אלגוריתמים מבוזרים - הרצאה 9 - בעיות גלובאליות

שרון מלטר, אתגר 17

21 בספטמבר 2024

תוכן עניינים

1 הקדמה

ראינו בהרצאה 1 שבמודל ה־ $LOCAL$ ניתן לפתור כל בעיה ב־ $O(D)$ סיבובים כש־ D הוא קוטר הגרף.

2 בעיות גלובליות

עד הרצאה זו התמקדנו בבעיות לוקליות, שאפשר לפתור אותן מהר יותר. כמו צביעה, MIS ו־ MDS. קיימות בעיות **גלובאליות** שהן בעיות המצריכות $\Omega(D)$ סיבובים. למשל, חישוב מרחקים בגרף, מציאת עץ פורש מינימלי, מציאת חתך מינימלי ו־ 2־צביעה של מסלול.
ניתן לפתור אותן ב־ $O(D)$ סיבובים במודל ה־ $LOCAL$ ע"י לימוד הגרף כולו.

מעבר לחוסר סקרנותנו לבעיות גלובליות, גם פחות התעניינו **בהגבלת** גודל ההודעות שניתן להעביר בין שכנים.
אם זאת, למדנו כי במודל $CONGEST$ הודעות מוגבלות ל־ $O(logn)$ ביטים.

עד כאן דיבורי סרק. נתחיל ללמוד על בעיות גלובליות;

3 בעיית $SSSP$

נתון שורש r ומטרתנו היא שכל צומת אחר ילמד את המרחק שלו מ־ r. (נתמקד בגרפים לא מכוונים ולא ממושקלים. כמו כן מרחק בין זוג צמתים הוא מס' הקשתות במסלול הקצר ביותר ביניהן)

3.1 חסם תחתון לבעיה

<u>טענה:</u> כל אלגוריתם במודל ה־ $LOCAL$ לבעיית $SSSP$ מצריך $\Omega(D)$ סיבובים.

3.1.1 הוכחה

נראה שני גרפים בעלי n צמתים שבהם סביבת $\frac{D}{2}$ של צומת v **נראית זהה** עבור קבוע c מתאים, למרות שהמרחק של v מ־ r שונה בשני הגרפים.

למשל: גרף שרוך G_1 וגרף שני G_2 הזהה לו מלבד לכך ש־ r, הצומת הראשון בשרוך, ישנו בן שני שהינו הצומת השכן של r ב־ G_1. . נסמן ב־ v את הצומת האחרון בשרוך (בקצה השני, בו לא נמצא r)
קוטר שני הגרפים הוא $n-1$ וסביבת $\frac{n}{2}$ של v נראית אותו הדבר.
כלומר, הפלט של v יהיה זהה, למרות שבגרף הראשון המרחק שלו מ־ r הוא $n-1$ ובשני המרחק שלו מ־ r הוא $n-2$.

קיבלנו סתירה ולכן כל אלגוריתם מצריך לפחות $\frac{n}{2}=\Omega(D)$ סיבובים.

(האלגוריתם מצריך $\Omega(D)$ סיבובים עבור גרף שרוך, לכן הוא יצטרך $\Omega(D)$ סיבובים תמיד, אחרת הוא לא פותר את הבעיה עבור קלטים מסוימים)

3.2 הערה על ת.ב 1 שאלה 3

חישוב עץ פורש מינימלי מצריך $\Omega(D)$ סיבובים :)
בשאלות מסוג זה צריך להראות שני גרפים בעלי n צמתים שבהם יש צומת v המקיים;

1. סביבות $\frac{D}{2}$ של v זהות בשני הגרפים (אותם מזהים ואותם משקלים), לכן בכל אלגוריתם שלוקח לכל היותר $\frac{D}{c}$ סיבובים, v יפעל באותו אופן בשני הגרפים.

2. הפלטים עבור v **צריכים להיות שונים** בפתרון פיזבילי לגרפים.

3.3 פתרון במודל $CONGEST$

טענה: ניתן לפתור את בעיית $SSSP$ עבור גרפים לא מכוונים ולא ממושקלים ב־ $O(D)$ סיבובים במודל ה־ $CONGEST$.

הוכחה: נוכיח בעזרת בניית אלגוריתם.
נחשב עץ BFS מהשורש r. בסיבוב ה־ 1, r יודיע לשכניו שהם במרחק 1 ממנו.
בסיבוב הבא, הצמתים במרחק 1 ישלחו הודעה לשכניהם. כל צומת שמקבל הודעה ולא למד על מרחקו עדיין, הוא ממרחק $i+1$, כך שהם ידעו שהם במרחק 2 מ־ r.
ואיך מחשבים את העץ BFS? ־ צומת במרחק i מ־ r משכן אחד לפחות. הוא יבחר באופן שרירותי את אחד מצמתים אלה כהורה שלו.

