Dense Distance Graph -ב Dijkstra

Note Title

הגדרת - 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 ל-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:

$SP(s)$:

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

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

$S=\emptyset$

While $S\neq V(G)$:

    $v \leftarrow Q.\text{ExtractMin}$

    Activate(v)

הפרוצדורה Activate עושה רלקסציה לכל הקשתות שיוצאות מ-v.

שימוש 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.

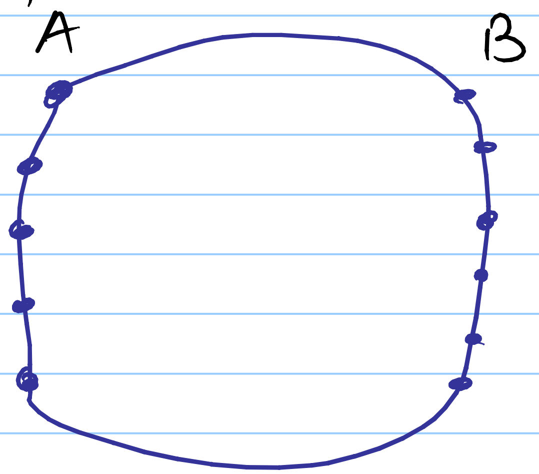
איך קיבלנו אלגוריתם מהיר יותר בשגרה של Bellman-Ford?

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

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

ראשית נראה איך מטפלים בתת-גרף יחיד ואח"כ נרכיב הכל יחד.

<u>הנחה</u>: הגרפים G_i הם גרפים דו-צדדיים שנראים כך:



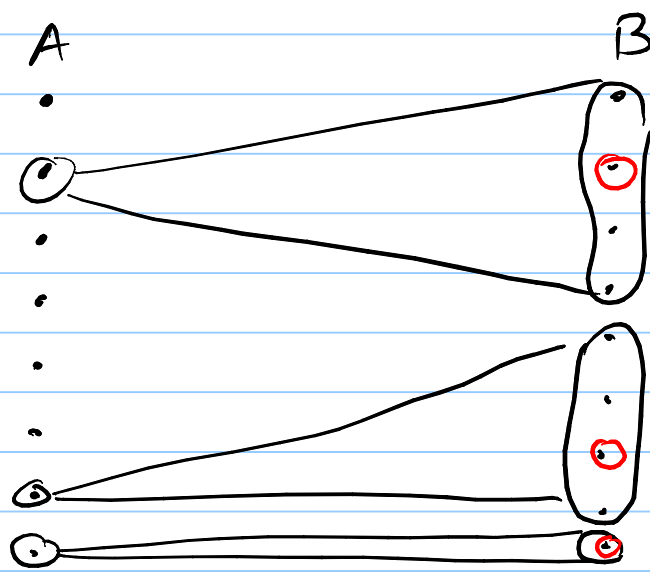
מסלולים מכוונים מ-A ל-B.

כלומר: Activate תמיד פועל על קדקד ב-A, ! 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'.

<u>הרעיון</u>: לא חייבים לעשות רלקסציה לכל הקשתות של ע. מספיק לדעת מיהו הקדקד ב-B שהמרחק אליו מינימאלי.



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

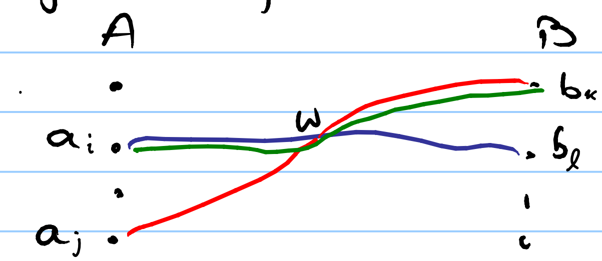
$$a = \underset{a'\in A}{\operatorname{argmin}}\; d(a') + c(a'b)$$

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

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

אם a_i הורה של b_l אז קיים $j\le i$ כך ש-a_j הורה של b_k.

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



נניח a_j הורה של b_k עבור $i<j$. מהמינימליות, המסלול הקצר ביותר מ-a_i ל-b_l

נחתך עם המסלול הקצר ביותר מ-a_j ל-b_k בקדקד כלשהו, w. מתקיים:

$d(a_i)+d(a_i,w)\le d(a_j)+d(a_j,w)$ ולכן $d(a_i)+d(a_i,b_k)\le d(a_i)+d(a_i,w)+d(w,b_k)\le$

$d(a_j)+d(a_j,w)+d(w,b_k)=d(a_j)+d(a_j,b_k)$

כלומר גם a_i הורה של b_k. □

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

של קדקדי B.

