אלגוריתמים 1: הרצאה 8

23 בדצמבר 2025

גיא יער־און

0.1 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, $2.37287\leq\omega$ כי כיום ידוע אלגוריתם שפותר בזמן זה, וכן נקבל סה"כ כי:

$$2\leq\omega\leq 2.371339$$

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

0.1.1 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. ברור לנו כי אלגוריתם קומבינטורי הוא אלגוריתם שלא ישתמש בהחסרות, כלומר בחיסור: רוצים להשתמש בכלים יותר "ישירים" - החסרה היא פעולה שמבטלת משהו שביצענו יותר מדי.

הנאיבי שעלותו $O(n^3)$ מאוד קל למימוש, וכן המהיר ביותר (הקומבינטורי) שקיים היום בסיבוכיות $O(\frac{n^3}{log^4n})$ של Yu.
נרצה להתמקד בBMM שאולי (!!!) יותר קלה מFMM אך בוודאי לא יותר קשה.

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

$$n^{2.9999...} <<< \frac{n^3}{log^4n}$$

אנחנו נתעלם מפקטורים של $log(n)$ וכו', נתעניין רק באקספוננט. מכאן נגדיר סימון חדש $\tilde{O}()$ שיציין התעלמות זאת. לדוגמה:

$$O(n^2logn) = \tilde{O}(n^2)$$

$$O(\frac{n^3}{log^4n}) = \tilde{O}(n^3)$$

בשנת 2024 הוכיחו כי קיים אלגוריתם קומבינטורי שזמן הריצה שלו הוא $O(\frac{n^3}{2^{\sqrt[7]{logn}}})$.

השערת הBMM: לכל קבוע $\varepsilon > 0$ לא קיים אלגוריתם קומבינטורי שזמן הריצה שלו הוא $O(n^{3-\varepsilon})$ לפתרון BMM.
נבחין, כי דרך אחרת לכתוב את ההשערה היא שכל אלגוריתם קומבינטורי לBMM דורש $n^{3-o(1)}$ זמן ($o(1)$ הקטן מחביא את הפקטורים של הlog)), נבחין כי $n^{3-o(1)} \notin n^{2.99999999}$.
בעזרת ההשערה הזו, ניתן להוכיח חסמים תחתונים לאלגוריתמים "קומבינטוריים" עבור כל מיני בעיות (כתלות בנכונות ההשערה).

מהי השערה? השערה יוצאת לפועל לאחר שניסו הרבה מאוד זמן לפתור בעיה ולא הצליחו. האם בהינתן השערה מסוימת, זה אומר שהיא בהכרח נכונה? לא. אם היה הוכחה להשערה - היא לא הייתה השערה. ולכן, אם בהמשך יוכיחו אותם הם כבר לא יהיו השערה. למשל, חוקי ניוטון הם רק השערה (אין הוכחה). אם כן: **ככל הנראה** צריך להשתמש בשיטות אלגוריתמיות חדשות (שיתכן שעוד לא המצאנו) על מנת להפריך את ההשערה. זה פותח עבורנו חקר של הסתכלות על משפחות מיוחדות של קלטים - זה רלוונטי עבור פרקטיקה: אם למשל בכפל מטריצות נתון כי $A = \{a_{ij} = 1\}$ (כל המטריצה אחדות) אזי הבעיה הייתה הופכת לקלה הרבה יותר - ב$O(n^2)$. זו משפחה מיוחדת של קלטים.
על סמך השערות ניתן להוכיח חסמים תחתונים מותנים. ובאנגלית: $conditional\ lower\ bounds$.
למשל, תחת ההנחה $P \neq NP$ אפשר להוכיח כי בעיות רבות אינן ניתנות לפתרון בזמן פולינומיאלי: אלא בהכרח בזמן אקספוננציאלי.
במדעי המחשב אנחנו די גרועים בהוכחת חסמים **תחתונים** לבעיות. אם כן, הזמן היחיד שכן הצלחנו להוכיח הוא חסם תחתון למיון מבוסס השוואות. אך, בהינתן השערה מסוימת נוכל כן להניח חסמים תחתונים כלשהם.

0.2 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.