שימו לב שהמרחק בין שני צמתים הוא לכל היותר n, כך שניתן תמיד לייצג אותו ב־ $O(logn)$ ביטים, כך שהאלגוריתם מציית למגבלת גודל ההודעות.

3.3.1 יעילות

האלגוריתם יתסתיים לאחר $O(D)$ סיבובים :)

3.3.2 הוכחת נכונות הפתרון

מדובר בהוכחה פשוטה, לכן לא נתעמק בה יתר על המידה.

רעיון:
נוכיח שאם צומת הוא במרחק i מ־ r, אז הוא יקבל הודעה לאחר i סיבובים.
הוא **לא יקבל הודעות** לפני כן, אחרת קיים מסלול קצר יותר מ־ r אליו.

הערה: ניתן לחשב עץ BFS ב־ $O(D)$ סיבובים ב־ $CONGEST$ גם אם לא נתון מראש שורש r.

הכיצד? ־ נגדיר את r להיות הצומת בעל המזהה המקסימלי (שאולי לא ידוע בהתחלה לכל הצמתים) ונריץ את האלגוריתם ל־ BFS מכל הצמתים במקביל כך שנשלח בהודעות גם את מזהה השורש ממנו התחיל האלגוריתם.
אזי אם צומת קיבל הודעות מכמה אלגוריתמים שונים, הוא ימשיך להריץ רק את האלגוריתם בעל המזהה המקסימלי שלמד עליו עד עכשיו.
אחרי $O(D)$ סיבובים נסיים לחשב עץ BFS מהשורש r בעל המזהה המקסימלי, וכל שאר האלגוריתמים יעצרו.

עד כאן בעיית $SSSP$.

4 בעיית $APSP$

נתמקד במקרה של גרפים לא ממושקלים ולא מכוונים.
בבעיה זו כל צומת צריך לדעת את מרחקו מכל שאר הצמתים בגרף.

נרצה לפתור את $APSP$ במודל ה־ $CONGEST$.

4.1 פתרון נאיבי

<u>אלגוריתם:</u> נריץ $SSSP$ מכל צומת בגרף.
סיבוכיות הזמן היא $O(D \cdot n)$.
לא משהו.

<u>הבחנה:</u> אם גודל ההודעות לא מוגבל, אז נוכל לחשב את הפתרון ב־ $O(D)$ סיבובים (כל הודעה תכיל את כל המרחקים של השולח)

<u>הערות</u>

- $\widetilde{\Omega}(n)$ זהו גם חסם תחתון ל־ $APSP$ ב־ $CONGEST$ (גם לקירובים)
- קיים אלגוריתם ב־ $\widetilde{O}(n)$ סיבובים, גם במקרה הממושקל.

4.2 פתרון $Peley,\ Roditty\ and\ Tal,\ 2012$

<u>רעיון:</u>
נרצה לבנות n עצי BFS במקביל, אבל בצורה חכמה, כך שלא "התנגשויות" בין הריצות השונות. נראה ש־ DFS הוא שימושי כדי לסנכרן את הריצות.

להלן המבנה הכללי של האלגוריתם;

1. מחשבים עץ BFS מצומת ספציפי v_0.
2. מחשבים DFS על העץ שקיבלנו ב־1.
   לכל צומת v, נסמן ב־ $T(v)$ את הסיבוב הראשון שבו הגענו אליו בסריקת DFS.
3. מחליטים על זמן התחלה גלובלי T_0 (נראה מאוחר יותר כיצד נגדיר אותו) וכל צומת v מתחיל לבנות עץ BFS בזמן $T_0 + 2 \cdot T(v)$.
4. בסיום, כל צומת יודע על המרחקים שלו **מכל** הצמתים.

נסיק את סיבוכיות הזמן של שלושת השלבים הראשונים;

1. שלב 1 לוקח $O(n)$ סיבובים.
2. שלב 2 לוקח $O(n)$ סיבובים.
3. נרצה להראות שבשלב 3 **אין התנגשויות** בין הריצות השונות. כלומר, אף צומת לא ייקבל יותר מהודעה אחת מאלגוריתם BFS אחד בכל סיבוב. מלמה זו נסיק את סיבוכיות שלב זה.

<u>טענה 1:</u>
לכל $v \in V$, מתקיים $T(v) < 2n$.

