ה-Dense Distance Graph ו-Dijkstra

תזכורת - Dense Distance Graph

הסיטואציה: גרף מישורי G מחולק לאזורים G_{out}, G_{in} ע"י מפריד מעגלי C. G_i' הוא גרף מלא שקדקודיו הם הקדקודים של C.

אורכה של הקשת uv ב-G_i' הוא המרחק מ-u ל-v ב-G_i.

$$G' = \bigcup_i G_i'$$

ראינו בשיעור הקודם איך עושים Bellman-Ford על G'.

אם ב-G' N קדקודים, מימוש נאיבי של BF לוקח $O(N^3)$ זמן. ראינו איך משתמשים בתכונת Monge כדי להשיג זמן ריצה של $O(N^2 \log N)$.

האם אפשר לעשות דבר דומה עם Dijkstra (בהנחה שמרחקים הם אי-שליליים)?

מימוש נאיבי ייקח $O(N^2 \log N)$

נראה שאפשר לקבל $O(N \log^2 N)$ (לא מספיק זמן להסתכל על כל הקשתות של G'!)

נזכר ב- Dijkstra:

$SP(s)$:

$d(s)=0$; $d(v)=\infty \ \forall v\neq s$

initialize heap Q of vertices with key $d(\cdot)$

$S=\emptyset$

While $S\neq V(G)$:

    $v \leftarrow Q$.ExtractMin

    Activate(v)

הפרוצדורה Activate עושה רלקסציה לכל הקשתות שיוצאות מ-v.

מימוש רגיל של Activate לוקח $O(N\log N)$ זמן

כי רלקסציה של כל קשת עלולה לגרום לעדכון ה- heap.

נראה שאפשר (תחת הנחה מקילה) לממש את Activate

בזמן $O(\log N)$!

למבנה הנתונים/אלגוריתם קוראים FR-Dijkstra, FR הם

Fakcharoenphol ו- Rao, ממציאי האלגוריתם.

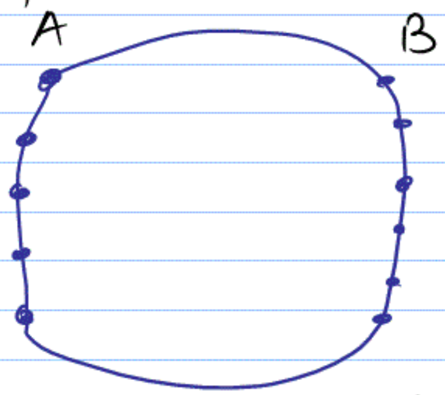
איך קיבלנו אלגוריתם מהיר יותר במקרה של Bellman-Ford ?

עשינו רלקסציה לקשתות בקבוצות - טיפלנו בכל G_i בנפרד, ואת הקשתות של כל G_i חילקנו לשתי קבוצות (שמאליות לימניות ושלשיות).

גם כאן, נחלק לתתי-גרף זרים בקשתות, אבל לא בקדקודים. בכל תת-גרף כזה נצליח לממש Activate ו- ExtractMin ביעילות, אבל נצטרך גם לתאם בין תתי-הגרפים. התאום יותר מורכב מאשר ב- BF כי הנכונות של Dijkstra מחייבת למצוא בכל שלב את הקדקד במרחק מינימלי (בכל הגרף, לא בתת-גרף מסוים) שטרם טיפלנו בו.

ראשית נראה איך מטפלים בתת-גרף יחיד ואחר כך נרכיב הכל יחד.

הנחה: הגרפים G_i הם גרפים דו-צדדיים שנראים כך:



מסלולים מכוונים מ-A ל-B.

כלומר: Activate תמיד פועל

על קדקד ב-A, ו-ExtractMin

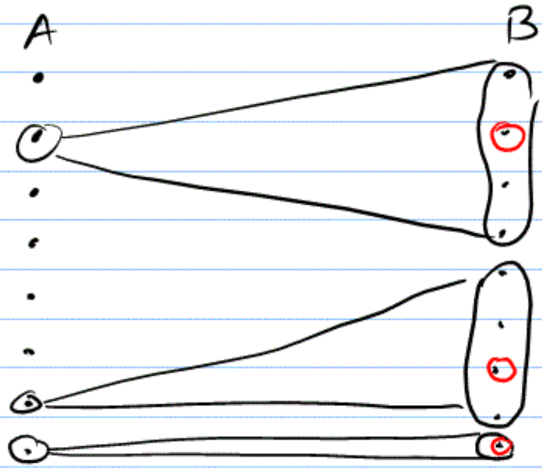
תמיד מחזיר קדקד ב-B.

