מבני נתונים 2

19 מרץ, 2000

שימושים לערימת פיבונצ'י

האלגוריתם של דייקסטרה למציאת מרחקים מינימליים בגרף

קלט: גרף מכוון וקשיר בן n צמתים עם משקולות w אי-שליליים על הקשתות וצומת מקור s.
פלט: לכל צומת, מרחקו המינימלי מ- s, כאשר המרחק מודגר כסכום המשקלים על פני מסלול מ-s לצומת.

האלגוריתם:
בכל רגע נחזיק קבוצת צמתים S של צמתים שמרחקם המינימלי מ- s ידוע ולא ישתנה בהמשך האלגוריתם. בנוסף נחזיק חזית: קבוצת כל הצמתים שהם שכנים של צמתים מ- S, אך אינם ב- S בעצמם.

אתחול: הכנס את s לחזית.
איטרציה: כל עוד החזית אינה ריקה, בצע:
1. מצא את הצומת v בחזית שמרחקו מ-s מינימלי.
2. הוצא את v מהחזית והוסף אותו ל- S.
3. עבור כל שכן u של v:
   - אם u בחזית, בדוק אם המסלול מ- s ל- u שעובר דרך v הוא קטן יותר מזה שידוע.
   - אם u איננו בחזית, הכנס אותו לחזית. מחיר המסלול המינימלי הידוע מ- s ל- u הוא מחיר המסלול העובר דרך v.

מימוש נאיבי:
נחזיק מערך של כל צמתי החזית. מציאת צומת בעל מרחק מינימלי מ-s בחזית ידרוש מעבר על כל צמתי החזית ולכן יקח זמן $O(n)$ במקרה הגרוע. יהא v הצומת בחזית שהוא בעל מרחק מינימלי מ-s. עדכון השכנים של v ייקח זמן $O(\deg(v))$.

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

מימוש יעיל:
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).

עדכון השכנים:
נסמן ב $dist(s,v)$ את 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 $\Delta = dist(s,v) + w(v,v_i) - dist(s,v_i)$.

סיבוכיות זמן:
אתחול: $O(n)$ (מסתכלים על השכנים של s).
ביצוע del_min: $O(\log n)$ לפעולה בודדת. מבצעים n פעולות כאלה לכל היותר. 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 זמן: $O(n\log n+|E|)$.

עץ פורש מינימום (MST)

הבעיה: נתון גרף קשיר לא מכוון עם משקולות לקשתות. מצא עץ פורש שסכום משקלי הקשתות בו הוא מינימלי. (עץ פורש מינימום, עץ פורש מינימום).

מסגרת כללית למציאת עצים פורשים:
נגדיר: אלגוריתם צובר הוא אלגוריתם שעובר על כל הקשתות, ולכל קשת מחליט האם להכניס אותה לעץ או לדחות אותה. האלגוריתם פועל במעבר אחד ואיננו משנה החלטות קודמות.

כל אלגוריתם צובר חייב לקיים את התכונה הבאה (שתסומן ב- *):
* בכל רגע קיים עץ פורש מינימום המכיל את כל הקשתות שהתקבלו ואינון מכיל אף קשת שנדחתה.

נגדיר: חתך בגרף הוא קבוצת קשתות שסילוקה מהגרף תשאיר גרף לא קשיר, כך שכל קשת שסולקה מחברת בין שני רכיבי קשירות שונים.

נתאר את הפעולה של אלגוריתם צובר כפעולת צביעה של הקשתות בכחול או אדום. הקשתות הכחולות נכנסות לעץ והאדומות נדחות. נראה שני כללים לצביעת קשתות.

הכלל הכחול: בחר חתך שבו אף קשת איננה כחולה ויש בו קשת שעדיין לא נצבעה. צבע בכחול את קשת החתך הקלה ביותר שעדיין לא נצבעה. (אם יש יותר מאחת, בחר אחת באופן שרירותי).

הכלל האדום: בחר מעגל פשוט שבו אף קשת אינה אדומה ויש בו קשת שעדיין לא נצבעה. צבע באדום את קשת המעגל הכבדה ביותר שעדיין לא נצבעה. (אם יש יותר מאחת, בחר אחת באופן שרירותי).

