ע"פ משקלן בסדר עולה,כך שההרצה של האלגוריתם של קרוסקל על G בהתאם ל-π_T תחזיר את T.

הערה חשובה. כאשר הוכחנו את הטענה הסתכלנו על עפ"מ כלשהו T של G והראינו עבורו סדר מיון כך שאלגוריתם קרוסקל מוציא את T כפלט. טעות נפוצה בשאלות מסוג זה היא נסיון להסתכל על עפ"מ שהוא פלט של אלגוריתם קרוסקל (או כל אלגוריתם שפותר את הבעיה לצורך העניין) ונסיון לטעון טענות לגביו. שימו לב כי עפ"מ של גרף הוא אובייקט מתמטי של הגרף, ויכולים להיות כמה עפמי"ם שונים. אלגוריתמים כמו פרים קרוסקל וכיוב הם תהליכים חישוביים שמוצאים אובייקט מתמטי שכזה (ביעילות), אך הוא ספציפי מבין כמה אפשריים, ולכן כאשר מתבקשים להוכיח טענה על אובייקט מתמטי שרירותי אסור לנו להניח שהוא פלט של אלגוריתם כזה או אחר (זהו מקרה פרטי של דגש הדורש מכם לשים לב כי ישנן שאלות בהן הטענה היא טענת לכל ולא טענת קיים, וכן להיפך).

מסקנה, לפי הטענה אם ידוע כי G מכיל קשתות ממשקלים שונים, בהכרח קיים סדר מיון יחיד ולכן קיים MST יחיד!

הוכחה. יהי T עפ"מ של G, נגדיר את π_T באופן הבא: π_T יהיה מיון חוקי בסדר עולה של E, <u>כאשר אם ישנן קשתות במשקל שווה ניתן עדיפות במיון לקשתות שנמצאות ב-T.</u>
כעת נרצה להוכיח שריצת האלגוריתם של קרוסקל על קשתות G בהתאם ל-π_T כפי שהוגדר למעלה, מחזירה את T כפלט:
נב"ש שלא, אזי האלגוריתם של קרוסקל מוציא כפלט עפ"מ אחר $\hat{T}$, כך ש: $\hat{T} \neq T$.
נביט בקשת הראשונה (ע"פ סדר המיון π_T) $e = (u, v)$ כך ש: $e \in \hat{T}, e \notin T$.
בגלל ש-T עפ"מ ובפרט עץ אזי קיים מסלול יחיד ב-T מ-u ל-v. נסמנו: $P_T = e_1, e_2, \ldots, e_k$.
נשים לב כי לפחות אחת מהקשתות ב-P_T לא נמצאת ב-$\hat{T}$ (אחרת קיים מעגל: $e_1, e_2, \ldots e_k, e$ ב-$\hat{T}$, בסתירה להיותו עץ). נסמן את הקשת הזו: $e_i = (x, y)$, ולכן $e_i \notin \hat{T}, e_i \in P_T \subseteq T$.

הבחנה: $w(e) < w(e_i)$.
הוכחה: נב"ש $w(e) \geq w(e_i)$. לפי הגדרת π_T, e_i מופיעה לפני e ב-π_T (מפני ש- $e_i \notin \hat{T}, e_i \in T$ וגם $w(e) \geq w(e_i)$).
מכיוון ש-$e_i \notin \hat{T}$, נסיק שכאשר האלגוריתם של קרוסקל בחן את e_i, e_i סגרה מעגל עם הקשתות שנבחרו לפניה ל-$\hat{T}$, ואותן הקשתות נבחרו ל-$\hat{T}$ לפני ש-e נבחרה ל-$\hat{T}$. ולכן כל הקשתות האלה נמצאות ב-T (לפי הגדרת e). מכאן שקיים מעגל ב-T (שמכיל את הקשת e_i) בסתירה לכך ש-T עץ.

כעת, נבנה מ-$T = (V, E_T)$ עץ אחר $T' = (V, E_{T'})$ שמשקלו קטן יותר וזו תהיה סתירה לכך ש: T עפ"מ של G:
נגדיר: $E_{T'} = \left(E_T \setminus \{e_i\}\right) \cup \{e\}$.
טענה: T' הוא עץ:
ברור כי $|E_{T'}| = |E_T| = |V| - 1$ ולכן מספיק שנראה כי תכונת הקשירות מתקיימת ב-T'.
נראה כי קיים מסלול ב-T' בין כל זוג קודקודים בעץ:
יהיו $s, t \in V$ בגלל ש-T עץ אנחנו יודעים שקיים מסלול יחיד בין s, t ב-T. נחלק למקרים:

מקרה 1: המסלול בין s ל-t ב-T לא משתמש בקשת e_i.
אזי אותו מסלול קיים בעץ T'.

מקרה 2: המסלול בין s ל-t ב-T משתמש בקשת e_i.
נניח כי המסלול הוא מהצורה הבאה: t $P_{s,t} = s \rightsquigarrow_{P_{s,x}} x \rightarrow y \rightsquigarrow_{P_{y,t}} t$.
תחילה נשים לב כי המסלולים $P_{s,x}, P_{y,t} \subseteq T'$ מפני שאינם משתמשים בקשת e_i.
כעת, נזכור שקיים ב-T מסלול מ-u ל-v, P_T שהקשת e_i נמצאת בו, אזי ניתן לפצל אותו לשלושה מסלולים:
$P_{u,x}$ החלק במסלול P_T מהקודקוד u עד לקודקוד x.
הקשת $e_i = (x, y)$.
$P_{y,v}$ החלק במסלול P_T מהקודקוד y עד לקודקוד v.
נשים לב כי $P_{u,x}, P_{y,v} \subseteq T$ וגם $e_i \notin P_{u,x}, P_{y,v}$ ולכן $P_{u,x}, P_{y,v} \subseteq T'$.
נבנה מסלול ב-T' שלא משתמש בקשת e_i בצורה הבאה:
$P'_{s,t} = s \rightsquigarrow_{P_{s,x}} x \rightsquigarrow_{P_{x,u}} u \rightarrow v \rightsquigarrow_{P_{v,y}} y \rightsquigarrow_{P_{y,t}} t$.

בסה"כ הראינו כי קיים מסלול בין כל זוג קודקודים ב-T' וגם כי $|E_{T'}| = |V| - 1$ לכן נסיק כי T' הוא עץ.
כעת כאשר הוכחנו כי T' הוא עץ נראה סתירה למינימליות של T:
מההבחנה אנחנו יודעים ש: $w(e) < w(e_i)$.
ולכן נקבל כי: $w(T') = w(T) + (w(e) - w(e_i)) < w(T)$.
בסתירה להיותו של T עפ"מ. □

4 הרצאות $3+4$ - $shorts\ path - SSSP$

כיצד מודדים מהו המסלול הקצר ביותר?

אם הגרף אינו ממושקל: עלות המסלול היא מס' הקשתות במסלול= אורך המסלול.
אם הגרף כן ממושקל: עלות של מסלול = סכום משקלי הקשתות שעל המסלול.

הגדרה: עבור $u, v \in V$ נסמן את העלות המינימלית של מסלול מu לv ב$\delta(u,v)$.
יהיה $P=(u, v_1, ..., v_{k-2}, v)$ המסלול הקצר ביותר בין u לv (אם קיים). אזי,
אם G ממושקל:

$$\delta(u,v) = \sum_{v \in P} w(v)$$

אם G אינו ממושקל:

$$\delta(u,v) = |P| = k-1$$

אם לא קיים מסלול בין u לv נגדיר:

$$\delta(u,v) = \infty$$

הגדרה: מסלול מu לv שעלותו היא $\delta(u,v)$ יקרא מסלול קצר ביותר.
הערה: יתכן שבין זוג קודקודים יש יותר ממסלול אחד קצר ביותר.

הערה: רוב האלגוריתמים שנראה בהרצאה יהיו עבור גרף מכוון. כיצד זה פותר את הבעיה עבור גרף שאינו מכוון? אם האלגוריתם יודע לפתור את הבעיה עבור גרף מכוון, נוכל להמיר כל גרף לא מכוון למולטי גרף מכוון: כך שכל קשת $a \longleftrightarrow b$ תתורגם לשתי קשתות $a \to b, b \to a$.
הערה: ישנם מקרים בהם יש אלגוריתם יותר מהיר עבור גרף לא מכוון.
הערה: ניתן להשתמש בפתרון עבור המקרה הממושקל למקרה הלא ממושקל, אם נגדיר פונקציית משקל קבועה. למשל כל הקשתות בעלות אחד.

4.1 בעיית מציאת המסלול הקצר ביותר

לבעיה זו יש מס' גרסאות. נשים לב כי הן מדורגות כעת מהקלה לקשה.

1. **זוג קודקודים** -
קלט: $G=(V,E)$ וזוג קודקודים $v, u \in V$.
פלט: לחשב את $\delta(u,v)$ ואולי אף למצוא מסלול מu לv שהוא קצר ביותר.
2. **מקור יחיד** $(Singal\ Source\ Shorts\ Paths\ (SSSP))$ -
קלט: $G=(V,E)$ וקודקוד $s \in V$ שיקרא קודקוד מקור.
הפלט: לחשב עבור כל $v \in V$ את $\delta(s,v)$ ואולי גם למצוא מסלול קצר ביותר מs לכל $v \in V$.
3. **כל הזוגות** $(All\ Pairs\ Shorts\ Paths\ (APSP))$ -
קלט: $G=(V,E)$
פלט: לכל $u, v \in V$ להחזיר את $\delta(u,v)$ ואולי אף למצוא את המסלול הקצר ביותר לכל $u, v \in V$.

הערה: בעיה 1 מוכלת בתוך בעיה 2, ועם זאת כפי שנראה בהמשך לא קיים אלגוריתם שפותר אותה יותר טוב מאת בעיה 2. כלומר, לא מצאו עדיין אלגוריתם יעיל יותר בסיבוכיות עבור בעיה 1. כמו כן, בעיה 2 מוכלת בבעיה 3 - אך כן זמן הריצה של בעיה 2 טוב יותר משל 3.

4.1.1 איך נראה פתרון בכל אחד מהגרסאות כשמחפשים את המסלול?

זוג יחיד: מסלול $P=(u,v_0,.....,v_{k-1},v)$, שיעלה $O(|V|)$ זכרון.
מקור יחיד: נאיבית, אפשר להחזיר $|V|$ מסלולים שונים, אחד עבור כל קודקוד מטרה. כלומר:

$$p_1=(s,....,v_1)$$

$$p_2=(s,...,v_2)$$

$$..$$

$$p_n=(s,...,v_n)$$

מה עלות הזכרון בפתרון זה? $\sum_{i=1}^{n}|P_i|\le|V|\times max\{|P_i|\}\le|V|\times|V|=O(|V|^2)$
כעת נראה שישנה אפשרות להחזיר את הפתרון בצורה שתשתמש בפחות מקום. לשם כך נשתמש בלמה החשובה מאוד הבאה -

למה 1: תת מסלול של מסלול קצר ביותר הוא גם מסלול קצר ביותר. כלומר, יהיה מסלול $P=(v,w_1,..w_r,x,...,y,z_1,..,z_t,u)$ קצר ביותר. אזי, המסלול בין x לy שמוכל במסלול P הוא גם הקצר ביותר.

הוכחה: יהי המסלול הקצר ביותר בין הקודקודים v וu: $P=(v,..,x,p_1,...,p_k,y,..,u)$. נב"ש כי המסלול בין x לy $P_{xy}=(x,p_1,...,p_k,u)$ אינו הקצר ביותר. כלומר, קיים מסלול $P_{xy2}=(x,u_1,....,u_m,u)$ בין x לy כך ש$m<k$, כלומר $|P_{xy2}|<|P_{xy}|$. אזי, נסתכל על המסלול $P'=(v,w_1,..,w_r,P_{xy2},z_1,..,z_t,u)$ קיבלנו מסלול בין v לu המקיים

$$|P'|=|P|-|P_{xy}|+|P_{xy2}|<|P|$$

בסתירה לכך שP הינו המסלול הקצר ביותר בין v לu.

כעת, נחזור לדון בשמירת הזכרון: בעת שמירת מסלול כלשהו, למשל $s\to v_4=(s,v_2,v_{14},v_3,v_{90},v_4)$ אנחנו שומרים מסלולים קצרים ביותר נוספים: בין $v_2\to v_{90}$ למשל, לפי הלמה שהוכחה לעיל גם הוא מסלול קצר ביותר.
כמו כן, נשים לב כי נוכל לקבל למשל שני מסלולים: $(s,v_{10},v_5,v_6,v_2),(s,v_{10},v_5,v_8,v_4,v_9)$ ולשרשר אותם למסלול יחיד כך:

$$(s,v_{10},v_{5\to v_6,v_2}^{\to v_8,v_4,v_9})$$

כלומר, ליצור צורה של עץ שs שורשו. סה"כ זו תהיה הטכניקה -
ניתן לייצג מסלולים קצרים ביותר מS לכל שאר הקודקודים בגרף באמצעות עץ מסלולים קצרים ביותר.

נשים לב - העץ לא מכיל את כל המסלולים הקצרים ביותר מs בגרף, כלומר: לכל $v \in V$ יהיה קיים מסלול קצר ביותר שייוצג בעץ מ$s \rightarrow v$, אך יתכן שקיימים שני מסלולים כאלו קצרים ביותר באותו משקל והעץ יבחר אחד מהם בדיוק שיופיע בו.

מסקנה: יתכן שישנם כמה עצי מסלולים קצרים ביותר.

לסיכום - בגרסת מקור יחיד אנחנו נחזיר **עץ מסלולים קצרים ביותר** שעלותו תהיה כגודל מס' הקודקודים בו, $O(|V|)$. נשים לב - קודקוד שהינו ראש העץ יהיה בלולאה עצמית עם עצמו, קודקודים שאין אליהם מסלול יסומנו אל $null$.

כל הזוגות $(APSP)$**:** במצב זה נרצה להחזיר מטריצה A בגודל $|V| \times |V|$, כשנרצה להחזיר את $\delta(v,u)$ אנחנו נייצג זאת במטריצה ע"י $A_{ij} = \delta(v_i, v_j)$, סה"כ יעלה $O(|V|^2)$ זכרון.
אם נהיה מעוניינים במסלולים - נרצה $|V|$ עצי מסלולים קצרים ביותר, וסה"כ $O(|V|^2)$ מקום.

4.2 אלגוריתם $SSSP - BFS$ במקרה הלא ממושקל

BFS היא סריקה לרוחב של העץ לפי רמות, מבצעים אותה באמצעות תור קדימויות כפי שראינו בקורס מבני נתונים.
האלגוריתם סורק את הצמתים בסדר שנקבע על פי מרחקם מהצומת ההתחלתי. **אלגוריתם זה מטפל ב$SSSP$ במקרה הלא ממושקל.** וכן, אורך המסלול נספר לפי מס' הקשתות.

האלגוריתם מאחסן 3 סוגי מידע לכל קודקוד:
1. $d[u]$ - אומדן לגבי $\delta(s,u)$. בסיום הריצה יתקיים $d[u] = \delta(s,u)$
2. $\pi[u]$ - כלי עזר לחישוב המסלולים הקצרים ביותר מs. בסיום הריצה הוא מצביע לאבא של u בעץ המסלולים הקצרים.
3. $Color[u]$- מעין מזהה:
   א. $w(hite)$= האלגוריתם עוד לא ביקר בu.
   ב. $g(ray)$= האלגוריתם ביקר בu ולא טיפל בו.
   ג. $b(lack)$= האלגוריתם סיים לטפל בu.

4.2.1 האלגוריתם $BFS=(G=(V,E),s)$

```
BFS(G = (V, E), s)
1  for each u ∈ V
2      d[u] ← ∞
3      π[u] ← NULL
4      color[u] ← w
5  d[s] ← 0
6  color[s] ← g
7  Q.Enqueue(s)
8  while Q ≠ ∅
9      u ← Q.Dequeue()
10     for each v ∈ ADJ[u]
11         if color[v] = w
12             color[v] ← g
13             d[v] ← d[u] + 1
14             π[v] ← u
15             Q.Enqueue(v)
16     color[u] ← b
```

האלגוריתם מאתחל בתחילה את $d[u]$ להיות אנסוף, את π להיות $null$ ואת כל הצבעים להיות w ־ לא ביקר. לאחר מכן נאתחל את s. נשים לב כי $d[s]=0$ כיוון שאורך המסלול הקצר ביותר מs לעצמו הוא אפס. תהליך האתחול מתרחש עד לשלב 7.

אנחנו משתמשים בתור $FIFO$, ומכניסים אליו את s. כעת כל עוד התור לא ריק אנחנו מבצעים: מוציאים מהתור את האיבר הראשון, עוברים על כל השכנים של הקודקוד שהוצאנו, אם הצבע שלהם לבן משמע לא ביקרנו אותו עוד, נסמן את הקודקודים באפור, נגדיר את הd שלהם להיות הd של הקודקוד שהוצאנו (שהיה השכן שלהם) ועוד אחד ־ כי יש קשת שנוספה למסלול,וכן נגדיר את אבא של כל הקודקודים האלו להיות u (הקודקוד שהוצאנו), לבסוף נכניס את כל השכנים הללו לתור.

בסיום, נגדיר את הצבע של הקודקוד שהוצאנו כb, סיימנו לטפל בו. וכעת, נעבור לטפל בקודקוד הבא בתור. כך ־ עד שהתור יתרוקן.

זמן הריצה: האתחול עולה $O(|V|)$ זמן, לאחר מכן מבצעים לולאה ־ נשים לב כי במהלך הלולאה אף קודקוד לא נצבע בלבן, ולכן כל קודקוד נכנס ויוצא מהתור לכל היותר פעם אחת. ולכן הפעולות $dequeue, enqueue$ מתבצעות פעם אחת לכל קודקוד. ומכאן, שכל הלולאה של ה$while$ מתבצעת לכל היותר $O(|V|)$ פעמים. באשר לעלות לולאת for על קודקוד u היא $O(deg(u))$, ולכן סה"כ זמן הריצה יהיה

$$|V|+\sum_{u\in U} deg(u)=|V|+2|E|=O(|V|+|E|)$$

וקיבלנו זמן לינארי. כמו כן נשים לב כי אסור להניח $|V|\le|E|$, אנחנו מדברים על גרף כללי G.

נשים לב: π מגדיר את עץ המסלולים הקצרים, כלומר ריצת BFS יכולה גם להחזיר לנו את עץ המסלולים הקצרים ביותר. באמצעות ערך π ניתן לבנות את עץ זה.

הערה: $O(|V|+|E|)$ הוא חסם תחתון לגודל הקלט ולכן אלגוריתם BFS הוא האופטימלי לפתרון הבעיה.

4.2.2 נכונות של BFS

המטרה היא להוכיח שבסוף ריצת $BFS(G,s)$ מתקיים כי $\forall u \in V : d[u]=\delta(s,u)$

למה 2: למת אי שוויון המשולש. יהי $G=(V,E)$ לא ממושקל, ויהי $s \in V$, ויהי $(u,v) \in E$, אזי

$$\delta(s,v) \leq \delta(s,u)+1$$

(כלומר, בהינתן המסלול $s \to ... \to u \to v$, המסלול הקצר ביותר לעבור מs אל v חסום במסלול הקצר ביותר לעבור מs אל u ועוד מעבר על הקשת $e=(u,v)$, נשים לב שזה אפשרות למסלול ויתכן שיש מסלול טוב יותר קצר יותר. מדברים על חסם בלבד!).

הוכחה: נחלק למקרים.

א. אם אין מסלול בין s לu: אזי בהכרח $\delta(s,v) \leq \infty + 1$, כנדרש.

ב. אם יש מסלול בין s לu: זה גורר שמתקיים מסלול בין s לv: המסלול בין s לu בתוספת הקשת (u,v). במקרה זה, עלות של המסלול הקצר ביותר מs לv לא יכול להיות גדול יותר מעלות המסלול מs לu בתוספת העלות של הקשת (u,v) כיוון שאחד המסלולים האפשריים מs לv הוא המסלול שבנינו כנ"ל, ולכן המסלול הקצר ביותר בוודאות יהיה או זה, או באורך קטן מזה. מסקנה $\Longleftarrow \delta(s,v) \leq \delta(s,u)+1$.

למה 3: לאחר הרצת $BFS(G,s)$ לכל $v \in V$ מתקיים $d[v] \geq \delta(s,v)$.

הוכחה: נבצע אינדוקציה על מס' פעולות $enqueue$ שיתבצעו באלגוריתם. נסמנם n.

בסיס: $n=1$, עבור s יתקיים $d[s]=0$ ועבור שאר הקודקודים u יתקיים $d[u]=\infty$ ואכן מתקיים התנאי.

צעד: נניח שהטענה נכונה עבור $n-1$ פעולות הכנסה. נוכיח לn.

שינוי הערך $d[u]$ יכול להתבצע רק בהלך מעבר על השכנים של u שצבעם כעת לבן. אם כן, יספיק להוכיח שעבור כל שכן של u, v שצבעו לבן יתקיים $d[v] \geq \delta(s,v)$

יהי v קודקוד כנ"ל. לאחר העדכון יתקיים -

$$d[v]=d[u]+1 \geq \delta(s,u)+1 \geq (*)\delta(s,v)$$

כאשר $(*)$ נכון לפי למת אי שיוויון המשולש והנחת האינדוקציה.

למה 4: בכל זמן באלגוריתם אם $Q=(v_1,...,v_r)$ מתקיימות שתי תכונות:

א. $d[v_1] \leq d[v_2] \leq ... \leq d[v_r]$

ב. $d[v_r] \leq 1+d[v_1]$ (כלומר לא יתכן שבתור בו זמנית ישנם יותר משתי שכבות במקביל).

הוכחה:

נוכיח את הטענה באינדוקציה על סדרת פעולות $enqueue, dequeue$

בסיס: תור מכיל רק את s ולכן בצומת אחד אכן מתקיימות שתי הלמות.

צעד:

1. $dequeue$: כעת התור נראה כך $Q=(v_2,...,v_r)$, אכן א' מתקיים כי אי השוויון בפרט יכול להתחיל מ$d[v_2]$.

אם כן,

$$d[v_r] \leq_{(*)} d[v_1] + 1 \leq_{(**)} d[v_2] + 1$$

כאשר $(*)$ מהנחת האינדוקציה ו$(**)$ מא'.

2. $enqueue$: כעת התור נראה $Q = (v_1, .., v_r, v_{r+1})$. נסמן v_0 את הצומת שיצא מהתור ובגינו נכנס v_{r+1}. מכאן $d[v_{r+1}] = d[v_0] + 1$, נפצל למקרים:

א. אם v_1 היה בתור בעת שv_0 יצא אזי בהכרח לפי א' $d[v_0] \leq d[v_1]$ ומכאן נקבל כי $d[v_{r+1}] = d[v_0] + 1 \leq d[v_1] + 1$ ומכאן שתכונה 2 מתקיימת.

- אם v_r היה בתור כשv_0 יצא אזי $d[v_r] \leq d[v_0] + 1 = d[v_{r+1}]$ ואז גם תכונה 1 מתקיימת.

- אם v_r לא היה בתור כשv_0 יצא אזי אזי v_r נכנס בגין v_0 ולכן $d[v_0] + 1 = d[v_r] = d[v_{r+1}]$ ו1 מתקיים גם כאן.

ב. אם v_1 לא היה בתור בעת שv_0 יצא, אזי כל הצמתים נכנסו בגין v_0 והערך $d[v_i]$ שלהם הוא $d[v_0] + 1$ ולכן גם 1 וגם 2 מתקיימים.

כנדרש.

מסקנה 5: אם $u \in V$ יוצא מQ לפני $v \in V$ תוך כדי ריצת $BFS(G, s)$ אזי $d[u] \leq d[v]$. (כלומר, ערכי הd של הקודקודים יכולים רק לעלות לאורך ההרצה).

הוכחה: נובע ישירות מלמה 4.

למה 6 (הוכחת נכונות BFS) : לאחר הרצת אלגוריתם BFS על גרף מכוון $G = (V, E)$ וקודקוד $s \in V$, מתקיים:

$$\forall u \in V : d[u] = \delta(s, u)$$

הוכחה: נניח בשלילה כי קיים צומת אחד לפחות עבורו אין שוויון. נסמן u הצומת עם $\delta(s, u)$ הכי קטן עבורו זה מתקיים. כלומר $d[u] \neq \delta(s, u)$. עבור v הקודם לu במק"ב מs לu בהכרח קיים שוויון $d[v] = \delta(s, v)$ (בהכרח $u \neq s$ כי $d[s] = 0$ לפי האלגוריתם ואכן $\delta(s, s) = 0$).

כאשר v יצא מהתור, הוא עובר על כל שכניו. אם u היה לבן, הוא היה מקבל את הערך הנכון של $d[u] = \delta(s, v) + 1$, ולכן u אינו לבן כלומר u קדם לv בתור. ולפי מסקנה 5: $d[u] \leq d[v] = \delta(s, v) < \delta(s, u)$, קיבלנו $d[u] < \delta(s, u)$ בסתירה ללמה 3.

4.3 אלגוריתם סריקת DFS

בהינתן גרף, סריקת DFS על הגרף היא סריקה לעומק.

הסריקה עוברת על כל הקודקודים של הגרף באופן הבא: כל עוד יש קודקוד שלא ביקרנו בו - נבקר בו. כאשר מבקרים בקודקוד, בודקים אם יש מישהו משכניו שטרם ביקרו בו - ואם כן מבקרים בו בקריאה רקורסיבית.

בהינתן גרף מכוון $G = (V, E)$ האלגוריתם DFS סורק את כל הקודקודים.

בדומה לBFS, האלגוריתם משייך לכל קודקוד צבע שמסמל את מצב הקודקוד:

b- שחור, ביקרנו וסיימנו לטפל בקודקוד.

w- לבן, טרם ביקרנו בקודקוד

g- אפור, ביקרנו אך טרם סיימנו לטפל בקודקוד.

בנוסף לכל קודקוד $u \in V$ שומר האלגוריתם שלושה ערכים:

א. $d(u)$ - זמן הגעה (צביעה באפור)

ב. $f(u)$- זמן עזיבה (צביעה בשחור)

ג. $\pi(u)$- קודקוד קודם. השדה π מגדיר לכל קודקוד קודקוד קודם, אשר אם נסתכל עליו כ"אבא" של הקודקוד הראשון נקבל אוסף של עצים, המכונה גם יער העומק. נשים לב שיער העומק

תלוי בריצה ספציפית של האלגוריתם הDFS ולגרף נתון יכולים להיות מס' יערי עומק שונים.

להלן האלגוריתם:

DFS(G)

1. for $u \in V$
   (א) $color[u] \leftarrow w, \pi[u] \leftarrow$ NULL
2. $t \leftarrow 0$
3. for $u \in V$
   (א) if $color[u] = w$
   (ב) then DFS-Visit(u)

DFS-Visit(u)

1. $color[u] \leftarrow g$
2. $d[u] \leftarrow t \leftarrow t + 1$
3. for $v \in adj(u)$
   (א) if $color[v] = w$
   (ב) then DFS-Visit(v)
      i. $\pi[v] \leftarrow u$
4. $color[u] \leftarrow b$
5. $f[u] \leftarrow t \leftarrow t + 1$

האלגוריתם ישמור כמשתנה גלובלי את t: שיאותחל בהתחלה לאפס ויגדל בהתאם לאלגוריתם. t ישמש לציין את זמן הכניסה הנוכחי. האלגוריתם הוא אלגוריתם רקורסיבי שעובר לעומק על כל הגרף.

הגדרה: יער העומק מוחזר אלינו כמערך π. נשים לב כי $null$ במערך מציין שורש של עץ. יער העומק מכיל עצים (מסלולים ארוכים) שונים.

סיבוכיות זמן ריצה: נראה כי כל צומת נקראת עם $DPS - Visit(u)$ פעם אחת בדיוק. בתחילה, האתחול עולה $O(|V|)$ וסה"כ מבצעים לולאה של $O(|V|)$ כפול $1 + deg(u)$ וסה"כ נקבל $\sum_{v \in V}(1 + deg(u)) = O(|V| + |E|)$ זמן הריצה של האלגוריתם. זמן הריצה לינארי.

הערה חשובה: לא מתואר כיצד אנחנו רצים על האלגוריתם, ולכן כל סדר של הקודקודים (כל עוד אנחנו יודעים אותם מראש) הינו חוקי. יתכנו $|V|!$ פרמוטציות אפשריות לריצת DFS.

משפט הסוגריים: בכל DFS על גרף $G = (V, E)$ מכוון או שלא מכוון,
עבור כל זוג קודקודים $v, u \in V$ מתקיים אחד מהשלושה הבאים (עבור האינטרוול של זמן ההתחלה עד זמן הסיום):

1. $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$
2. $[d[u], f[u]] \subseteq [d[v], f[v]]$
3. $[d[u], f[u]] \supseteq [d[v], f[v]]$

הערה. אם $v \neq u$ אזי בסעיפים 2 ו3 מדובר על הכלה ממש (לא שוויון)

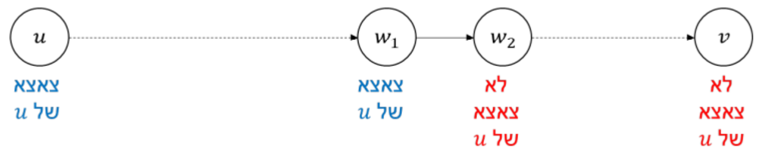
מסקנה: קודקוד v הוא צאצא של קודקוד u ביער העומק אמ"מ מקרה 3 מתקיים.

משפט המסלול הלבן: $v \in V$ הוא צאצא של $u \in V$ ביער העומק אמ"מ בזמן $d[u] - 1$ קיים בG מסלול מu לv שכל קודקודיו צבועים בלבן.

הוכחה:

$\Longleftarrow$: אם v צאצא של u ביער העומק, הרי קיים מסלול מv לu ביער העומק וכפרט בגרף G. כל הקודקודים על המסלול הזה הם צאצאים של u, ולכן ע"פ משפט לכל הקודקודים x הללו מתקיים $d[u] < d[x]$, כולם צבועים בלבן, לכן הראינו מסלול לבן מu לv. (נשים לב - המסלול הוא לבן וניתן לראות זאת באלגוריתם שיש קריאה של DFS אפ"מ הקודקוד בלבן).

$\Longrightarrow$: נניח כי בזמן $d[u] - 1$ קיים בG מסלול מu לv שכל קודקודיו צבועים בלבן. נניח בשלילה כי v אינו צאצא של u ביער העומק בזמן $d[u] - 1$. נסתכל על המסלול הנ"ל מu לv ויהי w_1 הקודקוד הכי קרוב לu על המסלול שהוא צאצא של u וw_2 הקודקוד העוקב במסלול (שבהכרח אינו צאצא של u - ראה



איור 1: המחשת ההוכחה

באיור), כיוון ש-w_1 צאצא של u מתקיים $d[u] \le d[w_1], f[w_1] \le f[u]$ בזמן שביקרנו ב-w_1 בוודאות האלגוריתם בודק את צבעו של w_2.
מכיוון ש-w_2 לבן (כי אחרת זה אומר שביקרנו בו במהלך הסריקה מ-u - והוא צאצא שלו), שהרי אנחנו עדיין בתוך סריקה זו) אזי בהכרח נבקר ב-w_2 וממילא $d[u] \le d[w_1] < d[w_2] < f[w_2] <$ $f[w_1] \le f[u]$ בסתירה להנחתנו. מכאן בהכרח v צאצא של u ביער העומק.

4.3.1 סיווג קשתות

ניתן לסווג את הקשתות בגרף בהתאם לריצת ה-DFS כך שכל קשת מסווגת לפי אחד מהסוגים הבאים, כך שאין קשת שנמצאת בשתי קבוצות :

1. קשת עץ: קשת מהצורה $(\pi(u), u)$ עבור $u \in V$ כלשהו. יש בדיוק $|v| - k$ קשתות כאלו באשר k הוא מס' העצים.
2. קשת אחורה: קשת מ-v לאב קדמון של v שנקרא לו u.
   צריך להתקיים $d[u] < d[v] < f[v] < f[u]$
3. קשת קדימה: קשת מ-v לצאצא לא ישיר של v, נקרא לו u.
   צריך להתקיים $d[v] < d[u] < f[u] < f[v]$
4. קשת חוצה: קשת מ-u לקודקוד שאינו צאצא ואינו אב קדמון של u. זה כל שאר הקודקודים, ה-DF של צמתי הקשת זרים.

4.4 גרף מכוון חסר מעגלים (DAG)

גרף שכזה נקרא DAG, כיצד נזהה האם גרף G הוא DAG?
טענה: גרף G הוא DAG $\iff$ בהרצת DFS אין קשתות אחוריות.
הוכחה: נוכיח קונטרה פוסיטיב.
נניח שיש מעגל ב-G. $\Longleftarrow$ נסמן u צומת ראשון שנקרא עם $DFS - visit$ במעגל. לכן יש מסלול לבן לשאר הצמתים במעגל. $\Longleftarrow$לפי משפט המסלול הלבן, צמתי המעגל הם צאצאים של u ביער העומק ויש לפחות קשת אחת מהם שחוזרת ל-u (מהגדרת מעגל).
נניח שיש קשת אחורית מ-v ל-u. בשילוב עם קשתות העץ מ-u ל-v (שקיימות כי v צאצא של u לפי הגדרת קשת אחורית) ונקבל כי יש מעגל בגרף המקורי. כנדרש.