<u>הוכחה 1:</u>
בסריקת ה־ DFS עוברים על כל קשת פעמיים, כיוון שמס' הקשתות בעץ הוא $n-1$. אזי מספר איטרציות ה־ DFS קטנה מ־ $2n$.

טענה 2:
לכל $v, w \in V$, אם $T(v) < T(w)$ אזי $d(v,w) \leqslant T(w) - T(v)$. (כאשר $d(v,w)$ הוא המרחק בין v, w)

הוכחנו טענה חמודה זו בקורס תכנון וניתוח אלגוריתמים.
אבל יאללה נוכיח;
הוכחה 2:
יהיו v, w כך ש־ $T(v) < T(w)$. בזמן הרצת ה־ DFS עברנו מ־ v ל־ w במסלול מאורך $T(w) - T(v)$. המסלול הקצר ביותר בין שני הצמתים מקיים $d(v,w) \leqslant T(w) - T(v)$.

למה:
משתי הטענות נקבל שאין התנגשויות בין ריצות ה־ BFS השונות.

הוכחת הלמה: נניח שיש התנגשויות, וההתנגשות הראשונה היא בין עצי ה־ BFS של v ו־ w בצומת x. (x קיבל הודעה מ־ v ומ־ w באותו סיבוב) הערה: כמובן שייתכן כי יש עוד התנגשויות באותו סיבוב.
כיוון שזו ההתנגשות הראשונה, מס' הסיבוב שבו קורית ההתנגשות הוא;

$$T_0 + 2 \cdot T(v) + d(v,x) = T_0 + 2 \cdot T(w) + d(w,x)$$

שימו לב כי $T_0 + 2 \cdot T(v)$ הוא זמן התחלת v ו־ $T_0 + 2 \cdot T(w)$ הוא זמן התחלת w.
נניח לה"כ כי $T(v) < T(w)$. נקבל:

$$d(v,x) - d(w,x) = 2 \cdot (T(w) - T(v)) \geqslant 2 \cdot d(v,w)$$

(אי־השוויון נובע מטענה 2)
מצד שני, מאי־שוויון המשולש, נובע כי־

$$d(v,x) \leqslant d(v,w) + d(w,x)$$

$$d(v,x) - d(w,x) \leqslant d(v,w)$$

קיבלנו סתירה, ולכן לא קיימות התנגשויות בין הרצות ה־ BFS־ים.

וכעת לשאלה שבערה בנו ־ איך מגדירים את T_0 ?
תשובה ־ כיוון שאין התנגשויות בשלב 3, וכיוון ש־ $T(v) < 2n$, לכל v מטענה 1, שלב 3 לוקח $O(n + D)$ סיבובים ($T_0 + 2 \cdot max_v\ T(v) + O(D) = O(n)$, כך ש־ $T_0 + 2 \cdot max_v\ T(v)$ הוא הזמן המקסימלי של BFS ו־ $O(D)$ זמן ריצת BFS יחיד)
כך שסה"כ זמן הריצה הוא $O(n)$, כך שניתן לקבוע את T_0 להיות הזמן שבו מסתיים שלב 2 ב־ $O(n)$.

4.3 סיכום

קיבלנו אלגוריתם ב־ $O(n)$ סיבובים ל־ $APSP$ ב־ $CONGEST$.
מסקנה: ניתן לחשב את D במדויק ב־ $O(n)$ סיבובים, מכיוון ש־ $D = max_{u,v}\ d(u,v)$.

הערות:

- יש חסם $\widetilde{\Omega}(n)$ לחישוב מדויק של הקוטר וגם לקירוב $\frac{2}{3} - \epsilon$. אם רוצים קירובים פחות טובים אז יש גם אלג' יותר יעילים.
- אפשר להשיג קירוב 2 לקוטר ב־ $O(D)$ סיבובים ע"י חישוב עץ BFS (תרגיל)
- ראינו שאפשר לחשב n עצי BFS במקביל ב־ $O(n)$ סיבובים. מה לגבי $k < n$?
- באופן כללי, אפשר לחשב k עצי BFS דטרמיניסטית ב־ $O(D + k)$ סיבובים (מצריך טכניקות אחרות, $pipelining$ חכם)
- אם מסתפקים באלג' רנדומי, טכניקה שמאפשרת להשיג את זה היא $random\ delays$. בשיטה זו במקום לבחור את זמן ההתחלה לפי DFS כמו שראינו, בוחרים אותם עם מרווחים אקראיים בין זה לזה. כלומר, כל צומת בוחר באופן רנדומי זמן התחלה מ־ $\{1, ..., k\}$.