הערה: - משמעות ההנחה שמטריצת המרחקים של G_i' הוא Monge.

- יהיו יותר משני G_i'. הקבוצה B של G_i' יכולה להיות הקבוצה A של G_j'.

הרעיון: לא חייבים לעשות רלקסציה לכל הקשתות של v.

מספיק לדעת מיהו הקדקד ב-B שהמרחק אליו מינימלי.



נאמר ש $a\in A$ הוא הורה של $b\in B$ אם:

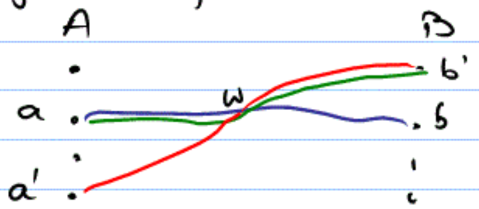
$$a = \underset{a'\in A}{\arg\min}\ d(a') + c(a'b)$$

נשמור לכל $a\in A$ שעשינו לו Activate מיהם ילדיו.

<u>למה</u> - ביחס ההורות אין חיתוכים. כלומר, עבור $i<j$, $k<l$

אם a_i הורה של b_l אז a_j אינו הורה של b_k.

הוכחה - בציור. דומה להוכחת תכונת ה- Monge בשיעור הקודם



אם a הורה של b, אז הוא גם ההורה של קדקד w

בגלל המישורים השווים ולכן $d(a)$ + אורך המסלול הירוק אינו גדול

מ- $d(a')$ + אורך המסלול האדום.

כלומר a הורה של b'. $\square$

<u>מסקנה</u> - קבוצת הילדים של b של $a\in A$ היא תת-סדרה רציפה

של קדקדי B.

מספיק לשמור לכל $a\in A$ את קצות האינטרוול של ילדיו ב- B.

את האינטרוולים נייצג באמצעות שלשות (a,b_1,b_2)

נאחסן את השלשות בעץ חיפוש בינארי T לפי יחס סדר לקסיקוגרפי.

כמו כן, נשמור לכל $a \in A$ מי מבין הילדים שלו הוא זה עבורו הגודל

$d(b) = d(a) + c(ab)$ מינימלי (הנציג של a)

<u>אבחנה</u>: אם $b_i\, b_{i+1} \ldots b_j$ ילדים של a, אזי הילד בעל

מרחק מינימלי תלוי ב- i, j, אבל לא ב- $d(a)$

ננצל זאת: לפני תחילת האלגוריתם, בזמן שבונים את G_i',

נבנה מבנה נתונים שלכל a, i, j מחזיר את

האינדקס $\underset{i \le k \le j}{\operatorname{argmin}}\ c(ab_k)$

(זמן בניה כזמן הבניה של G_i', זמן שאילתא $\log(N)$)

קיימת בניה מהירה יותר שמנצלת את תכונת ה Monge של G_i'

כדי לתמוך ב ExtractMin, נשמור את הנציגים בערימה H_i

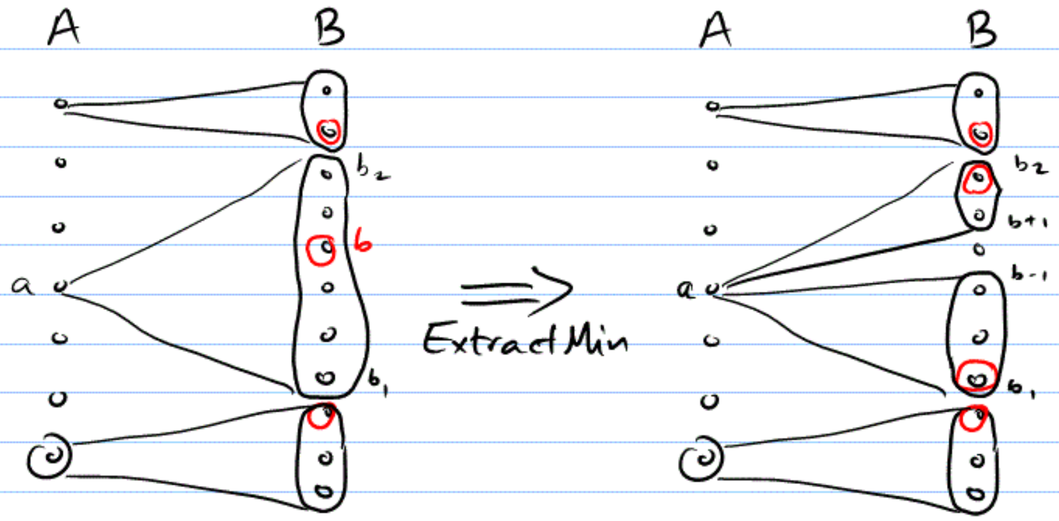
כך שב- $O(\log N)$ זמן, נוכל למצוא את הקדקד $b \in B$