האלגוריתם הצובר הגנרי: כל עוד יש בגרף קשת לא צבועה, הפעל את הכלל הכחול או האדום.

טענה: אם יש בגרף קשת לא צבועה, ניתן להפעיל את אחד הכללים.
הוכחה: תהא $e=(u,v)$ קשת לא צבועה. נסמן ב- B את קב' הצמתים שיש אליהם מסלול כחול מ- u. אם $v\in B$, אז e סוגרת מעגל שבו כל הקשתות פרט ל- e כחולות, וניתן להפעיל את הכלל האדום. אחרת, אוסף הקשתות שבדיוק צומת קצה אחד שלהן נמצא ב- B מהווה חתך שאין בו אף קשת כחולה (מהבניה) ויש בו קשת לא צבועה (e). מכאן שאפשר להפעיל את הכלל האדום.
משי"ל

טענה: האלגוריתם הגנרי מקיים את התכונה *.
הוכחה: באינדוקציה על שלבי האלגוריתם. בהתחלה אין קשתות צבועות והתנאי מתקיים באופן ריק. נניח שבסוף הצעד ה- i התכונה מתקיימת, ויהא T עץ פורש מינימום שמראה זאת. נראה שקיים עץ פורש מינימום T' המראה ש-* מתקיימת גם בסוף הצעד ה- i+1. נסמן ב- $w(e)$ את משקל הקשת e.

מקרה 1 : מתבצע הכלל הכחול

יהא C החתך שנבחר, ותהא e הקשת מתוכו שנצבעה. אם $e \in T$, נבחר $T'=T$. אחרת, קיימת קשת $\tilde{e} \in T \cap C$. הקשתות ב T אינן אדומות, והקשתות ב C אינן כחולות. לכן $\tilde{e}$ אינה צבועה. מהעובדה שהכלל הכחול בחר ב- e ולא ב- $\tilde{e}$ נובע $w(e) \le w(\tilde{e})$. לכן, $T' = (T - \{\tilde{e}\}) \cup \{e\}$ הוא עץ פורש מינימום כנדרש.

מקרה 2 : מתבצע הכלל האדום

יהא C המעגל שנבחר, ותהא e הקשת מתוכו שנצבעה. אם $e \notin T$, נבחר $T'=T$. אחרת, קיימת קשת $\tilde{e} \in C - T$. הקשתות ב C אינן אדומות, והקשתות מחוץ ל- T אינן כחולות. לכן $\tilde{e}$ אינה צבועה. מהעובדה שהכלל האדום בחר ב- e ולא ב- $\tilde{e}$ נובע $w(e) \ge w(\tilde{e})$. לכן, $T' = (T - \{e\}) \cup \{\tilde{e}\}$ הוא עץ פורש מינימום כנדרש.

משי"ל

טענה: כל אלגוריתם צובר הוא מקרה פרטי של האלגוריתם הגנרי. (Kruskal, Prim, ...)

האלגוריתם של Prim:

יהא s צומת כלשהו. נגדל עץ מ-s. בכל שלב, יש בידנו עץ חלקי T וחזית של צמתים שהם שכנים של צמתי T אך אינם שייכים ל-T.

אתחול : הכנס את s ל-T ואת שכניו לחזית.

איטרציה : כל עוד החזית אינה ריקה, בצע :

1. יהא v המשקל המינימלי של קשת בין צומת בחזית וצומת ב-T, כך ש-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.

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

מימוש בעזרת ערמת פיבונאצ'י : נשמור את החזית בערמת פיבונצ'י. שוב, בעזרת אותו ניתוח כמו באלגוריתם של דייקסטרה, נקבל סיבוכיות הזמן : $O\left(n\log n+|E|\right)$.

אלגוריתם משופר

נניח, ש- $|E| < n\log n$. (במקרים אחרים האלגוריתם הקודם שלקח זמן $O\left(n\log n+|E|\right)$ 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 איננו 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 $n_{i+1}k_i \le 2E_i \le 2E$.

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

$$k_1 = 2^{2E/n}$$
$$k_2 = 2^{2^{2E/n}}$$
...
$$k_{final} \le n$$

<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 n סביר, $\log n \le 5$.
עבור c>1, נקבל: $\beta(cn,n) \le \log^* n$, ולכן האלגוריתם יותר יעיל מהאלגוריתם של Prim.