כיצד משתמשים ב- MSSP

- <u>מסלולים קצרים ביותר.</u>

נתבונן בצומת v. לכל חץ d שראשו v, יש קבוצה רציפה של שורשים (קדקדים של f_∞) עבורם d נמצא בעץ המסלולים הקצרים.

כלומר, החצים של v מחלקים את הקדקדים של f_∞ לאינטרוואלים רציפים.
נשמור לכל קדקד v ייצוג של האינטרוואלים הללו. בהינתן קדקד v ושורש r, הייצוג מחזיר את החץ d של v שהאינטרוואל שלו מכיל את r.
כלומר, אנו יודעים מהו החץ האחרון במסלול קצר ביותר מ-r ל-v.
בצורה זו ניתן לשחזר המסלול כולו. כמה זמן זה יקח?
אם היינו משתמשים בעץ בינארי רגיל לייצוג האינטרוואלים הללו, כל שאילתא הייתה לוקחת $O(\log|f_\infty|) = O(\log n)$. מכיוון שהאינטרוואלים הם של שלמים שגדולים לכל היותר $n>|f_\infty|$, ניתן לעשות זאת בזמן $O(\log\log n)$
[Van Emde Boas 77, Melhorn & Naher 90]
הזמן לשחזור מסלול באורך l יהיה $O(l \log\log n)$.

<u>מרחקים</u> -

אם נתונה לנו רשימה של k זוגות (r_j, v_j), כאשר $r_j\in f_\infty, v_j\in V$,
ניתן לקבל את המרחקים בין הזוגות תוך כדי ריצת האלגוריתם.
לשם כך צריך לייצג גם את העץ T בעץ דינאמי שתומך ב Link, Cut,
הוספה קבועה לכל הצאצאים של קדקד, ושאילתת מרחק של קדקד.
כאשר מקבלים את T_i, שואלים מהם המרחקים לכל הקדקדים בזוגות בהם r_j הוא השורש של T_i.
הזמן: $O(n\log n)$ להרצת MSSP ועוד $O(k\log n)$ לעבור k שאילתות לעץ הדינאמי של T.

באופן כללי יותר, ניתן להשתמש בטכניקה של persistence [Driscoll, Sarnak, Sleator, ושל Tarjan 89]

כדי לשמור את ההיסטוריה של האלגוריתם בצורה 'נגישה'.

סיבוכיות המקום עולה ל- $O(n \log n)$, אבל אז אפשר לגשת לכל

ההיסטוריה של האלגוריתם ב- $O(\log n)$ זמן. כך שבאופן מעשי

ניתן לקבל מרחק משורש לכל קדקד ב- $O(\log n)$ זמן (amortized).

מסלולים קצרים ביותר בגרף עם אורכים שליליים

הנחה: אין מעגלים שאורכם שלילי (אחרת מרחקים אינם מוגדרים)
האלגוריתם שנלמד מגלה מעגל שלילי אם קיים.

תזכורת: Bellman-Ford

חזור $n-1$ פעמים:
- בצע רילקסציה לכל הקשתות

נכונות:
באיטרציה ה-k מחושבים מרחקים במסלולים שבהם לכל היותר k קשתות.

זמן ריצה: $O(n \cdot m)$, כלומר $O(n^2)$ בגרף מישורי.

היום נראה אלגוריתם שרץ בזמן $O(n \log^2 n)$

(הטוב ביותר שידוע היום: $O(n \log^2 n / \log\log n)$)

תזכורת מהשיעור הקודם ומהתרגיל: עלויות (slack costs)

$p(v)$ - פונקצית מחיר $\quad p: V \to \mathbb{R}$

אורך חדש $\quad c_p(uv) = c(uv) + p(u) - p(v)$

"בכמה המסלול דרך uv יקר מהמסלול עפ"י p?"

מהתרגיל: - שימוש באורכים מעודכנים אינו משנה מסלולים קצרים.
- האורך של מסלול מ-s ל-t משתנה ב- $p(s)-p(t)$
- אם p מייצגת מרחקים בגרף אז c_p אי-שליליות.

$\Leftarrow$ אם יודעים מרחקים מקדקד כלשהו v, ניתן לחשב מרחקים מכל קדקד s באמצעות Dijkstra בזמן $O(n)$.

הרעיון: לשלב האבחנה הזו עם פתרון רקורסיבי באמצעות ספרטורים.