בעל מרחק מינימלי. נוציא את b מהערימה H_i, ונפצל

את השלשה (a, b_1, b_2) ש- b היה הנציג שלה לשלשות

$(a, b_1, b-1)$, $(a, b+1, b_2)$, ונכניס ל- H_i את הנציגים

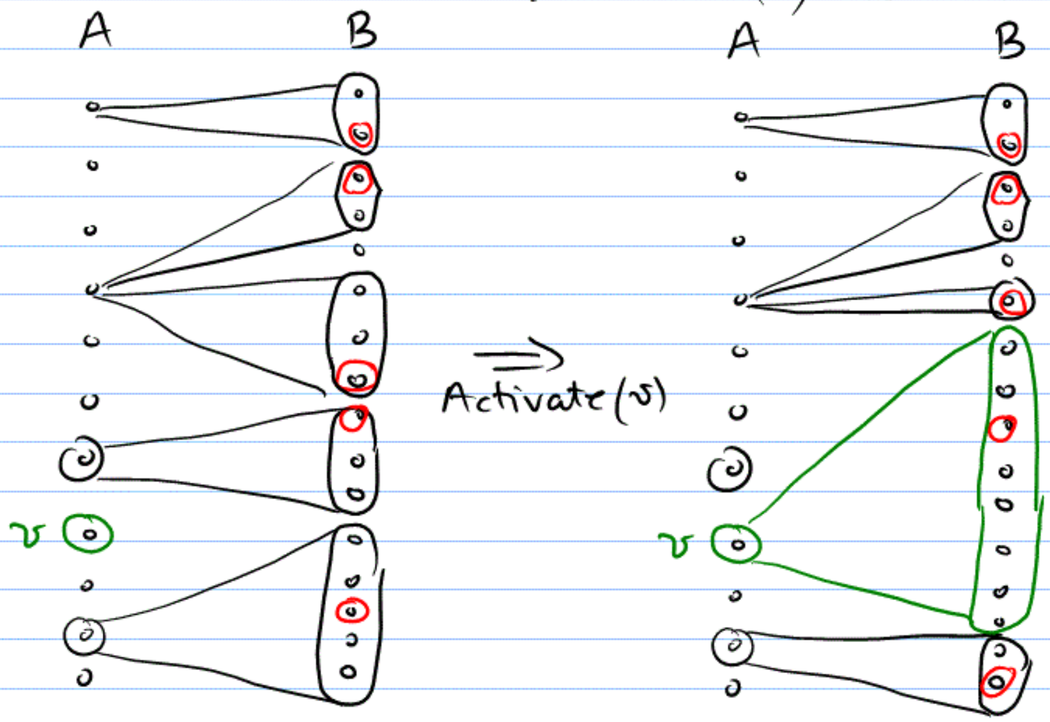
של שלשות אלו.



הזמן לביצוע ExtractMin :

מספר קבוע של פעולות על הערימה H_i:	$O(\log N)$
מספר קבוע של פעולות בעץ T	$O(\log N)$
מציאת מספר קבוע של נציגים	$O(\log N)$
	$O(\log N)$

כיצד מממשים את $Activate(v)$?



מוצאים את השלשה t ב-T שבאה אחרי v.

עוברים על השלשות ב-T החל מ-t, עד שמוצאים שלשה $t'=(a,b_1,b_2)$

עבורה $d(a)+c(ab_2) < d(v)+c(vb_2)$

u הופך להיות ההורה של כל הצמתים ב-B בשלשות בין t ל-t'
(לא כולל t')

u אינו ההורה של b_1.

מוצאים היכן בדיוק עובר הגבול בין השלשה של u לשלשה של a,
ע"י חיפוש בינארי בין b_1 ל-b_2.

מוצאים את הצמתים של השכבה שהתעדכנו, מכניסים אותם ל H_i

ומוציאים מ-H_i את הצמתים של שכבה שנמחקו.

מבצעים תהליך סימטרי עבור השכבה הקודמת ל-v ב-T.

הזמן לביצוע $Activate(v)$:

מציאת t — $O(\log N)$

הכנסת השכבה החדשה של v ל-T — $O(\log N)$

אבל עוברים בדרך על הרבה שכבות.....