מסקנה: כעת בהינתן הרצת DFS, נוכל לבדוק בקלות האם יש קשתות אחוריות (נעבור על כל הקשתות), ואם אין, משמעות הדבר שאין מעגלים. כלומר אלגוריתם לבדיקה האם יש ב-G מעגל בעלות $O(|E| + |V|)$.

4.5 מיון טופולוגי

הגדרה: גרף מכוון ללא מעגלים (גמ"ל) הוא גרף מכוון שלא מכיל מעגלים.

בהינתן גמ"ל נרצה סידור מיוחד של הקודקודים משמאל לימין שבו כל הקשתות הן בכיוון משמאל לימין וממילא כל מסלול מכוון הוא משמאל לימין.

הגדרה: מיון טופולוגי של קודקודי הגרף $G=(V,E)$ שהוא גמ"ל הוא סידור $(v_1,...,v_n)$ של הקודקודים בV כך שלכל קשת $(v_i,v_j)\in E$ מתקיים $i<j$.

$Topoloogical-Sort(G)$:
1. הרץ DFS (המחשב את $f[u]$ לכל $u\in V$)
2. החזר את קודקודי V בסדר יורד של $f[u]$

זמן ריצה: ישירות מDFS, זמן של $O(|E|+|V|)$

למה 6: בסיום ריצת $Topoloogical-Sort(G)$ לכל קשת $(v_i,v_j)\in E$ מתקיים $i<j$
הוכחה:
תהי $(v_i,v_j)\in E$ ונוכיח $i<j$ אחרי המיון הטופולוגי. כלומר $f[v_i]>f[v_j]$.
א. אם $d[v_i]<d[v_j]$ אזי בזמן $d[v_i]-1$ יש מסלול לבן (הקשת (v_i,v_j) בלבד) מv_i לv_j. לכן ע"פ משפט המסלול הלבן ביער העומק v_j צאצא של v_i כלומר $d[v_i]<d[v_j]<f[v_j]<f[v_i]$
ב. אם $d[v_j]<d[v_i]$ לפי משפט הסוגריים ישנן שתי אפשרויות.
1. $d[v_j]<f[v_j]<d[v_i]<f[v_i]$ - במקרה זה אכן המשפט תכף ומתקיים הדרוש
2. $d[v_j]<d[v_i]<f[v_i]<f[v_j]$ - במקרה זה ע"פ נקבל כי v_i צאצא של v_j ביער העומק. בפרט קיים מסלול מכוון מv_j לv_i. אבל, יש לנו גם את הקשת (v_i,v_j) ונקבל מעגל (שרשור המסלול והקשת) בסתירה לכך שG הוא DAG.

טענה: לגרף קיים מיון טופולוגי $\Longleftrightarrow$ הגרף הוא גמל.

4.6 רכיבים קשירים היטב (G^{SCC})

בגרפים לא מכוונים הגדרנו רכיבי קשירות כקבוצות מקסימומיות כך שקיים מסלול בין כל זוג קודקודים בקבוצה. בגרפים מכוונים המסלולים חייבים להיות מכוונים.

הגדרה: רכיב קשיר היטב\חזק בגרף $G=(V,E)$ מכוון הוא קבוצה מקסימומית $C\subseteq V$ כך שלכל $u,v\in C$ מתקיים $v\rightsquigarrow u$ וגם $u\rightsquigarrow v$.

הגדרה: גרף $G=(V,E)$ יקרא קשיר היטב אם ורק אם G מכיל בדיוק רכיב קשיר היטב אחד.

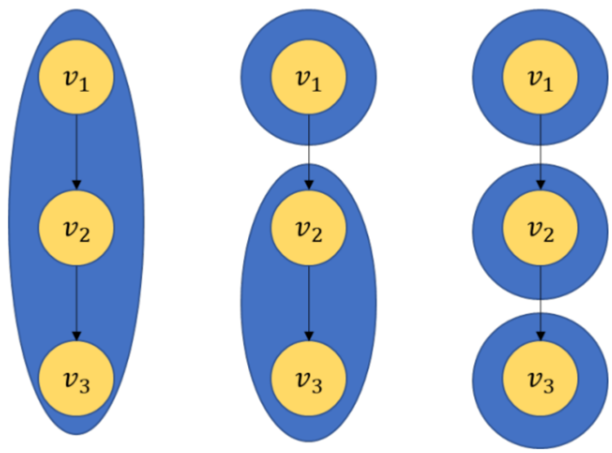
הגדרה: יהי $G=(V,E)$ גרף מכוון. נגדיר את גרף הרכיבים הקשירים היטב כגרף $G^{SCC}=(V^{SCC},E^{SCC})$ כאשר V^{SCC} היא קבוצת הרכיבים הקשירים היטב של G. בנוסף עבור שני רכיבים קשירים היטב שונים $C_1,C_2\in V^{SCC}$ מתקיים $(C_1,C_2)\in E^{SCC}$ אם ורק אם קיימים $v_1\in C_1$ ו$v_2\in C_2$ כך ש$(v_1,v_2)\in E$.

הלמה הבאה מצביעה על תכונה חשובה של גרף ה-SCC שיכולה להיות שימושית במקרים רבים, בעיקר כאשר רוצים להשתמש באלגוריתמים שעובדים רק על גמ"ל (כמו מיון טופולוגי) עם גרפים שמכילים מעגלים.

למה 10: יהי $G=(V,E)$ גרף מכוון (שכמובן ייתכן וכולל מעגלים). G^{SCC} הינו גרף מכוון ללא מעגלים. (נב"ש שיש בו מעגל, נקבל שהיה רכיב קשירות גדול יותר שיכולנו ליצור בסתירה).

נראה אלגוריתם לינארי למציאת רכיבי הקשירות החזקה.

דיון על האינטואיטציה. נתחיל מלחשוב על אלגוריתם הDFS. כאשר מריצים את אלגוריתם הDFS על גרף לא מכוון ־ מקבלים יער, שבו העצים הם בדיוק רכיבי הקשירות של G. (למה? כי מרגע שהפונקציה הראשית הביאה אותנו לקודקוד, אנו סורקים ע"פ משפט המסלול הלבן את כל מי שאיתו באותו רכיב. ומצד שני כמובן שלא ניתן להגיע מרכיב אחד לשני בלי לחזור ללולאה בפונקציה הראשית.) מה יקרה אם נריץ את אלגוריתם ה־DFS על גרף מכוון? אנו כמובן עוברים לדבר על רכיבי קשירות חזקה. באופן דומה למצב בגרף לא מכוון ־ מרגע שהאלגוריתם מגיע לקודקוד כלשהו, הוא יסרוק את כל הרכיב הקשיר החזק שלו (ע"פ משפט המסלול הלבן ־ כל הקודקודים ברכיב הקשיר החזק יהיו בעץ), אולם ייתכן שיהיו בעץ גם קודקודים נוספים. ראו דוגמא לחלוקה ע"פ הרצות שונות באיור . ננסח את ההבחנה באופן פורמלי:



משפט 11: יהי $G = (V, E)$ ויהי $C \subseteq V$ רכיב קשיר חזק בG. לאחר ריצת DFS על G כל קודקודי C נמצאים באותו עץ ביער העומק.

הוכחה: יהי $v \in C$ הקודקוד הראשון שמגיעים אליו בהרצת DFS מבין כל קודקודי C. לכן בזמן $d[v]-1$ קיים מסלול לבן מ־v לכל קודקוד אחר $u \in C$ ברכיב, וממילא ע"פ משפט המסלול הלבן u צאצא של v ביער העומק, ובפרט נמצא באותו עץ, וממילא כל קודוקדי C נמצאים באותו העץ.

הערה: נשים לב שכל רכיב היטב יהיה ביחד בעץ הDFS, אם כן יתכן שכמה רכיבים קשירים היטב יהיו יחד באותו עץ ביער העומק, ועל כך נצטרך להתגבר באלגוריתם.
כעת, המטרה היא למצוא שיטה להריץ את ה־DFS בסדר מיוחד שבו נקבל גם את הצד השני ־ לא רק שכל רכיב קשיר חזק נמצא בעץ אחד אלא גם שבכל עץ יש בדיוק רכיב קשיר חזק אחד. לצורך כך נגדיר מהו גרף משוחלף

הגדרה: יהי $G = (V, E)$ גרף מכוון. הגרף המשוחלף בגרף G ונסמנו $G^T = (V, E^T)$ הוא הגרף המתקבל מהיפוך כל קשת בגרף G. כלומר, $(u, v) \in E \iff (v, u) \in E^T$. באופן שקול, הגרף הוא הגרף שמטריצת השכנויות שלו היא שיחלוף של מטריצת השכנויות של הגרף המקורי.
נעיר כי האלגוריתם שנראה מחזיר את קודקודי G^{SCC}.

$Strongly\ Connected\ Component(G)$:

א. הרץ $DFS(G)$ (המחשב תוך כדי ריצתו את $f[u]$ לכל $u \in V$)

ב. חשב את G^T
ג. הרץ $DFS(G^T)$ כאשר בלולאה הראשית בפונקציה DFS עבור על הקודקודים ע"פ סדר יורד של $f[u]$.
ד. דווח על כל עץ ביער העומק כרכיב קשיר היטב.

מדוע אינטואטיבית הרעיון עובד? ראינו כבר כי אם נעבור בכיוון מסויים, למשל בגרף המכוון $a \to b \to c$ מימין לשמאל נקבל אכן שלושה רכיבי קשירות שונים. אם נעבור משמאל לימין בDFS נקבל רכיב קשירות אחד. נרצה תמיד לעבור בכיוון השני. אנחנו מגדירים את G^T ואז מריצים dfs עליו לפי סדר ה$f[u]$ בסדר יורד. כלומר - זמן הסיום האחרון יתחיל ממנו ב$DFS(G^T)$. נשים לב ששום אלגוריתם אחר לא יעבוד - באינטואיציה נראה כי בהכרח בהינתן שני רכיבי קשירות זמן סיום של צד אחד יהיה לפני השני, ולכן אם נהפוך את הקשת בהכרח באלגוריתם זה לא נשתמש באותו עץ בכמה רכיבי קשירות.

זמן ריצה: DFS עלותו $O(|V|+|E|)$, חישוב G^T עלותו $O(|E|)$ והרצה נוספת של DFS עלותה $O(|E|+|V|)$ וכן ד' יעלה עוד זמן לינארי וסה"כ סיבוכיות האלגוריתם $O(|E|+|V|)$.

4.7 מציאת גרף דו צדדי

קלט: גרף דו צדדי $G=(V,E)$ קשיר.
פלט: חלוקה של V לR וL כך שאין קשתות בתוך L ולא בתוך R.

האלגוריתם:
א. הרץ BFS מקודקוד s שרירותי.
ב. כל מי שבמרחק זוגי יהיה מL וכל מי שבמרחק אי זוגי יהיה בR.

הערה. אפשר כמובן להכליל את הבעיה לגרף לא קשיר, ולהריץ את האלגוריתם על רכיבי הקשירות שלו.

טענה (הוכחת נכונות): יהי $s \in L$. אזי לכל $u \in V$ מתקיים $u \in R \iff \delta(u,s)$ אי זוגי.
הוכחה: באינדוקציה על המרחק.
בסיס: עבור $\delta(u,s)=0$ בהכרח $u=s$ ואכן זוגי ולא יתכן שהוא בR, אם $\delta(u,s)=1$ אזי בהכרח כי לא קיימת קשת ביניהם, בהכרח $u \in L$ כי הקשת בצד השני.
צעד: נניח שמתקיים עבור $k-1$. נוכיח עבור k.
נוכיח עבור k זוגי אחרת הוכחה דומה. יהי $u \in V$ כך ש$\delta(s,u)=k$. נסתכל על הקודקוד הקודם לu במסלול הקצר ביותר, נסמנו v. בהכרח $\delta(s,v)=k-1$ ומהנחת האינדוקציה אכן $v \in R$ ולכן $u \in L$ כי הקשת ביניהם בהכרח עוברת אל L, כנדרש.

4.8 מציאת עץ מסלולים קצרים ביותר בגמ"ל

קלט: גרף $G=(V,E)$ גמ"ל עם פונקציית משקלים $w: E \to \mathbb{R}$ וקודקוד מטרה s.
פלט: $SSSP$ (מערך מרחקים)

נפתור בתכנות דינמי.

$$f(u)=\left\{\begin{array}{cc} 0 & u=s \\ min_{\{v|(v,u)\in E\}}\{f(v)+w(v,u)\} & o.w \end{array}\right\}$$

נכונות הנוסחה: יהי $P=(s,v_1,...v_k,u)$ מסלול קצר ביותר מs לu. נשים לב כי הקודקוד v_k שהוא הקודם לu מקיים $(u_k,u)\in E$ ולכן נשקל כאחת האפשרויות בביטוי שהפונקציה מנסה להביא

למינימום. נשים לב כי העלות של המסלול P היא $w(P) = \sum_{e \in P} w(e) = w(P') + w(v_k, u)$ כאשר $P' = (s, ..., v_k)$ הוא מק"ב בעצמו.
נשים לב כי ביחידה הבאה: 4.9 נציע רעיון די דומה עבור גרף כללי אך זה לא יעבוד מהסיבה שמתוארת מטה. מדוע כאן יעבוד? כי אין מעגלים.

עלות זמן ריצה: כדי לחשב את הנוסחה ביעילות לכל קודקוד נרצה לשים לב כי ערך הפונקציה של קודקוד תלויה אך ורק בערכי הפונקציה של השכנים הנכנסים של הקודקוד. נזכור שעבור גמל ניתן לחשב מיון טופולוגי של הקודקודים שמבטיח לנו שכל השכנים הנכנסים של הקודקוד יופיעו בסדר המיון לפני הקודקוד, לכן, נוכל לחשב מיון טופולוגי ולאתחל את הערך של הקודקוד s להיות 0. ואז לחשב עבור כל קודקוד את ערך הנוסחה הרקורסיבית עבורו עפ סדר המיון הטופולוגי. סדר מיון זה מבטיח לנו כי ערכי הפונקציה עבור כל השכנים הנכנסים של הקודקוד חושבו לפני שמנסים לחשב את ערך הפונקציה עבור הקודקוד ולכן עלות החישוב של הנוסחה עבור כל קודקוד $O(degv)$ במקרה הגרוע. עפ למת לחיצת הידיים נקבל כי עלות זמן הריצה הכוללת של האלגוריתם יחד עם המיון הטופולוגי הינה $O(|E| + |V|)$.

4.9 $SSSP$ בגרפים ממושקלים

קלט: גרף $G = (V, E)$, $s \in V$ ופונקציית משקל $w : E \to \mathbb{R}$.
פלט: $\forall v \in V : \delta(s, v)$

4.9.1 נסיון ראשון - תכנות דינמי

נזכר כי במקרה הממושקל:

$$\delta(u, v) := \left\{ \begin{array}{cc} min_{p=u \rightsquigarrow v}\{w(p)\} & \exists p = u \rightsquigarrow v \\ \infty & o.w \end{array} \right\}$$

ראינו בלמה 1, שתת מסלול של מסלול קצר ביותר הוא גם כן מסלול קצר ביותר. זה נכון גם עבור גרפים ממושקלים. אם כן, נשים לב שנוכל לקבל כאן רעיון לאלגוריתם רקורסיבי: אם נרצה לחשב את המסלול הקצר ביותר מs אל u, וידוע כי שכניו של u הינם $u_1, .., u_4$ אזי נחשב את המסלול $s \to u_1$ למשל, ונוסיף משקל קשת. כלומר -

$$f(u) = min\{f(u_1) + w(u_1, u), f(u_2) + w(u_2, u), f(u_3) + w(u_3, u), f(u_4) + w(u_4, u)\}$$

נשים לב כי אם יש מסלול אל u, בפרט ישנו קצר ביותר, והוא בוודאות יעבור קודם לכן אצל אחד השכנים של u (לפי אותה למה). ובמילים אחרות: המסלול הקצר ביותר אל u חייב לעבור אצל אחד מהקודקודים השכנים של u.
נתבונן בנוסחה הבאה, באשר $f(u) = \delta(s, u)$:

$$f(u) = \left\{ \begin{array}{cc} 0 & u = s \\ min\{min_{(v,u) \in E} \quad (\{f(v) + w(v, u)\}), \infty\} & o.w \end{array} \right\}$$

הנוסחה די ברורה, נשים לב לשני דברים:
א. אנחנו מבצעים השוואה בין הביטוי לבין אנסוף - יתכן שלא קיים מסלול בין s לu.
ב. נשים לב כי הגרף מכוון, אנחנו עוברים על הקשתות (v, u) כלומר הקשתות שנכנסות אל u.

הלוואי וזה היה עובד, זה היה קל מאוד. באלגוריתם הנ"ל יש בעיה. מה הבעיה?

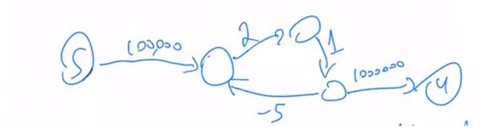
נניח שיש מעגל wvu וישנה קשת $s \to v, s \to w$. כשנחשב ברקורסיה את המסלול הקצר ביותר מs אל v, נצטרך לחשב את $f(w)$ כי יש ביניהם קשת, כשנחשב את $f(w)$ נגלה שאנחנו צריכים לחשב את $f(u)$ כי ישנה קשת $u \to w$, ואז שוב צריך לחשב את $f(u)$. וכשנצטרך לחשב אותו - חוזר חלילה. נתקע בלופ: הרקורסיה תקרוס, והבוס שלך יפטר אותך. באסה.

מסקנה: בגרף מכוון, ללא מעגלים, נוסחה זו תעבוד בסיבוכיות $O(|V| + |E|)$.

4.9.2 סוגי מעגלים

כפי שראינו, מעגלים עושים לנו בעיות. ישנם מספר סוגי מעגלים:

א. מעגל שלילי - מעגל כמו המעגל המתואר כאן לעיל הוא מעגל שלילי. הליכה על מעגל שכזה תמיד תהיה טובה לנו כי נעבוד $1 + 2$ וירד לנו $5-$ כלומר בכל סיבוב אנחנו נרוויח $2-$. במצב כזה, אם נלך אנסוף פעמים על המעגל נגיע למשקל שהוא מאוד נמוך - מינוס אנסוף.



לכן, אם יש מעגל שלילי מs לu בG נגדיר: $\delta(s, u) = -\infty$.

כעת, נניח כי בגרף יכולים להיות משקלים שליליים אך אין מעגלים שליליים.

ב. מעגל אפס - מעגל שסכום המשקלים שלו הוא אפס, ואין סיבה לכאורה לעבור בו. למשל מעגל כמו שמתואר מעלה, במקום 1 שיהיה הערך 3. נראה כי סכומו יהיה $2 + 3 - 5 = 0$. מעגל כזה לא מפריע לנו.

ג. מעגל חיובי - מעגל שסכום הערכים על הקשתות שלו חיובי. אין סיבה לעבור עליו במציאת מסלול קצר ביותר.

מסקנה: אם אין מעגלים שליליים, ניתן להניח שקיים מסלול קצר ביותר שהוא מסלול פשוט. (מדוע? כי לא נרצה לעבור במעגל חיובי, ועל מעגל אפס אפשר לדלג).

מכאן, ניתן להניח כי במסלול קצר ביותר יש לכל היותר $|V| - 1$ קשתות. (זהו מסלול פשוט: אין בו מעגלים, ולכן על כל קודקוד ניתן לעבור לכל היותר פעם אחת. במקרה הגרוע ביותר עברנו על כל $|V|$ הקודקודים, יש ביניהם $|V| - 1$ קשתות).

נשים לב - גם אם אין מעגלים שליליים, אנחנו עדיין באותה בעיה שנתקלנו בה באשר ניסינו לעבוד בתכנות דינמי. אנחנו לא מפרקים את הבעיה לתתי בעיות שם - תמיד נתקע בלופ.

4.9.3 הקלת קשתות $-Relaxtions$

רעיון האלגוריתם: לכל $u \in V$ האלגוריתם מתחזק ערך $d[u]$ שהוא חסם עליון על $\delta(s, u)$. כלומר, $d[u] \geq \delta(s, u)$.

האלגו' על פני ריצתו יוריד ערכי d עד ש(בתקווה) כל ערכי הd מקיימים:

$$\forall u \in V : d[u] = \delta(s, u)$$

כמו כן, נשתמש במשתנה $\pi[u]$ להגדיר את עץ המסלולים הקצרים ביותר.

כך יראה האתחול:

```
INITIALIZE-SINGLE-SOURCE(G = (V, E), s)
for each vertex v ∈ V
      d[v] ← ∞
      π[v] ← NULL
d[s] ← 0
```

$**$ נשים לב כי $d[s]=0$ היא הנחה לגיטימית, כלומר $\delta(s,s)=0$, הדרך היחידה ש$\delta(s,s)\neq 0$ היא שיהיה מעגל שלילי שמתחיל ונגמר בs, ואז $\delta(s,s)<0$, אך **אנחנו מניחים שאין מעגלים שליליים.**

נשים לב: אנחנו עובדים לפי קונבציה שs מצביע אל $null$ בעץ, וכן יתכנו קודקודים נוספים שמצביעים אל $null$ (מי שלא נגישים אל s), איך נבדיל ביניהם לבין s? הd שלהם יהיה אנסוף ו$d[s]=0$.

למת אי שוויון המשולש (במקרה הממושקל): עבור $G=(V,E)$ עם פונקציית משקל $w:E\to\mathbb{R}$ וכן $s\in V$ ו$(u,v)\in E$ מתקיים:

$$\delta(s,v)\leq\delta(s,u)+w(u,v)$$

מסקנה: נניח כי אכן $d[u]\geq\delta(s,u)$, אזי

$$d[u]+w(u,v)\geq\delta(s,u)+w(u,v)\geq\delta(s,v)$$

אזי, אם $d[v]>d[u]+w(u,v)$ אזי ניתן להקטין את $d[v]$ להיות $d[u]+w(u,v)$. מדוע? כי במקרה זה, כיוון ש$d[u]+w(u,v)\geq\delta(s,v)$, עדיין אם נגדיר $d[v]=d[u]+w(u,v)$ יתקיים $d[v]\geq\delta(s,v)$ כפי שרצינו. פעולת הקטנה זו נקראת **פעולת הקלה.**

(עוד אינטואיציה: נניח כי $d[u]=100, d[v]=200$, וכן ישנה קשת $e:u\to v$ כך ש$w(e)=30$, אכן מתקיים $d[v]>d[u]+w(u,v)$, ולכן כדאי לשפר את $d[v]$ ולהקטין אותו להיות המסלול של $d[u]$ ועוד הקשת ששווה 30, שכן ערך המסלול ירד.)

פסאודו לפעולת $relax$ **- ההקלה:**

```
RELAX((u, v), w)
if d[v] > d[u] + w(u, v)
      d[v] ← d[u] + w(u, v)
      π[v] ← u
```

נשים לב כי אם בחרנו לבצע הקלה, שינינו את אבא של v להיות u ולכן π משתנה.

4.9.4 אלגוריתם מבוסס הקלות והוכחת נכונות של בלמן פורד

אלגוריתם מבוסס הקלות הוא אלגוריתם שמאתחל ערכי d,π פעם אחת בעזרת קריאה ל- $Initalize-$
$Single-Source$, ולאחר מכן כל העדכונים לערכי d יתבצעו רק בעזרת פעולות של $relax$.

המטרה: להראות כי האלגוריתם מבצע סדרה של הקלות שמובילות לכך ש:

$$\forall u \in V : d[u] = \delta(s,u)$$

כלל הטענות הבאות יהיו נכונות לכל אלגוריתם מבוסס הקלות.

למה 7 (**למת חוסר המסלול**): לאחר ביצוע $ISS(G,s)$ (אתחול) באלגוריתם מבוסס הקלות, אם אין מסלול מS ל$v \in V$ בG אזי תמיד מתקיים $\delta(s,v) = d[v] = \infty$.

הוכחה: ההוכחה נובעת ישירות מלמה 9 שכאן למטה. בעת האתחול יתבצע $d[v] = \infty$. כמו כן, אם אין מסלול מתקיים $\delta(s,v) = \infty$. מתקיים $d[v] = \delta(s,v)$ ולפי למה 9 מהרגע שזה מתקיים הערך לא ישתנה יותר, ואכן מתקיים כדרוש.

למה 8 (**למת הקלת מסלול**) : נניח כי המסלול $P = (v_0,...,v_k)$ הוא מסלול קצר ביותר עם $s = v_0$. נניח שבזמן ריצת אלגוריתם מבוסס הקלות לאחר ביצוע $ISS(G,s)$, סדרת ההקלות שהאלגוריתם מבצע מכילה את סדרת הקשתות של p כתת סדרה לפי סדרן בP. (כלומר, לאחר ביצוע פעולות רילקס על כל (v_i,v_{i+1}) לכל $0 \le i \le k-1$ לפי הסדר **אך לא בהכרח ברצף**) אזי, בסיום ריצת האלגוריתם מתקיים $d[v_k] = \delta(s,v_k)$. (כלומר, הבעיה נפתרה עבור קודקוד v_k).

הוכחה: נראה כי לאחר ביצוע הקלה על הקשת הi (בתת הסדרה) מתקיים $d[v_i] = \delta(s,v_i)$. נוכיח באינדוקציה על i.

בסיס: $i = 0$, כלומר משמעות הדבר היא $s = v_0$, אכן בתחילה מתקיים $d[s] = \delta(s,s) = 0$. ע"פ למה 9, הערך של $d[s]$ לא ישתנה עוד לעולם.

צעד: ע"פ הנחת האינדוקציה, אחרי ההקלה על הקשת הi בתת סדרה מתקיים שביצענו הקלות על הקשת הראשונה, השנייה וכו' עד הi ולכן מתקיים $\forall_{0\le k\le i} : d[v_k] = \delta(s,v_k)$

בהקלה ה$i+1$, לפי למה 10, מתקיים כי $\delta[v_{i+1}] = \delta(s,v_{i+1})$, כי אכן ביצענו הקלה על כל הקשתות הקודמות ובפרט על זו הi, ולכן לאחר ביצוע הקלה יתקיים השוויון. כנדרש.

נשים לב. למה 8 טוענת שאם יש לי את המסלול הקצר ביותר, אזי אני יודע שפתרתי את הבעיה עבור v_k. מדוע צריך את זה? נסתכל על המסקנה הבאה -

מסקנה: נשים לב כי לפי למה 8 ולפי למה 1, אם פתרנו את הבעיה עבור v_k, אזי פתרנו את הבעיה לכל $0 \le i < k$, כלומר $\forall_{0\le i<k} : d[v_i] = \delta(s,v_i)$. כלומר, אם נריץ פעם אחת נפתור את הבעיה עבור v_1. אם נריץ פעמיים, נפתור את הבעיה עבור v_2. וכן אם נריץ עבור $|V|-1$ נקבל את הפתרון להבעיה.

למה 9 (**למת החסם העליון של** $d[v]$): באלגוריתם מבוסס הקלות, לאחר ביצוע האתחול $ISS(G,s)$ לכל קודקוד $v \in V$ תמיד מתקיים $d[v] \ge \delta(s,v)$. בנוסף, מהרגע שבו האלגוריתם מציב $d[v] = \delta(s,v)$, הערך של $d[v]$ לא ישתנה יותר עד לסיום ריצת האלגוריתם.

הוכחה: נוכיח באינדוקציה על מס' פעולות ההקלה שהאלגוריתם מבצע. נסמנם n.

בסיס: $n = 0$, $\forall v \in V/s$ אם האלגוריתם לא ביצע עדיין הקלות מתקיים $d[v] = \infty \ge \delta(s,v)$.
וכן עבור $v = s$ מתקיים $d[s] = 0 = d(s,s)$ לפי הגדרת האתחול, ומתקיים $\delta(s,s) = 0$ כי אין מעגלים שליליים.

צעד: נניח שלפני ביצוע הקלה $Relax((u,v),w)$ מתקיים שלכל $x \in V$ $d[x] \ge \delta(s,x)$.

אם פעולת ה-$relax$ לא שינתה שום ערך של d, לפי הנחת האינדוקציה עדיין מתקיים כי : $\forall x \in V$ $.d[x] \geq \delta(s,x)$

אם פעולת ה-$relax$ כן שינתה ערך של d, היא יכולה לשנות רק את הערך של $d[v]$, בפרט יתקיים כעת כי $d[v] = d[u] + w(u,v)$, מתקיים כי לפי הנחת האינדוקציה כי $d[u] \geq \delta(s,u)$, ולכן נקבל $d[v] = d[u] + w(u,v) \geq \delta(s,u) + w(u,w) \geq (*)\delta(s,v)$, באשר $(*)$ מתקיים לפי אי שוויון המשולש. סה"כ $d[v] \geq \delta(s,v)$ כנדרש.

נשים לב כי עלינו להוכיח דבר נוסף, מהרגע שנציב ערך $d[v] = \delta(s,v)$ הערך של $d[v]$ לא ישתנה יותר. אם כן, כיוון שפעולות $relax$ רק מקטינות ערכי d, והוכחנו הרי כי $\forall u \in V : d[u] \geq \delta(s,u)$ אזי ברגע ש-$d[u] = \delta(s,u)$ הערך של $d[u]$ לא יכול להשתנות יותר כי $d[u]$ לא יכול לגדול, ו-$d[u]$ לא יכול להיות קטן יותר כי $d[u] \geq \delta(s,u)$, סה"כ ערכו לא משתנה.

למה 10 (**תכונת ההתכנסות**): נניח שהמסלול $s \rightsquigarrow u \rightarrow v$ הוא מסלול קצר ביותר מ-s ל-v. אזי, לאחר ביצוע $ISS(G,s)$ באלגוריתם מבוסס הקלות, מתקיים שאם $d[u] = \delta(s,u)$ מתישהו לפני ביצוע הקלה על הקשת (u,v), אזי לאחר ביצוע הקלה על הקשת מתקיים $d[v] = \delta(s,v)$. (כלומר, אם המסלול מ-$u \rightsquigarrow s$ כבר ידוע לי, לאחר ביצוע הקלה אחת אנחנו נדע את $v \rightsquigarrow s$ כי הרי סה"כ מוסיפים הליכה על קשת אחת).

הוכחה: ע"פ למה 9, תמיד מתקיים $d[v] \geq \delta(s,v)$, נפצל למקרים.

א. אם לפני ביצוע ההקלה התקיים $d[v] = \delta(s,v)$ אזי סיימנו, כי הערך לא ישתנה יותר לפי למה 9.

ב. אחרת, כלומר $d[v] > \delta(s,v)$. מכאן, $d[v] > \delta(s,v) = \delta(s,u) + w(u,v) = d[u] + w(u,v)$ כלומר קיבלנו $d[v] > d[u] + w(u,v)$, זהו בדיוק התנאי שאנחנו בודקים בביצוע הקלה. התנאי הזה מתקיים, ולכן אנחנו מעדכנים לאחר ביצוע ההקלה את $d[v] = d[u] + w(u,v)$, כמו כן מתקיים $d[u] = \delta(s,u)$ ולכן

$$d[v] = d[u] + w(u,v) = \delta(s,u) + w(u,v) = \delta(s,v)$$

ומלמה 9, ערך $d[v]$ לא ישתנה עוד. כנדרש.

4.9.5 האלגוריתם של בלמן פורד

נשים לב כי לפי למה 8, אם נדע את המסלול הקצר ביותר מ-s אל v נוכל לדעת את $\delta(s,v)$. ובכן, לשם כך נצטרך לפתור את הבעיה עצמה - המסלול הקצר ביותר. עם זאת, נשים לב כי אם נבצע הקלה לכל קשתות הגרף - לכל קשת פעם אחת: בוודאות ביצענו הקלה על הקשת הראשונה במסלול הקצר ביותר, אך לא ידוע לנו באיזה שלב של ההקלות ביצענו עליה הקלה. כלומר שאנחנו יודעים כי $d[v_1] = \delta(s,v_1)$ לפי למה 8. אם נבצע זאת שוב, הקלות לכל הקשתות בגרף, כעת בוודאות ביצענו הקלה בקשת $v_1 \rightarrow v_2$ ולפי אותה למה נדע כי $d[v_2] = \delta(s,v_2)$. באופן דומה ומחזורי: נבצע את התהליך שוב ושוב, עד שנקבל לאחר k איטרציות - עבור כל קודקוד u שיש אליו מסלול קצר ביותר מ-s עם לכל היותר k קשתות, יתקיים $.d[u] = \delta(s,u)$

כמה איטרציות אנחנו צריכים? בהינתן ההנחה שאין מעגלים שליליים, לכל קודקוד $u \in V$ קיים מסלול קצר ביותר מ-s אל u שהוא מסלול פשוט, במסלול כזה יתקיים כי אורכו $\leq |V| - 1$. ולכן נצטרך לבצע לכל היותר $|V| - 1$ איטרציות.