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

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

נחזיק 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 $b_i\, b_{i+1} \dots b_j$ 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

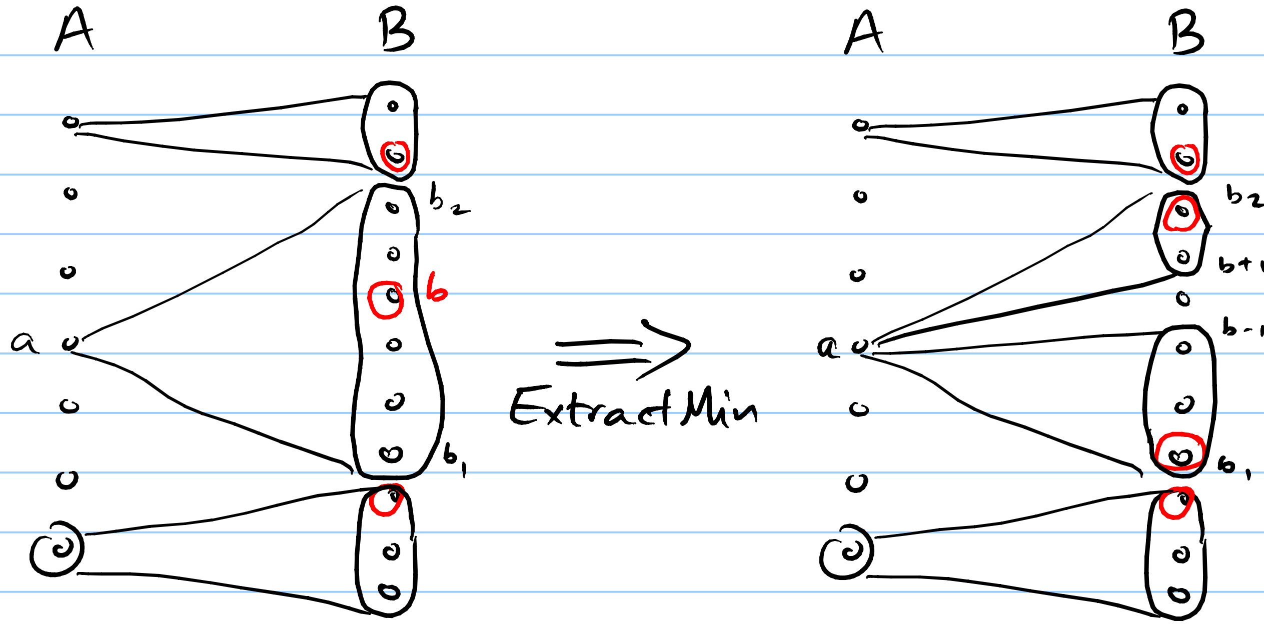
כך שב- $O(\log N)$ זמן, נוכל למצוא את הקדקד $b \in B$

בעל מרחק מינימלי. נוציא את b מהערימה H_i, נפצל

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

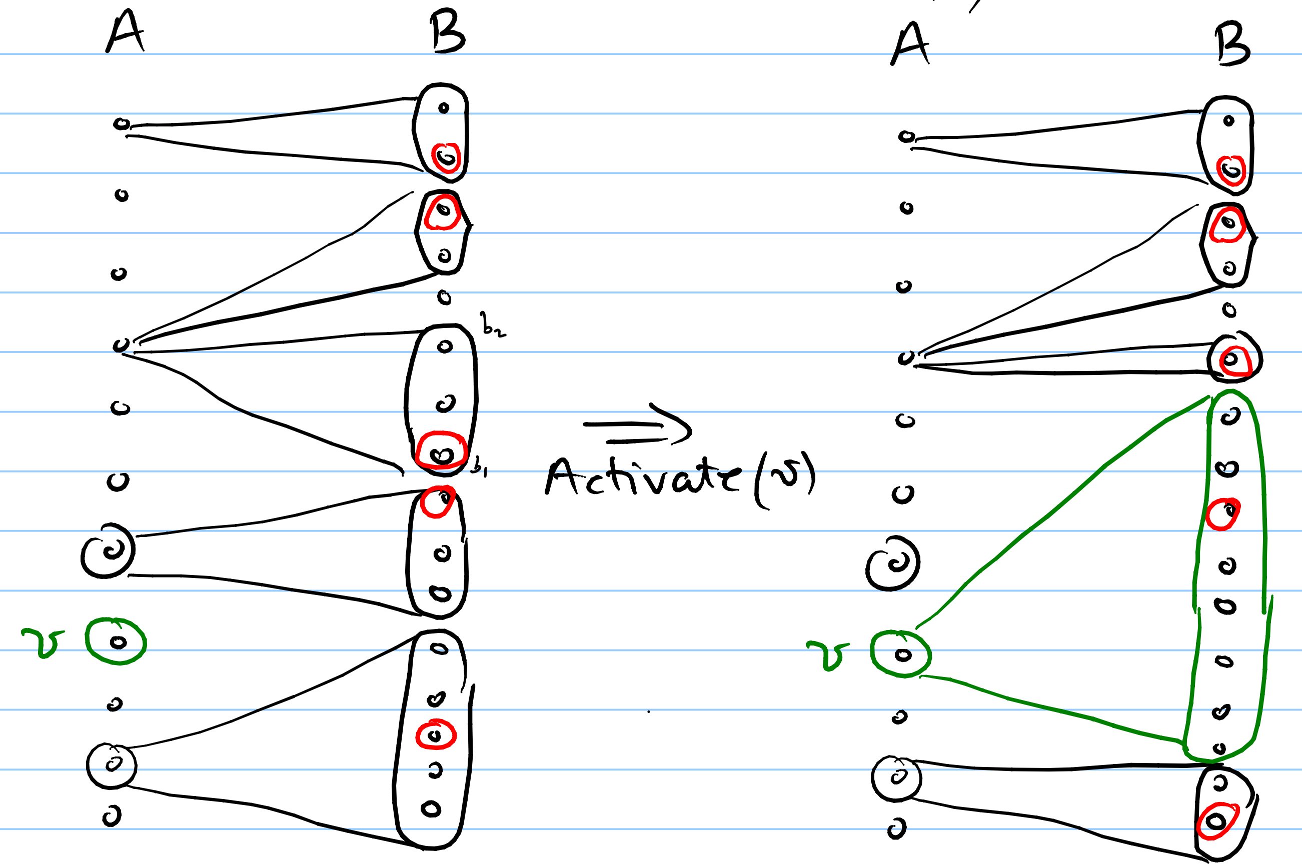
של שלשות אלו.



