מבני נתונים 2

הרצאה 3

האלגוריתם של דייקסטרה

<u>קלט</u>: גרף מכוון עם משקולות w אי-שליליים על קשתות וצומת מקור s. נסמן ב-n את מספר הצמתים בגרף.

<u>פלט</u>: מרחקי כל הצמתים מ-s.

<u>האלגוריתם</u>:

בכל רגע נחזיק קבוצת צמתים S שמרחקם מ-s ידוע ולא ישתנה בהמשך האלגוריתם. בנוסף נחזיק חזית: קבוצת כל הצמתים השכנים לצמתי S.

<u>אתחול</u>: נכניס את s לחזית

<u>אישרציה</u>: כל עוד החזית אינה ריקה, בצע:

1. מצא צומת בעל מרחק מינימלי מ-s בחזית.
2. הוצא את הצומת מהחזית והוסף אותו ל-S.
3. עדכן את השכנים של הצומת (או הוסף אותם לחזית, אם אינם בחזית כרגע).

<u>מימוש נאיבי</u>:

נחזיק מערך של כל צמתי החזית.

מציאת צומת בעל מרחק מינימלי מ-s בחזית ייקח זמן $O(n)$.

יהי v הצומת בעל מרחק מינימלי בחזית. עדכון השכנים של v ייקח זמן $O(\deg(v))$.

סה"כ זמן ביצוע האלגוריתם: $O(n) + \sum_v O(\deg(v)) = O(n^2 + 2|E|)$.

<u>מימוש יעיל</u>:

נחזיק את צמתי החזית בערמת פיבונצ'י, ובנוסף נחזיק מערך של מצביעים מצמתי הגרף לצמתי הערמה (עבור כל צומת, הכניסה המתאימה במערך תצביע על הצומת המתאים בערמה).

.<u>עדכון השכנים</u>:

יהי $dist(s,v)$ - המרחק (נכון לעכשיו) של v מ-s. לכל שכן v_i של v, אם $dist(s,v)+w(v,v_i)<dist(s,v_i)$, נבצע: $decrease(v_i,\Delta)$, כאשר $\Delta = dist(is,v)+w(v,v_i)-dist(s,v_i)$.

<u>סיבוכיות הזמן</u>:

אתחול: O(n) (מסתכלים על השכנים של s).

ביצוע del_min: O($\log n$) לפעולה בודדת. מבצעים n פעולות כאלה. לכן סה"כ: $O(n\log n)$

פעולת insert: O(1) לפעולה בודדת. מבצעים n פעולות כאלה. לכן, סה"כ O(n).

פעולת decrease: O(1) לפעולה בודדת. מבצעים $|E|$ פעולות כאלה. לכן, סה"כ: $O(|E|)$.

קיבלנו סיבוכיות הזמן: $O(n\log n+|E|)$.

<u>עץ פורש מינימום (MST)</u>

<u>הבעיה</u>: נתון גרף קשיר לא מכוון עם משקולות לקשתות. מצא עץ פורש במשקל מינימום.

<u>האלגוריתם של Prim</u>:

יהי s צומת כלשהו. נגדל עץ ב-s. בכל שלב, יש בידנו עץ חלקי T וחזית של צמתים ששכנים לצמתי T אך אינם שייכים ל-T.

<u>אתחול</u>: הכנס את s ל-T ואת שכניו לחזית.

<u>אינדוקציה</u>: כל עוד החזית אינה ריקה, בצע:

1. יהי $c(v)$ המשקל המינימלי של קשת בין צומת בחזית וצומת ב-T, ו-v הוא צומת בחזית בו פוגעת הקשת.
2. הוצא את v מהחזית והוסף אותו ל-T.
3. עדכן את כל שכני v, ואם יש צורך, הוסף אותם לחזית.

<u>מימוש נאיבי</u>: כמו קודם, נחזיק את החזית במערך. ע"י ניתוח זהה לזה של האלגוריתם של דייקסטרה, נקבל סיבוכיות הזמן: $O(n^2+|E|)$.

<u>מימוש בעזרת ערמת פיבונאצ'י</u>: נשמור את החזית בערמת פיבונצ'י.

שוב, בעזרת אותו ניתוח כמו באלגוריתם של דייקסטרה, נקבל סיבוכיות הזמן: $O(n\log n+|E|)$.

אלגוריתם משופר

נניח, ש- $|E| < n\log n$. (במקרים אחרים האלגוריתם הקודם שלקח זמן $O(n\log n + |E|)$ היה אופטימלי).

האלגוריתם יעבוד בפאזות. בפאזה i נקבע פרמטר k_i, ונדאג שגודל כל ערמה בפאזה זאת לא יעלה על k_i.

בכל פאזה נגדל יער של עצים זרים, כך שכל צומת שייך לאחד העצים בדיוק. נגדל כל עץ עד שיהיו לו k_i שכנים, או שנמזג אותו עם עץ אחר שגידלנו לפני כן.

לעץ הנוכחי נחזיק את כל שכניו בערמת פיבונצ'י.

צעד:

כל עוד גודל הערמה לא עולה על k_i, בצע:

1. הוצא מהערמה את הצומת בעל משקל מינימלי, v.
2. אם v שייך לעץ אחר, הוסף את הקשת שמובילה ל-v והפסק לגדל את העץ.
3. אחרת, עדכן את הערמה לכל שכני v.