האלגוריתם: $SP(G, s)$

1 - מצא מפריד מעגלי $C \Rightarrow G_{in}, G_{out}$

2 - בחר קדקד כלשהו $r \in C$

3 - חשב δ_i: מרחקים קצרים מ-r ב-G_i (רקורסיבית $SP(G_i, r)$)

4 - חשב A_i: מרחקים בין כל זוג קדקדים של C ב-G_i
(באמצעות MSSP, δ_i הוא הפוט' ההתחלתי)

- יהי G_i' הגרף המלא על הקדקדים של C, כאשר אורך הקשת uv
נתון ע"י המרחק מ-u ל-v ב-G_i (נתון ב-A_i)

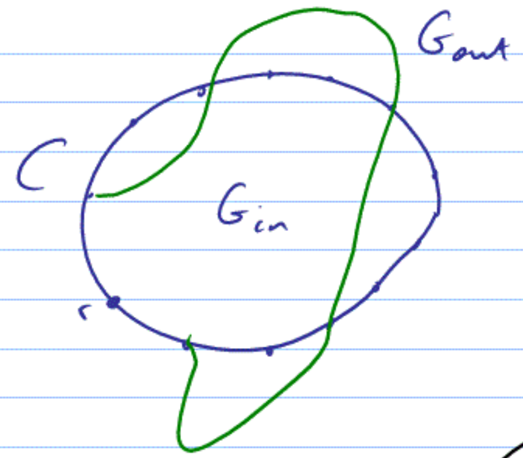
5 - חשב B: מרחקים קצרים ב-G מ-r לכל הקדקדים של C
ע"י Bellman-Ford בגרף $G' = G'_{in} \cup G'_{out}$

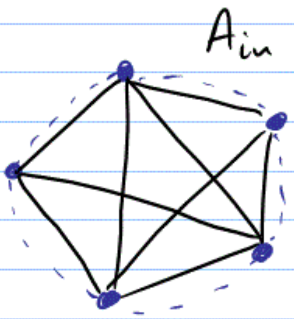
נקרא G'
Dense Distance Graph

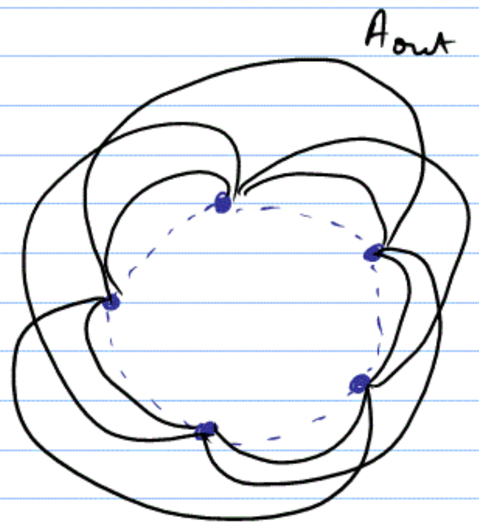
6 - חשב δ_i': מרחקים קצרים ב-G מ-r לכל הקדקדים של G_i
ע"י Dijkstra בגרף G_i, עם פונקציית מחיר δ_i ואתחול
המרחקים לקדקדים של C לערכים ע"פ B.

- יהי: $\delta' = \delta'_{in} \cup \delta'_{out}$ מרחקים מ-r ב-G

7 - חשב מרחקים מ-s ב-G ע"י Dijkstra ופונקציית מחיר δ'.







זמן ריצה:

1. ספרטור -	G	$O(n)$
4. MSSP -	G_i	$\sum_i O(n_i \log n_i) = O(n \log n)$
5. Bellman-Ford	G'	$O(\sqrt{n})$ קודקודים, $O(n)$ קשתות $\Rightarrow O(n^{3/2})$
6. Dijkstra	G_i	$\sum_i O(n_i) = O(n)$
7. Dijkstra	G	$O(n)$

$$O(n^{3/2})$$

$$T(n) = T(n_1) + T(n_2) + O(n^{3/2}) = O(n^{3/2})$$

כבר יותר טוב מ $O(n^2)$, אבל הבחנות ... $O(n \log^2 n)$

[illegible] העקרון הוא חישוב Bellman-Ford ב-G'.

בשלב האיטרציה ה-k של BF, $B(u)$ הוא המרחק מ-r ל-u שמשתמש בלכל היותר k קשתות.
לעשות רלקסציה לכל הקשתות של G' שקול למצוא:

$$\forall v \in C \qquad B(v) = \min_{u \in C} \begin{Bmatrix} B(u) + A_{in}(u,v), \\ B(u) + A_{out}(u,v) \end{Bmatrix}$$

אם נגדיר $A'_i = B(u) + A_i(u,v)$

A'_i היא מטריצה עם $O(\sqrt{n})$ שורות ועמודות.