הערה. אנחנו תמיד נדע את $d[v_1]$ בהתחלה כי יתקיים התנאי של אי השוויון לפי האלגוריתם כי $d[s] = 0$, זה לא בהכרח יקרה עבור שאר הקודקודים כי נקבל אי שוויון של $\infty < \infty + w()$ שאיננו נכון בהכרח.

להלן האלגוריתם:

בתחילה, נאתחל. לאחר מכן במשך $|V| - 1$ פעמים אנחנו נעבור על כל הקשתות בגרף ונבצע הקלה.

```
Bellman-Ford(G = (V, E), w, s)
1  Initialize-Single-Source(G = (V, E), s)
2  for i = 1 to |V| - 1
3        for (u, v) ∈ E
4              Relax((u, v), w)
```

סיבוכיות זמן הריצה: $O(|V| \times |E|)$

נכונות האלגוריתם: נניח וקיים מסלול, אזי קיים גם מסלול קצר ביותר (אחרת ראינו כי הערך יהיה אנסוף כבר בהתחלה, ולא ישתנה), אנחנו יודעים לחשב את $d[v_1]$ שבמסלול לפי הערה כאן לעיל, ומשם לפי למה 10, לאחר ביצוע הקלה אחת אנחנו יודעים גם את $d[v_2] = \delta(s, v_2)$. וכן הלאה ממשיכים עם הקלה ואח"כ למה 10 ולבסוף אנו יודעים את $d[v]$.

נשים לב: בDAG ישנו מיון טופולוגי, ולכן ניתן להחליט שנעבור בסדר מסויים על הגרף. וכך, לא נצטרך בכל שלב באלגוריתם בשורה **3** לעבור על כל הקשתות, נוכל לעבור בכל קודקוד רק על הקודקודים שיוצאים ימינה. נשים לב כי לקודקוד הראשון יש אפשרויות ללכת רק ימינה וכן הלאה על ההבאים. נקבל כי סיבוכיות זמן הריצה היא $O(|E| + |V|)$.

הערה סופר חשובה: כיוון שאמרנו שמסלול ארוך ביותר יהיה בגודל $|V| - 1$ כי הוא פשוט (לא נרצה לעבור על מעגלים). ניתן למצוא דרך לזהות מעגל שלילי - ניתן להריץ בלמן פורד ולהוסיף איטרציה מס' $|V|$. נעבור בה על הקשתות. אם עדיין אפשר לעשות הקלה לאחת הקשתות ישר נחזיר שקיים מעגל שלילי. מדוע? כי אם אפשר לשפר, מתכונה זו בדיוק, זה מסלול בגודל $|V|$, מעגל שלילי.

4.10 האלגוריתם של דייקסטרה

4.10.1 הגדרת הבעיה ומבוא

קלט: גרף $G = (V, E)$ ממושקל עם פונקציית משקל $w : E \to \mathbb{R}^+$ (משקלים חיוביים בלבד), וקודקוד מקור $s \in V$

פלט: $\forall u \in V : \delta(s, u)$ וכן להחזיר עץ מסלולים קצר ביותר מs.

באלגוריתם של פרים, למציאת MST, בכל איטרציה האלגוריתם בוחר את הקשת בעלת המשקל הנמוך ביותר שחוצה את החתך ומוסיף אותה לעץ. האלגוריתם של דייקסטרה פועל באופן דומה. בכל איטרציה, האלגוריתם של דייקסטרה בוחר את הקשת הטובה ביותר להוסיף לעץ המסלולים. השוני בין האלגוריתמים - מה זה אומר "הטובה ביותר".

הגדרה: נסמן בS את קבוצת הקודקודים שנמצאת כרגע בעץ המסלולים הקצרים ביותר. מסלול P יקרא "מסלול מיוחד" אם כל הקודקודים במסלול P נמצאים בS חוץ מהקודקוד האחרון שלא נמצא בS. ייתכנו כמה מסלולים מיוחדים P כנ"ל.

האלגוריתם של דייקסטרה בכל איטרציה בוחר את הקודקוד u, $u \notin S$ כך שיש מסלול מיוחד מs לu שכרגע הוא מסלול מיוחד קצר ביותר ביחס לכל המסלולים המיוחדים. כלומר, אנחנו בוחרים קודקוד u שלא נמצא בתוך קבוצת הקודקודים בעץ, והקודקוד u הזה שאנחנו בוחרים הוא זה שהמסלול מs

אליו הוא הקצר ביותר ביחס ל**כל המסלולים המיוחדים**. זוהי נקודה מהותית ־ הוא בוחר ביחס לכל המסלולים המיוחדים, לא ביחס לכל מי שמסתיים בu. האלגוריתם מוסיף את הקשת האחרונה על המסלול (זאת שנראית כרגע "הטובה ביותר") המיוחד P לעץ ומוסיף את u לS.

נחדד: אנחנו נסתכל על קבוצה S שהיא כל הקודקודים שכרגע בעץ המסלולים הקצרים ביותר. נרצה בכל פעם לבחור את הקודקוד שהוא לא בS, והמסלול הקצר ביותר אליו הוא הקצר ביותר וכן המסלול מs אל אותו קודקוד הוא מסלול שכל הקודקודים בו נמצאים בS, פרט לקודקוד האחרון שנמצא ב$V\setminus S$ והקשת האחרונה במסלול היא קשת שחוצה את החתך.

4.10.2 האלגוריתם

האלגוריתם של דייקסטרה הוא אלגוריתם מבוסס הקלות, לכן הוא מחזיק ערכי d וπ וכן מבצע ISS בתחילת הריצה ומוריד את ערכי d רק באמצעות $relax$. נשים לב כי נאתחל בהתחלה את מבנה הנתונים Q עם v ובתחילה s יצא ממנו. המפתח יהיה $d[u]$ לכל קודקוד.
להלן האלגוריתם:

```
DIJKSTRA(G = (V, E), w, s)
1  INITIALIZE-SINGLE-SOURCE(G = (V, E), s)
2  S ← ∅
3  Q.Init(V)
4  while Q ≠ ∅
5       u ← Q.extract_min()
6       Add u to S
7       for each v ∈ ADJ[u]
8            RELAX((u, v), w)
```

כאלגוריתם מבסס הקלות מתחילים ב$Iss(G,s)$ ויוצרים קבוצה ריקה S. כמו כן, אנו משתמשים בתור קדימויות Q. מאתחלים אותו ראשית. נגדיר כי מפתחות התור הם ערכי d. בכל שלב, כל עוד התור לא ריק (בתחילה הוא מאותחל עם כל איברי V כל הקודקודים), אנחנו נוציא את זה עם המפתח המינימלי (הd הכי קטן, המרחק הכי קטן..), נכניס אותו אל S, ונעבור על כל שכניו ונבצע להם הקלה.

ישנה נקודה שצריך לשים לב אליה ־ אם הערך של $d[v]$ יורד, משמעות הדבר היא כי צריך לבצע $decrease\_key$ על v בQ. כלומר, נשנה את פעולת $relax$ שאם נכנסים אל if בתוך רילקס, אזי מוסיפים $Q.dec\_key(v, d[v])$.

הקודקוד הראשון שיצא מQ הוא s כי $d[s] = 0$ ושל כל השאר הוא אנסוף. וממנו יבנה העץ.

סיבוכיות זמן הריצה: נראה כי יש לנו אתחול, ידוע כי הוא עולה $O(|V|)$ זמן. הלולאה מתבצעת $|V|$ פעמים, חלק 7 מתבצע $deg(u)$ פעמים ולכן כמכלול $\sum_{v\in V} deg(v) = O(|E|)$ מס' הפעמים

שיכולה להתבצע שורה 8, וכן בכל אחד מפעולות ה$relax$ אנחנו עלולים לבצע $decrease\_key$. מכאן ש:

ISS שיעלה $O(|V|)$
ביצוע $init$
$|V|$ פעמים $extract\_min$
$|E|$ פעמים $decrease\_key$
וזמן הריצה הוא:

$$O(init + |V| + |V| \times extract\_min + |E| \times decrease\_key)$$

ערימה בינארית:

$$O(|V| + |V| + |V| \times log(|V|) + |E| \times log(|V|)) = O(|V|log(|V|) + |E|log(|V|))$$

ערימת פיבונאצ'י:

$$O(|V| + |V| + |V| \times log(|V|) + |E| \times O(1))) = O(|V|log(|V|) + |E|)$$

הערה חשובה: האלגוריתם מאתחל בהתחלה את Q להיות כל קבוצת הקודקודים. תמיד בתחילת האלגוריתם s יצא קודם!!, בכל שלב נוציא את האיבר שהמפתח שלו הכי קטן. לבסוף נסיים ללא איברים בתור קדימויות ונסיים.

4.10.3 הוכחת נכונות של דייקסטרה

האלגוריתם של דייקסטרה הוא אלגוריתם חמדני. וניתן להוכיח למה בחירה חמדנית וכו'. אך הפעם נשתמש בהוכחה יותר ישירה.

הערה. אינווריאנטה היא טענה שנרצה להוכיח שתמיד מתקיימת.

אינווריאנטה 1: כל פעם שהאלגוריתם של דייקסטרה מגיע לשורה 4 בפסודו קוד, אז לכל $u \in S$ מתקיים $d[u] = \delta(s, u)$ (כלומר, כל מי שהכנסנו כבר לקבוצה S ונכנס לעץ, אנחנו יודעים את המרחק הקצר ביותר עבורו. אם זה נכון, זה בפרט יהיה נכון כאשר נגיע לשורה 4 בפעם האחרונה, ואז יתקיים לכל הקודקודים כי $d[u] = \delta(u, s)$ והוכחנו את הטענה).

הוכחה: ההוכחה היא באינדוקציה על מס' הפעמים שהאלגוריתם בריצה מסויימת מגיע לשורה 4. נספנו n.

בסיס: $n = 0$, בשלב זה כל מה שבוצע עד כה באלגוריתם היה אתחול בלבד. הקבוצה S ריקה ובפרט האינווריאנטה מתקיימת באופן ריק.

צעד: נראה שכל פעם שהאלגוריתם מוסיף קודקוד $s \to u$ מתקיים $d[u] = \delta(s, u)$. יחד עם למה 10, נקבל שמהרגע שבו u התווסף אל S תמיד יתקיים $d[u] = \delta(s, u)$.

נניח בשלילה כי u מתווסף אל S גדולה, אך $d[u] \neq \delta(s, u)$. בה"כ u הוא הקודקוד הראשון שמתווסף אל S שעבורו $d[u] \neq \delta(s, u)$. ולכן כל הקודקודים שקדמו לu, יקיימו $d[k] = \delta(s, k)$.

נשים לב כי $s \neq u$ כי כבר באתחול $d[s] = \delta(s, s) = 0$ (כי אין מעגלים שליליים כי אין קשתות עם משקל שלילי) ולפי למה 10 הערך לא ישתנה לעולם. מכאן שלא יתכן ש$s = u$.

מכאן נסיק כי לפני שu התווסף אל S, בוודאות $S \neq \emptyset$ (בוודאות s יהיה שם). כמו כן, בהכרח קיים מסלול מs אל u כי אחרת לפי למה 7 (למת חוסר המסלול) יתקיים כי $d[u] = \infty = \delta(s,u)$, ולכן בוודאות יש מסלול $u \rightsquigarrow s$. כלומר $\delta(s,u) < \infty$.
נסמן P' מסלול קצר ביותר מs אל u בG (קיים כזה), נסמן בy את הקודקוד הראשון בP' שנמצא בV/S. בהחלט יתכן כי $y = u$. כמו כן, נסמן את הקודקוד לפני y במסלול בx, ולכן $x \in S$. כלומר המסלול הינו $P' = (s, ..., x, y, ..., u)$
מכאן שבזמן שu הצטרף לS, אנו יודעים כי $d[x] = \delta(s,x)$ (כי הנחנו שu הוא הראשון עבורו אי השוויון קרה).
נסתכל על מה האלגוריתם עשה ברגע שx הצטרף אל S: האלגוריתם בשלב ההוא עבר על כל הקשתות שיוצאות מx ובפרט על הקשת (x,y) וביצע עליהן הקלות. מלמה 10 נובע, שלאחר הקלה על הקשת (x,y) נקבל כי $d[y] = \delta(s,y)$ (כי הרישא של P' שמסתיימת בy היא מסלול קצר ביותר מs אל u, ובפרט תת מסלול של מסלול קצר ביותר הוא מסלול קצר ביותר, ולכן המסלול עד x הוא הקצר ביותר, ומכאן לפי למה קודמת).
מכאן, $d[y] = \delta(s,y) \leq \delta(s,u) \leq d[u]$ (כי אנחנו ללא משקלים שליליים, והמסלול יכל רק לגדול)
מצד שני, האלגוריתם בחר בu להצטרף אל S כאשר y הוא אחת מהאופציות , לכן בשלב זה $d[u] \leq d[y]$ (אחרת y היה נבחר) - נשים לב כי y הוגדר להיות ב$V \backslash S$ ולכן בהכרח u יצא קודם מהתור לפני y.
סה"כ קיבלנו $d[u] \leq d[y]$ וכן $d[y] \leq d[u]$ ומכאן $d[u] = d[y]$, כלומר $d[y] = \delta(s,y) \leq \delta(s,u) \leq d[u] = d[y]$ בפרט מתקיים כי כל השוויונות באמצע מתקיימים ונקבל $d[u] = \delta(s,u)$, בסתירה לכך שהנחנו אי שוויון ביניהם. כנדרש.

4.11 סיכום

ישנם שלושה אלגוריתמים שונים לבעיית $shorts\ paths$.
אלגוריתם BFS: מטפל ב$SSSP$ במקרה הלא ממושקל. סיבוכיות זמן הריצה שלו $O(|E|+|V|)$
האלגוריתם של בלמן פורד: מטפל ב$SSSP$ במקרה הממושקל. מניח שיתכנו משקלים שליליים אך לא יתכנו מעגלים שליליים. סיבוכיות זמן הריצה שלו $O(|E| \times |V|)$
האלגוריתם של דייקסטרה: מטפל ב$SSSP$ במקרה הממושקל, אך מניח שלא יתכנו משקלים שליליים. סיבוכיות זמן הריצה שלו $O(|V|log|V|+|E|)$ עם פיבונאצ'י ועם ערימה בינארית $O(|V|log(|V|)+|E|log(|V|))$. **אלגוריתם חמדן.**

5 הרצאה 5: $shorts\ path - APSP$

5.1 מבוא לכפל מטריצות מהיר

קלט: שתי מטריצות $A, B \in \mathbb{F}^{n \times n}$. נסמנן: $A = \{a_{ij}\}, B = \{b_{ij}\}$
פלט: לחשב את $C = A \times B$, כלומר את $C = \{c_{ij}\}$

דרך נאיבית לפתרון הבעיה: $C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$, כל תא יעלה לחישוב לפי הנוסחה $O(n)$ והמטריצה בגודל $n \times n$ כלומר ישנם n^2 תאים ומכאן עלות האלגוריתם $O(n^3)$.

אחת מהשאלות החשובות בעולם התאוריה של מדעי המחשב, היא מהו הזמן הכי מהיר שבו ניתן לחשב את המטריצה C.
הבחנה ראשונה: זמן מינימלי לפתרון הבעיה הינו $\Omega(n^2)$ כיוון שגודל הפלט הינו $O(n^2)$. וכן הוא לינארי בגודל הקלט שלו.

האלגוריתם של שטרסן: רץ בזמן $O(n^{log_2 7}) = O(n^{2.81})$ - האלגוריתם שביצענו בעצמנו בתרגיל (1) שאלה (1), מגדירים כפלים חדשים ומצליחים להגיע לנוסחת הנסיגה $T(n) = 7T(\frac{n}{2}) + O(n^2)$.

בשנת 1986 נפל דבר, ו$Coppersmith–Winograd$ הצליחו להגיע לסיבוכיות זמן של $O(n^{2.376})$.
לאחר מכן הסיבוכיות ירדה ל$O(n^{2.37287})$

נסמן בω את האקספוננט של האלגוריתם הכי מהיר שקיים לפתרון הבעיה. בהכרח, $\omega \leq 2.37287$ כי כיום ידוע אלגוריתם שפותר בזמן זה, וכן נקבל סה"כ כי:

$$2 \leq \omega \leq 2.37287$$

מכאן, האלגוריתם הכי מהיר לכפל מטריצות ריבועיות מגודל $n \times n$ עולה $O(n^{\omega})$ זמן.

אנחנו לא נלמד על כפל מטריצות מהיר בקורס. עם זאת, ישנם הרבה אלגוריתמים שמשתמשים בכפל מטריצות מהיר בתור פרוצדורה. כלומר, כ"קופסה שחורה", אליה נכנס קלט, מתבצע FMM $(Fast\ Matrix\ Multication)$, ויוצא פלט. נשתמש כהנחה שכפל מטריצות עלותה $O(n^{\omega})$ ונעזר בכך להוריד זמן ריצה של אלגוריתמים.
כעת נראה את האלגוריתם של סיידל, באשר $G = (V, E)$ הוא גרף לא מכוון ולא ממושקל, ונרצה לפתור את $APSP$ על G שיעשה בדיוק אותו דבר שתיארנו כאן ויגיע לסיבוכיות זמן של $O(|V|^{\omega} log(v))$. כמובן שאם בעתיד הערך של ω יקטן, גם הזמן של סיידל יקטן כי הוא משתמש בכפל מטריצות באופן ישיר.

5.2 הגדרת $APSP$

קלט: גרף $G = (V, E)$ מכוון/ לא מכוון ופונקציית משקלים $w : E \to \mathbb{R}$. מניחים כי $V = \{1, ..., n\}$.

פלט: שתי מטריצות

1. מטריצת מרחקים $D = (d_{i,j})$ מגודל $|V| \times |V|$ כך ש$d_{i,j} = \delta(i, j)$
2. מטריצת קודמים $\pi = (\pi_{i,j})$ מגודל $|V| \times |V|$ כך ש$\pi_{i,j} = null$ אם $i = j$ או שאין מסלול מi לj, אחרת $\pi_{i,j}$ הוא קודקוד שקודם לj במסלול הקצר ביותר מi לj.

פתרון נאיבי: אם יש קשתות שליליות אך אין מעגל שלילי, להריץ את בלמן פורד מכל קודקוד ולקבל זמן ריצה של $O(|V| \times |E|) \times |V| = O(|V|^2|E|)$

הנחה: נניח כי הגרף מיוצג ע"י מטריצת שכנויות.

5.3 האלגוריתם של $Floyd - Warshall$

הערה: האלגוריתם יניח שיתכנו קשתות שליליות, אך אין מעגל שלילי.
האלגוריתם פותר את הבעיה באמצעות תכנות דינמי. נרצה לחשב $|V|^2$ מרחקים - בין כל זוג (i, j). נרצה להגדיר נוסחת נסיגה לכל מסלול שבכל שלב מתבססת על קלט קטן יותר. הקלט יהיה הקודקודים בהם ניתן להעזר במסלול.

הגדרה: יהי $P = (v_1, ..., v_\ell)$ מסלול. אזי, הקודקודים $v_2, ..., v_{\ell-1}$ הם קודקודי ביניים של P.

נגדיר את נוסחת הנסיגה באמצעות קודקודים הביניים בהם ניתן להשתמש. כלומר, עבור הרמה הk אנו רוצים לדעת את אורכי המסלולים הקצרים ביותר מבין זוגות המסלולים בגרף כאשר מותר להעזר כקודקודי ביניים רק בקודקודים $\{1, ..., k\}$.
יהי P מסלול קצר ביותר מi לj מבין המסלולים המשתמשים בקודקודים $\{1, ..., k\}$. נראה כי כיוון שG לא מכיל מעגל שלילי אנו יכולים להניח בה"כ כי P מסלול פשוט. ובהכרח מתקיים אחד מהשניים:

א. אם k אינו קודקוד ביניים בP אזי המסלול P משתמש רק בקודקודים $\{1,...,k-1\}$ ובהכרח אורכו קצר ביותר מבין כלל המסלולים המשתמשים ב$\{1,....,k-1\}$ (ואפשר למצוא את אורכו ברקורסיה)

ב.אם k הינו קודקוד ביניים בP, אזי ניתן לחלק את P לשני מסלולים. מסלול p_1 מi אל k ומסלול p_2 מk אל j. כל אחד מהמסלולים p_1, p_2 לא מכיל את k כקודקוד ביניים (הוא הקצה) ולכן בהכרח משתמש רק בקודקודים מתוך $\{1,...,k-1\}$ כקודקודי ביניים. כיוון שתת מסלול של מסלול קצר ביותר הוא מסלול קצר ביותר,בהכרח p_1 הוא מסלול קצר ביותר מבין המסלולים המשתמשים ב$\{1,...,k-1\}$ מi אל k וp_2 מסלול קצר ביותר מבין המסלולים המשתמשים ב$\{1,...,k-1\}$ מk אל j.

לפיכך, נגדיר את המטריצות $D^{(0)}, D^{(1)}, ..., D^{(n)}$ כך ש$d_{ij}^{(k)}$ הוא אורך המסלול הקצר ביותר מi לj מבין המסלולים שקודקודי הביניים שלהם מהקבוצה $\{1,...,k\}$. נגדיר זאת כך:

$$d_{ij}^{(k)} = \left\{ \begin{array}{cc} w_{ij} & k=0 \\ min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)} & k>0 \end{array} \right\}$$

אכן הנוסחה מתאימה למה שתיארנו קודם. בדיוק שני אפשרויות לפנינו. להלן האלגוריתם:

Ployd-Warshall(W):
1. n→rows[W]
2. $D^{(0)} = W$
3. for $k = 1$ to n
a. for $i = 1$ to n
b. for $j = 1$ to n
$d_{ij}^{(k)} = min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$
4. return $D^{(n)}$

הסבר: אנו מקבלים פונקציית משקלים ומגדירים את $D^{(0)}$ להיות פונקציית המשקלים עצמה, כלומר כל איבר אם משתמשים רק באיברים מ1 עד... אפס? אין כאלו איברים - לכן הדרך לעבור מi לj באשר לא משתמשים בקודקודי ביניים זה בדיוק אם קיימת קשת בניהם. לאחר מכן רצים בשלוש לולאות וממש מחשבים את נוסחת הנסיגה. לבסוף כמובן מחזירים את $D^{(n)}$ שהיא המטריצה שמשתמשת באיברים $\{1,...,n\}$

זמן הריצה: $O(|V|^3)$ כמובן.

כיצד מחשבים את π?

א. תוך כדי החישוב של מטריצת המרחקים ניתן לזכור את הבחירות במטריצות עזר $\pi^{(k)}$

ב. ניתן לבנות את המטריצה π מהמטריצה $D^{(n)}$ בזמן $O(n^3)$. כל תא $\pi_{i,j}$ יעלה $O(|V|)$ זמן וסה"כ אכן נגיע ל$O(|V|^3)$. נראה כי $\pi_{ij} \in \{x|d_{ij} = d_{ix} + w_{xi}\}$. נסתכל על זוג קודקודים (i,j). ידוע כי d_{ij} שומר את המרחק בניהם,כלומר את אורך המסלול הקצר ביותר מi לj.אם $w(i,j) = d(i,j)$ אזי מסלול מתאים הוא כמובן פשוט הקשת בניהם. אחרת, אנו יודעים שקיים מסלול אחר קצר יותר. נסמן את הקודקוד הקודם לj במסלול כx ואזי מתקיים $d_{ij} = d_{ix} + w(x,j)$ ואם כך נוכל לסמן $\pi_{ij} = x$. לכן במקרה הגרוע ביותר בהינתן $D^{(n)}$ החישוב של π_{ij} דורש מעבר על כל הקודקודים והבדיקה הנ"ל. סה"כ אכן $O(|V|^3)$ לחישוב π.

5.4 הסגור הטרנזטיבי של גרף מכוון

עבור גרף מכוון $G = (V,E)$ הסגור הטרנזטיבי $G^* = (V, E^*)$ הוא גרף על אותם קודקודים כך שאם בG יש מסלול מקודקוד i לקודקוד j אזי בG^* ישנה קשת בניהם.

$$E^* = \{(u,v) | u \rightsquigarrow_G v\}$$

דרך אחת לחישוב E^* היא להשתמש באלגוריתם של פלויד וורשל, עם פונקציית המשקל $w(e) = 1$ לכל $e \in E$ (וכך אנו מודדים אורך של מסלול ולא צלעות). לבסוף, כל זוג שהמרחק בניהם יהיה קטן מאנסוף ניצור לו קשת מתאימה בE^*.

מבחינה פרקטית,ניתן לשפר ולהשתמש במטריצות בוליאניות.כל כניסה של המטריצה יכולה להיות רק אפס או אחד. במקרה זה החישוב יהיה

$$t_{ij}^{(k)} = \left\{ \begin{array}{cc} (i,j) \in E \vee (i=j) & k=0 \\ t_{ij}^{(k-1)}, \vee (t_{ik}^{(k-1)} \wedge d_{kj}^{(k-1)}) & k>0 \end{array} \right\}$$

האלגוריתם מקביל לחלוטין לאלגוריתם לחישוב מרחקים, השיפור הוא רק בכך שהמחשב צריך לבצע פעולות על ביטים במקום על מספרים.

סה"כ זמן הריצה לחישוב הסגור הטרנזטיבי הינו $O(|V|^3)$

דרך אחרת לחישוב הסגור הטרנטזיבי היא להריץ BFS מכל קודקוד ולקבל זמן ריצה $O(|V|(|V| + |E|)$

5.4.1 חישוב הסגור הטרנטיזיבי בזמן $O(|V|^\omega)$

נזכר כי בהרצאה לקחנו את מטריצת השכניות D והעלנו בחזקה. D^k זה המטריצה של כל המסלולים באורך k בדיוק בגרף. אם כן, נשים לב כי סגור טרנזטיבי הוא כל המסלולים עד לאורך k כלשהו. לכן הרעיון יהיה להוסיף אחדות על האלכסון. כך, אם קיים מסלול בין שני קודקודים, נוכל תמיד ללכת בלולאה העצמית שבגרף. ופורמלית נסתכל על המטריצה $A \vee I$ באשר I היא מטריצת היחידה, ו$A = D$. אנחנו יודעים שלאחר $|V|$ פעולות כפל נקבל את כל המסלולים עד אורך $|V|$. נראה כי סך הכל אם נעלה את המטריצה $(A \vee I)^{|V|}$ בחזקת $|V|$ זה יעלה לפי מבנ"ת $O(log|V|)$ כפול הזמן לכפל מטריצות וסה"כ $O(|V|^\omega log|V|)$. כעת ־ נתמקד בדרך להוריד את הlog.

קלט: גרף מכוון $G = (V,E)$

פלט: סגור טרנזטיבי $G^* = (V, E^*)$

בשביל להפטר מהlog **נניח כמה הנחות לגיטימיות** לכל גרף:

1. G הוא DAG (בזמן $O(|E| + |V|)$ ניתן לחשב את G^{SCC} והוא בהכרח DAG, לכן אם G לא DAG נוכל לפתור עבור G^{SCC}, שכן בתוך כל רכיב קשיר אנו יודעים כיצד נראה הסגור קלין (בהכל יש קשתות אחדות במטריצה כי יש מסלול))
2. G ממויין טופולוגית (אם לא, ניתן למיין טופולוגית ב$O(|E| + |V|)$).
3. V הוא חזקה שלמה של 2. (ברור כי ניתן להניח כי זאת שכן נוכל להוסיף קודקודי דמה ־ כמו אפסים)

נשים לב כי כיוון שהוא ממויין טופולוגית, זו בהכרח מטריצה משולשית עליונה (מתחת לאלכסון יש אפסים כי לא יתכנו קשתות בכיוון זה). נחלק את השאר לX,Y,Z באשר X וZ מטריצות משולשות עליונות.

$$(A \vee I) = \begin{pmatrix} & X & & Y \\ & & & \\ 0 & 0 & & Z \\ 0 & 0 & & \end{pmatrix}$$

טענה.

$$(A \vee V)^* = \begin{pmatrix} X^* & X^* \times Y \times Z^* \\ \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} & Z^* \end{pmatrix}$$

כלומר, בשביל לחשב את החזקה כל שנדרש הוא לבצע הפרד ומשול ־ לחשב את החלקים השונים של המטריצה. נקבל הפרד ומשול קלאסי בזמן ריצה: $n = |V|$

$$T(n) = 2T(\frac{n}{2}) + O(n^\omega) = O(n^\omega)$$

וקיבלנו את זמן הריצה הנדרש.

הערה. מה זה אומר קשת בגרף המקורי באזור Y? אם דולק שם ביט, מדובר בקשת בהכרח שמתחילה בצד X ונגמרת בZ. בהכרח ־ ולכן במטריצה שבחזקה, בהכרח כופלים את X^* (מתחילים בX), מכפילים בY בשביל לעבור על הקשת בין האזורים ולבסוף מכפילים בZ^* בשביל לסיים בZ.

5.5 האלגוריתם של $Jhonson$

הערה: האלגוריתם יניח שיתכנו קשתות שליליות, אך אין מעגל שלילי.

ראינו כי פלויד וורשל רץ בזמן $O(|V|^3)$ ומטפל במקרה הממושקל ללא מעגלים שליליים אך עם קשתות שליליות.

כמו כן ראינו כי אם ישנן קשתות שליליות אך אין מעגלים שליליים ניתן להריץ בלמן פורד מכל קודקוד ולקבל $O(|V|^2|E|)$

אם בגרף אין קשתות שליליות, דייקסטרה רץ בזמן $O(|V|log|V| + |E|)$ ואם נריץ דייקסטרה מכל קודקוד נקבל $O(|V|^2log|V| + |E||V|)$.

מה אם הגרף מכיל קשתות שליליות אך לא מעגל שלילי? נרצה להריץ דייקסטרה מכל קודקוד.
אבל: צריך לעשות שינוי בגרף ־ שלא יכיל קשתות שליליות. אחרת: אי אפשר להשתמש בדייקסטרה.
נגדיר פונקציית משקל חדשה $\hat{w}$ שתקיים:

א. $\hat{w} : E \to \mathbb{R}^+$

ב. מסלול p מקודקוד u לv יהיה קצר ביותר לפי $\hat{w}$ $\iff$ מסלול p מקודקוד u לv יהיה קצר ביותר לפי w.

תהי $h : V \to \mathbb{R}$ פונקציית משקל לקודקודים. נגדיר:

$$\hat{w}(u, v) = w(u, v) + h(u) - h(v)$$

למה 3. יהי $G = (V, E)$ גרף מכוון עם פונקציית משקל $w : E \to \mathbb{R}$ ותהי $h : V \to \mathbb{R}$ ולכל $(u, v) \in E$ נגדיר $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$.
יהי $P = (v_0, ..., v_k)$ מסלול מv_0 לv_k. אזי P מסלול מינימלי ע"פ w אמ"מ P מינימלי עבור $\hat{w}$.בנוסף, G מכיל מעגל שלילי ע"פ w אמ"מ G מכיל מעגל שלילי ע"פ $\hat{w}$.

הוכחה: נרצה להוכיח כי מתקיים $\hat{w}(p) = w(p) + h(v_0) - h(v_k)$

$$\hat{w}(p) = \sum_{1 \le i \le k} \hat{w}(v_{i-1}, v_i) = \sum_{1 \le i \le k} w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i) =$$

$$\sum_{1 \le i \le k} w(v_{i-1}, v_i) + \sum_{1 \le i \le k} h(v_{i-1}) - h(v_i) = w(p) + h(v_0) - h(v_k)$$

שכן קיבלנו סכום טלסקופי שמצטמצם. נראה כי פיתוח זה נכון עבור כל מסלול p מv_0 לv_k. כלומר המשקל של כל מסלול מv_0 לv_k ע"פ $\hat{w}$ הוא בדיוק המשקל לפי w בתוספת $h(v_0) - h(v_k)$. ערך זה איננו תלוי במסלול, שכן קבוע מראש, ולכן בהכרח אם p מינימלי לפי $\hat{w}$ בבירור מינימלי לפי w ולהפך.

בפרט, אם p הוא מעגל מתקיים $v_0 = v_k$ ואז $\hat{w}(p) = w(p)$ ולכן אם מעגל הוא שלילי לפי w הוא שלילי לפי $\hat{w}$. כנדרש.

מסקנה: אם נחשב את מטריצת המרחקים לפי $\hat{w}$ נוכל במעבר אחד על המטריצה ($(O(|V|^2)$) לחשב את מטריצת המרחקים לפי w. אם כן, כל שנותר הוא למצוא משקל לקודקודים כך ש$\hat{w}$ תהיה תמיד חיובית.