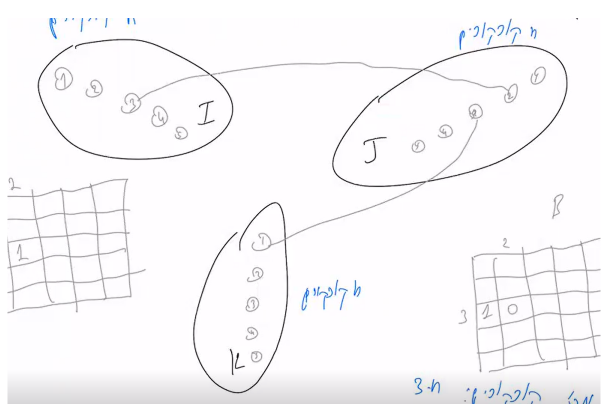
1. האם בG יש משולש?
2. אם בG יש משולש, מצא אחד שכזה.
3. דווח על כל המשולשים שיש בG.

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: $\exists_{1\le i\le n} M^3[i][i]=1$ (באלכסון) $\iff$ 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** 2)**:** 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: $C=A\times B$ באשר 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. כלומר $(n^2 \leq |E| \leq 3n^2)$

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 $\iff$ $C[i][k] = 1$. (C היא מטריצת הפלט של הכפל הבוליאני).

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 לt 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? n^2 לכל היותר 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

$$(t^3+n^2)\times\tilde{O}((3\frac{n}{t})^{3-\varepsilon})=$$

$$3^{3-\varepsilon}(t^3(\frac{n}{t})^{3-\varepsilon}+n^2(\frac{n}{t})^{3-\varepsilon})=$$

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:

$$n^2\times n^{\frac{3-\varepsilon}{3}}+(n^{\frac{2}{3}})^3n^{\frac{3-\varepsilon}{3}}=\tilde{O}(n^{3-\frac{\varepsilon}{3}})$$

לסיכום, יצרנו אלגוריתם שיודע להחזיר את מטריצת הכפל בזמן הריצה של $\tilde{O}(n^{\tilde{3-\frac{\varepsilon}{3}}})$ באשר השתמשנו באלגוריתם שמזהה משולשים.

הערה נוספת. נבחין כי באמצעות (והאינטואיציה) לחלוקה ל־t היא להקטין את גודל הקלט של הגרף שמריצים עליו את האלגוריתם למציאת משולשים בכל שלב ־ ולכן רצינו להקטין את גודל הגרף הזה בכל שלב. אם כן, אנחנו בכל שלב בוחרים חלק $(x,y,z)\in I\times J\times K$ ובכל אחד מהם יש t חלקים ולכן סה"כ t^3.

נבחין, כי הנחנו שהאלגוריתם גם מזהה משולש וגם מחזיר אותו. מדוע? נתבונן בלמה הבאה:

למה 3. תהי $T(X) = \Omega(X)$ מונוטונית עולה (לא יורדת). אם קיים אלגוריתם TD (זיהוי משולשים) שרץ בזמן $T(|V|)$ אזי קיים אלגוריתם שמוצא את המשולש בזמן $O(T(|V|))$.

הוכחה. הערה ־ ההוכחה לא תהיה פורמלית. (זו ההוכחה שניתנה בכיתה).

הרעיון יהיה הרעיון הבא: נקח את הגרף ונחלק אותו ל4 חלקים בגודל $\frac{n}{4}$: V_1, V_2, V_3, V_4. נבחין כי אם ישנו משולש בגרף ־ הוא לא יכול להיות ב4 החלקים והוא נמצא כל פעם ב3 מתוכם. לכן, אנחנו בכל שלב נוריד חלק אחד מן V_i ונבדוק האם קיים משולש באחד מ4 החלקים: אם נמצא אחד כזה ־ באחת מן 4 האיטרציות נמצא משולש, נפטר בכל שלב מרבע מהגרף: ולכן נריץ שוב ברקורסיה. **להלן האלגוריתם:**

$Algo$**:**

לכל V_i : הרץ TD על $V \setminus V_i$. אם יש משולש ־ המשך ברקורסיה והחזר את התשובה.

זמן הריצה: $\hat{T}(|V|) = 4T(\frac{3}{4}|V|) + \hat{T}(\frac{3}{4}|V|) = \hat{T}(\frac{3}{4}|V|) + O(T|V|)$ ולפי משפט האב נקבל כי $\hat{T}(|V|) = O(T(|V|))$.