בסיום כל פאזה, נכווץ כל עץ שגידלנו לצומת בודד.

<u>המטרה</u>: לכווץ את כל העצים בזמן $O(|E|)$.

יש לגלות, לאיזה עץ שייך כל צומת. נבצע DFS מכל השורשים של העצים שלא מוזגו. לכל צומת נסמן לאיזה עץ הוא שייך.

נותר: להיפטר מחוגים עצמיים וקשתות מקבילות.

1. חוגים עצמיים: נעבור על כל הקשתות. לכל קשת (u,v), אם לשני הקצוות שלה יש אותו סימון (הם שייכים לאותו עץ), הקשת הנה חוג עצמי בגרף המכווץ. נמחק אותה.
2. קשתות מקבילות:

לכל קשת $e = (u,v)$, נסמן ב-$l(e)$ את הסימון המינימלי מבין סימוני u, v, וב-$h(e)$ את הסימון המקסימלי מביניהם. נמיין את כל הקשתות לקסיקוגרפית ב-Bucket Sort קודם לפי $h(e)$, ולאחר מכן לפי $l(e)$. נשים לב כי ה-Bucket Sort הוא יציב ולכן לא הורס סדר קודם. קשתות שמחברות בין אותו זוג עצים תהיינה סמוכות ברשימה הממוינת. כך נוכל לזהות אילו קשתות מחברות בין אותו זוג עצים, כלומר הינן מקבילות בגרף המכווץ. מבין הקשתות המקבילות נשאיר רק קשת עם משקל מינימלי ונמחק את שאר הקשתות המקבילות לה מהגרף המכווץ.

<u>סיבוכיות הזמן</u>:

זמן כיווץ הגרף - $O(|E|)$.

גידול העצים:

יהי n_i מס' הצמתים בפאזה ה-i.

נבצע :

- $O(n_i)$ פעולות del_min, וכל אחת תיקח זמן משוערך $O(\log k_i)$
- $O(|E|)$ פעולות insert, (כל קשת תורמת לכל היותר שתי פעולת insert) וכל פעולה תיקח זמן משוערך $O(1)$.
- $O(|E|)$ פעולות decrease, וכל אחת תיקח זמן משוערך $O(1)$.

סה"כ הזמן לפאזה ה-i: $O(|E|+n_i \log k_i)$.

נרצה: $n_i \log k_i = O(|E|)$.

לכן, נבחר: $k_i = 2^{2|E|/n_i}$.

קיבלנו: זמן פאזה הוא $|E|+n_i \log 2^{2|E|/n_i} = |E|+2|E| = 3|E|$.

מהו מספר הפאזות?

<u>אבחנה</u>: בסוף הפאזה ה-i יש n_{i+1} עצים. (כל עץ בפאזה הנוכחית הופך להיות צומת בפאזה הבאה). כל אחד מעצים אלה נוגע (מכיל או שכן) בלפחות k_i קשתות. לכן אם נסמן את מס' הקשתות בפאזה ה-i ב-E_i, כיון שכל קשת נוגעת לכל היותר בשני עצים, נקבל כי $n_{i+1}k_i \le 2E_i \le 2E$.

$$n_{i+1} \le \frac{2E_i}{k_i}$$

$$k_{i+1} = 2^{2E/n_{i+1}} \ge 2^{2E/(2E/k_i)} = 2^{k_i}$$

$$k_1 = 2^{2E/n}$$

$$k_2 = 2^{2^{2E/n}}$$

$$\dots$$

$$k_{final} \le n$$

<u>מס' הפאזות:</u> $\min\{i : \log^{(i)} n \le 2|E|/n\}$.

פונקציה זאת נקראת $\beta(|E|,n)$.

<u>משפט</u>: ניתן למצוא עץ פורש מינימום בזמן $O(|E|\beta(|E|,n))$.

(הוכחנו כבר).

נחקור את פונקציה β

א. עבור $|E| > n\log n$:

$$\beta(|E|,n) = \min\{i : \log^{(i)} n \le 2|E|/n\}$$
$$\beta(|E|,n) = 1$$

ב. עבור $|E| = cn$

$$\beta(cn,n) = \min\{i : \log^{(i)} n \le 2c\}$$

עבור c=1 נקבל: $\beta(cn,n) = \min\{i : \log^{(i)} n \le 2\}$

לפונקציה זו קוראים $\log^*$ ($\log^* n = \min\{i : \log^{(i)} n \le 2\}$).

מכיון ש-

$$2^2 = 4$$
$$2^{2^2} = 2^4 = 16$$
$$2^{2^{2^2}} = 2^{16} = 65{,}536$$
$$2^{2^{2^{2^2}}} = 2^{65{,}536} = \left(2^{10}\right)^{6{,}553.6} = 1024^{6{,}553.6} \approx 10^{19{,}661}$$

לכל מספר סביר, פונקציה זו מקבלת ערך לא גדול מ- 5.

עבור c>1, נקבל: $\beta(cn,n) \le \log^* n$, ולכן האלגוריתם יותר יעיל מהאלגוריתם של Prim.