אנו רוצים שהאלגוריתם יזהה אם יש מעגלים שליליים, ואם אין מעגלים שליליים, ימצא את המסלולים הקצרים ביותר. איזה אלגוריתם אנו מכירים שמבצע זיהוי מעגלים שליליים, ויודע להחזיר תשובה משמעותית אם אין מעגלים שליליים? האלגוריתם של בלמן־פורד! הפלט של אלגוריתם של בלמן־פורד משייך לכל קודקוד מספר ־ המרחק שלו מקודקוד המקור. הדבר היחיד שנותר להגדיר הוא מיהו קודקוד המקור לאלגוריתם.

נוסיף לגרף קודקוד חדש s ונוסיף קשת מכוונת מs לכל קודקוד שמשקלה יהיה אפס.

פורמלית, נגדיר $G' = (V \cup \{s\}, E')$ באשר $E' = \{E \cup \{(s,u) | u \in V\}\}$ וכן $w' : E \to \mathbb{R}$ המוגדרת כך:

$$w'(u,v) = \left\{ \begin{array}{cc} w(u,v) & (u,v) \in E \\ 0 & u = s \end{array} \right\}$$

כפי שכבר נאמר, הרצת האלגוריתם של בלמן פורד תחשב לכל קודקוד $v \in V$ את המרחק $\delta_{G'}(s,v)$ ולאחר מכן נגדיר $h(v) = \delta_{G'}(s,v)$.

למען האינטואיציה: נשים לב כי לכל קודקוד קיים מסלול מs לv שמשקלו אפס. לכן בהכרח קיים מסלול, אם קיים קטן יותר אזי בהכרח שהוא במשקל שלילי. נראה כי

$$\hat{w}(u,v) = w(u,v) + h(u) - h(v) = w(u,v) + \delta_{G'}(s,u) - \delta_{G'}(s,v)$$

עלינו להוכיח כי $\hat{w}(u,v) \ge 0$. נסתכל על המסלול הקצר ביותר בG' מs אל v. כיוון שישנה קשת מu אל v (הרי אנו מסתכלים על משקל בקשת זו, שמוגדר רק על קשתות שקיימות) ע"פ אי שוויון המשולש מתקיים $\delta_{G'}(s,v) \le \delta_{G'}(s,u) + w(u,v)$ ובהעברת אגף נקבל $0 \le w(u,v) + \delta_{G'}(s,u) - \delta_{G'}(s,v) = \hat{w}(u,v)$

סה"כ ־ הוכחנו כי $\hat{w}$ היא פונקצייה שתמיד אי שלילית, וכן היא משמרת מסלולים קצרים ביותר. והרעיון של ג'ונסון ללא ספק עבד.

להלן האלגוריתם:

אלגוריתם 1 ג'ונסון($G=(V,E)$)

(א) צור קודקוד חדש s והוסף קשתות במשקל 0 מ־s לכל קודקוד $v \in V$.

(ב) הרץ את האלגוריתם של בלמן־פורד מקודקוד s.

(ג) אם האלגוריתם של בלמן־פורד מצא מעגל שלילי ־

i. החזר "מעגל שלילי".

(ד) אחרת

i. לכל $u \in V$

א'. שמור $h(v)=\delta(s,v)$

ii. הגדר לכל $(u,v)\in E$

א'. $\hat{w}(u,v)=w(u,v)+\delta(s,u)-\delta(s,v)$

iii. תהי $D=(d_{uv})$ מטריצה בגודל $n\times n$

iv. לכל $u \in V$

א'. הרץ את האלגוריתם של דייקסטרה עם הפונקציה $\hat{w}$ מ־u כקודקוד מקור ושמור את $\hat{\delta}(u,v)$

ב'. עבור כל $v\in V$

ג'. $\hat{\delta}(u,v)+h(v)-h(u)\to d_{uv}$

v. החזר את D

זמן הריצה: האלגוריתם מריץ בלמן פורד, ולאחר מכן $|V|$ פעמים מריץ דייקסטרה. סה"כ זמן הריצה $O(|V||E|+|V|^2log|V|)$

5.6 האלגוריתם של $Seidel$

5.6.1 הגדרת הבעיה

קלט: גרף $G=(V,E)$ לא מכוון ולא ממושקל.
פלט: $APSP$.

נשתמש בהנחה מקלה ־ הגרף G קשיר. מכאן נקבל $|V|-1\le|E|$. ניתן להניח זאת כי אם הגרף לא קשיר, נוכל לפתור את הבעיה עבור כל רכיב קשירות בנפרד (שכן מחשבים מרחק מסלול קצר ביותר בין קודקודים.)

כמה זמן לוקח להמיר גרף שמיוצג ע"י רשימת שכנויות לייצוג ע"י מטריצת שכנויות? $O(|V|^2)$.
נניח כי הגרף מיוצג ע"י מטריצת שכנויות. כדקלמן ־

$$A_{u,v}:=\left\{\begin{array}{cc}1 & (u,v)\in E\\ 0 & else\end{array}\right\}$$

המטריצה A סימטרית, כיוון שG אינו מכוון ולכן $A_{u.v}=A_{v,u}$

5.6.2 כפל מטריצות בוליאני

בכפל מטריצות רגיל מתקיים לפי עדי בן צבי ־ $C_{ij}=\sum_{k=1}^{n}A_{ik}B_{kj}$.

בכפל מטריצות בוליאני (BMM) שנקרא $Boolean\ Metrix\ Multipcation$:

$$C_{ij}=\bigvee_{k=1}^{n}a_{ik}\wedge b_{kj}$$

כלומר, כאשר מייצגים מטריצות בוליאניות אנחנו מסתכלים על המיקום הij. הוא מכפלה של השורה הi במטריצה A והעמודה הj במטריצה B, מספיק שיהיה קיים אינדקס אחד k עבורו הערך הk בשורה הi והערך הk בעמודה הj הוא 1, אזי נגדיר $c_{ij} = 1$. אחרת, $c_{ij} = 0$.

נראה כי ניתן לחשב BMM באמצעות FMM: מדוע? במטריצות בוליאניות, נקבל ערך ששונה מאפס אמ"מ היה ערך k עבורו $a_{ik}, b_{kj} \neq 0$. מכאן שאם יצא לנו ערך $c_{ij} \neq 0$ נגדירו כעת 1, ואם יצא 0 הוא ישאר אפס. עלות BMM תהיה $O(n^{\omega})$.

באלגוריתם של סיידל, אנחנו נשתמש במטריצה A שהינה מטריצת שכנויות, ונגדיר את המטריצה:

$$A' = A^2 \vee A$$

באשר A^2 היא מטריצת השכנויות שמוכפלת בעצמה, בכפל בוליאני.

נרצה להבין מה המשמעות של A^2. מכפלה של מטריצת השכנויות עם עצמה: A^2_{ij} משמעות הדבר היא שלקחנו את השורה הi: כל השכנים של v_i, ולקחנו את השורה הj - כל השכנים של v_j, וחישבנו:

$$A^2_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge a_{kj}$$

כלומר - מסתכלים על כל השכנים של i, כל השכנים של j: אם מוצאים k מסויים שהוא שכן של שניהם, המשמעות היא שיש מסלול באורך 2 בין i לj.

טענה: במיקום $A^2_{i,j} = 1$ אמ"מ קיים מסלול באורך 2 בין קודקוד i לקודקוד j.

המשמעות היא, שבגרף שמיוצג ע"י A^2 (נתייחס אליה גם כמטריצת שכנויות) יש קשת בין קודקוד i לקודקוד j אמ"מ יש מסלול באורך 2 בין i לj.

נזכר כי: רצינו להגדיר את המטריצה כך $A' = A^2 \vee A$. A^2 משמעות הדבר היא עבור כל המסלולים באורך 2 וA עבור קשתות המקוריות. כאשר נבצע or בין המטריצות, במטריצת הפלט A' יש 1 אמ"מ בין קודקוד i לj יש מסלול באורך 1 (קשת) או מסלול באורך 2.

טענה: $A'_{ij} = 1$ אמ"מ בG קיים מסלול באורך 1 או 2 בין הקודקודים i לj.

הערה חשובה: $A^2_{ii} = 1$ אמ"מ לצומת יש שכן שכן משמעות הדבר כי קיים k עבורו $A_{ik} = 1$ וגם $A_{ki} = 1$. אם נסתכל על זוג הקודקודים $i - j$ עם הקשת בניהם, נראה כי המסלול $i \to j \to i$ הוא גם מסלול באורך 2. שחוזר למקום. משמעות הדבר הינה, שבA^2 על כל האלכסון יהיה אחדות. ומה זה אומר אם קיים מסלול באורך 1 מi לi? שישנה לולאה עצמית. ולכן - **תמיד אנחנו נאפס את האלכסון הראשי של A^2.**

5.6.3 טענות $1, 2$

הגדרה: נגדיר $\delta'(u,v)$ - אורך המסלול הקצר ביותר מu לv בG', כאשר G' הוא הגרף המיוצג ע"י A'.

טענה 1: $\delta'(u,v) = \left\lceil \frac{\delta(u,v)}{2} \right\rceil$

אינטואיציה לטענה: ראינו כי בגרף G' יש קשת בין קודקודים שבG היה בניהם מסלול באורך 2, משמעות הדבר היא שעל כל צלע שהולכים בG' צריך ללכת פי 2 בגרף G. כלומר $x' = 2x$ ומכאן $x = \frac{x'}{2}$.

הוכחה: נסמן בP מסלול קצר ביותר מu לv בG.

אם האורך של p הוא $2k$ עבור $k \in \mathbb{N}$ אזי קיים מסלול מאורך k בין u לv בG', נסמנו P'. כיוון שהמסלול P' מדלג על כל קודקוד שני בP. כלומר אם $P = (v_0, ..., v_{2k-1})$ אזי $P' = (v_0, v_2, v_4, ..., v_{2k-2})$. עלינו להראות כי P' הוא מסלול קצר ביותר בין u לv. נב"ש כי קיים מסלול $\hat{p} = (v_0, ..., v_{m-1})$ כאשר $m < k$. אזי המסלול $p'' = (v_0, u_0, v_1, u_1, ..., v_{m-1}u_{,m-1})$ קיים בG (כי בG' אנחנו יכולים לדלג על קודקוד, בG לא), כלומר מצאנו מסלול בין u לv בG שאורכו $2m < 2k$ בסתירה לכך שP הוא המסלול הקצר ביותר. מסקנה - P' מסלול קצר ביותר בין u לv בG'.

ולכן אכן $\delta'(u,v) = k = \frac{2k}{2} = \frac{\delta(u,v)}{2}$

ההוכחה עבור המקרה האי זוגי - דומה מאוד.

מסקנה: אם למשל $\delta'(u,v) = 4$ אזי $\delta(u,v) = 7 \vee 8$. נרצה להבין מתי $\delta(u,v)$ יקבל איזה ערך. אם $\delta(u,v)$ זוגי הוא יקבל 8, ואם הוא אי זוגי הוא יקבל 7. **נרצה לפתח שיטה שתבדוק האם $\delta(u,v)$ הוא זוגי או אי זוגי.**

נסתכל על קודקוד w שהוא שכן של v. כמו כן, ישנו מסלול קצר ביותר בין u לv. וכן, קיים מסלול קצר ביותר בין u לw וכן לפי א"ש המשולש: $\delta(u,v) - 1 \leq \delta(u,w) \leq \delta(u,v) + 1$ (נשים לב כי באי השוויון השתמשנו פעמיים). מכאן נחלק למקרים:

א. **מקרה ראשון -** $[\delta(u,v) = \delta(u,w)]mod2$ **:** כלומר או ששניהם זוגיים, או ששניהם אי זוגיים.

במקרה זה אנו רואים שחייב להיות כי $\delta(u,v) = \delta(u,w)$. מדוע? מאי שוויון המשולש - ראינו כי ההפרש $|\delta(u,w) - \delta(u,v)| \leq 1$ ומכאן שההפרש שווה ל0 או 1. עם זאת, הזוגיות/אי זוגיות שלהם שווה ולכן לא יתכן שההפרש בניהם הוא אחד. מכאן ההפרש בניהם אפס - כלומר $\delta(u,v) = \delta(u,w)$. כמו כן, זה אומר כי w לא נמצא על המסלול הקצר ביותר מu לv. כמו כן, מכאן נקבל גם כי $\delta'(u,v) = \delta'(u,w)$.

ב. **מקרה שני -** $\delta(u,v)$ **זוגי ו**$\delta(u,w)$ **אי זוגי:**

$$\delta'(u,w) = \left\lceil \frac{\delta(u,w)}{2} \right\rceil =_{\delta(u,w)-odd} \frac{\delta(u,w)+1}{2} \geq_{\delta(u,v)-1 \leq \delta(u,w)} \frac{\delta(u,v)-1+1}{2} = \frac{\delta(u,v)}{2} = \delta'(u,v)$$

כלומר - $\delta'(u,w) \geq \delta'(u,v)$

ג. **מקרה שלישי -** $\delta(u,v)$ **אי זוגי ו**$\delta(u,w)$ **זוגי:**

$$\delta'(u,v) = \left\lceil \frac{\delta(u,v)}{2} \right\rceil =_{\delta(u,v)-odd} \frac{\delta(u,v)+1}{2} \geq_{\delta(u,v) \geq \delta(u,w)-1} \frac{\delta(u,w)-1+1}{2} = \frac{\delta(u,w)}{2} = \delta'(u,w)$$

כלומר - $\delta'(u,v) \geq \delta'(u,w)$

מסקנה: אם $\delta(u,v)$ זוגי, אזי לכל שכן w של v מתקיים $\delta'(u,w) \geq \delta'(u,v)$
אם $\delta(u,v)$ אי זוגי, אזי לכל שכן w של v מתקיים $\delta'(u,w) \leq \delta'(u,v)$

מסקנה מהמסקנה - אם אנחנו יכולים לחשב את כל δ', אנו יודעים לבדוק יחס גדול-קטן בניהם. ומכאן: אנחנו יכולים להכריע האם δ זוגי או שאי זוגי. פרט לבעיה אחת - מה קורה

אם $\delta'(u,v)=\delta'(u,w)$? לשם כך נצטרך להעזר בטענה הבאה.

אנו נמצאים במקרה של $\delta(u,v)$ אי זוגי ו$\delta(u,w)$ זוגי. יודעים כי $\delta'(u,v)\geq\delta'(u,w)$. נראה כי אם נסתכל על שכן מסויים מאוד w של v, נצליח להגיע במקרה זה למסקנה מעט אחרת. נסתכל על המסלול הקצר ביותר מu לv. נסמן בx את השכן של v על המסלול (זה שנמצא קודקוד אחד לפניו במסלול הקצר ביותר). מתכונות מסלולים קצרים ביותר (למה 1) מתקיים

$$\delta(u,v)=\delta(u,x)+1$$

שקול לחלוטין:

$$2k=\delta(u,x)=\delta(u,v)-1$$

נסמן את הביטוי הנ"ל ב$2k$. אכן $\delta(u,x)$ זוגי כי $w=x$ במקרה זה. מכאן $\delta(u,v)=2k+1$. לפי ההגדרה:

$$\delta'(u,x)=\left\lceil\frac{\delta(u,x)}{2}\right\rceil=\left\lceil\frac{2k}{2}\right\rceil=\lceil k\rceil=k$$

$$\delta'(u,v)=\left\lceil\frac{\delta(u,v)}{2}\right\rceil=\left\lceil\frac{2k+1}{2}\right\rceil=\left\lceil k+\frac{1}{2}\right\rceil=k+1$$

מכאן ניתן לראות כי $\delta'(u,x)<\delta'(u,v)$. כלומר - בהינתן שנדע למצוא את השכן של v על המסלול הקצר ביותר מu לv. נוכל לדעת כי במקרה בו $\delta(u,v)$ אי זוגי ו$\delta(u,w)$ זוגי אם נסתכל על אותו שכן כזה x, יתקיים $\delta'(u,x)<\delta'(u,v)$ (ולא יתקיים שוויון בניהם), נשתמש בזה בהוכחה הבאה.

הרעיון של סיידל היה להסתכל על כל השכנים יחד של v.

טענה 2: $\delta(u,v)$ זוגי $\Longleftrightarrow\sum_{w\in\Gamma(v)}\delta'(u,w)\geq deg(v)\times\delta'(u,v)$

(מדוע אנחנו זקוקים לטענה זו? אנו יודעים להכריע אודות הדרגה. אם נצליח לחשב (ונצליח, באופן רקורסיבי) את הΣ הנ"ל, נדע להכריע האם $\delta(u,v)$ הינו זוגי. אם לא - הוא בהכרח אי זוגי.)

הוכחה:

$\Longleftarrow$ אם $\delta(u,v)$ זוגי - אזי לכל שכן w של v יתקיים $\delta'(u,w)\geq\delta'(u,v)$ (כפי שראינו קודם לכן), ולכן

$$\sum_{w\in\Gamma(v)}\delta'(u,w)\geq\sum_{w\in\Gamma(v)}\delta'(u,v)=deg(v)\delta'(u,v)$$

$\Longrightarrow$ נראה קונטרה פוזיטיב. נניח $\delta(u,v)$ אי זוגי. אזי כפי שראינו מעלה במקרה זה לv יש שכן x שעבורו $\delta'(u,x)<\delta'(u,v)$ וכן לכל שכן w של v מתקיים $\delta'(u,w)\leq\delta'(u,v)$

$$\sum_{w\in\Gamma(v)}\delta'(u,w)=\delta'(u,x)+\sum_{w\in\Gamma(v)\setminus x}\delta'(u,w)$$

החלק הימני של הביטוי $(deg(v)-1)\times\delta'(u,v) \geq$ וכן החלק השמאלי $\delta'(u,v) >$ ולכן נקבל

$$\sum_{w\in\Gamma(v)}\delta'(u,w)=\delta'(u,x)+\sum_{w\in\Gamma(v)\setminus x}\delta'(u,w)<\delta'(u,v)deg(v)$$

כפי שרצינו להראות.

5.6.4 האלגוריתם

עוד לפני שנדון באלגוריתם - נרצה לראות אלגוריתם גנרי:

```
ALG1(A)
if A is all 1s except for the diagonal
      return A
else Δ' ← ALG1(A² ∨ A)
      for u, v ∈ V
            if δ(u, v) is odd
                  δ(u, v) = 2δ'(u, v) − 1
            else δ(u, v) = 2δ'(u, v)
      return Δ
```

האלגוריתם של סיידל יתבסס על אלגוריתם גנרי זה. אלגוריתם זה פשוט יותר להבנה - האלגוריתם מקבל את מטריצת השכנויות A. האלגוריתם יחזיר מטריצה Δ באשר:

$$\Delta_{i,j}=(\delta(i,j))$$

ראשית, ישנו תנאי עצירה: אם G הוא קליקה, נרצה להחזיר את A. (מדוע? בקליקה לכל $u\neq v\in V$ מתקיים $\delta(u,v)=1$, במקרה זה מטריצת השכנויות של הקליקה הינה 0 באלכסון ובכל שאר המקומות 1. במצב זה - זה בדיוק מטריצת ה$APSP$ שנרצה להחזיר).

אחרת, אנחנו נרצה לגשת שוב לאלגוריתם עם $A'=A^2\vee A$. ערך זה, יכנס לΔ'. נשים לב כי זה יתבצע בתחילת האלגוריתם שוב ושוב, עד שנקבל $(\Delta')^n$ עבור n כלשהו, באשר הוא קליקה.

לאחר מכן, נעבור על כל זוג קודקודים בV. אם $\delta(u,v)$ הוא אי זוגי, אזי $\delta(u,v)=2\delta'(u,v)-1$. אחרת, $\delta(u,v)=2\delta'(u,v)$ [נובע ישירות מטענה 1 שראינו], לבסוף מחזירים את Δ. הרעיון של סיידל התבסס על כיצד אנחנו מכריעים אודות הזוגיות של $\delta(u,v)$ בשביל לבצע מה שעשינו באלגוריתם הגנרי.

כעת, נראה את האלגוריתם של סיידל:

```
SEIDEL(A)
1  if A is all 1s except for the diagonal
2        return A
3  else Δ' ← SEIDEL(A² ∨ A)
4        M ← Δ' · A
5        for u, v ∈ V
6              if m_{u,v} < deg(v)δ'(u, v)
7                    δ(u, v) = 2δ'(u, v) − 1
8              else δ(u, v) = 2δ'(u, v)
9        return Δ
```

האלגוריתם זהה בהתחלה לגנרי, ובשורה 6 אנחנו משתמשים בטענה 2: אם $m_{u,v} < deg(v)\delta'(u,v)$ אזי אנחנו במקרה האי זוגי. מכאן, ישירות לפי טענה 2 אפשר להבין לבד מהו $m_{u,v}$: $\sum_{w\in\Gamma(v)} \delta'(u,w)$.

בשורה 4 אנחנו מגדירים את M: היא מכפלה של מטריצת השכנויות המקורית A עם המטריצה Δ' - נראה כי במיקום הu,v ישנה מכפלה של כל השורה הu בΔ' עם העמודה הv בA. נראה כי כיוון שהמכפלה בוליאנית, המכפלה של העמודה והשורה יתנו לנו את סכום כל הδ' של הקודקודים שהם שכנים של v. מדוע? השורה הu היא שורה שמחזיקה את האיברים $(\delta'(u,v_1),....,\delta'(u,v_n))$, מכאן שערך שכזה יכנס למכפלה אמ"מ v_i הוא שכן של v. מכאן שהמכפלה הנ"ל תניב את $\sum_{w\in\Gamma(v)} \delta'(u,w)$. כלומר,

$$m_{u,v} = \sum_{w\in\Gamma(v)} \delta'(u,w)$$

ומכאן קיבלנו את האלגוריתם שמבוסס ישירות על טענה 2 - אם $\sum_{w\in\Gamma(v)} \delta'(u,w) < deg(v)\delta'(u,v)$ המשמעות היא כי $\delta(u,v)$ אי זוגי ולכן $\delta(u,v) = 2\delta'(u,v) - 1$. אחרת, $\delta(u,v)$ זוגי ולכן $\delta(u,v) = 2\delta'(u,v)$.

סיבוכיות זמן ריצה:

נשים לב כי ניתן לחשב את דרגות הקודקודים ב$O(|V|^2)$ זמן באופן טריוואלי - ניצור מערך של קודקודים, עבור כל קודקוד $v \in V$ נעבור על השורה המתאימה במטריצת השכנויות ונסכום את מס' הקודקודים u שקיימת $e = (v,u)$ בניהם. נעבור על $O(|V|)$ קודקודים ובכל פעם שכזו נעבור על $O(|V|)$ קודקודים וסה"כ זמן הריצה יהיה $O(|V|^2)$.

מכאן, שבבירור ניתן לחשב את שורה 6 באלגוריתם. נראה כי שורות $4 - 8$ באלגוריתם עולות $O(|V|^\omega)$ זמן, כיוון שמבצעים מעבר על $|V|$ קודקודים ובהם מבצעים פעולות שהינם $O(1)$ וכן כופלים בBMM שתי מטריצות, מה שעולה $O(|V|^\omega)$ זמן. סה"כ $|V| + |V|^\omega = O(|V|^\omega)$ כי $\omega \geq 2 > 1$.

מהי נוסחת הנסיגה של האלגוריתם? מתי נגיע לתנאי העצירה?

הבחנה: אם אורך המסלול הקצר ביותר בG הוא ℓ אזי בG' אורך המסלול הקצר ביותר מu לv הוא $\lceil \frac{\ell}{2} \rceil$. מכאן, שלאחר $log(|V|)$ קריאות רקורסיביות אורך המסלול הקצר ביותר הגרף הוא

קליקה. מדוע? אורך המסלול הקצר ביותר הוא לכל היותר $|V|-1$ קשתות. אנחנו נרצה לדעת מתי נגיע לקליקה ־ כלומר מתי אורך המסלול הקצר ביותר יהיה 1. מכאן ש

$$\frac{|V|-1}{2^n}=1 \Longleftrightarrow |V|-1=2^n \Longleftrightarrow n=log(|V|-1)=O(log|V|)$$

וסה"כ נקבל כי סיבוכיות זמן הריצה היא מס' האיטרציות $O(log|V|)$ כפול הזמן בכל אטרציה $O(|V|^\omega)$ ונקבל ־

$$O(|V|^\omega log|V|)<O(|V|^3)$$

הבחנה. אם ידוע כי האורך הכי גדול של מסלול בגרף בין שני קודקודים הוא d, אזי סיבוכיות זמן הריצה של האלגוריתם הינה $O(|V|^\omega log(d))$.

הבחנה. גם אם נדע אלגוריתם טוב יותר לBMM, בכל מקרה בכל איטרציה מחשבים את M שהיא לא כפל מטריצות בוליאניות, ולכן בכל מקרה זה זמן הריצה.

5.7 A^*

נתון גרף מכוון וממושקל $G=(V,E)$ וכן שני קודקודים $s,t\in V$. נרצה למצוב מק"ב מs לt. נניח כי $w:E\to\mathbb{R}^+$.

דייקסטרה ירוץ כאן בזמן $O(|V|log|V|+|E|)$.

נרצה לפתור כרגע בעיה בין $source$ ל$target$: רק בין s לt. נשים לב כי טענו שלא קיים פתרון לבעיה זו של מקור יחיד בזמן יותר טוב מ$SSSP$. אך אנחנו לא נדבר על המקרה הגרוע ביותר.

איך נעזר בדייקסטרה? נרצה לעצור באשר t יוצא מהתור כי אז הוכחנו שמצאנו מק"ב מs אל t ־ ואין לנו צורך בהמשך הריצה של דייקסטרה.

כעת נדגיש: הרעיון של A^* הינו רעיון אלגוריתמי וניתן להגדיר אלגוריתמיים שפועלים לפי טכניקה A^*, אנחנו לא נראה אלגוריתם ישיר שפותר A^* אלא כיצד משתמשים בA^* להגדרת בעיות ופתרונם.

ננסה לחשוב על הכיוון הבא ־ נגדיר את פונקציית הפוטנציאל $P:V\to\mathbb{R}$. מטרת הפונקציה היא כשצומת v קרוב אל t אזי P קטן יותר. ואז, מה שנעשה באלגוריתם של דייקסטרה יהיה להוסיף לכל $d[v]$ את $p(v)$ כלומר $d[v]=d[v]+p(v)$, וכך אנחנו בתחילת הריצה של דייקסטרה נתעדף את הצמתים שקרובים אל t. נשים לב ־ כל התוספת הנ"ל פוגעת בכל הנכונות של דייקסטרה. לכן ־ לא נשתמש בזה.

נגדיר פונקציית משקל חדשה:

$$w'(u,v)=w(u,v)+p(v)-p(u)$$

נראה כי

$$w'(P_{u\to v})=w(P_{u\to v})+p(v)-p(u)$$

ולכן כל המסלולים בין u לv עם אותה התוספת $p(v)-p(u)$ ולכן הסדר ביניהם נשאר, ובפרט למק"ב.

עם זאת, יש המון בעיות שאנו זקוקים לטפל בהם - בין צמתים שונים (למשל u לv וu לw) לא מובטח שנשמר הסדר. כמו כן, בשביל שדייקסטרה יעבור נרצה כי $w'(u,v) \geq 0$.

הגדרה: אם לאחר ההתמרה (הגדרת w') כל $w' \geq 0$ אזי p (פונקציית הפוטנציאל) הינה פיזיבילית. כלומר לכל $(u,v) \in E$ $w'(u,v) \geq 0 \iff p$ פיזיבילית.

למה 1: אם p פיזיבילית, ויהי $t \in V$ כך ש $p(t) \leq 0$, אזי $\forall v \in V$: $p(v) \leq \delta(v,t)$

הוכחה: יהי מסלול מv אל t.

$$0 \leq w'(P_{v\to t}) = w(P_{v\to t}) + p(t) - p(v) \leq w(P_{v\to t}) - p(v)$$

וסה"כ נקבל $p(v) \leq w(P_{v\to t})$. בפרט, זה נכון עבור המסלול הקצר ביותר כלומר $p(v) \leq \delta(v,t)$

מה קורה באשר $p(v) = \delta(v,t)$ לכל קודקוד?

$$p_{s\to t} = (s, v_1, ..., v_k, t)$$

נסתכל על (v_i, v_{i+1}) במסלול.

$$w'(v_i, v_{i+1}) = w(v_i, v_{i+1}) + \delta(v_{i+1}, t) - \delta(v_i, t) = \delta(v_i, t) - \delta(v_i, t) = 0$$

משום ש$w(v_i, v_{i+1}) + \delta(v_{i+1}, t) = \delta(v_i, t)$. ולכן - כל קשת במק"ב מ$s \to t$ תהיה במשקל 0.
נשים לב - כיוון שכל המשקלים על המסלול הנ"ל הינם אפס, דייקסטרה ישר ירוץ על המסלול שלנו מs לt. הוא ראשית יוציא את s עם $d[s] = 0$ ואז יבצע סדרת הקלות קודם כל על המק"ב שלנו מs לt. מכאן, שהשאיפה שלנו היא למצוא פונקציית פוטנציאל שמקרבת את $p(v)$ כמה שיותר אל $\delta(v,t)$. (שכן אנחנו לא יודעים את $\delta(v,t)$ שכן בשביל לדעת אותו צריך להריץ דייקסטרה, ואנחנו לא מעוניינים לעשות זאת. נשים לב כי את $\delta(v,t)$ נוכל למצוא אם נריץ דייקסטרה על G^T מt.)

מסקנה מהחישוב: אם נמצא ערך $p(v)$ קרוב מאוד ל$\delta(v,t)$ אזי ערך הקשת יהיה קרוב יותר לאפס.

למה 2: אם p פיזיבלי ו$p(t) > 0$ אזי ניתן להפוך לp' פיזיבלי עם $p'(t) \leq 0$.
הוכחה:
נגדיר

$$p'(v) = p(v) - p(t)$$

ברור כי עבור $p'(t) = 0$. כל שניתן להראות הוא כי p' פיזיבילי. תהי $(u,v) \in E$. אזי,

$$w''(u,v) = w(u,v) + p'(v) - p'(u) = w(u,v) + p(v) - p(t) - p(u) + p(t) = w'(u,v) \geq 0$$

שכן מראש הנחנו כי p פיזיבלי ולכן $w'(u,v) \geq 0$.

דוגמה ראשונה לשימוש ב-A^*:
נרצה לבדוק מרחק קצר ביותר בין שני צמתים במישור. אם ניתן לשכן צמתים במישור, נסתכל על מפה דו מימדית וצומת (x_1,y_1) ו-(x_2,y_2). נסמן $||g(u)-g(v)||$ כמרחק האוקלידי בין הקודקודים והוא $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$.
אם נסתכל על פונקציית פוטנציאל $p(v)=||g(v)-g(t)||$ אזי בהכרח $p(t)=||g(t)-g(t)||=0$.
מכאן $p(t) \geq 0$ ונראה כי היא פיזיבילית -

$$w'(u,v)=w(u,v)+p(v)-p(u)=w(u,v)+||g(v)-g(t)||-||g(u)-g(t)||$$

נראה כי אם נדרוש $w(u,v) \geq ||g(u)-g(v)||$ (**ברור שהכביש יהיה ארוך יותר מהמרחק האווירי**).
נניח שזה מתקיים, אזי

$$\geq ||g(u)-g(v)||+||g(v)-g(t)||-||g(u)-g(t)|| \geq 0$$

באשר המעבר האחרון מגיע מאי שוויון המשולש על מרחקים אוקלידים.

דוגמה שנייה לשימוש ב-A^*: נניח צומת מרכזי f וידוע $\delta(v,f)$ מכל $v \in V$ אליו.)כלומר כמה זמן לוקח לי להגיע מכל קודקוד ליעד מרכזי). נראה כי אני כן מבזבז נתונים ומפעיל דייקסטרה אל f מכל קודקוד אך רוצה להשתמש בנתון זה לדעת יותר - למשל, בהינתן המידע הזה, לאיזה צומת אני מעוניין להגיע מהמיקום הנוכחי שלי, כך שסה"כ המרחק ממני ל-f יהיה קצר ביותר.
בהינתן מידע זה נרצה להגדיר את הפונקציה הבאה:

$$p(v)=\delta(v,f)-\delta(t,f)$$

אכן $p(t)=0$. נוכיח פיזיבלי:

$$w'(u,v)=w(u,v)+p(v)-p(u)=w(u,v)+\delta(v,f)-\delta(t,f)+\delta(t,f)-\delta(u,f)=$$

$$w(u,v)+\delta(v,f)-\delta(u,f) \geq 0$$

שכן המעבר נובע אוטומטית מאי שוויון המשולש.

מסקנה: ניתן להריץ את האלגוריתם של דייקסטרה אם מגדירים לו פונקציית פוטנציאל טובה, פיזיבילית, ולקבל שבאופן ישיר האלגוריתם ראשית יבצע את המסלול שאני מעוניין בו, מה שיוריד את זמן הריצה.

5.7.1 הגדרה פורמלית

קלט: גרף מכוון $G = (V, E)$ עם פונקציית משקל $w : E \to \mathbb{R}$ ופונקציית פוטנציאל $p : V \to \mathbb{R}^+$ (פונקציית הערכות של המרחק לt) ושני קודקודים s, t
פלט: מסלול קצר ביותר מs לt.