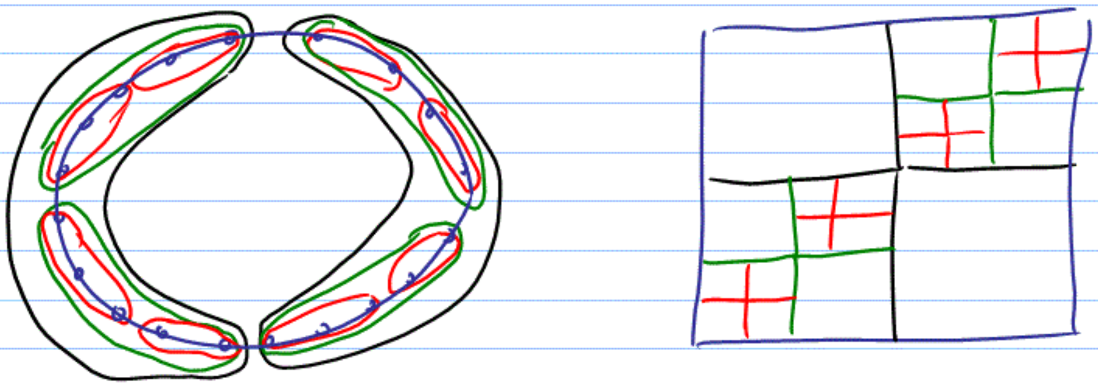
כל שכבה שעוברים עליה, פרט לשתיים נמחקת. הזמן לטפל בכל שכבה כזו (למצוא אותה ב-T, לעדכן את הצמתים ב-H_i...) הוא $O(\log N)$. נחייב את עלות הטיפול לזמן ההכנסה של השכבה (ב Activate או ב-ExtractMin), שבכל מקרה היה $O(\log N)$.

כלומר הזמן ל-$Activate(v)$ הוא $O(\log N)$ amortized.

נקרא למבנה הנתונים שתארנו Monge Heap

הסרת ההנחה:

בהינתן G_i', נפרק אותו לגרפים דו-צדדיים.



יש $2+4+\dots+N = O(N)$ גרפים דו-צדדיים (בגדלים שונים)

כל קדקד נמצא ב-$O(\log N)$ גרפים דו-צדדיים.

לכל גרף דו צדדי יהיה Monge heap משלו, שתומך ב-Activate וב-ExtractMin.

כדי לממש את Dijkstra, כל גרף דו צדדי יתרום את הנציג המינימלי שלו לערימה הגלובלית Q של Dijkstra.

כלומר, ב-Q יהיו $O(N)=O(\sqrt{n})$ איברים.

יש לנו כאן יותר מעותק אחד של כל קדקד. כשמממשים Dijkstra עותקים שונים של אותו קודקוד יכולים להופיע כמינימום הגלובלי. ב-Dijkstra כל קדקד מופיע בדיוק פעם אחת כמינימום הגלובלי. רק ההופעה הראשונה היא אמיתית. השאר הן תוצאה של השכפול ואין צורך לעשות דבר, פרט להוצאת העותק מה-Monge heap בו הופיע.

FR-Dijkstra(G'_{in}, G'_{out}, s)

$d(s)=0$; $d(v)=\infty \quad \forall v \neq s$

decompose into bipartite graphs

initialize a Monge heap for each bipartite graph

initialize heap Q w/ min represetative from each Monge heap

$S = \{s\}$

While $S \neq V(G)$:

  $v, M, d_v \leftarrow Q$. ExtractMin

  M. ExtractMin להוציא את v מ- M

  Q. updateHeap (M. FindMin) לעדכן את הנציג של M ב-Q

  if $v \notin S$: אם זהו המרחק הראשון של v שהוצא מינימום גלובלי, צריך לסמן ב-v

    $d(v) \leftarrow d_v$

    $S \leftarrow S \cup \{v\}$

    for each M that contains v: הפעל את v בכל הגרפים שמכילים אותו

      M. Activate (v)

      Q. UpdateHeap (M. FindMin) ועדכן את הערימה הגלובלית

זמן ריצה :

כל עותק של G נבדק מספר לא יותר פעם אחת כמינימום

של הערימה Q. לכן מספר האיטרציות $O(N \log N)$

כל פעולה $Q.ExtractMin$

$M.ExtractMin$

$M.Activate$

$M.FindMin$

לוקחת $O(\log N)$, כך שסך הכל זמן הריצה

$$O(N \log^2 N) = O(\sqrt{n} \log^2 n)$$

כפי שטענו.