נבחין כעת, כי בסוף נגיע ל4 חלקים בגודל 1 (תנאי העצירה) ונמצא את המשולש הספציפי. סה"כ נשים לב שבאותו זמן של האם קיים משולש ניתן למצוא אותו גם, כנדרש.

0.3 TD (מציאת משולשים) בגרפים דלילים (קל כבד)

ראינו כי $O(|V|^{\omega})$ בגרף כללי. נראה כעת אלגוריתם כגרף עם מעט קשות, כתלות ב$|E|$. זה לא אלגוריתם קומבינטורי אלא רגיל.

נגדיר פרמטר בשם τ (בהמשך נקבע אותו ממש).

הגדרה: נאמר כי קודקוד u הוא קודקוד כבד אם $deg(u) \geq \tau$. אחרת ־ נאמר כי u הוא קודקוד קל.

טענה: מס' הקודקודים הכבדים הוא לכל היותר $\frac{2|E|}{\tau}$.

הוכחה: נניח בשלילה כי מס' הקודקודים הכבדים הינו $> \frac{2|E|}{\tau}$ ונקבל כי סכום דרגותיהם הינו $> \frac{2|E|}{\tau} \times \tau = 2|E|$ בסתירה ללמת לחיצת הידיים.

נראה אלגוריתם שמאזן את שני המקרים ־ קודקודים קלים וכבדים. כלומר, נראה אלגוריתם לקלים ולכבדים והאלגוריתם בסוף יאזן את שניהם.

עבור u קודקוד קל (וסה"כ עבור הקלים): נעבור על כל זוגות השכנים $x, y \in N(u)$: לכל היותר $deg^2(u)$ זוגות שכאלו. וכן, אם $(x, y) \in E$ נחזיר כן: ישנו משולש. (שכן, x שכן של u וגם y שכן של u וישנה קשת (x, y) וסה"כ משולש x, y, u, x). זמן הריצה של החלק הזה:

$$O(\sum_{u} deg^2(u))$$

עבור כל הכבדים יחד: יהי $\hat{G}$ הגרף שמושרה ע"י כל הקודקודים הכבדים. מתקיים $|\hat{V}| \leq \frac{2|E|}{\tau}$. נשתמש ב$FMM$ על $\hat{G}$ (נעלה בשלישית ונעזר בטענה שבאלכסון יש אחד אמ"מ משולש). נקבל זמן ריצה: $O((\frac{|E|}{\tau})^{\omega})$.

סה"כ, נריץ את שני האלגוריתמים יחד. נבחין כי ישנו שני סוגים של משולשים: משולש שכל הקודקודים בו כבדים. נגלה אותו בשלב של הFMM: אחרת, יש לפחות קודקוד אחד קל במשולש ונגלה אותו בשלב השני שנריץ עבור הקלים.

סה"כ, זמן הריצה יהיה:

$$O(\sum_u deg^2(u)) + O((\frac{|E|}{\tau})^\omega)$$

$$O(\sum_u deg^2(u)) \leq O(\sum_u \tau \times deg(u)) = \tau \sum_u deg(u) = 2\tau|E|$$

ולכן זמן הריצה:

$$2\tau|E| + (\frac{|E|}{\tau})^\omega$$

נשווה כעת את שני הביטויים:

$$\tau|E| = (\frac{|E|}{\tau})^\omega \Longrightarrow \tau^{\omega+1} = |E|^{\omega-1} \Longrightarrow \tau = |E|^{\frac{\omega-1}{\omega+1}}$$

מכאן, שסה"כ זמן ריצת האלגוריתם:

$$(\frac{|E|}{\tau})^\omega = (\frac{|E|}{|E|^{\frac{\omega-1}{\omega+1}}})^\omega = (|E|^{1-\frac{\omega-1}{\omega+1}})^\omega = O(|E|^{\frac{2\omega}{\omega+1}})$$

הבהרה. ניתן לבצע את האלגוריתם בכל גרף, יתכן שהוא לא כדאי בגרפים שאינם דלילים. נבחין כי אם יום אחד יגלו כי אכן $\omega = 2$ אזי האלגוריתם רץ בזמן ריצה $O(|E|^{\frac{4}{3}})$ - מדהים!