טענה. אם לכל $u \in V$ $p(u) = \delta(u, t)$ אזי סריקת A^* תבקר רק בקודקודים על גבי מסלול קצר ביותר מs לt.

הוכחה. נזכר כי המשקל של כל קשת הוא $\hat{w}(u, v) = w(u, v) + p(v) - p(u)$, לכל קודקוד u במסלול קצר ביותר מs לt מתקיים כי $\hat{\delta}(s, u) = \delta(s, u) + p(u) - p(s) = \delta(s, u) + \delta(u, t) -$ $\delta(s, t) = \delta(s, t) - \delta(s, t) = 0$ (המרחק לפי $\hat{w}$) שכן אנו יודעים שאכן u על המסלול. משמע - כל קשת שנמצאת על המסלול הקצר ביותר עלותה אפס.

אם כן, לכל קודקוד v שאינו במק"ב מתקיים $\hat{\delta}(s, v) = \delta(s, v) + p(v) - p(s) = \delta(s, v) +$ $\delta(v, t) - \delta(s, t) > 0$ שכן בהכרח $\delta(s, v) + \delta(v, t) > \delta(s, t)$ שכן v אינו על המסלול הקצר ביותר.

מסקנה - דייקסטרה יבקר קודם בקודקודים שמשקלם אפס, ולכן בהכרח יעבור ראשית על המסלול שלנו מs לt ויתעדף אותו. שכן דייקסטרה מתעדף לפי הקרבה אל s.

נשים לב - לא נובע מטענה זו שזמן הריצה יהיה O של אורך המסלול, שכן יתכן מצב אחד בדיוק בעייתי: אם כל הקשתות בגרף הם אפסים, במקרה הזה דייקסטרה לא בהכרח יתעדף את הקודקודים על המסלול שלנו. לכן במקרה הגרוע ביותר נגיע שוב לזמן הריצה של דייקסטרה. אם כן - הטענה כן אומרת שלא נבקר בקודקוד שלא במק"ב.

הגדרה. תהי פונקציית פוטנציאל $p : V \to \mathbb{R}$, p תקרא קבילה אם $\forall u \in V : p(u) \leq \delta(u, t)$

טענה. אם לכל $u \in V$ $p(u) \leq \delta(u, t)$ אזי סריקת A^* קבילה - כלומר A^* לא תבקר בקודקוד $u \in V$ כך ש$\delta(s, u) + p(u) > \delta(s, t)$ (כלומר יתכן שנבקר בקודקוד לא טוב - אבל לא נבקר בקודקוד ממש לא טוב).

הוכחה.

לכל קודקוד u במסלול קצר ביותר מs לt מתקיים כי $\hat{\delta}(s, u) = \delta(s, u) + p(u) - p(s) \leq$ $\delta(s, u) + \delta(u, t) - p(s) = \delta(s, t) - p(s)$ (המרחק לפי $\hat{w}$)

אם כן, לכל קודקוד v שאינו במק"ב. נב"ש שביקרנו בקודקוד v שאינו במק"ב שמקיים $\delta(s, v) +$ $p(v) > \delta(s, t)$. אם כן,

$$\delta(\hat{s, v}) = \delta(s, v) + p(v) - p(s) > \delta(s, t) - p(s)$$

כלומר, כעת קיבלנו קודקוד שגדול מערך $\delta(s, t) - p(s)$ אך ראינו שקודקודים על המק"ב הם לכל היותר בערך זה ולכן אין סיבה שנבקר בו. בסתירה.

טענה. אם לכל $u \in V$ $p(u) \leq 2\delta(u, t)$ אזי סריקת A^* לא תבקר בקודקוד $u \in V$ כך ש$\delta(s, u) + p(u) > 2\delta(s, t)$

הוכחה.

לכל קודקוד u במסלול קצר ביותר מs לt מתקיים כי $\hat{\delta}(s, u) = \delta(s, u) + p(u) - p(s) \leq$ $\delta(s, u) + 2\delta(u, t) - p(s) = 2\delta(s, t) - p(s)$ (המרחק לפי $\hat{w}$)

אם כן, לכל קודקוד v שאינו במק"ב. נב"ש שביקרנו בקודקוד v שאינו במק"ב שמקיים $\delta(s, v) +$ $p(v) > 2\delta(s, t)$. אם כן,

$$\hat{\delta(s, v)} = \delta(s,v) + p(v) - p(s) > 2\delta(s,t) - p(s)$$

כלומר, כעת קיבלנו קודקוד שגדול מערך $2\delta(s,t) - p(s)$ אך ראינו שקודקודים על המק"ב הם לכל היותר בערך זה ולכן אין סיבה שנבקר בו. בסתירה.

6 הרצאה 6: רשתות זרימה $(Network\ Flow)$

מעט מוטיבציה: נניח ואנחנו חברה שרוצה להעביר נפט נוזלי מנקודה A לנקודה B. בנינו מראש רשת של צינורות שמאפשרות העברה שכזו. בצינור אפשר להכניס את הנפט מצד אחד והוא יכול לצאת מן הצד השני. לכל צינור, ישנה קיבולת אחרת. וכן: לכל צינור ישנו קוטר שונה, קוטר גדול יותר מאפשר להזרים יותר נפט דרך הצינור. אפשר לדמיין את הצינור כקשת בגרף מכוון. היא יכולה לנוע בדיוק בכיוון אחד. ממקור הנפט, בנינו מערך של צינורות שונים. כמו כן, יתכן שעוברים שני צינורות בין שתי נקודות: אחד לכל כיוון. המטרה שלנו היא להעביר כמה שיותר נפט מקודקוד ההתחלה s אל קודקוד היעד t. בכל צינור, אפשר להעביר עד מס' ליטרים מסויים.

המטרה שלנו היא בהינתן רשת הצינורות, לחשב כמה "נפט" ניתן להזרים בשנייה ברשת. נשים לב כי המושג להזרים לא הוגדר היטב.

נשים לב כי בהינתן צנור כנ"ל $s \to a_1 \to a_2 \to a_3 \to a_4 \to t$, אם בכל הצינורות אפשר להזרים מאה, אך בצינור $a_3 \to a_4$ אפשר להזרים רק 2 למשל, זה לא עוזר לי: אני נתקע מאחור עם 98 ליטר נפט. מערכת ה"ביוב" תתקע, אי אפשר לצבור במיקום מסויים נפט/ מים שיצטברו. לכן אסור מראש להעביר שם(!) מאה. נשים לב שצריך מראש לדעת מה כמות הליטר המקסימלית שמותר לנו להעביר במסלול, שלא יווצר מצב של להתקע.

נשים לב כי תתכן למשל רשת זרימה $s \to a_1 \to a_{2\to a_5 \to a_6}^{\to a_3 \to a_4} \to t$ מסוג זה, באשר נקודה a_2 היא נקודת פיצול. נניח שכל הצינורות בגודל קיבולת 100, אך $a_2 \to a_3$ בקיבולת 2, וכן $a_2 \to a_5$ בקיבולת 50. במצב זה, נוכל להעביר 2 ליטר בצינורות דרך $s \to a_1 \to a_2 \to a_3 \to a_4 \to t$ ונוכל להעביר עוד 50 דרך $s \to a_1 \to a_2 \to a_5 \to a_6 \to t$. וסה"כ הקיבולת ברשת הזרימה תהיה $50 + 2 = 52$.

נחדד: אין כאן משמעות לזמן, אלא בהינתן רשת זרימה, כמה אפשר להעביר בה בכל מצב שהוא. וכן: כמות ה"נפט" שנכנס אל קודקוד ברשת הזרימה u שווה לכמות ה"נפט" שיוצא מהקודקוד u. המקום היחיד שממנו יכול להיווצר נפט הוא בs, והמקום היחיד שיכול להשאר בו/ לצאת נפט: קודקוד t.

דוגמאות לשימוש: להבין מהו קצב העברת המידע האפשרי בין שני מחשבים ברשת מחשבים בזמן העברת קובץ ענק בין המחשבים. דוגמה נוספת היא לחשב איזה מסילות רכבת ניתן להפציץ בעלות הקטנה ביותר על מנת למנוע מעבר של ציוד מנקודה אחת לשנייה.

6.1 הגדרה פורמלית של זרימה

הגדרה: רשת זרימה היא גרף מכוון $G = (V, E)$ עם קודקוד מקור $s \in V$ וקודקוד יעד $t \in V$. ופונקציית קיבולת $C : V \times V \to \mathbb{R}^+ \cup \{0\}$
באשר $(u,v) \in E \iff C(u,v) > 0$.

הבהרה. הקיבולת יכולה להיות מס' ממשי חיובי או אפס. היא אפס אם אין קשת בין הקודקודים, אחרת: היא גדולה ממש מאפס.

נפרמל,

$$C(u,v) = \left\{ \begin{array}{cc} x \in \mathbb{R}^+ & (u,v) \in E \\ 0 & (u,v) \notin E \end{array} \right\}$$

נתבונן בשתי הגדרות, שקולות עבור **זרימה**. ההגדרה הראשונה יותר אינטואטיבית, והשנייה פחות (אך תעזור לנו בהמשך עם המתמטיקה).

הגדרה ראשונה: זרימה ברשת זרימה $G=(V,E)$ עם פונקציית קיבולת C, היא פונקציה $f: V\times V\to\mathbb{R}^+\cup\{0\}$ שמקיימת את התנאים הבאים (נדגיש ־ פונקציית הזרימה היא מה שאנו מזרימים בפועל על הרשת):

1. אילוצי קיבולת:

$$\forall u,v\in V: 0\le f(u,v)\le c(u,v)$$

כלומר, בהכרח הזרימה תהיה גדולה־שווה מאפס (לפי הגדרת הקיבולת, אם אין קשת היא אפס). וכן הזרימה לא תוכל לעבור לעולם את הקיבולת (כי לא נוכל לעולם להעביר את הזרימה דרך הרשת).

2. שימור זרימה: סכום הזרימה שנכנס שווה לסכום הזרימה שיוצא. חוק שימור הזרימה.

$$\forall v\in V\backslash\{s,t\}: \sum_{u\in V} f(u,v)=\sum_{u\in V} f(v,u)$$

הערך של זרימה f הוא:

$$|f|=\sum_{v\in V} f(s,v)-\sum_{v\in V} f(v,s)$$

כלומר, סכום כל ערכי הזרימה של הקודקודים שיוצאים מs, פחות כל ערכי הזרימה שנכנסים אל s. נשים לב כי יתכן שיכול לזרום חזרה אל s זרם. סה"כ ערך זה זה הערך שיצא מs, בניקוי מה שחזר. כלומר: ממש הסכום נטו שיוצא לבסוף.

הגדרה שנייה: זרימה ברשת זרימה $G=(V,E)$ עם פונקציית קיבולת C, היא פונקציה $f: V\times V\to\mathbb{R}$ שמקיימת את התנאים הבאים:

1. אילוצי קיבולת:

$$\forall u,v\in V: f(u,v)\le c(u,v)$$

נשים לב כי בהגדרה זו, יתכן כי הזרם יהיה שלילי. הוא רק לא יכול לעבור את הקיבולת. זה מאוד מוזר מבחינה מתמטית: ההסבר לכך יהיה הסימטריה מטה.

2. סימטריה:

$$\forall u,v\in V: f(u,v)=-f(v,u)$$

נשים לב שזו הגדרה מאוד אינטואטיבית וחשובה. אם יעבור לנו בצד מסויים 6 יחידות, בצד ההפוך אליו עבר -6. באופן דומה: אם מישהו הביא לי 100 שקל, אצלי עלה 100 שקל ואצלו ירד 100 שקל.

3. **שימור זרימה:** בהגדרה זו אנחנו מסתכלים רק על הקשתות שיוצאות מקודקוד מסויים, ונאמר שסכום הזרימה שלהם הוא אפס.

$$\forall v \in V \backslash \{s,t\} : \sum_{u \in V} f(v,u) = 0$$

הסבר: אנו מעוניינים כי אם נכנס מצד אחד של v סכום זרם של 12 למשל, מהכיוון שנכנס אל v בצד השני של הקודקוד יזרום סכום זרם של -12. (וכן כמובן שבסוף יתכן שזרמו מאותו צד של הקודקוד סכום שהתאפס לאפס. בכל מקרה: מדובר בקשתות שנכנסות אל v). נשים לב כי עדיין ישנו שימור זרימה כמו בהגדרה הקודמת, אבל מהגדרת הסימטריה צריך לייצג זאת מתמטית קצת שונה. - ניתן לומר כי אם מסתכלים על כמה שיוצא, אם מצד אחד יוצא 15 ערך זרם, נראה שנכנס אליו גם 15 (אך בכיוון השני מאיפה שנכנס, יצא -15 בגלל סימטריות) ולכן אכן $0 = 15 + 15-$.
הערך של זרימה f הוא:

$$|f| = \sum_{v \in V} f(s,v)$$

כעת, ערך הזרימה יוגדר להיות כל מה שיוצא מs, ושוב נשים לב שזה שקול אינטואטיבית לבעיה הקודמת, מהסימטריה, אם נכנס חזרה 5 יחידות אל s זה כאילו יצא מs -5 (זו הסימטריה). לכן אם יצא מs 15 למשל, ונכנס 2. זה שקול לכך שיצא 15 מs ויצא -2 (מסימטריה) ולכן ערך הזרימה הוא $15 - 2 = 13$.

בקורס נשתמש רק בהגדרה השנייה. ההגדרה הראשונה לטובת אינטואיציה בלבד.

הערה. נשים לב כי הפונקציה $f = 0$ היא גם פונקציית זרימה.

6.2 הגדרת הבעיה

קלט: רשת זרימה $G = (V,E)$ עם פונקציית קיבולת $C : V \times V \to \mathbb{R}^+ \cup \{0\}$
פלט: זרימה f בG בעלת ערך גדול ביותר מבין כל הזרימות האפשריות. $Max\ flow$.

6.3 תכונות של זרימה

נרצה להרחיב את ההגדרה של זרימה לקבוצות.
הגדרה: יהיו $X, Y \subseteq V$. נגדיר את הזרימה בין שתי קבוצות הקודקודים כך:

$$f(X,Y) = \sum_{x \in X} \sum_{y \in Y} f(x,y)$$

טענה: $f(\{s\}, V) = |f|$.
הוכחה:

$$f(\{s\}, V) = f(s,V) = \sum_{v \in V} f(s,v) = |f|$$

טענה 4**:** תהי f זרימה ברשת זרימה $G = (V,E)$. אזי,
1. $\forall X \subseteq V : f(X,X) = 0$
2. $\forall X, Y \subseteq V : f(X,Y) = -f(Y,X)$

3. $\forall X,Y,Z\subseteq V\wedge X\cap Y=\emptyset$ מתקיים:

a. $f(X\cup Y,Z)=f(X,Z)+f(Y,Z)$

b. $f(Z,X\cup Y)=f(Z,X)+f(Z,Y)$

הוכחה:

תהי f רשת זרימה. ויהיו $X,Y,Z\subseteq V$.

1.

$$f(X,X)=\sum_{x\in X}\sum_{x_2\in x}f(x,x_2)=\sum_{x\in X}0=0$$

2.

$$f(X,Y)=\sum_{x\in X}\sum_{y\in Y}f(x,y)=\sum_{x\in X}\sum_{y\in Y}-f(y,x)=-\sum_{y\in Y}\sum_{x\in X}f(y,x)=-f(Y,X)$$

3. נניח כי $X\cap Y=\emptyset$.

$$f(X\cup Y,Z)=\sum_{w\in X\cup Y}\sum_{z\in Z}f(w,z)=_{X\cap Y=\emptyset}\sum_{x\in X}\sum_{z\in Z}f(x,z)+\sum_{y\in Y}\sum_{z\in Z}f(y,z)=f(X,Z)+f(Y,Z)$$

למה 5: תהי f זרימה ברשת זרימה $G=(V,E)$. אזי,

$$|f|=f(V,t)$$

הוכחה:

$$f(V,V\backslash\{s,t\})=-f(V\backslash\{s,t\},V)=-\sum_{x\in V\backslash\{s,t\}}\sum_{y\in V}f(x,y)=-\sum_{x\in V\backslash\{s,t\}}0=0$$

כמו כן נשים לב כי

$$V=\{s,t\}\cup(V\backslash\{s,t\})$$

וכן שתי קבוצות אלו זרות. כעת לפי טענה 4 נוכל לומר כי:

$$|f|=f(s,V)=f(V,V)-f(V\backslash\{s\},V)=0-f(V\backslash\{s\},V)=f(V,V\backslash\{s\})$$

$$=f(V,V\backslash\{s,t\})+f(V,t)=0+f(V,t)$$

וסה"כ קיבלנו $|f|=f(V,t)$ כנדרש.

מסקנה: ראינו כי $f(s,V)=|f|$ ומלמה 5 ראינו כי $|f|=f(V,t)$ ונקבל כי $f(V,t)=f(s,V)$.
כלומר: סך הזרם שיוצא מs שווה לזרם שנכנס אל t.

הגדרה: חתך (s,t) הוא חתך $(S,T)=(S,V\backslash S)$ כאשר $s\in S$ וכן $t\in T$. נשים לב כי G הינו מכוון ולכן קשת שחוצה את החתך היא קשת שעוברת מS אל T (הקשתות בכיוון השני לא נקראות כאלו ולא מעניינות אותנו). כמו כן בהכרח $\emptyset\neq S\subset V$.

למה 6: יהי $G=(V,E)$ רשת זרימה עם פונקציית קיבולת C. ותהי f זרימה בG. יהי (S,T) חתך (s,t) של G. אז,

$$|f|=f(S,T)$$

(כלומר, אם נסתכל על הזרימה משמאל S אל T, סכום הזרימות הללו הוא בדיוק ערך הזרימה).

הוכחה:
נשים לב כי $S\cap T=\emptyset$ וכן $T\cup S=V$. כמו כן,

$$f(S\backslash\{s\},V)=\sum_{x\in S\backslash\{s\}}\sum_{y\in V}f(x,y)=_{(*)}0$$

כיוון ש$x\neq s,t$, לפי חוק שימור הזרימה (3).

$$f(S,T)=f(S,V)-f(S,S)=f(S,V)-0=f(S,V)=f(S\backslash\{s\},V)+f(s,V)=0+f(s,V)=|f|$$

כנדרש.

הגדרה: קיבולת בין קבוצות הינה

$$C(S,T)=\sum_{x\in S}\sum_{y\in T}c(x,y)$$

למה 7. יהיו S,T. אזי, מתקיים $f(S,T)\leq C(S,T)$
הוכחה:

$$f(S,T)=\sum_{x\in S}\sum_{y\in T}f(x,y)\leq\sum_{x\in S}\sum_{y\in T}c(x,y)=C(S,T)$$

מסקנה. לכל חתך גודל הזרימה זהה לפי למה 6 ולכל זרימה בחתך חסומה ע"י הקיבול של החתך, לכן כל זרימה חסומה ע"י כל החתכים ובפרט הקטן ביותר, ובפרט הזרימה הגדולה. לכן ערך הזרימה המקסימלי, יהיה בהכרח קטן שווה מהקיבול הקטן ביותר. **כלומר,** $Max\ flow\leq Min\ cut$**.**

6.4 שיטת פורד־פלקרסון

נניח שאנחנו מתחילים מ$f = 0$ (כפי שהערנו קודם זו אכן זרימה). כלומר: 0 $\forall_{v,u\in V} f(u,v) =$. מכאן ש$|f| = 0$.

נניח שאנחנו מסתכלים על רשת זרימה $G = (V, E)$ ומצאנו מסלול כלשהו בין s לt. אזי, ברור מכאן כי הזרימה המקסימלית האפשרית באותו המסלול: הוא ערך הזרימה המינימלי שמופיע על המסלול. כלומר אם מצאנו מסלול $s \rightarrow_{100} a_1 \rightarrow_{50} a_2 \rightarrow_{14} \rightarrow t$ אזי הזרימה המקסימלית האפשרית במסלול הינה 14.

נרצה להגדיר פונקציית זרימה כך -

יהי מסלול P. אם $(x, y) \in P$ היא קשת עם $C(x,y) = min_{(u,v)\in P}\{C(u,v)\}$ אזי

$$f'(u,v) = \left\{ \begin{array}{cc} C(x,y) & (u,v) \in P \\ -C(x,y) & (v,u) \in P \\ 0 & o.w \end{array} \right\}$$

$$|f'| = C(x,y)$$

וכן נשים לב שאכן ערך הזרימה הינו $C(x,y)$ כי זה בדיוק ה14 עליו דיברנו קודם בדוגמה. ערך הקיבולת הקטן ביותר על המסלול הינו ערך הזרימה.

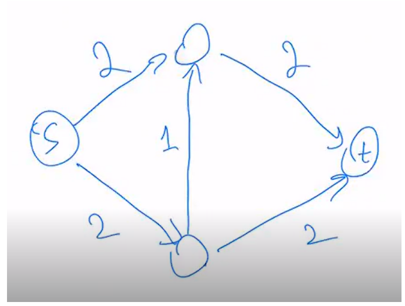
נראה כי אכן שיפרנו את הזרימה מ0 ל$C(x,y)$.

יתכן כי ישנם הרבה מסלולים זרים (בקשתות) מs לt ואז אפשר להפעיל את הרעיון שעשינו קודם על כל המסלולים. אם כן, זה לא מכסה את כל המקרים. באופן כללי יכול להיות שהיינו רוצים להשתמש בקשת אחת עבור יותר ממסלול אחד (נקודת פיצול למשל). לשם כך צריך לפתח מנגנון שיאפשר לקחת את המינימלי בכל מסלול, אך להזרים יותר בידיעה שניתן להתפצל לאורך המסלול.

הגדרה: קשת $(x, y) \in E$ תקרא רוויה תחת זרימה f אם $f(x,y) = C(x,y)$.
הערה. נשים לב כי אם קשת איננה רוויה, אזי ניתן להשתמש בקיבולת הנותרת.

רעיון: כל עוד קיים מסלול מs לt שקשתותיו אינן רוויות, "נזרים" על מסלול זה "כמה שאפשר". נשים לב שהרעיון הוא בגדר רעיון ולא מוגדר היטב. עם זאת: זה לא יעבוד.

נסתכל על הדוגמה הבאה:



נראה כי נרצה ללכת במסלול שבצורת Z מs מטה, עובר ב1 ומסיים בt. הערך המינימלי במסלול זה הינו 1. וקיבלנו כי הקשת 1 רוויה. המסלול היחיד שנותר לנו מt לs שקשתותיו אינן רוויות זה להתחיל מs מעלה, ללכת על המסלול של שתי הקשתות שערכן 2. נשים לב שעל מסלול זה נוכל להזרים

1 בלבד כי הקשת 2 שמגיעה מלמעלה אל t זרם בה כבר 1 (מהמסלול הקודם). מכאן שנסתכל על המסלול האחרון שלא כל הקשתות בו רוויות, המסלול שמתחיל בs מטה ועובר רק בקשתות שערכן 2. שוב: הקשת 2 שמגיעה מלמטה אל t השתמשה באחד ולכן ניתן להזרים במסלול זה 1. סה"כ הזרמנו 1 בכל מסלול והיו לנו 3 מסלולים וקיבלנו כי ערך הזרימה הינו $|f| = 3$. עם זאת: היה ניתן להזרים 4 במעבר ישיר של 2 מs מעלה ומטה. מסקנה: הרעיון לא טוב. והבעיה ־ הרבה יותר קשה משחשבנו.

6.5 הרשת השיורית

נשים לב כי הבעיה בדוגמה שהראנו קודם, היא שהאלגוריתם קודם בחר את המסלול שעובר דרך 1. אם הוא לא היה בוחר במסלול זה, או היה מנסה לבחור אותו אחרון: הפתרון כן היה עובד באשר לדוגמה הספציפית הקודמת. נראה כי אלגוריתם חמדן בוחר החלטה ואחר כך חייב לעמוד בה, הוא לא יכול להתחרט. במצב של קודם, היינו שמחים אם לאחר הבחירה במסלול שעובר ב1 הוא היה יכול להתחרט. מכאן נגיע להגדרה הבאה.

הגדרה: יהי $G = (V, E)$ רשת זרימה עם פונקציית קיבולת C. ותהי f זרימה בG. הקיבולת השיורית של C תחת f היא הפונקציה

$$C_f(u, v) = C(u, v) - f(u, v)$$

כלומר: כמה עוד יש לנו להזרים בקשת מסויימת.

הגדרה: הרשת השיורית של G תחת f היא רשת זרימה $G_f = (V, E_f)$ שפונקציית הקיבולת שלה הינה C_f. וכן $E_f = \{(u, v)|C_f(u, v) > 0\}$. נשים לב כי אכן פונקציית הקיבולת מקיימת $C_f \geq 0$ שכן תמיד $C(u, v) \geq f(u, v)$ כלומר $C_f(u, v) \geq 0$.

כלומר: קבוצת הקשתות זה כל הקשתות שעוד ניתן להזרים בהן.

נשים לב כי אמרנו שתתכן זרימה שלילית. מכאן: אם בכיוון $x \to t$ זרם 2, ולא הייתה קשת בכיוון ההפוך. כלומר $f(x, y) = 2$, נראה כי בכיוון השני לא עברה זרימה ולכן $C(y, x) = 0$. מכאן נקבל כי $c_f(y, x) = C(y, x) - f(y, x) = 0 - f(y, x) = f(x, y) = 2$. כלומר, אם בכיוון מסויים ברשת זרם ערך כלשהו, 2 במקרה שלנו, בכיוון ההפוך יזרום אותו ערך ברשת השיורית. ומכאן המסקנה: יתכן כי ברשת השיורית ייתכנו קשתות נוספות שלא היו בגרף המקורי.

נראה כי כתוצאה מהרשת השיורית, כעת פיתחנו מנגנון ל"חזרה אחורה" במידה ולא מעוניינים במה שבחרנו. כעת יש את האפשרות ללכת בכיוון הנגדי ולחפש מסלול שישלים. מתי נדע לעצור? כשאין מסלול מs לt: כשכל הקשתות נכנסות מs ולא יוצאות ממנו. הרעיון יהיה לשפר מסלולים על הרשת השיורית עד שלא ניתן יהיה לעשות זאת.

הגדרה: בהינתן מסלול P ברשת השיורית G_f נגדיר (את הקיבולת השיורית המינימלית) כך:

$$C_f(P) = min\{c_f(u, v)|(u, v) \in P\}$$

6.6 שיטת פורד־פלקרסון

להלן האלגוריתם:

$\textsc{Ford-Fulkerson}(G=(V,E),s,t,c)$

initialize $f(u,v)=0$ for all $u,v\in V$

$G_f \leftarrow G,\ c_f \leftarrow c$

while there exists a path P from s to t in G_f

    $c_f(P) \leftarrow \min_{(u,v)\in P}\{c_f(u,v)\}$

    for each edge $(u,v)\in P$

        $f(u,v) \leftarrow f(u,v) + c_f(P)$

        $c_f(u,v) \leftarrow c_f(u,v) - c_f(P)$

        $f(v,u) \leftarrow f(v,u) - c_f(P)$

        $c_f(v,u) \leftarrow c_f(v,u) + c_f(P)$

        update E_f

Return f

מה קורה באלגוריתם? האלגוריתם מקבל פונקציית קיבולת, רשת זרימה וקודקוד מקור ויעד. בתחילה: האלגוריתם מאתחל את פונקציית הזרימה להיות אפס עבור כל הקודקודים. כמו כן: מאתחלים את רשת הזרימה השיורית להיות בתחילה רשת הזרימה עצמה ואת הקיבולת השיורית להיות הקיבולת. לאחר מכן נכנסים אל לולאה שמתבצעת כל עוד קיים מסלול מs לt ברשת השיורית.

מגדירים את $C_f(P)$ כפי שהוגדר לעיל, עוברים על כל זוג קודקודים במסלול P, מוסיפים ל$f(u,v)$ את $c_f(P)$ שגילינו קודם לכן (כמה כעת אפשר לעבור בו) ובאופן דומה מורידים אותו מ$f(v,u)$ וכן מגדירים את הרשת השיורית באופן דומה ונגדי: מ$c_f(u,v)$ אנו מורידים את $c_f(P)$ (כי כעת יש שם פחות זרם שניתן להעביר) ואל $c_f(v,u)$ אנחנו מוסיפים את $c_f(P)$ (כי יש יותר זרם שניתן להעביר).

לבסוף: מעדכנים את הקשתות E_f (יתכן שיש קשתות כעת שמוסיפים או לחלופין מורידים). כלומר כל מי שהקיבולת השיורית שלו התאפסה צריך להעיף, מי שקודם לכן היה אפס וכעת לא: צריך להכניסו לרשת השיורית. פעולה זו היא למעשה העדכון של G_f.

הגדרה: למסלול P אנחנו נקרא "מסלול שיפור". וכן אנחנו משתמשים בP בשביל לשפר את f.

6.7 נכונות האלגוריתם וזמן הריצה

נתבונן בבעיית חתך מינימום:

קלט: גרף $G=(V,E)$ מכוון. ושני קודקודים $s,t\in V$ וכן פונקציית משקל $w:E\to\mathbb{R}^+$

פלט: חתך (s,t) שסכום קשתות שעוברות מS לT הוא כמה שיותר קטן.

בשביל לפתור את בעיית זרימת המינימום נרצה לפתור בעיה של זרימה ברשת שנגדיר, ועל מנת לראות שזה אכן פותר את הבעיה נוכיח את נכונות האלגוריתם של פורד (ואז כבר קיבלנו את הנכונות שרצינו).

למה 7: תהי $G=(V,E)$ רשת זרימה עם פונקציית קיבולת C ותהי f רשת זרימה בG. תהי (S,T) חתך (s,t) בG. אזי, $|f|\leq C(S,T)$
(כלומר, ערך הזרימה בגרף יהיה קטן־שווה מסכום הקיבולות של הקשתות שחוצות את החתך משמאל S אל ימין T)

הוכחה: תהי $G=(V,E)$ רשת זרימה עם פונקציית קיבולת C ותהי f רשת זרימה בG. תהי (S,T) חתך (s,t) בG.
ראינו כבר כי $|f|=f(S,T)$ בלפה 6. לכן

$$|f|=f(S,T)=\sum_{x\in S}\sum_{y\in T}f(x,y)\leq_{(*)}\sum_{x\in S}\sum_{y\in T}C(x,y)=C(S,T)$$

באשר $(*)$ זה כיוון שתמיד מתקיים כי $f(x,y)\leq C(x,y)$, כלומר הזרימה היא לכל היותר בגודל הקיבולת. כנדרש.

סימון: נסמן את זרימת המקסימום $|f^*|$.
מסקנה: ערך כל זרימה שהיא $|f|$ יהיה קטן או שווה מחתך ה(s,t) המינימלי.

6.7.1 משפט $max-flow-min-cut$

משפט 8: תהי $G=(V,E)$ רשת זרימה עם פונקציית קיבולת C ותהי f זרימה בG. אזי, כל התנאים הבאים שקולים:
א. f זרימת מקסימום.
ב. בG_f אין מסלול שיפור
ג. קיים חתך (s,t) שנסמנו (S,T) בG כך ש$|f|=C(S,T)$.

מסקנה: זרימת מקסימום $=$ חתך (S,T) מינימום (!)
מסקנה שנייה: המשפט מוכיח את נכונות האלגוריתם, כיוון שא' גורר את ב' באמ"מ. אכן אם אין מסלול שיפור מצאנו את זרימת המקסימום.

הוכחה:
א $\Longleftarrow$ ב: נניח כי f זרימת מקסימום. נניח בשלילה כי f זרימת מקסימום ובG_f יש מסלול שיפור P. מכאן, ניתן להשתמש בP על מנת להגדיל את ערך הזרימה ולכן f אינה זרימת מקסימום, בסתירה.
ב $\Longleftarrow$ ג: נניח כי בG_f אין מסלול שיפור. נגדיר את (S,T) כדקלמן:

$$S=\{v\in G_f|\exists P=(s,...,v)\}$$

$$T=\{v\in G_f|not\exists P=(s,...v)\}$$

כלומר T היא קבוצת הקודקודים שקיים מסלול מs אליהם, וT זו הקבוצה שלא קיים מסלול מs אליהם.
נראה כי אכן קיים מסלול מs אל s ולכן $s\in S$ ולכן לא קיים מסלול מs אל t כי אין מסלולי שיפור ולכן $t\in T$. ולכן אכן (S,T) שהוגדר חתך (s,t).