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 ממשים 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 $d(a)+c(ab_2) < d(v)+c(vb_2)$

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

v אינו ההורה של b_1.

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

מוציאים את הנציגים של השלשה שהתעדכנו, מכניסים אותם ל H_i

ומוציאים מ-H_i את הנציגים של שלשות שנמחקו.

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

הזמן לביצוע $\text{Activate}(v)$:

מציאת t $\qquad O(\log N)$

הכנסת השלשה החדשה של v ל-T $\qquad O(\log N)$

אבל עוברים בדרך על הרבה שלשות.....

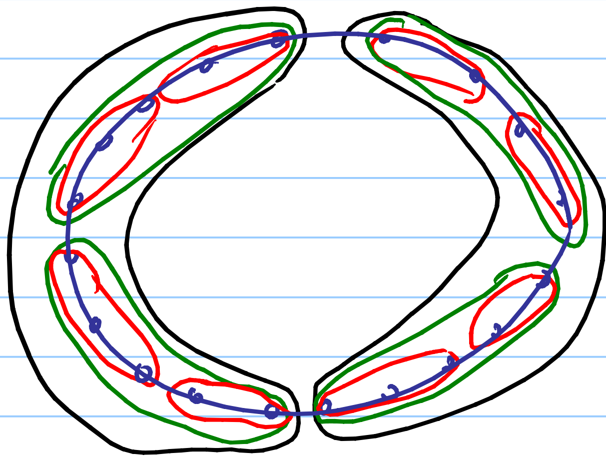
כל שלשה שעוברים עליה, פרט לשתיים נמחקת. הזמן לטפל בכל שלשה כזו (למצוא אותה ב-T, לעדכן את הנציג ב-H_i...) הוא $O(\log N)$. נחייב את עלות הטיפול לזמן ההכנסה של השלשה (ב Activate או ב-ExtractMin), שבכל מקרה היה $O(\log N)$.

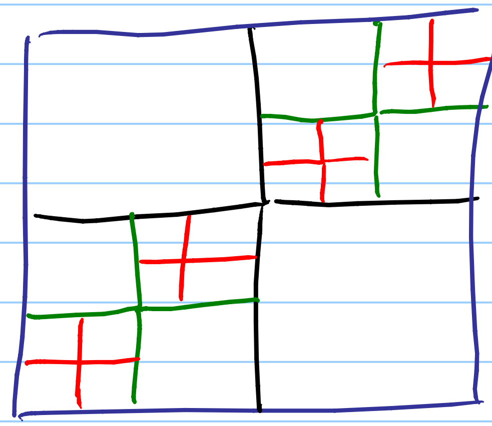
כלומר הזמן ל-$\text{Activate}(v)$ הוא $O(\log N)$ amortized.

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 כעת 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 representative from each Monge heap

$S = \{s\}$

While $S \neq V(G)$:

$v, M, d_v \leftarrow Q.\text{ExtractMin}$

הקודקוד
ה-Monge heap
שמכיל את v כקודקוד
המרחק של v ב-M

$M.\text{ExtractMin}$ — להוציא את v מ-M

$Q.\text{updateHeap}(M.\text{FindMin})$ — לעדכן את הקודקוד של M ב-Q

if $v \notin S$:

אם זהו המופע הראשון של v שהוצא
מינימום גלובלי צריך לטפל 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

$M.\text{Activate}(v)$

$Q.\text{UpdateHeap}(M.\text{FindMin})$

ועדכן
את
הערימה
הגלובלית

זמן ריצה:

כל עותק של כל קדקד מוצא לכל היותר 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.

<u>הרחבה:</u>

ניתן לשלב מספר מוגבל ($O(\sqrt{n})$) של קשתות נוספות, עם אורכים

אי-שליליים כלשהם (לא חייבים להגיע מגרף מישורי.)

לקשתות אלה אפשר לעשות רולקסציה רגילה (כמו ב-Dijkstra קלאסי),

או לחלופין, לחשוב על כל קשת כזו כעל Monge heap

עם קדקד אחד בכל צד.