טענה 9: לכל $u\in S$ ו$v\in T$ מתקיים $f(u,v)=C(u,v)$

הוכחה: מצד אחד תמיד מתקיים $f(u,v) \le C(u,v)$. נניח בשלילה כי $f(u,v) < C(u,v)$. כלומר,

$$C_f(u,v) = c(u,v) - f(u,v) > 0$$

ומכאן קיבלנו כי $(u,v) \in E_f$. ע"פ הגדרת S, יש מסלול מs לu בG_f. מסלול זה יחד עם הקשת $(u,v) \in E_f$ יוצר מסלול מs לv בG_f בסתירה לכך ש$v \notin S$.
כעת,

$$|f| = f(S,T) =_{def} \sum_{x\in S}\sum_{y \in T} f(x,y) =_{Lemma9} \sum_{x\in S}\sum_{y\in T} C(x,y) = C(S,T)$$

ג $\Longleftarrow$ א: נניח שקיים חתך (s,t) שנסמנו (S,T) בG כך ש$|f| = C(S,T)$. אז, נניח בשלילה כי f איננה זרימת מקסימום. כלומר, קיימת פונקציית זרימה f' כך ש$|f| < |f'|$. מכאן ש$|f'| > |f| = C(S,T)$ בסתירה ללמה 7 כי $f(S,T) \le C(S,T)$. מכאן שבהכרח f זרימת מקסימום.

כנדרש.

6.7.2 סיבוכיות זמן הריצה (פורד פרקלסון)

באופן כללי, השיטה של פורד פרקלסון עלולה שלא להסתיים לעולם. עם זאת, אם כל הקיבולות הם מספרים שלמים: האלגוריתם כן יסתיים.

מכאן נובע, שבכל איטרציה הזרימה תשתפר בלפחות אחד. ולכן, אם הזרימה המקסימלית הינה $|f^*|$ אזי לכל היותר לאחר $|f^*|$ איטרציות האלגוריתם יסיים.

נראה כי בכל איטרציה אנו נדרשים למצוא מסלול - למשל באמצעות dfs. זה יעלה $O(|E_f|) = O(|E|)$ וכן מתבצעים עדכונים על המסלול שעלותם $O(|V|)$ סה"כ כל איטרציה עולה $O(|E|+|V|)$.

הנחה: נניח כי כל הקודקודים בV נמצאים על מסלול כלשהו מs לt בגרף המקורי. אחרת, אפשר בזמן לינארי להוציא את אלו שלא נמצאים ונקבל מכאן כי $|V|-1 \le |E|$ ולכן כל איטרציה עלותה $O(|E|+|V|) \le O(|E|)$.

מכאן נקבל כי זמן הריצה הינו: $O(|f^*| \times |E|)$

למה 10: תהי רשת זרימה $G=(V,E)$ עם פונקציית קיבולת C וזרימה f, אזי מתקיים
f' זרימה בG_f $\Longleftrightarrow$ $f+f'$ זרימה בG.
מסקנה: f' זרימת מקסימום בG_f $\Longleftrightarrow$ $f+f'$ זרימת מקסימום בG.

נשים לב, גם אם כל הקיבולים הם רציונליים ניתן להגדיר את זמן הריצה הנ"ל שכן ניתן בתחילה למצוא את המכנה המשותף ולהכפיל בו, להריץ אלגו רגיל על מס' שלמים ובסוף לחלק חזרה במכנה המשותף.

7 הרצאה 7: זרימה - $Dinic$

ראינו בהרצאה הקודמת את השיטה של פורד למציאת רשת זרימה בעלות של $O(|f^*| \times |E|)$ אם הקיבולות מספרים שלמים.

אלגוריתם נוסף שראינו הוא אדמונס קארפ שרץ בסיבוכיות זמן $O(|V| \times |E|^2)$. כעת נראה אלגוריתם של $Dinic$ שרץ בסיבוכיות זמן $O(|V|^2 \times |E|)$.

7.1 האלגוריתם של אדמונס קארפ

האלגוריתם של אדמונס קארפ הוא צורת מימוש לשיטה של פורד־פרקלסון. בכל שלב אנחנו נמצא מסלול שיפור ברשת השיורית בעל מספר מינימלי של קשתות, מציאת המסלול מתבצעת על ידי הרצת BFS מs עד שמגיעים לt. ברור כי כל שיפור מסלול עלותו $O(|V|+|E|) = O(|E|)$ זמן (כי מניחים קשירות), ולאחר שנוכיח כי האלגוריתם מוצא את זרימת המקסימום בתוך לכל היותר $O(|E| \times |V|)$ איטרציות נקבל כי סיבוכיות זמן הריצה של האלגוריתם הינה $O(|E|^2 \times |V|)$.

אלגוריתם 1 אדמונדס־קארפ$(G = (V, E), s, t, c)$

---

1. לכל קשת $(u, v) \in E$
   (א) $0 \to f[u, v] \to f[v, u]$
2. כל עוד קיים מסלול ברשת השיורית G_f מ־s ל־t.
   (א) הרץ BFS מ־s עד מציאת t. ויהי p המסלול שנמצא בעץ המסלולים הקצרים ביותר מ־s ל־t.
   (ב) $\min \{c_f(u, v) \mid (u, v) \in p\} \to c_f(p)$
   (ג) לכל קשת $(u, v) \in p$
      i. $f[u, v] - c_f(p) \to f[u, v]$
      ii. $-f[u, v] \to f[v, u]$

האלגוריתם משתמש ברעיון של פורד וממשש אותו שונה. נרצה להוכיח נכונותו.

הגדרה: נסמן $\delta_f(u, v)$ כאורך המסלול הקצר ביותר בין u ל v בG_f.

למה 1: תהי f' זרימה המתקבלת מזרימה f ע"י שיפור על גבי מסלול באורך הקצר ביותר מs לt בG_f. אזי לכל $u \in V$ מתקיים $\delta_f(s, u) \leq \delta_{f'}(s, u)$
(כלומר, במסלול השיפור קודקודים רק מתרחקים מקודקוד המקור).

הוכחה: נניח בשלילה שזהו לא המצב. כלומר קיים קודקוד $u \in V$ המקיים $\delta_f(s, u) > \delta_{f'}(s, u)$. ייתכנו מס' קודקודים כנ"ל ובה"כ u הוא קודקוד במרחק מינימלי מs ב$G_{f'}$ שעבורו זה מתקיים. יהי p מסלול קצר ביותר מs לu לאחר השיפור, כלומר ברשת $G_{f'}$. ויהי v הקודקוד הקודם לu בP. מתקיים $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$ וכן $\delta_f(s, v) \leq \delta_{f'}(s, v)$. נחלק למקרים -

א. אם $f(v, u) < c(v, u)$ אזי הקשת (v, u) קיימת גם בG_f. ולכן

$$\delta_f(s, u) \leq \delta_f(s, v) + 1 \leq \delta_{f'}(s, v) + 1 = \delta_{f'}(s, u)$$

בסתירה לכך ש$\delta_f(s, u) > \delta_{f'}(s, u)$.

ב. אם $f(v, u) = c(v, u)$ אזי $(v, u) \notin E_f$ כלומר בהכרח השיפור שעשינו עבר בקשת (u, v)- בכיוון ההפוך לv, u. כיוון שהשיפור נעשה על פני מסלול p' שהוא קצר ביותר מs לt הרי כל תת מסלול שלו הוא קצר ביותר ובפרט הקשת (u, v) היא על מסלול קצר ביותר מs לu בG_f. לכן

$$\delta_f(s, u) = \delta_f(s, v) - 1 \leq \delta_{f'}(s, v) = \delta_{f'}(s, u) - 1 < \delta_{f'}(s, u)$$

ושוב בסתירה להנחתנו כי $\delta_f(s, u) > \delta_{f'}(s, u)$.

מסקנה 2: תהי f' זרימה המתקבלת מזרימה f על ידי שיפור על גבי מסלול באורך קצר ביותר מs לt בG_f. אזי, לכל $u \in V$ מתקיים $\delta_f(u,t) \leq \delta_{f'}(u,t)$

למה 3: תהי f' זרימה המתקבלת מזרימה f ע"י שיפור על גבי מסלול באורך קצר ביותר מs לt בG_f. אם $\delta_f(s,t) = \delta_{f'}(s,t)$ אזי כל מסלול קצר ביותר מs לt ב$G_{f'}$ הוא גם מסלול קצר ביותר מs לt בG_f. (כלומר, אם יש שוויון שכזה אזי לאחר השיפור לא נוצרו מסלולים קצרים ביותר חדשים)

הוכחה: נגדיר מושג חדש של קשתות חדשות בגרף - קשת (v,u) היא קשת חדשה אם ורק אם (u,v) הייתה בG_f ומסלול השיפור כלל אותה. כיוון שמסלול השיפור הוא מסלול מאורך קצר ביותר מובטח כי $\delta_f(s,v) = \delta_f(s,u) + 1$.

יהי P מסלול קצר ביותר מs לt ב$G_{f'}$. נניח בשלילה כי P לא נמצא בG_f. אזי P בהכרח מכיל קשת חדשה (ייתכן שיותר מאחת). תהי (v,u) קשת חדשה שהיא במסלול P.

לפי למה 1 מתקיים $\delta_{f'}(s,v) \geq \delta_f(s,v)$ ולפי מסקנה 2 נקבל כי $\delta_{f'}(u,t) \geq \delta_f(u,t)$. כיוון ש$P$ מסלול קצר ביותר מs לt ב$G_{f'}$ ו$(v,u) \in P$ אזי

$$\delta_{f'}(s,t) = |P| = \delta_{f'}(s,v) + 1 + \delta_{f'}(u,t)$$

מצד שני כיוון ש(u,v) היא קשת על מסלול השיפור שהוא מסלול מאורך קצר ביותר בG_f מתקיים כי $\delta_f(s,v) = \delta_f(s,u) + 1$. ולכן:

$$\delta_{f'}(s,t) = |P| = \delta_{f'}(s,v) + 1 + \delta_{f'}(u,t) \geq$$

$$\delta_f(s,v) + 1 + \delta_f(u,t) = \delta_f(s,u) + 1 + 1 + \delta_f(u,t) =$$

$$\delta_f(s,t) + 2 > \delta_f(s,t)$$

וקיבלנו $\delta_{f'}(s,t) > \delta_f(s,t)$ בסתירה לכך שהם שווים.

למה 4: תהי G רשת זרימה וf זרימה כלשהי. נתבונן באלגוריתם אשר משפר על גבי מסלולים מאורך קצר ביותר מs לt רק כל עוד אורכם הוא $\delta_f(s,t)$. אזי, מרגע שקשת (u,v) נהיית רוויה ע"י האלגוריתם, קשת זאת לא תהיה בשימוש על ידי אף מסלול שיפור אחר במהלך ריצת האלגוריתם.

כעת, נסתכל על ריצת האלגוריתם אדמונס קארפ, ונסתכל על כל מסלול השיפור שאורכם ℓ. נקרא לאיטרציות ששיפרו אותם: הפאזה הℓ של האלגוריתם.

כלומר: הפאזה הℓ באלגוריתם של Ek היא סדרת האיטרציות שבהן אורך המסלול הקצר ביותר הוא ℓ קשתות. פאזה יכולה להיות ריקה.

כל מסלול שיפור בפאזה הℓ ניתן לשייך לקשת אחת (לפחות) אותה הוא הפך לרוויה. לפי הלמה, מובטח שכל קשת תשוייך למסלול אחד בפאזה הℓ לכל היותר. ולכן סה"כ בפאזה הℓ יכולות להיות לכל היותר $|E|$ איטרציות.

מסקנה 5: יש לכל היותר $|E|$ איטרציות בכל פאזה.

בנוסף, כיוון שיש לכל היותר $|V|-1$ מרחקים אפשריים של s וt, מס' הפאזות הינו $O(|V|)$. מכאן סה"כ מס' האיטרציות של האלגוריתם הינו לכל היותר $O(|E| \times |V|)$. כמו כן, כל איטרציה דורשת הרצת BFS כפי שכבר אמרנו, שעלותה $O(|E|)$ ונקבל את סיבוכיות זמן הריצה: $O(|E|^2|V|)$.

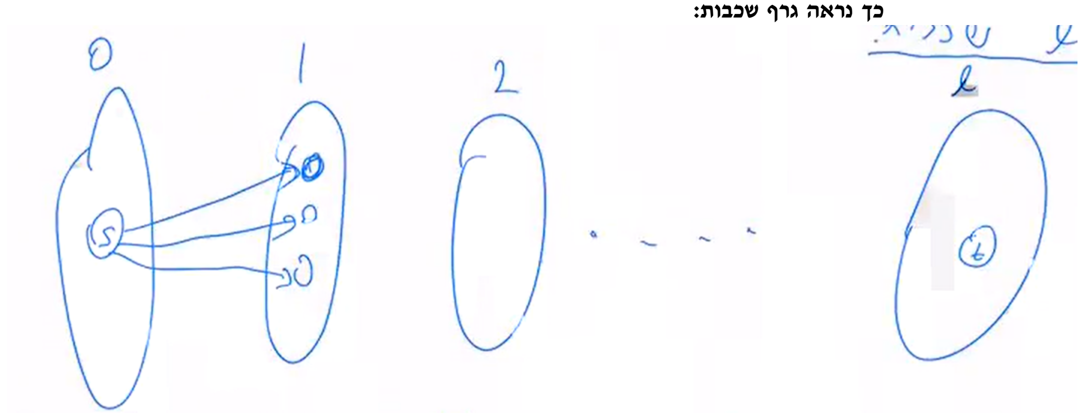
7.2 גרף השכבות

באלגוריתם של דיניץ' יש עדיין $O(|V|)$ פאזות. כל פאזה תעלה סה"כ $O(|V| \times |E|)$ זמן. ואז זמן הריצה יהיה $O(|V|^2|E|)$.

הגדרה: יהי $G = (V, E)$ גרף מכוון עם קודקוד מקור s וקודקוד יעד t, נסמן בℓ את $\delta(s, t)$ (מס' הקשתות במסלול הקצר ביותר). נגדיר באמצעותו את גרף השכבות, בו יש $\ell + 1$ שכבות, ובשכבה הi יהיו כל הקודקודים $u \in V$ כך ש$\delta_f(s, u) = i$ וגם u נמצא על מסלול קצר ביותר מs אל t. (כלומר, הוא תת מסלול של מסלול קצר ביותר).

סה"כ, נקבל את השכבות $0, 1, ..., \ell$, נשים לב כי בהכרח בשכבה 0 ישנו את s ורק את s. כמו כן, בהכרח בשכבה ℓ ישנו את קודקוד t ורק אותו שכן $\ell = \delta(s, t)$ לפי הגדרה. שכן, נב"ש שישנו קודקוד אחר $x \in \ell$ שאינו t, אזי אורך המסלול הקצר ביותר מs אל x הוא ℓ וכן אם הוא שם הוא חלק ממסלול קצר ביותר מs אל t, בסתירה כי המסלול הקצר ביותר שהוא חל בו אל t הוא באורך $\ell + 1$, ואז בהכרח זה גורר קיום שכבה $\ell + 1$ בסתירה.

כך נראה גרף שכבות:



הקודקודים שנכנסים אל ℓ הם קודקודים מהשכבה $\ell - 1$ שנמצאים על המסלול הקצר ביותר בדרך אל t באורך $\ell - 1$ ואם נוסיף את הקשת שאיתה הם נכנסים אל ℓ אזי נקבל מסלול קצר ביותר באורך $\ell = \delta(s, t)$ בדיוק.

הקשתות בין השכבות הן קשתות שנמצאות על איזשהו מסלול קצר ביותר מs אל t.

נשים לב כי **לא כל הקודקודים נמצאים בגרף השכבות** - רק קודקודים שנמצאים על קשתות שנמצאות על אחד מן המסלולים הקצרים ביותר מs אל t. באופן דומה, לא כל הקשתות נמצאות בגרף השכבות אלא רק הקשתות שבאחד המסלולים הקצרים ביותר.

הרעיון באלגוריתם של דיניץ, יהיה בתחילת כל פאזה לבנות גרף שכבות L מG_f.

נזכר מהי פאזה - הסתכלנו על ריצת האלגוריתם אדמונס קארפ, וכן הסתכלנו על כל מסלולי השיפור שאורכם ℓ (יתכנו כמה כאלו). נקרא לאיטרציות ששיפרו אותם: הפאזה הℓ של האלגוריתם. כלומר: הפאזה הℓ באלגוריתם של Ek היא סדרת האיטרציות שבהן אורך המסלול הקצר ביותר הוא ℓ קשתות. פאזה יכולה להיות ריקה. **ובקצרה: בכל איטרציה אנחנו מוצאים מסלול מגדיל באמצעות BFS המסלול הקצר ביותר יש לו אורך d, כל עוד האורך הזה לא משתנה - אנחנו באותה פאזה ברגע שהאורך גדל $(d \to d + 1)$ - פאזה חדשה מתחילה.**

לפי למה 4, שבתחילת כל פאזה, בG_f נמצאים כל המסלולים הקצרים ביותר שהאלגוריתם ימצא תוך כדי הפאזה. לכן, לפי ההגדרה של L: גרף השכבות L מכיל את כל המסלולים הקצרים ביותר מs אל t בG_f.

בתחילה אנו בונים את גרף השכבות, לוקחים את G_f ובונים את גרף השכבות. כמה עולה לבנות את גרף השכבות? **זמן הבניה של גרף השכבות הוא $O(|V|+|E|) = O(|E|)$ - כי יש קשירות.** כיצד? מריצים BFS מs, ומריצים BFS מt על G^T. כעת, לפי טענה שקשת (u, v) נמצאת במסלול קצר ביותר אמ"מ $\delta(s,t) = \delta(s,u) + 1 + \delta(v,t)$, נוכל לבדוק לכל $(u,v) \in E$ האם מתקיים השוויון ואם כן היא על מסלול קצר ביותר מs אל t ונוסיף אותה לגרף השכבות. סה"כ בניית גרף השכבות תעלה $O(|E|+|V|)$, אך $|V| = O(|E|)$ לכן סה"כ $O(|E|)$.

7.3 מציאת מסלול

כיצד מוצאים מסלול קצר ביותר בעזרת גרף השכבות?

מתחילים מs שבשכבה 0, ואנו יודעים כי כל קשת מs תוביל אותנו לקודקוד שנמצא בשכבה הראשונה. בדומה, בשכבה 1 לא משנה איזה קשת נבחר נעבור לקודקוד בשכבה השניה. באופן כללי, אם אנו בשכבה הi, וישנו קודקוד v_i, אנו יודעים כי גם אם ישנם הרבה קשתות מהשכבה הi לשכבה ה$i+1$, לא משנה איזה קשת נבחר היא תמיד נמצאת על מסלול קצר ביותר כלשהו מs אל t. לפי הבניה של L, כל קשת $e \in L$ נמצאת על מסלול קצר ביותר מs אל t.

לכן, בחירה של קשת שרירותית שיוצאת מקודקוד u בשכבה הi בהכרח תוביל לקודקוד שנמצא בשכבה ה$i+1$.

לכן, האלגוריתם למציאת מסלול קצר ביותר כלשהו מs לt מאוד פשוט:

א. נאתחל את $s \to u$. כל עוד $u \neq t$ בחר קשת שרירותית שיוצאת מu, (u, v)

ב. נוסיף את (u, v) למסלול

ג. ועדכן $u = v$ וחזור לשלב א'.

ד. לבסוף, החזר את המסלול.

כמה זמן לוקח למצוא מסלול קצר ביותר מs לt שכזה? נראה כי אנחנו בוחרים כל אחת מהקשתות, לכן הזמן שאני משקיע בכל שכבה הינה $O(1)$ זמן. אם כן, זמן הריצה הוא מס' השכבות, אם $\ell+1$ הוא מס' השכבות זמן הריצה יהיה $O(\ell)$.

אז מה קורה בתחילת האלגוריתם? בשלב הראשון של הפאזה, בנינו את גרף השכבות L שיעלה $O(|E|)$ זמן.

כעת, נחפש מסלול קצר ביותר מs אל t בL. אנו יודעים כי מסלול קצר ביותר מs אל t בL הוא מסלול קצר ביותר גם בG_f. זה יעלה $O(\ell) = O(|V|)$ כי אורך המסלול הוא לכל היותר $|V| - 1$.

הרעיון יהיה, להמשיך לחפש מסלולים קצרים ביותר בL. אך - ישנה בעיה. הבעיה היא שלאחר שמצאנו את המסלול הראשון, חלק מהקשתות נהיות רוויות וצריכות להמחק מגרף השכבות L. יתכן גם שצריכים למחוק קודקודים מסוימים מהגרף. למה זה חשוב לנו? אמרנו שאנחנו יכולים לבחור קשת שרירותית בעת שמצאנו מסלול קצר ביותר - למה אמרנו שניתן לבחור שרירותית? כי לא משנה איזה קשת נבחר, תמיד בצד השני יהיה קודקוד שמגיעים אליו ומשם ממשיכים, אך אם מחקנו קשת באיטרציה הקודמת יתכן (מאוד) שהגישה של לקחת שרירותית לא תעזור לנו ואנחנו נתקע - כי הקשת השרירותית אליה הלכנו, היא קשת שממנה אין להתקדם (היה באיטרציה הקודמת דרך להתקדם, אך מחקנו את הקשת).

לכן, עלינו לפתח מנגנון שיבטיח שגם לאחר מציאת מסלול, גרף השכבות יהיה גרף מעודכן שעודנו גרף שכבות (ואז כן נוכל לקחת קשת שרירותית).

7.4 עדכון גרף השכבות

נסתכל על קשת $u \to v$ שצריכה להמחק מהגרף. מה יכול לקרות?

מבחינת u, יתכן שישנה קשת נוספת שיוצאת מu , למשל (u, x) - אז המחיקה של (u, v) לא משפיעה על u כי באיטרציה הבאה ישנן דרכים אחרים להתקדם למשל דרך x. אך, מה אם הקשת

היחידה שיוצאת מu היא (u,v)? אם נמחוק אותה כעת - זה אומר שאין לu איך להתקדם אל t בהמשך כי אולי יש הרבה קשתות שנכנסות אליו אך אין קשתות שיוצאות ממנו. הקושי בזה הוא שבשלב לפני u, אם נבחור בקשת אל u שרירותית אנחנו נתקע כי מu אין איך להתקדם.

במצב כזה, אם $deg_{out}(u) = 0$, נרצה למחוק את u מL ולמחוק את כל הקשתות (w,u) שנכנסות אל u. ומה, אם היה קודקוד x שיש לו קשת (x,u), וכעת מחקנו את u, והדרך היחידה לצאת מx הייתה דרך הקשת (x,u), כעת מחקנו את (x,u) כי היא נכנסת אל u ומחקנו את u - אז כעת אנחנו צריכים למחוק גם את x והקשתות שיוצאות ממנו: ומכאן שזה יכול להיות תהליך ארוך מאוד, כל המחיקה הזו ברקורסיה שנפטרת מכל המבויים הסתומים.

מה באשר לv בקשת $u \to v$ שמחקנו? יתכן כעת שהקשת היחידה שנכנסת אל v הייתה (u,v), במצב זה יתקיים כי $deg_{in}(v) = 0$, ולפי הגדרת גרף השכבות נצטרך למחוק את v ואת צלעותיו.

נגדיר זאת פורמלי:

הגדרה:
1. מבוי סתום הוא מצב של קודקוד $v \neq t$ כך ש$deg_{out}(v) = 0$.
2. מבוי סתום הוא מצב של קודקוד $v \neq s$ כך ש$deg_{in}(v) = 0$.

עדכון של L בעקבות מחיקה של קשת: כל עוד קיים מבוי סתום כלשהו, מחק אותו ואת כל קשתותיו.

כמה זמן לוקח הטיפול במבוי סתום?

הבחנה: כל קודקוד נהיה מבוי סתום פעם אחת בדיוק לכל היותר בפאזה. כיוון, שברגע שנמחוק אותו הוא לא יחזור להיות מבוי סתום. בנוסף, כל קשת נמחקת לכל היותר פעם אחת.

מכאן שסה"כ ישנם $O(|V|)$ קודקודים שנמחקו ו$O(|E|)$ קשתות שנמחקו. אם כן, ההחלטה כיצד למחוק היא לוקאלית - אין צורך בסריקה נוספת של הגרף בעת שמצאנו מבוי סתום, אנחנו מוחקים את השכנים ומי שקרוב אליו, אין לנו סיבה לסרוק את כל הגרף לחפש את הבעיה הבעיה לוקאלית.

לכן, סה"כ עלות כל העדכונים של L בפאזה אחת עולה $O(|E| + |V|) = O(|E|)$ זמן.

7.5 האלגוריתם של $Dinic$

להלן האלגוריתם של דיניק:

```
DINIC(G = (V, E), s, t, c)
initialize f(u, v) = 0 for all u, v ∈ V
while there exists a path from s to t in G_f
    build layer graph L
    while there exists a path P from s to t in L
        augment f on P
        update L by continuously removing dead-ends
```

הסבר על האלגוריתם:

בדומה לשיטה של פורד־פרקלסון, מאתחלים בהתחלה את הזרימה להיות $f = 0$. ההבדל בין האלגוריתמים של פורד־פרקלסון לשל דיניץ' יהיה במציאת מסלול השיפור שכאן תהיה מאוד מסויימת.

לאחר מכן, נכנסים ללולאת $while$ כל עוד ישנו מסלול מs לt בG_f (בדיקה באמצעות BFS) , בדומה לתנאי של פרקלסון.

בכל פאזה של האלגוריתם:

א. בונים גרף שכבות L

ב. מתחילים איטרציה: כל עוד קיים מסלול P מs אל t בL:

1. אם מצאנו ־ משפרים את f על המסלול. (זהה לתהליך שקורה אצל פורד פרלקסון). מבצעים את הפעולות הבאות ־

$$
\begin{aligned}
& c_f(P) \leftarrow \min_{(u,v)\in P}\{c_f(u,v)\} \\
& \textbf{for } \text{each edge } (u,v) \in P \\
& \qquad f(u,v) \leftarrow f(u,v) + c_f(P) \\
& \qquad c_f(u,v) \leftarrow c_f(u,v) - c_f(P) \\
& \qquad f(v,u) \leftarrow f(v,u) - c_f(P) \\
& \qquad c_f(v,u) \leftarrow c_f(v,u) + c_f(P) \\
& \qquad \text{update } E_f
\end{aligned}
$$

2. לאחר ששיפרנו ־ מעדכנים את L כמו שאמרנו קודם לכן: מורידים את המבויים הסתומים.

נכונות האלגוריתם ־ נובעת ישירות מפורד פרקלסון. נשים לב שלפי אדמונס קארפ יהיו לנו לכל היותר $|V|-1$ גרפי שכבות.

מה באשר לזמן הריצה?

האתחול עלותו $O(|V|^2)$. ישנם $O(|V|)$ פאזות ולכן שלב ב' יתבצע $O(|V|)$ פעמים. כל שלב שכזה:

בניית גרף שכבות עלותו $O(|E|)$

לאחר מכן נכנסים ללולאת $while$ של איטרציות. כל איטרציה עולה לבדיקה האם קיים מסלול ב$O(|V|)$ + שיפור על המסלול ב$O(\ell)$ זמן, וכן עדכון גרף השכבות עלותו $O(|E|)$ על כל הפאזה (!) ־ לא כל איטרציה.

כמה איטרציות ישנן בכל פאזה? נניח שמס' זה הוא k. אם ישנן k איטרציות בפאזה ־ אזי כל פאזה תעלה $O(|E| + k \times \ell)$ באשר ℓ הוא מס' השכבות כאשר $\ell = O(|V|)$. כמו כן, ישנם $O(|V|)$ פאזות, ונקבל כי זמן הריצה הוא:

$$|V| \times (|E| + k \times \ell) \leq |V| \times (|E| + |E||V|) = O(|V|^2|E|)$$

כיוון ש$\ell \leq |V|-1$ וכן $k \leq |E|$ (שכן ברגע שקשת נהיית רוויה בפאזה מסוימת, היא לעולם לא תהיה חלק ממסלול שיפור נוסף. לכן בהכרח ישנם לכל היותר $|E|$ מסלולי שיפור בכל פאזה ־ איטרציות).

8 הרצאה 8: BMM, משולשים וקל כבד

8.1 BMM

בכפל מטריצות רגיל מתקיים לפי עדי בן צבי ־ $C_{ij} = \sum_{k=1}^{n} A_{ik}B_{kj}$.

בכפל מטריצות בוליאני (BMM) שנקרא $Boolean\ Metrix\ Multipcation$:

$$C_{ij} = \bigvee_{k=1}^{n} a_{ik} \wedge b_{kj}$$

כלומר, כאשר מייצגים מטריצות בוליאניות אנחנו מסתכלים על המיקום הij. הוא מכפלה של השורה הi במטריצה A והעמודה הj במטריצה B, מספיק שיהיה קיים אינדקס אחד k עבורו הערך הk בשורה הi והערך הk בעמודה הj הוא 1, אזי נגדיר $c_{ij} = 1$. אחרת, $c_{ij} = 0$.

נסמן בω את האקספוננט של האלגוריתם הכי מהיר שקיים לפתרון בעיית כפל המטריצות. בהכרח, $\omega \leq 2.37287$ כי כיום ידוע אלגוריתם שפותר בזמן זה, וכן נקבל סה"כ כי:

$$2 \leq \omega \leq 2.37287$$

מכאן, האלגוריתם הכי מהיר לכפל מטריצות ריבועיות מגודל $n \times n$ עולה $O(n^{\omega})$ זמן.

8.1.1 BMM קומבינטורי והשערת הBMM

המושג של אלגוריתם קומבינטורי אינו מוגדר היטב. עם זאת, ניתן לומר מהו אלגוריתם לא קומבינטורי.
האלגוריתמים של FMM משתמשים בפעולות חיסור והמקדמים הקבועים בזמן הריצה בדרך כלל גדולים מאוד (!). מכאן - שבדרך כלל זה לא מעשי להשתמש באלגוריתמים האלו (פרט לאלגוריתם של שטרסן).
מצד שני, אלגוריתמים "קומבינטוריים" נוטים להיות מהירים מבחינת מקדמי הקבועים וגם הם יחסית אלגוריתמים פשוטים. ומדוע זה טוב? קל לממש אותו.
הנאיבי שעלותו $O(n^3)$ מאוד קל למימוש, וכן המהיר ביותר (הקומבינטורי) שקיים היום בסיבוכיות $O(\frac{n^3}{log^4 n})$ של Yu.
אחת מהשאלות הכי מעניינות ופתוחות כיום בעולם הקומבינטוריקה ומדעי המחשב היא -
האם קיים אלגוריתם " קומבינטורי" לBMM שזמן הריצה שלו הינו $O(n^{3-\varepsilon})$ באשר $\varepsilon > 0$?
נראה כי אפילו אם ימצאו אלגוריתם שרץ בסיבוכיות זמן $O(n^{2.9999})$ בהכרח יתקיים

$$n^{2.9999} <<< \frac{n^3}{log^4 n}$$

השערה: לכל קבוע $\varepsilon > 0$ לא קיים אלגוריתם קומבינטורי שזמן הריצה שלו הוא $O(n^{3-\varepsilon})$ לפתרון BMM.
בעזרת ההשערה הזו, ניתן להוכיח חסמים תחתונים לאלגוריתמים "קומבינטוריים" עבור כל מיני בעיות (כתלות בנכונות ההשערה).

אנחנו נתעלם מפקטורים של $log(n)$ וכו', נתעניין רק באקספוננט. מכאן נגדיר סימון חדש $\tilde{O}()$ שיציין התעלמות זאת. לדוגמה:

$$O(n^2 logn) = \tilde{O}(n^2)$$

$$O(\frac{n^3}{log^4 n}) = \tilde{O}(n^3)$$

8.2 זיהוי משולשים בגרף

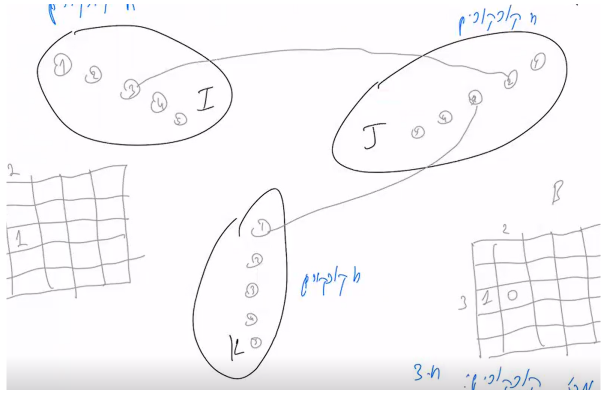
הגדרה: יהי $G=(V,E)$ גרף. מסלול (v_0,v_1,v_2,v_0) יקרא משולש. כלומר, משולש הוא מעגל בגודל 3.

קלט: $G=(V,E)$ גרף לא מכוון.
פלט: ישנם מס' אפשרויות.
1. האם בG יש משולש?
2. אם בG יש משולש, מצא אחד שכזה.
3. דווח על כל המשולשים שיש בG.

אנחנו נתמקד בבעיה 1: האם בG יש משולש?
נניח כי G מיוצגת ע"י מטריצת שכנויות M. ניתן לבדוק אם בG יש משולש ע"י בדיקה של M^3 בכפל BMM.
טענה: $\exists_{1\le i\le n} M[i][i]=1$ (באלכסון) $\Longleftrightarrow$ בG יש משולש.
מסקנה: ניתן לזהות משולש ע"י 2 חישובים של BMM, בצורה לא קומבינטורית נוכל לטעון שסיבוכיות האלגוריתם הינה $O(n^\omega)$. בצורה קומבינטורית, נטען כי קיים אלגוריתם קומבינטורי בזמן $\tilde{O}(|V|^3)$.

שאלה: האם ניתן לזהות משולש בגרף G בזמן יותר מהיר פולינומית?
טענה (משפט 5): אם קיים אלגוריתם קומבינטורי שפותר זיהוי משולשים בזמן $\tilde{O}(|V|^{3-\varepsilon})$ אזי קיים אלגוריתם קומבינטורי שפותר את BMM בזמן $\tilde{O}(n^{3-\frac{\varepsilon}{3}})$.
מסקנה: על סמך השערת הBMM הקומבינטורית, לא קיים אלגוריתם קומבינטורי שפותר זיהוי משולשים בזמן $\tilde{O}(|V|^{3-\varepsilon})$.

הוכחה:
קלט: A,B מטריצות בוליאניות מגודל $n\times n$ וכן אלגוריתם קומבינטורי שפותר זיהוי משולשים בזמן $\tilde{O}(|V|^{3-\varepsilon})$
פלט: $C=A\times V$ כאשר הכפל הוא BMM
נבנה גרף תלת צדדי (כמו דו צדדי, רק של שלושה חלקים), אשר החלקים יקראו I,J,K ובכל חלק מס' שווה של קודקודים ונוסיף קשת בין $v\in I$ לבין $u\in J$ אם כשנסתכל על מטריצת הקלט מתקיים בה $A[v][u]=1$, ובדומה תהיה קשת בין $v\in I$ לבין $u\in K$ אם $B[v][u]=1$.
פורמלית מעט יותר: לכל $A_{ij}=1$ נוסיף קשת מהקודקוד הi בI לקודקוד הj בקבוצה J. באותו אופן, לכל $B_{ij}=1$ נוסיף קשת מהקודקוד הi בJ לבין הקודקוד הj בK.



קיבלנו גרף חדש, מתקיים בו $|V| = 3n$ וכן $|E| = O(n^2)$ (שכן מס' הצלעות הוא כמס' ה1 בA ובB, במקרה הגרוע המטריצות כולן אחדות.)
כמו כן, נוסיף את כל הקשתות האפשריות בין K לI. מכאן קיבלנו שבין I לK יש ממס' n^2 צלעות - בכל אחד מהם יש n קודקודים.

טענה - עבור $i \in I$ ו$k \in K$ הקשת (i,k) נמצאת במשולש $\iff C[i][k] = 1$.

הוכחה:

נניח כי הקשת (i,k) נמצאת במשולש. אזי, קיים אינדקס j עבורו קיימות הקשתות (k,j) וכן (i,j).
מכאן, לפי ההגדרה $A[i,j] = A[j,i] = 1$ וכן $B[k,j] = B[j,k] = 1$.

$$C_{ik} = \bigvee_{m=1}^{n} a_{im} \wedge b_{mk}$$

בפרט עבור $m = j$ נסתכל ונקבל: $a_{ij} \wedge b_{jk} = 1 \wedge 1 = 1$ ונקבל:

$$C_{ik} = (\bigvee_{m=1\neq j}^{n} a_{im} \wedge b_{mk}) \wedge 1 = 1$$

ומכאן $C[i][k] = 1$.
נניח כי $C[i][k] = 1$. אזי, קיים אינדקס j עבורו $a_{ij} = 1 \wedge b_{jk} = 1$ כלומר קיימת קשת $(i,j), (j,k)$. אנו יודעים כי הקשת (i,k) קיימת כי כל הקשתות בין I וK קיימות, מכאן שמקיום שתי הקשתות האחרות נקבל כי (i,k) במשולש.

נסיון ראשון (לא יעבוד):

א. אתחל $C = 0$

ב. כל עוד בG יש משולש $(i,j,k) \in I \times J \times K$:

1. קבע $c_{i,k} = 1$
2. הורד את הקשת (i,k) מהגרף.

הבחנה ראשונה: מס' האיטרציות הוא $O(n^2)$, שכן כל איטרציה מורידה קשת אחת בין I לK - ולכן אחרי $O(n^2)$ פעמים אין יותר משולשים. כלומר, זמן הריצה הוא n^2 כפול זמן הריצה של אלגוריתם לזיהוי משולשים. מכאן נקבל:

$$\tilde{O}(|V|^{3-\varepsilon}) \times n^2 \Longrightarrow_{|V|=3n} \tilde{O}(3^{3-\varepsilon} \times n^{3-\varepsilon+2}) = \tilde{O}(n^{5-\varepsilon})$$

וזמן הריצה לא תואם למה שרצינו לעשות. חבל. בעיה נוספת: האלגוריתם שאנחנו ניסינו לבנות מוצא משולש, האלגוריתם עליו דיברנו כ"קופסא שחורה" יודע **לזהות משולש.**

כעת נרצה לשפר את זמן הריצה. נשתמש באותם I, J, K אך הפעם נחלק כל אחד מהם ל-t חלקים: $(I_1, ..., I_t), (J_1, ..., J_t), (K_1, ..., K_t)$, בכל חלק יש $\Theta(\frac{n}{t})$ קודקודים.

האלגוריתם:

א. נאתחל $C = 0$

ב. עבור כל שלשה (I_x, J_y, K_z) נריץ את האלגוריתם הקודם:

כל עוד קיים משולש בגרף שמושרה ע"י השלשה הנוכחית $(i, j, k) \in I_x, J_y, K_z$:

1. קבע $C_{ik} = 1$
2. נוריד את (i, k) מהגרף G.

ובעברית: תחילה נריץ את (I_1, J_1, K_1). אם נמצא משולש בין חלקים אלו אנחנו נוריד את הקשת (i, k) הרלוונטית. לאחר מכן נריץ את האלגוריתם שוב **על אותה שלשה.** אם נמצא עוד משולשים - נבצע תהליך דומה גם כן. לאחר מכן נריץ את (I_1, J_2, K_1) וכן הלאה - על כל שלשה.

נכונות: האלגוריתם עובד מאותה הסיבה שהקודם עובד - מוצאים את כל המשולשים בגרפים המושרים, ובפרט כל הגרפים המושרים מהווים יחד את הגרף כולו G.

זמן הריצה:

כמה שלשות יש בגרף? יש t אפשרויות ב-i, j, k. מכאן שיש t^3 שלשות.

נאמר שהרצה של אלגוריתם למציאת משולש נכשלה אם התשובה היא שאין משולש, ואחרת היא הצליחה.

כל כשלון מוביל לשלשה חדשה, וכל הצלחה קובעת ערך ב-C. כמה פעמים ניתן להגיע לכשלון? לכל היותר t^3 כשלונות. כמה פעמים אפשר להצליח? לכל היותר n^2 כמס' התאים ב-C וכן לא יתכן שתהיה הצלחה פעמיים על אותו תא, כי אנחנו מורידים את הקשת (i, k). כמו כן, בכל אחת מ-t^3 השלשות ישנם $\frac{n}{t}$ קודקודים עליהם מופעל האלגוריתם.

מס' הפעמים שהאלגוריתם מריץ את האלגוריתם לזיהוי משולשים הינו $t^3 + n^2$. כמו כן, את האלגוריתם הנ"ל הוא מפעיל בכל פעם על גרף מגודל $3\frac{n}{t}$, מכאן שזמן הריצה יהיה

$$(t^3 \times +n^2) \times \tilde{O}((3\frac{n}{t})^{3-\varepsilon}) =$$

$$3^{3-\varepsilon}(t^3(\frac{n}{t})^{3-\varepsilon} + n^2(\frac{n}{t})^{3-\varepsilon}) =$$

אם נשווה את שני הביטויים נקבל כי

$$t^3(\frac{n}{t})^{3-\varepsilon} = n^2(\frac{n}{t})^{3-\varepsilon}$$

$$t^3 = n^2 \Longrightarrow t = n^{\frac{2}{3}}$$

עבור $t = n^{\frac{2}{3}}$ נקבל את זמן הריצה:

$$n^2 \times n^{\frac{3-\varepsilon}{3}} + (n^{\frac{2}{3}})^3 n^{\frac{3-\varepsilon}{3}} = \tilde{O}(n^{3-\frac{\varepsilon}{3}})$$

לסיכום, יצרנו אלגוריתם שיודע להחזיר את מטריצת הכפל בזמן הריצה של $\tilde{O}(n^{3-\frac{\varepsilon}{3}})$ באשר השתמשנו באלגוריתם שמזהה משולשים.

9 הרצאה 9: אלגוריתמים רנדומיים

9.1 מבוא והגדרה

הגדרה: יהי אלגוריתם רנדומי A. אלגוריתם רנדומי הוא אלגוריתם שמשתמש במחרוזת אקראית r כקלט. יש לכך פירושים שונים - בקורס אלגוריתמים 1: אנחנו מניחים כי כל מספר בr הוא מספר אקראי שמתפלג באופן אחיד מטווח המספרים השלמים $[0,n]$ או $[1,n]$ באשר n הוא גודל הקלט של האלגוריתם A.

דוגמה. אם הקלט שלנו הינו גרף $G=(V,E)$ אזי $n=|E|+|V|$ ונקבל שכל מספר בR הוא ממשי בטווח $[0,n]$ שמתפלג בו אחיד.

בעת שאנחנו מריצים אלגוריתמיים דטרמיניסטיים, הנכונות והסיבוכיות ברורים ואפשריים להוכחה. עם זאת, באלגוריתם רנדומי יכולים לקרות דברים שונים.

א. הריצה של אלגוריתם רנדומי A יכולה לרוץ בזמן שונה. יתכן כי זמן הריצה יהיה תלוי במחרוזת האקראית r

ב. האלגוריתם A יכול להחזיר בסוף הריצה שלו תשובה שונה. כלומר, יהיו P_1, P_2 ריצות שונות של A. יתכן כי $P_1(A) \neq P_2(A)$.

משתנים מקריים:

א. זמן הריצה של אלגוריתם רנדומי יהיה משתנה מקרי שתלוי בr.

ב. הנכונות של האלגוריתם (והצלחתו) היא גם כן משתנה מקרי שתלוי בr.

9.1.1 אלגוריתמי מונטה קרלו ואלגוריתמי לאס וגאס

הגדרה: אלגוריתמי מונטה קרלו $(Monte-Carlo)$ הם אלגוריתמים רנדומיים שזמן הריצה שלהם הוא דטרמיניסטי (ניתן לחסום אותו), אך הנכונות היא משתנה מקרי. כלומר, האלגוריתם עלול לטעות ולא להצליח. נרצה לנתח את הסיכוי לטעות: שיהיה כמה שיותר קטן.

בהינתן גודל קלט n, נרצה כי הסיכוי לטעות $\leq \frac{1}{n^\alpha}$ עבור $\alpha \geq 1$ קבוע.

אם אכן הסיכוי לטעות $\leq \frac{1}{n^\alpha}$ אזי אנחנו נאמר שהאלגוריתם צודק בסיכוי גבוה $\leq 1-\frac{1}{n^\alpha}$.

הגדרה: אלגוריתמי לאס וגאס $(Las-Vegas)$ הם אלגוריתמים רנדומיים שזמן הריצה שלהם הוא משתנה מקרי, והנכונות היא דטרמיניסטית. כלומר, האלגוריתם תמיד צודק אך זמן הריצה משתנה. באלגוריתמי לאס וגאס אנחנו נרצה לנתח את התכונות ההסתברויות של זמן הריצה. למשל: לחשב את תוחלת זמן הריצה, או חסם עליון לזמן הריצה בסיכוי גבוה.

הגדרה: אלגוריתמי אטלנטיק סיטי הם אלגוריתמים שגם זמן הריצה וגם הנכונות הינם משתנים מקריים. (לא נתעסק בהם בקורס).

9.2 וידוא כפל מטריצות

קלט: 3 מטריצות בינאריות מגודל $n \times n$. נסמנן A, B, C.
פלט: לבדוק האם $C = A \times B$ באשר הכפל הוא במודולו 2.

הערה. כפל במודולו 2 הכוונה היא שכפל הוא כמו AND וחיבור הוא כמו XOR. כלומר:
$1 \times 1 = 1, 1 \times 0 = 0, 0 \times 0 = 0, 0 \times 1 = 0$ וכן $0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 0, 0 + 0 = 0$.

אלגוריתם נאיבי: נכפיל את A בB ונבדוק האם אכן הפלט הינו C. ראינו כבר שההכפלה תעלה $O(n^{\omega})$. וזה - לא ממש טוב לנו.

9.2.1 נסיון ראשון

נראה אלגוריתם מונטה קרלו שזמן הריצה שלו הינו $O(n^2)$ שטועה בסיכוי $\leq \frac{1}{2}$.

להלן האלגוריתם:

```
VERIFY-BINARY-MM-BASIC(A, B, C)
1  Pick a random vector V⃗ = (v_1, v_2, ..., v_n) of bits
2  V⃗_B ← B · V⃗
3  V⃗_AB ← A · V⃗_B
4  if V_AB = C · V⃗
5       return "true"
6  else return "false"
```

האלגוריתם די ברור. הוא מבצע את המכפלה באמצעות וקטור ערכים אקראיים המורכב מביטים, בשני שלבים, ולאחר מכן בודק האם המכפלה הזו שקולה למכפלה של המטריצה C בוקטור. אם כן: מחזיר אמת, אחרת מחזיר שקר. נשים לב כי אם $C = AB$ אזי בהכרח לכל וקטור $\vec{v}$ יתקיים
$C \times \vec{v} = AB \times \vec{v}$
נשים לב כי אם $C = AB$ האלגוריתם יהיה צודק תמיד, אם $C \neq AB$ אבל בחירה של וקטור לא טוב יוביל את האלגוריתם לטעות. מצב זה נקרא $false\ positive$.

סיבוכיות זמן הריצה: נראה כי חישוב כל אחד מהשלבים 2 ו3 יעלה באופן נאיבי $O(n^2)$, כל אחד מהשלבים פולט וקטור בגודל n. שלב 4 מבצע שוב כפל שעולה $O(n^2)$. לבסוף הבדיקה האם V_{AB} שווה למכפלה זו עולה $O(n)$ שכן מעבר על n ערכי וקטור. סה"כ סיבוכיות האלגוריתם $O(3n^2 + n) = O(n^2)$.

ניתוח הסיכוי לטעות:
נגדיר $D = C - AB$. נניח $C \neq AB$. אזי המטריצה $D \neq 0$. נסמנה $D = \{d_{ij}\}$
משמעות הדבר: קיים $d_{ij} = 1$. (לפחות אחד שכזה).

הגדרה: נאמר כי וקטור $\vec{v}$ הוא רע אם מתקיים $D \times \vec{v} = (C - AB) \times \vec{v} = 0$ (וקטור שכזה יגרום לאלגוריתם שלנו לטעות). אם $\vec{v}$ אינו רע, אזי נאמר כי $\vec{v}$ הוא טוב. כלומר: אם $D \times \vec{v} \neq 0$.

למה 1: מספר הוקטורים הטובים הוא לפחות כמספר הוקטורים הרעים. (תחת הנחה ש$D \neq 0$)
מסקנה: אם בחרנו וקטור באקראי, והלמה אכן נכונה (מיד נוכיח), אזי הסיכוי לבחור וקטור טוב הוא לפחות $\frac{1}{2}$.

הוכחה: נתאר פונקציה חד חד ערכית שממפה וקטורים רעים לוקטורים טובים. מכאן שבהכרח יתקיים כי $|bad-vectors| \leq |good-vectors|$ לפי בדידה, ואז סיימנו את ההוכחה.

יהי $\vec{v}$ וקטור רע. כלומר, $D \times \vec{v} = 0$. כלומר, $\forall_{1\leq \ell \leq n} : \sum_{k=1}^{n} d_{\ell k} \times v_k = 0$

נזכר כי קיים $d_{ij} = 1$, כיוון שהנחנו $D \neq 0$ (אם אכן $D = 0$ אזי אין סיכוי לטעות).

נגדיר וקטור

$$\vec{w_j} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ .. \\ 0 \end{pmatrix}$$

באשר הגדרת הוקטור הינה 0 פרט למיקום j (אותו j מ$d_{ij} = 1$) בו יהיה 1.

נזכיר. יצאנו מתוך נקודת הנחה ש$\vec{v}$ הינו רע (כלומר, הוא עבד עלינו). נראה כי הוקטור $\vec{v'} = \vec{v} + \vec{v_j}$ הוא וקטור טוב.

כלומר, נוכיח כי $D \times \vec{v'} \neq 0$.

נסמן
$$\vec{v'} = \begin{pmatrix} v'_1 \\ v'_2 \\ .. \\ v'_j \\ .. \\ v'_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ .. \\ v_j \\ .. \\ v_n \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ .. \\ w_j \\ .. \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ .. \\ v_j + 1 \\ .. \\ v_n \end{pmatrix}$$

נסתכל על $(D \times \vec{v})_i$. נקבל לפי הגדרה כי -

$$(D \times \vec{v'})_i = \sum_{k=1}^{n} d_{ik} v'_k = \sum_{k=1}^{n} d_{ik}(v_k + w_k) = \sum_{k=1}^{n} d_{ik} v_k + \sum_{k=1}^{n} d_{ik} w_k$$

נשים לב כי $\sum_{k=1}^{n} d_{ik} v_k = 0$ כפי שהסברנו מעלה (כי הנחנו $D \times \vec{v}$ שווה לאפס, וטענו לגבי ℓ בפרט $\ell = i$ יתקיים).

כלומר:

$$(D \times \vec{v'})_i = \sum_{k=1}^{n} d_{ik} w_k = d_{ij} w_j + 0 + 0 + ... + 0 = 1 \times 1 = 1$$

באשר כל שאר הערכים הינם אפס פרט ל$w_j = 1$, וכן $d_{ij} = 1$. סה"כ קיבלנו כי $(D \times \vec{v})_i = 1$ ולכן $D \times \vec{v}' \neq 0$.

סה"כ קיבלנו פונקציה:

$$f(\vec{v}) = \vec{v} + \vec{w_j}$$

נוכיח כי f חד חד ערכית.

נניח בשלילה כי f אינה חד חד ערכית. כלומר, קיימים $X = \begin{pmatrix} x_1 \\ x_2 \\ .. \\ .. \\ x_{n-1} \\ x_n \end{pmatrix}$ ו $Y = \begin{pmatrix} y_1 \\ y_2 \\ .. \\ .. \\ y_{n-1} \\ y_n \end{pmatrix}$ באשר $X \neq Y$. עבורם $f(X) = f(Y)$.
נסמן, $X' = f(X) = X + w_j$ וכן $Y' = f(Y) = Y + w_j$, קיבלנו כי $\forall_{1 \leq \ell \leq n}$ מתקיים $x'_\ell = y'_\ell$. נשים לב כי הוספה או הזזה של w_j בוקטור בסך הכל משנה את הביט הj בוקטור. מכאן שאם $X' = Y'$ אזי לא יתכן כי $X = Y$. בסתירה. (בסה"כ מוסיפים בשתי הצדדים אותו דבר, ולכן בהכרח נקבל $X = Y$, בסתירה). מסקנה: f חד חד ערכית.

9.2.2 נסיון שני ־ אמפליפקציה $(Amplification)$

הרעיון הוא: נריץ את האלגוריתם הבסיסי k פעמים. נתבונן באלגוריתם הבא:

```
Verify-Binary-MM(A, B, C)
k ← α log n
for i = 1 to k
    if Verify-Binary-MM-Basic(A, B, C) = "false"
        return "false"
return "true"
```

אם באחת האיטרציות יוחזר לנו $false$ מהאלגוריתם הבסיסי, אז האלגוריתם יחזיר שקר. כלומר: בשביל שהאלגוריתם יטעה בכל האלגוריתם, ויחזיר אמת למרות שלא מתקיים $C = AB$ אזי צריך לבחור k פעמים וקטור רע.
נגדיר את המאורע: A= בכל איטרציה אנחנו בוחרים וקטור רע. נשים לב כי הסיכוי לטעות בכל איטרציה בלתי תלוי באחרות. לכן נגדיר B כמאורע של הסיכוי לטעות באיטרציה אחת.

$$Pr[A] = (Pr[B])^k \leq (\frac{1}{2})^k = \frac{1}{2^k}$$

נרצה כי זמן הריצה יהיה $\frac{1}{n^\alpha}$ עבור $\alpha \geq 2$ קבוע. זה יקרה באשר $\frac{1}{2^k} = \frac{1}{n^\alpha}$ כלומר $n^\alpha = 2^k$ ומכאן $k = log(n^\alpha) = \alpha log(n)$.
ואכן אם נגדיר את k להיות $\alpha log(n)$ נקבל כי

$$Pr[A] \leq \frac{1}{2^k} = \frac{1}{2^{\alpha log(n)}} = \frac{1}{n^\alpha}$$

סיבוכיות זמן הריצה: נראה כי הלולאה מתבצעת k פעמים, בכל איטרציה קוראים לאלגוריתם הבסיסי שעלותו $O(n^2)$ ונקבל $O(n^2 k) = O(\alpha n^2 logn) = O(n^2 logn)$.

9.3 מיון מהיר ־ $Quick\ Sort$: מבוא

בעיית המיון:

קלט: מערך $A=[a_1,...,a_n]$ של מספרים ממשיים (שונים ־ לא מהותי, מהותי מתמטית מבחינת ההוכחות).

פלט: מיון של A בסדרר עולה.

מיון מהיר:

בחירה של איבר אקראי a_r, ויצירת $pertiton$ סביב a_r. כל מה שמימינו יהיה גדול ממנו וכל מה שמשמאלו יהיה קטן ממנו. ואת הצד הימני והצד השמאלי, פותרים איך לא: ברקורסיה. לאחר הרקורסיה נקבל את L ממויין, a_r בניהם ואת R ממויין וסה"כ נשרשר את שני המערכים ונקבל את A ממויין.

באלגוריתם הקלאסי r נבחר באופן אקראי אחיד בטווח האינדקסים השלמים $[1,n]$. חשוב שנשים לב ־ תמיד האלגוריתם מצליח למיין. מה שלא תמיד קורה: הוא זמן הריצה שמשתנה. ומכאן ־ **מדובר באלגוריתם לאס וגאס**.

נסמן ב$T(n)$ את זמן הריצה על קלט בגודל n. במקרה הגרוע: $r=1$ או $r=n$ ואז אנחנו צריכים לבצע מיון על קבוצה כלשהי בגודל $n-1$ (בוודאות L או G היא קבוצה ריקה). בשני המקרים הללו נקבל כי $T(n)=T(n-1)+O(n)=O(n^2)$ כאשר הקלט קטן באחד, וכן נדרש עוד $O(n)$ לביצוע ה$partion$ (האיחוד לבסוף).

באופן כללי ־

$$T(n)=T(|L|)+T(|G)+O(n)$$

ומכאן כי במקרה הגרוע עלות האלגוריתם יכולה להגיע ל$O(n^2)$.

מקרה טוב עבורנו ־ a_r הינו החציון של A. במקרה זה נקבל:

$$T(n)=T(\frac{n}{2})+T(\frac{n}{2})+O(n)=2T(\frac{n}{2})+O(n)\in O(nlogn)$$

כלומר: אם החלוקה תהיה תמיד בדיוק בחצי, אזי עלות זמן הריצה של האלגוריתם תהיה $O(nlogn)$.

נשים לב כי ־ לא צריך חלוקה עד כדי כך טובה. גם אם החלוקה היא כזו שגודל כל צד הוא לפחות רבע מn, זמן הריצה יצא עדיין $O(nlogn)$. ויותר מזה ־ אם כל צד הוא לפחות $\frac{n}{a}$ עבור a קבוע כלשהו, אזי עדיין זמן הריצה יהיה $O(nlogn)$. הבחנה זו, תוביל אותנו לגרסה מעט שונה של $Quick\ Sort$.

9.4 מיון מהיר פרנואידי

האלגוריתם מבצע שינוי מאוד פשוט באלגוריתם המקורי: לאחר שמבצעים חלוקה, אם $|L|<\frac{n}{4}$ או $|G|<\frac{n}{4}$ אזי האלגוריתם מבטל את בחירת ה$Pivot$ הנוכחי ומחפש $Pivot$ חדש. (באותו האופן).

זמן ריצה: אנחנו נרצה לנתח את תוחלת זמן הריצה, שכן זמן הריצה אינו קבוע. חשוב להדגיש שכל חישובי $T(n)$ קודם לכן נבעו מאינטואיציה, ואינם היו מדוייקים. נרצה לחשב כעת את תוחלת זמן הריצה שהיא הגורם המכריע באלגוריתמי לאס וגאס.

נזכר בכלל התוחלת $E[X+Y]=E[X]+E[Y]$.

$$E[T(n)]=E[T(|L|)+T(|G|)+T(pivot)]=E[T(|L|)]+E[T(|G|)]+E[T(pivot)]$$

באשר $T(pivot)$ זהו הזמן שלוקח למצוא $pivot$ טוב. נשים לב כי $|G| + |L| = n - 1$. מכאן:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + E[T(pivot)]$$

נתמקד כעת ב$E[T(pivot)]$. נשים לב כי $T(pivot)$ הינו מס' הפעמים שמחפשים $pivot$ כפול $O(n)$, שכן בכל חיפוש $Pivot$ סורקים את המערך פעם אחת (בשביל למצוא מי מעליו ומתחתיו ולגלות את גדלי הקבוצות). נזכר כי $E[aX] = aE[X]$ ומכאן שנסמן T: מס' הפעמים שמחפשים $Pivot$. נקבל:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + O(n) \times E[T]$$

בחירת ה$Pivot$ היא ניסוי שמבצעים שוב ושוב, עד שמצליחים למצוא $Pivot$ טוב. מספר הנסיונות עד ההצלחה מתפלג גאומטרית. ולכן אם ההצלחה בכל ניסוי הינה p, אזי תוחלת מס' הנסיונות עד ההצלחה הינה $\frac{1}{p}$. נחשב כעת את p. על מנת ש$Pivot$ יהיה רע, הוא צריך להיות או ברבע האיברים הכי קטנים כי אז $|L| < \frac{n}{4}$. או ברבע האיברים הכי גדולים, ואז $|G| < \frac{n}{4}$. כלומר: בשביל ש$pivot$ יהיה טוב הוא צריך להיות בטווח $[\frac{n}{4}+1, \frac{3n}{4}-1]$. נשים לב כי אלו בדיוק חצי מהאפשרויות ל$pivot$. מכאן המסקנה כי

$$p = Pr[Good - Pivot] = \frac{1}{2}$$

מכאן, $E[T] = \frac{1}{p} = \frac{1}{\frac{1}{2}} = 2$. סה"כ קיבלנו עד כה:

$$E[T(n)] = E[T(n-1-|G|] + E[T(|G|)] + 2 \times O(n)$$

נגדיר פונקציה $T'(k) = E[T(k)]$ ונסמן $|G| = x$. נקבל

$$E[T(n)] = T'(n-1-x) + T'(x) + 2 \times O(n)$$

כיוון שאנו יודעים כי $\frac{n}{4} \le x \le \frac{3n}{4}$ נוכל לראות כי $E[T(n)] = T'(n) = O(nlogn)$. כי: $\frac{n}{4} \le x \le \frac{3n}{4} \Longrightarrow -\frac{n}{4} \ge -x \ge -\frac{3n}{4} \Longrightarrow n-1-\frac{n}{4} \ge n-1-x \ge n-1-\frac{3n}{4} \Longrightarrow \frac{3n}{4}-1 \ge n-1-x \ge \frac{n}{4}-1$ ונקבל כי

$$E[T(n)] \le T(\frac{3n}{4}-1) + T(\frac{3n}{4}) + 2O(n) \le 2T(\frac{3n}{4}) + O(n)$$

והביטוי מימין חסום ב$O(nlogn)$ עם עץ רקורסיה או סתם אינדוקציה שנחסוך כעת. מש"ל.

9.5 תוחלת זמן הריצה של מיון מהיר "קלאסי"

בהתייחס לאלגוריתם הכללי של מיון מהיר, נרצה לחסום את תוחלת מס' ההשוואות.

מתי האלגוריתם מבצע השוואות? האם האלגוריתם משווה בין כל זוג איברים? האלגוריתם לא משווה בין כל זוג איברים. ננסה להבין למה: האלגוריתם מקבל כקלט את המערך A ובוחר r. נניח כי בצד L ישנו a_i ובצד R ישנו a_j. אזי נשים לב: a_i וa_j לא ישוו זה מול זה לעולם.

אזי, מתי האלגוריתם כן ישווה בין a_i לa_j? נהפוך את צורת ההסתכלות. אמרנו כי $A = (a_1, ..., a_n)$. נסמן את הפלט $(y_1, ..., y_n)$. (בהכרח מתקיים $y_1 < y_2 < ... < y_n$).

כעת נשאל - מתי האלגוריתם משווה בין y_i לy_j? זה קורה מתי שy_i או y_j נבחרו ל$pivot$ ועד לאותו רגע, עבור כל $i \leq k \leq j$ אף אחד מהאיברים y_k לא נבחר להיות $pivot$. (אם נבחר - הם לא ישוו). כל עוד לא נבחר y_k שכזה להיות ה$pivot$ אזי y_i וy_j יהיו יחד בקריאה הרקורסיבית.

אם בפעם הראשונה שהאלגוריתם בוחר $pivot$ מתוך האיברים $y_i, y_{i+1}, .., y_j$ הוא בחירה של y_i או y_j אזי באותה בחירה האלגוריתם משווה בין y_i לבין y_j.

נגדיר משתנה מקרי ונסמן מאורע A: האלגוריתם משווה בין y_i לy_j.

$$X_{ij} = \left\{ \begin{array}{cc} 1 & A: exsist \\ 0 & o.w \end{array} \right\}$$

$$E[T(n)] = (*)E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

$(*)$-נשים לב כי זהו בדיוק מס' ההשוואות.

$$E[X_{ij}] = 1 \times Pr[x_{ij} = 1] + 0 \times Pr[x_{ij} = 0] = Pr[x_{ij} = 1]$$

כפי שאמרנו, ההסתברות ש$Pr[x_{ij} = 1]$ היא ההסתברות שתהיה השוואה בין y_i לבין y_j. אמרנו קודם שהם ישוו רק אם כל האיברים ביניהם לא נבחרו להיות $Pivot$ ואחד מהם נבחר. בטווח ישנם $j - i + 1$ איברים. מתוכם, שני איברים אם נבחרים ראשונים (y_i, y_j) יגררו ש$X_{ij} = 1$. הבחירה כאן אקראית - יוניפורמית. הסיכוי שx_i או x_j יבחרו ראשונים בטווח הינו $\frac{2}{j-i+1}$. מכאן $E[x_{ij}] = Pr[x_{ij} = 1] = \frac{2}{j-i+1}$. נקבל

$$E[T(n)] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = 2\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} =$$

$$2\sum_{i=1}^{n-1}(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + .. + \frac{1}{n-i+1}) \leq 2\sum_{i=1}^{n-1} \sum_{\ell=1}^{n-j+1} \frac{1}{\ell} = 2\sum_{i=1}^{n-1} ln(n-i+1)$$

$$\leq 2\sum_{i=1}^{n-1} log(n) = O(nlogn)$$

כנדרש.

מסקנה: מיון מהיר מבצע בתוחלת $O(nlogn)$ השוואות.

9.6 מיון דלי $(Bucket\ Sort)$

האלגוריתם הינו דטרמיניסטי. אם כן, נניח שהקלט מיוצר בצורה אקראית. נפתור את בעיית המיון כאשר כל מספר בקלט נבחר בהתפלגות אחידה בקטע $[0,1]$. הרעיון יהיה שהאלגוריתם אינו מבוסס השוואות, ולכן בתוחלת נצליח לרדת מ$O(nlogn)$. עם זאת במקרה הכי גרוע נגיע לסיבוכיות כזו גם.
האלגוריתם יעבוד כך: נקח את טווח המספרים $[0,1]$ ונחלק אותו לn דליים. כלומר $[0,\frac{1}{n},\frac{2}{n},\frac{3}{n},...,\frac{n-1}{n},1]$.
נתחיל "לזרוק" את האיברים אחד אחד. לכל דלי יכנסו מס' של איברים. בתוחלת: בכל דלי, יש איבר אחד. שכן יש n מספרים והסיכוי של כל איבר להכנס לתא הוא $\frac{1}{n}$ ליפול בתא מסויים. מכאן שתוחלת מס' האיברים בדלי תהיה $n \times \frac{1}{n} = 1$. אם תאורטית - בכל דלי נפל איבר אחד, אזי קיבלנו מיון של האיברים שכן הדליים בסדר עולה. אם לא, ויתכן מצב תאורטי שבו נפלו הרבה בתא אחד, אז אנחנו בבעיה. פסודו לאלגוריתם יראה כך -

מיון דלי $(A=(a_0,..,a_n))$
- נכניס כל איבר לדלי המתאים (האיבר נבחר מראש אקראית!)
- נמיין כל דלי באמצעות מיון הכנסה
- נשרשר את הדליים לפי סדרם

זמן הריצה:
נסמן $B_i=[\frac{i-1}{n},\frac{i}{n}]$ הדלי הi. נסמן בn_i את מס' האיברים בB_i. נראה כי n_i הוא משתנה מקרי. כמו כן $\sum_i n_i = n$. נראה כי

$$\forall_{1\le i\le n}: E[n_i]=\frac{1}{n}\times n=1$$

להכניס כל איבר לדלי המתאים עולה $O(n)$ זמן. שרשור הדליים גם כן יעלה $O(n)$ זמן. נראה כי השלב השני, מיון כל דלי באמצעות מיון הכנסה, יעלה לכל n_i $O(n_i^2)$ כי משתמשים במיון הכנסה.
מכאן, זמן הריצה הינו $O(n)+\sum_{i=1}^{n} n_i^2$
נסתכל על תוחלת זמן הריצה.

$$E[n+\sum_{i=1}^{n} n_i^2]=E[n]+\sum_{i=1}^{n} E[n_i^2]=n+\sum_{i=1}^{n} E[n_i]^2$$

נתמקד בבאקט B_i ובn_i. נשים לב שהאלגוריתם לוקח n איברים וכל אחד מהם מצליח להכנס אל הבאקט B_i בסיכוי $\frac{1}{n}$. n_i סופר את מספר ההצלחות. כלומר: יש לנו ניסוי שחוזר n פעמים עם סיכוי הצלחה $p=\frac{1}{n}$. ספירת מס' ההצלחות היא התפלגות בינומית $n_i \sim Bin(n,\frac{1}{n})$. מכאן ש

$$E[n_i]=\frac{1}{n}\times n=1$$

$$Var[n_i]=np(1-p)=1-\frac{1}{n}$$

נזכר כי $Var[n_i]=E[n_i^2]-E[n_i]^2$ ומכאן נוכל לקבל כי

$$E[n_i^2]=Var[n_i]+E[n_i]^2=1-\frac{1}{n}+1^2=2-\frac{1}{n}$$

סה"כ נחזור מעלה ונקבל

$$E[n+\sum_{i=1}^{n}n_i^2]=n+\sum_{i=1}^{n}E[n_i]^2=n+n(2-\frac{1}{n})=n+2n-1=3n-1=O(n)$$

כלומר, תוחלת זמן הריצה הינה $O(n)$.

נותרת השאלה: למה בחרנו במיון הכנסה? באמצעות מניפולציות מתמטיות קל לטפל בn_i^2 הרבה יותר מ$n_i log(n_i)$, כמו כן: בחרנו תאים מאוד קטנים ואנחנו מצפים שלא יהיה שם הרבה ערכים. מיון הכנסה הוא המיון הכי טוב עבור קלטים מאוד מאוד קטנים ־ הקבועים די קטנים. לעומת זאת, האלגוריתמים של $O(nlogn)$ מחזיקים קבועים גדולים ונהיים יעילים רק עבור n גדול.

10 הרצאה 10: שבירת סימטריה

10.1 מבוא והגדרת הבעיה

נניח כי יש לנו שני אנשים: אליס ובוב. שניהם נמצאים בשיחת זום, ואצל שניהם המצלמות כבויות. כל אחד מהם רוצה לומר משהו אל הצד השני. למשל: אליס רוצה לומר לבוב שקוראים לה אליס, ובדומה בוב רוצה להגיד לאליס ששמו הוא בוב. יותר מזה: יתכן שלשניהם קוראים אליס (לא בהכרח שהם בשם שונה). אם שניהם ידברו בו זמנית, הם יעלו על הקול אחד של השני ולא יצליחו לשמוע את הקול. המטרה היא להגיע למצב שרק אחת "משדרת" קול.

נשים לב כי כל אלגוריתם דטרמיניסטי לא יצליח לפתור את הבעיה. מדוע? מהו אלגוריתם דטרמיניסטי? אלגוריתם דטרמיניסטי הוא קוד כתוב שידוע מראש. ולכן אם בוב ואליס ישתמשו בקוד שנמצא במחשב שלהם בו זמנית, הוא יגיד להם לעשות אותו הדבר בדיוק. מכאן שאנחנו חייבים להשתמש ברנדומיות.

בעזרת מטבע רנדומי אצל כל אחד מהמשתתפים ניתן להצליח. נניח כי נגדיר את הטלת 1 להיות שהמשתתף מדבר ו0 שהמשתתף שותק. נראה כי במקרה של אליס ובוב מס' האפשרויות להטלת המטבע הינו:

$$00,11,01,10$$

מצב טוב עבורנו הוא $01,10$. ומכאן מה הסיכוי להצלחה ושיהיה משתתף אחד בדיוק שמדבר? $\frac{1}{2}$.

מכאן אפשר לקבל אלגוריתם: כל עוד אין הצלחה ־ הטל מטבע, אם יוצא אחד אז תשדר.

נראה כי מס' הנסיונות עד להצלחה הראשונה הוא משתנה מקרי שמתפלג גאומטרית, ולכן תוחלת מס' הנסיונות תהיה $2=\frac{1}{\frac{1}{2}}=\frac{1}{p}$.

ניתן גם לבדוק אחרי כמה סיבובים תהיה הצלחה בסיכוי גבוה. נראה כי כדי שלא תהיה הצלחה בk סיבובים ההסתברות הינה:

$$\frac{1}{2}\cdot\frac{1}{2}\cdot....\cdot\frac{1}{2}=\frac{1}{2^k}$$

עבור $k=clogn$ עבור $c>1$ קבוע, נקבל כי הסיכוי לא להצליח בk נסיונות הוא:

$$\frac{1}{2^k}=\frac{1}{2^{clogn}}=\frac{1}{n^c}$$

כלומר, הסיכוי לא להצליח קטן פולינומית. ומכאן שיש הצלחה בסיכוי די גבוה.

אפשר להרחיב את הבעיה, מה אם יש שלושה אנשים ואנו רוצים לשבור סימטריה? נרצה שכל אחד ינסה לשדר בסיכוי שליש. בשביל שתהיה הצלחה בניסוי אחד: צריך שאחד ישדר, ושניים אחרים ישתקו. הסיכוי לכך (מתפלג בינומית) הינו:

$$\binom{3}{1} \times p \times (1-p)^2 = 3 \times \frac{1}{3} \times (\frac{2}{3})^2 = \frac{4}{9}$$

ומכאן שבתוחלת לאחר $\frac{1}{\frac{4}{9}} = \frac{9}{4} = 2.25$ סיבובים תהיה הצלחה.

וכמובן איך לא, מה קורה כאשר ישנם n שחקנים? נשים לב (גם כשיש 3 שחקנים) שהשחקנים זקוקים לדעת מהו n. נניח כי כל אחד מהם מנסה לשדר בסיכוי p. הסיכוי לשידור הינו:

$$Pr[Success] = \binom{n}{1} \times p \times (1-p)^{n-1}$$

נרצה למקסם את הסיכוי להצלחה, כלומר p. מכאן שהסיכוי להצלחה המקסימלי יתקבל כאשר $p = \frac{1}{n}$ (גזירה פשוטה מראה זאת)
מכאן שהסיכוי להצלחה הינו:

$$Pr[Success] = \binom{n}{1} \times \frac{1}{n} \times (1-\frac{1}{n})^{n-1} = (1-\frac{1}{n})^{n-1} = \frac{(1-\frac{1}{n})^n}{1-\frac{1}{n}} \leq \frac{e^{-1}}{1-\frac{1}{n}} =$$

$$\Longrightarrow Pr[Success] \leq \frac{n}{e(n-1)} \leq \frac{1}{e}$$

ולכן תוחלת מס' הנסיונות עד להצלחה תהיה **בערך** $e = \frac{1}{\frac{1}{e}}$.

10.2 המודל המבוזר המקומי

נניח שיש n מחשבים ברשת מחשבים, כל קודקוד ייצג מחשב וישנן קשתות בין מחשבים (גרף). כל קשת היא חיבור רשת בין מחשבים. לכל מחשב יש כוח חישוב אינסופי. כלומר – אנחנו נניח שכל מחשב באופן מקומי יכול להריץ אלגוריתמים מאוד מסובכים שזמן הריצה שלהם מאוד גבוה: בחינם. כלומר: תאורטית, כל מחשב יכול לפתוח בעיות NP קשות בשנייה. בכל יחידת זמן, כל מחשב יכול לשלוח הודעות גדולות כרצונו לכל אחד משכניו (זה יעלה סיבוב אחד של תקשורת). נראה כי בזמן שקודקוד אחד שולח הודעה לשכניו, גם שאר הקודקודים שולחים הודעה לשכניהם. כלומר: שליחת ההודעות מתבצעת במקביל.

במודל הזה, נמדוד את היעילות של האלגוריתמים שלנו באמצעות מס' סבבי התקשורת.
למשל, בהינתן גרף כנ"ל:

$$a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow .... \rightarrow a_d$$

אנחנו נאלץ לd סבבי תקשורת, בסבב הראשון ההודעה תצא מa_1 אל a_2, בשלב השני מa_2 לa_3 וכן הלאה. אין לנו ברירה אחרת כאן כיוון שאין לנו דרכי קיצור.

בחלק זה של הקורס: נסמן בn את מס' הקודקודים, ונניח כי כל מחשב יודע את n. לכל מחשב **אין** id (יש id אם מרשים שימוש ב$random$).

במודל המבוזר המקומי, ישנן שתי בעיות של שבירת סימטריה שיוכלות לעניין אותנו.

צביעה: המטרה היא לצבוע את קודקודי הגרף (לתת מספרים שהם צבעים) כך שלכל קשת שני הקודקודים צבועים בצבעים שונים. נשים לב שללא רנדומיות לא ניתן לפתור את הבעיה, שכן יתכן גרף סימטרי עם שני קודקודים v_1, v_2. לכל אחד מהם יש שלושה שכנים נוספים. אז מבחינת כל קודקוד יש אותו, יש לו שלושה שכנים ויש קודקוד נוסף שיש קשת בניהם שגם לו יש שלושה קודקודים. אין שוני בניהם ולכן הם יבצעו את אותה ההחלטה באלגוריתם דטרמיניסטי.

מציאת קבוצה בלתי תלויה מקסימלית: בהינתן גרף רוצה לבחור תת קבוצה של הקודקודים שהיא 1. מקסימלית (במובן הלוקאלי: כלומר לא ניתן להוסיף עוד קודקוד ולהשאר בקבוצה בת"ל) , 2. אין זוג שכנים שנבחר. במדעי המחשב לרוב מדברים על קבוצה בלתי תלויה מקסימומית (מקסימום)- כאן רוצים למצוא את הקבוצה הבלתי תלויה בגודל הכי גדול. - זו בעיה הרבה יותר קשה.

10.3 בעיית הצביעה

נזכר כי בגרף דו צדדי ניתן תמיד לצבוע אותו בשני צבעים, גרף תלת צדדי ניתן לצביעה בשלושה צבעים. גרף n-צדדי ניתן לצביעה בn צבעים. (שכן די ברור הרעיון אין קודקודים בתוך כל צד אז אפשר לצבוע כל צד בצבעים שונים).

באופן כללי, הקושי הוא במציאת המספר הקטן ביותר של צבעים שבהם ניתן לצבוע את הגרף באופן חוקי. מדובר בבעיה מאוד קשה. לא ניתן לפתור אותה באופן דטרמיניסטי בפחות מ$O(2^n)$. ואף - מאמינים כי לא ניתן להגיע לזמן פולינומי. אם כן, יש משפחות של גרפים שניתן לצבוע אותם באופן פולינומי. גרף דו צדדי למשל.

נסמן בΔ את הדרגה הכי גדולה בגרף. כלומר, לכל קודקוד $v \in V$ ישנה דרגה $\Delta \leq deg(v)$.
הבחנה: ניתן לצבוע את הגרף באופן חוקי ב$\Delta + 1$ צבעים. מדוע? נתחיל בקודקוד מדרגה d כלשהי, בהכרח $d \leq \Delta$, גם אם $d = \Delta$ הוא משוייך לΔ קודקודים שכל אחד מהם תפס צבע אחר, במקרה הגרוע ביותר שאכן $d = \Delta$ עדיין הוא יוכל להשתמש בצבע האחרון. באופן כללי הוא יוכל להשתמש ב$\Delta + 1 - d \geq \Delta + 1 - \Delta = 1$ צבעים.

10.4 צביעה במודל המבוזר המקומי

נניח שיש לנו קודקוד v ויש לו שכנים $v_1, .., v_k$. הבעיה של הקודקוד v הוא שהוא לא יודע כיצד שכניו פועלים. למשל: אם קודקוד v היה יודע ששכניו אינם שכנים אחד של השני - אזי קודקוד v היה רוצה לומר להם: תצבעו כולכם באותו הצבע, ואני אצבע את עצמי בצבע שונה משלכם.

כרעיון בסיסי מאוד - אני יכול להחליט שהקודקוד v ישלח הודעה $(v_1, v_2, ..., v_k)$ לכל שכניו. כמו כן: כל שכן ישלח הודעה לכל השכנים שלו עצמו. ואז - אני אקבל את ההודעות של כל שכניי, ואוכל להסתכל האם באחת ההודעות אני מזהה קודקוד שכבר יש לי. כלומר: האם השכנים שלי הם גם שכנים אחד של השני. נשים לב שקצת רימינו - שכן טענו כי לקודקודים אין id, אז איך נוכל לשלוח הודעה שכזו? נדבר על תהליך בחירת ID שקורה במקביל עבור כל הקודקודים.

בחירת ID:

1. כל קודקוד בוחר id בין המספרים $[1, 2, ..., n^{c+2}]$ עבור c קבוע.

2. שלח ID לכל השכנים. (שכל אחד ידע את ה$Id's$ של השכנים שלו)

נרצה לבדוק מה הסיכוי שישנם שני קודקודים עם אותו id.

$$Pr[sameID] = 1 \times \frac{1}{n^{c+2}} = \frac{1}{n^{c+2}}$$

נסמן בA_{uv} את המאורע שבו u וv בחרו את אותו Id. מכאן $Pr[A_{uv}] = \frac{1}{n^{c+2}}$. נסתכל על המאורע הבא, שמשמעותו שאף אחד לא בחר את אותו id.

$$Pr[\overline{\bigcup_{u,v \in V} A_{uv}}] = 1 - Pr[\bigcup_{u,v \in V} A_{uv}]$$

נראה כי

$$Pr[\bigcup_{u,v \in V} A_{uv}] \leq \sum_{u,v \in V} Pr[A_{uv}] \leq n^2 \times \frac{1}{n^{c+2}} = \frac{1}{n^c}$$

וקיבלנו כי

$$Pr[\overline{\bigcup_{u,v \in V} A_{uv}}] = 1 - Pr[\bigcup_{u,v \in V} A_{uv}] = 1 - \frac{1}{n^c}$$

ולכן הסיכוי לטעות קטן פולינומית, והסיכוי להצלחה גדול מאוד פולינומית.

סה"כ קיבלנו כי באמצעות טכניקה רנדומית פשוטה, יצרנו לכל קודקוד ID. מכאן: יש משמעות לשליחת הודעה לכל השכנים של רשימת השכנים.

נראה כי ישנו קושי - בהחלט יכול להיות שאפילו שלשכנים שלי אין קשתות ביניהם, עדיין לא ניתן לצבוע את כל השכנים באותו צבע. הרבה דוגמאות יכולות להעיד על כך: הרעיון הזה פשוט מדי, וחושב לוקלית מקומית אך הגרף הרבה יותר גדול מזה. בחירה מקומית יכולה להשפיע על הגרף כולו באופן שלא ציפינו. יתרה מזאת - כל מעגל אי זוגי דורש לפחות 3 צבעים. ושיטה זו במעגל תניב 2 צבעים. וכן: יש קושי להבין האם קודקוד נמצא במעגל אי זוגי. שכן יתכן כי גודל המעגל האי זוגי מאוד מאוד גדול ויקח המון זמן והודעות להבין שאנחנו נמצאים בכזה.

אנחנו נרצה לצבוע את הגרף באמצעות 2Δ צבעים. (ראינו כבר כי ניתן להשתמש ב$\Delta + 1$ צבעים, אך המטרה באלגוריתם שנראה בהרצאה הוא לא משהו חדשני - אלא להבין את המודל המבוזר המקומי)

10.5 אלגוריתם צביעה

כעת נתאר את האלגוריתם שיצבע את הגרף באמצעות 2Δ צבעים.

1. נבחר צבע באקראי מבין $[1, ..., 2\Delta]$. נשים לב - ישנה כאן הנחה סמויה: כל קודקוד $v \in V$ מכיר את Δ.
2. נשווה עם השכנים. ישנם שני מקרים -

א. אף שכן לא בחר את הצבע שאנחנו בחרנו: במקרה זה, אנחנו הצלחנו. נחליט שזה הצבע שלנו.

ב. אחרת, קיים לפחות שכן אחד שבחר את הצבע שאנחנו בחרנו, במקרה זה אנחנו ננסה לבחור שוב את הצבע. (נחזור ל1).

נראה את האלגוריתם עבור קודקוד יחיד ־

Color (Δ):
while(true):
-pick random color from $[1, ..., 2\Delta] \rightarrow c$
- send c to neighbors
- recieve colors of neighbors
-if there is no neighbor with color c so return.

נשים לב שבבדיקה אנחנו בודקים את כל השכנים של הקודקוד, ולא רק את אלו ש"אקטיביים" כרגע. כלומר ־ בודקים גם את אלו שסיימנו לצבוע את הקודקוד שלהם.

10.5.1 נכונות האלגוריתם

מה הסיכוי שהבדיקה בשורה 5 תצליח? כלומר: שאין שכן שגם בחר את הצבע c.
לכל קודקוד יש דרגה $d \leq \Delta$ ולכן לא משנה איזה צבעים השכנים בחרו, תמיד יש לפחות $2\Delta - d$ צבעים פנויים. שורה 5 בהכרח מצליחה אם הצבע c שנבחר הוא אחד מהצבעים הפנויים. נסמן בx את מס' הצבעים הפנויים. בהכרח $x \geq 2\Delta - d$.

$$Pr[SuccessLine5] = \frac{x}{2\Delta} \geq \frac{2\Delta - d}{2\Delta} = 1 - \frac{d}{2\Delta} \geq_{\Delta \geq d} 1 - \frac{\Delta}{2\Delta} = \frac{1}{2}$$

כלומר, הסיכוי להצליח בשורה 5 הוא גדול שווה מ$\frac{1}{2}$.
כעת, נרצה לחסום את מס' הסיבובים שהאלגוריתם עולה עד לצביעה חוקית של כל הגרף.

נסמן בV_i את קבוצת הקודקודים שעדיין לא נצבעו אחרי i איטרציות של האלגוריתם. בהכרח לפי הגדרה $V_0 = V$. נראה כי

$$\forall u \in V : Pr[u \in V_i] \leq \frac{1}{2^i}$$

כיוון שהסיכוי שקודקוד יהיה בקבוצה, אזי בכל האיטרציות הקודמת היה כשלון בשורה 5. שכן אנו יודעים שכשלון בשורה 5 קטן שווה מסיכוי חצי.

$$E[|V_i|] = \sum_{u \in V} Pr[u \in V_i] \leq \frac{n}{2^i}$$

לכן, אחרי $log(n) + 1$ איטרציות נקבל כי

$$E[|V_{logn+1}|] \leq \frac{n}{2^{logn+1}} = \frac{1}{2}$$

כלומר מס' הקודקודים בתוחלת לאחר $logn + 1$ איטרציות הוא חצי. נראה כי ישנה טעות נפוצה בשלב זה: להגיד מכאן נובע כי מס' האיטרציות עד שכל הקודקוד צבועים הוא לכל היותר $logn + 1$". נראה כי זה שהתוחלת היא לכל היותר חצי (לא יתכן הרי חצי איבר), לא אומר שלאחר $logn + 1$

איטרציות הקבוצה תהיה ריקה. בהתחשב בתובנה הזו: כיצד נמשיך מכאן? נראה כי תמיד יתקיים לפי האלגוריתם וההסתברות שראינו קודם כי ־

$$E[|V_i|] \leq \frac{1}{2}|V_{i-1}|$$

נזכר באי שוויון מרקוב. שאומר את הטענה הבאה: $Pr[X \geq t] \leq \frac{E[X]}{t}$. מכאן, נרצה לחשב מה הסיכוי שגודל של קבוצה יהיה גדול שווה מ$\frac{3}{4}|V_{i-1}|$.

$$Pr[|V_i| \geq \frac{3}{4}|V_{i-1}] \leq \frac{E[V_i]}{\frac{3}{4}|V_{i-1}|} \leq \frac{\frac{1}{2}|V_{i-1}|}{\frac{3}{4}|V_{i-1}|} = \frac{2}{3}$$

מכאן שסיכוי זה הוא לכל היותר $\frac{2}{3}$. ומכאן:

$$Pr[|V_i| < \frac{3}{4}|V_{i-1}] = 1 - Pr[|V_i| \geq \frac{3}{4}|V_{i-1}] \geq 1 - \frac{2}{3} = \frac{1}{3}$$

כלומר, הסיכוי שבאיטרציה הi הצלחנו לצבוע לפחות רבע מהקודקודים שלא היו צבועים קודם באיטרציה ה$i-1$ היא לפחות סיכוי של $\frac{1}{3}$. המספר שליש הוא קבוע, וזה מה שחשוב כאן. נסמן $p = \frac{1}{3}$.

נאמר שאיטרציה היא "טובה" אם היא הצליחה לצבוע לפחות רבע מהקודקודים שלא היו צבועים בתחילת האיטרציה. ורעה=לא טובה.

בהינתן שהיו k איטרציות טובות, נשארנו עם $\geq (\frac{3}{4})^k \times n$ מהקודקודים.

נרצה כי:

$$(\frac{3}{4})^k \times n < 1 \Longrightarrow n < (\frac{4}{3})^k \Longrightarrow log_{\frac{4}{3}}(n) < k$$

כלומר אם $k = log_{\frac{4}{3}}(n) + 1$ אזי הצלחנו לצבוע את כל הגרף. כלומר אם נקח k כנ"ל כמס' האיטרציות הטובות אנחנו סיימנו.

קודם לכן ראינו כי $Pr[|V_i| < \frac{3}{4}|V_{i-1}] \geq \frac{1}{3}$, כלומר הסיכוי שתהיה איטרציה טובה הוא לפחות שליש. לכן, בתוחלת (התפלגות גאומטרית) עם $p \geq \frac{1}{3}$ מס' האיטרציות שיש ברצף עד שמקבלים איטרציה טובה ראשונה הוא $\frac{1}{p} \leq 3$.

ומכאן: מס' האיטרציות הטובות הינו $log_{\frac{4}{3}}(n)+1$, כפול 3 כמס' האיטרציות שקורות עד שמקבלים את האיטרציה הטובה הבאה. סה"כ נקבל כי

$$3log_{\frac{4}{3}}(n) + 3$$

הוא תוחלת מס' האיטרציות עד שאין קודקודים לא צבועים. ואכן, מס' האיטרציות שהאלגוריתם יעשה תהיה $O(logn)$.

כעת, נרצה **גם** לבחון את הנכונות במקרה הגרוע ביותר. ראינו כי

$$E[|V_i|] \leq \frac{n}{2^i}$$

נרצה לבדוק מה הסיכוי שלא סיימנו עבור $i = clogn$ עבור $c > 1$ קבוע.

$$Pr[|V_{clogn}| < 1] = 1 - Pr[|V_{clogn}| \geq 1]$$

$$Pr[|V_{clogn}| \geq 1]_{markov} \leq \frac{E[|V_{clogn}|]}{1} \leq \frac{n}{2^{clogn}} = \frac{1}{n^c}$$

ונקבל כי

$$Pr[|V_{clogn}| < 1] = 1 - Pr[|V_{clogn}| \geq 1] \geq 1 - \frac{1}{n^c}$$

כלומר, הסיכוי שלא סיימנו קטן פולינומית. ולכן הסיכוי להצלחה יחסית טוב.

11 הרצאה 11: חסמי $Chernoff$

12 הרצאה 12: אלגוריתמים רנדומיים בגרפים

13 סיכום אלגוריתמים שראינו בקורס + זמני ריצה ("קופסאות שחורות")

אלגוריתם FFT: בהינתן שני פולינומים, A וB מדרגה חסומה n, מחשב את פולינום המכפלה $C = A \times B$ בזמן $O(nlogn)$.

אלגוריתם מרחק האמינג: בהינתן מחרוזת P באורך n ותבנית בגודל m מחזיר את מרחק ההאמינג של המחרוזת מהתבנית עבור כל היסט שלה. זמן הריצה $O(nlogm)$.

אלגוריתם למציאת MST:

1. האלגוריתם של פרים ־ משתמש בתור קדימויות ועלותו באמצעות עץ פיבונאצ'י הינה $O(|V|log|V| + |E|)$
2. האלגוריתם של קרוסקל ־ משתמש ביוניון פיינד ועלותו $O(|E|(\alpha|V|) + sort(|E|))$

אלגוריתם $Shorts\ Path$:

אלגוריתם BFS: מטפל ב$SSSP$ במקרה הלא ממושקל. סיבוכיות זמן הריצה שלו $O(|E|+|V|)$

אלגוריתם DFS: סריקה לעומק של הגרף G, בזמן $O(|E| + |V|)$. מאפשר למשל בדיקה של האם קיים בגרף מעגל (לאחר הרצת האלגוריתם נבדוק האם יש קשתות אחוריות, אם כן אז יש מעגל).

אלגוריתם למציאת מעגל בגרף: מריצים DFS ובודקים במהלך ההרצה האם יש קשתות אחוריות. אם מצאנו כזו: דווח על מעגל. אחרת, אין. זמן ריצה $O(|E| + |V|)$

אלגוריתם למציאת רכיבי הקשירות חוזקה: בגרף מכוון, רוצים למצוא את רכיבי הקשירות חוזקה. בזמן $O(|E| + |V|)$

אלגוריתם מיון טופולוגי: לכל קשת (v_i, v_j) יתקיים בסוף המיון $i < j$. זמן הריצה $O(|E|+|V|)$

אלגוריתם גרף דו צדדי: בהינתן גרף $G = (V, E)$ דו"צ, מחזיר את הקבוצות R, L. (מחזיר אחת כאלו, יתכנו כמובן כמה). סיבוכיות זמן ריצה של $O(|E| + |V|)$

אלגוריתם למציאת $SSSP$ בDAG: בהינתן גרף מכוון חסר מעגלים, מחשב $SSSP$ באמצעות תכנות דינמי בסיבוכיות זמן ריצה $O(|E| + |V|)$.

האלגוריתם של בלמן פורד: מטפל ב$SSSP$ במקרה הממושקל. מניח שיתכנו משקלים שליליים אך לא יתכנו מעגלים שליליים. סיבוכיות זמן הריצה שלו $O(|E| \times |V|)$

האלגוריתם של דייקסטרה: מטפל ב$SSSP$ במקרה הממושקל, אך מניח שלא יתכנו משקלים שליליים. סיבוכיות זמן הריצה שלו $O(|V|log|V|+|E|)$ עם פיבונאצ'י ועם ערימה בינארית $O(|V|log(|V|)+|E|log(|V|))$.

האלגוריתם של פלויד וורשל: מטפל ב$APSP$ במקרה הממושקל, מניח שיתכנו משקלים שליליים אך לא יתכנו מעגלים שליליים. עובד בתכנות דינמי עם נוסחת נסיגה פשוטה למדי, ומחזיר את $D^{(n)}$ מטריצת המרחקים וπ מטריצת הקודמים. רץ בסיבוכיות זמן ריצה $O(|V|^3)$.

האלגוריתם של ג'ונסון: מטפל ב$APSP$ במקרה הממושקל, מניח שיתכנו משקלים שליליים אך לא יתכנו מעגלים שליליים. "מרמה" ומשתמש באלגוריתמים קיימים. תחילה מריץ בלמן פורד. אם מצא מעגל שלילי מפסיק מיד. לאחר מכן משתמש ב$\delta(u,v)$ שחישב בלמן פורד למען הגדרת פונקציית משקל על הקודקודים, ובאמצעותה פונקציית משקל חדשה על הקשתות. פונקציית המשקל המוגדרת החדשה $\hat{w}$ היא אי שלילית ולכן הוא מריץ $|V|$ פעמים דייקסטרה. סה"כ זמן הריצה $O(|V||E|+|V|^2log|V|)$.

אלגוריתם לחישוב הסגור הטרנטזיבי של גרף G^*: משתמשים ב$APSP$ (של פלויד וורשל למשל) ומקבלים זמן ריצה של $O(|V|^3)$.

אלגוריתם לחישוב סגור טרנזטיבי של גרף מכוון G^* באמצעות כפל מטריצות: משתמשים בהפרד ומשול ומקבלים זמן ריצה של $O(|V|^\omega)$.

אלגוריתם כפל מהיר (FMM): מחשב את תוצאת המכפלה של שתי מטריצות A,B בעלות זמן $O(n^\omega)$ באשר $2 \leq \omega \leq 2.37287$

האלגוריתם של סיידל: מחשב את $APSP$ במקרה הלא ממושקל באשר הגרף G לא מכוון, מחזיר מטריצה $\Delta_{i,j} = \delta(v_i, v_j)$ בסיבוכיות זמן $O(|V|^\omega log|V|)$.

רשתות זרימה:

השיטה של פורד־פלקרסון: מוצא את הזרימה המקסימלית ברשת זרימה $G = (V,E)$ נתונה. שקול למציאת חתך (s,t) מינימלי. מניח שכל הקיבולות הם ערכים שלמים. עלות זמן הריצה היא $O(|f^*| \times |E|)$ באשר $|f^*|$ הוא גודל הזרימה המקסימלי.

האלגוריתם של אדמונס קארפ: מוצא זרימה מקסימלית בסיבוכיות $O(|V| \times |E|^2)$

האלגוריתם של $Dinic$: מוצא זרימה מקסימלית בסיבוכיות $O(|V|^2 \times |E|)$

אלגוריתמים רנדומיים:

אלגוריתמי מונטה קרלו:

א. אלגוריתם לוידוא כפל מטריצות:

1. אלגוריתם בסיסי ־ מסתמך על העובדה שאם $C = AB$ אזי $Cv = ABv$ ובודק את תוצאות הוקטור. סיבוכיות זמן ריצה $O(n^2)$ וסיכוי לטעות $\geq \frac{1}{2}$.

2. האלגוריתם השני ־ מריץ את האלגוריתם הראשון $k = \alpha log(n)$ פעמים, סיבוכיות זמן הריצה הינה $O(n^2 logn)$ והסיכוי לטעות הוא לכל היותר $\frac{1}{n^\alpha}$ באשר $\alpha \geq 2$ קבוע.

אלגוריתמי לאס וגאס:

א. אלגוריתם מיון מהיר ־ הוכחנו בקורס כי מס' ההשואות של אלגוריתם מיון מהיר קלאסי בתוחלת הינו $O(nlogn)$.

ב. אלגוריתם מיון מהיר פרנואידי ־ אלגוריתם מיון מהיר שמוודא בכל שלב שכל צד בחר לפחות $\frac{n}{4}$ איברים. הוכחנו כי זמן הריצה שלו בתוחלת הינו $O(nlogn)$.

מיון דלי: אלגוריתם דטרמיניסטי שמקבל קלט רנדומי, מחלק את הקלט לבאקטים שונים ובהתאם לכך מבצע בכל באקט מיון הכנסה ולבסוף משרשר את הבאקטים. תוחלת זמן הריצה של המיון הינו $O(n)$.

בעיית הצביעה:

14 טריקים ושטיקים

יתמלאו כאן טריקים ושטיקים ששמתי לב אליהם, לא סופי כמובן:

א. **עפ"מ:** אם רוצים למצוא MST בעל משקל מקסימום, כל שצריך לעשות הוא להגדיר $w'(e) = -w(e)$, וישירות מקבלים MST מקסימלי לפי w'.

ב. **כללי בגרפים:**

1. להוסיף קודקוד (או כמה) חדשים לגרף. טריק שימושי הוא להוסיף קודקוד s עם פונקציית משקל 0 לכל הקשתות מs. כמו שג'ונסון עושה.
2. לשכפל את הגרף - לצורך כלשהו. גם להסתכל על גרף תלת שכבתי, יכול לתרום לנו.
3. להגדיר G^T.
4. לשים לב - אם קשת (u, v) נמצאת על מסלול קצר ביותר מs אל t אזי בהכרח מתקיים:

i) לפי הגרסה הלא ממושקלת - $\delta(s,t) = \delta(s,u) + 1 + \delta(v,t)$

ii) לפי הגרסה הממושקלת - $\delta(s,t) = \delta(s,u) + w(u,v) + \delta(v,t)$

5. קשת אחורית אמ"מ יש מעגל. סופר חשוב. כיצד מוצאים קשת אחורית? אם במהלך מציאת (u, v) גילנו את v באשר אנחנו עדיין מבקרים בu.

ג. **טריקים טובים ממבנ"ת:** העלאה בחזקה $O(logn)$, מיון בסיס $O(n)$. עץ AVL לחיפוש ב$O(logn)$.

ד.