אז צעד של BF מוצא את האיבר המינימלי בכל עמודה של A'_i

באופן נאיבי זה לוקח $O(n)$ זמן (כמספר האיברים ב-A'_i)

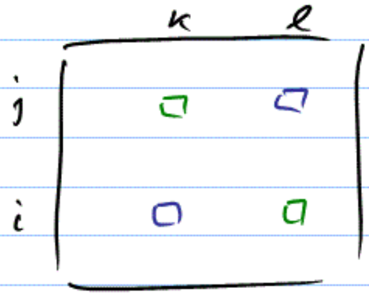
נראה שניתן לעשות זאת ב- $O(\sqrt{n} \log n)$ זמן

(למעשה אפשר ממש ב-$O(\sqrt{n})$ SMAWK).

עיקר: תכונת Monge

מטריצה $A_{n\times n}$ נקראת Monge אם עבור כל $i<j$ ו $k<l$ קיים

$$A_{ik} + A_{jl} \geq A_{il} + A_{jk}$$



תהי: $\mathcal{E}(j) = \arg\min_i A_{ij}$ סדרת האינדקסים של השורות בהן מתקבלים האיברים המינימליים בעמודה

למה - אם A מטריצת Monge אז הסדרה $\mathcal{E}$ אינה עולה.

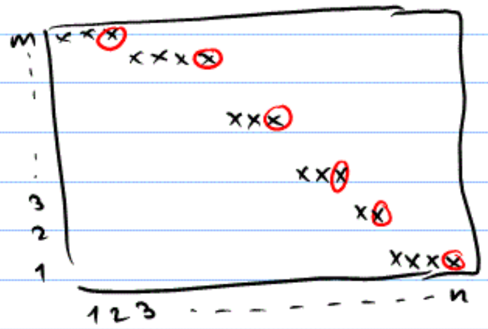
הוכחה: נניח $\mathcal{E}(k)=i$. כלומר, עבור $j>i$,

$$A_{ik} \leq A_{jk}$$

מכיוון שעבור $l>k$ $\quad A_{ik}+A_{jl} \geq A_{il}+A_{jk}$

חייב להתקיים $\quad A_{jl} \geq A_{il}$

כלומר $\mathcal{E}(l) \leq i$ $\square$



למשל, כך יכולה להראות סדרת האינדקסים של האיברים המינימליים בעמודה.

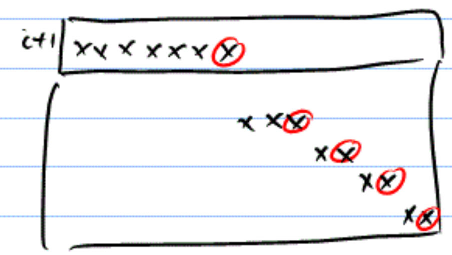
נגדיר breakpoint של A להיות זוג (i,k) כך ש $E(k)=i$ אבל $E(k+1)\neq i$

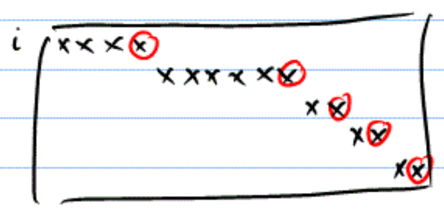
נשים לב שאם ידועה סדרת ה-breakpoints, אז ניתן למצוא את כל האיברים המינימליים של כל העמודות בזמן לינארי במספר העמודות.

כיצד מוצאים את ה breakpoints של A?

נניח שמצאנו את ה-breakpoints של המטריצה המתקבלת מ-i השורות הראשונות של A.

לכל BP (j,k) בסדר עולה של העמודות, נשווה את $A_{i+1,k}$! $A_{j,k}$





מהלמה:

אם $A_{i+1,k} < A_{j,k}$ אז לכל עמודה $\ell > k$ המינימום נמצא בשורה $i+1$

$\Leftarrow$ ה-BP (j,k) מתבטל

אם $A_{i+1,k} \geq A_{j,k}$ אז לכל עמודה $\ell < k$ המינימום אינו בשורה $i+1$.

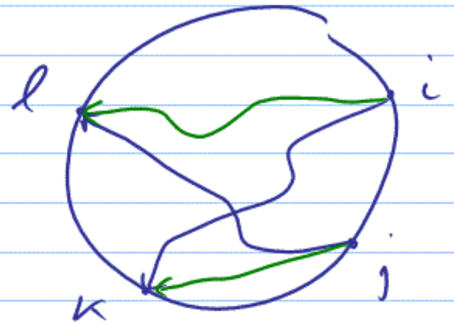
$\Leftarrow$ אפשר להפסיק להשוות

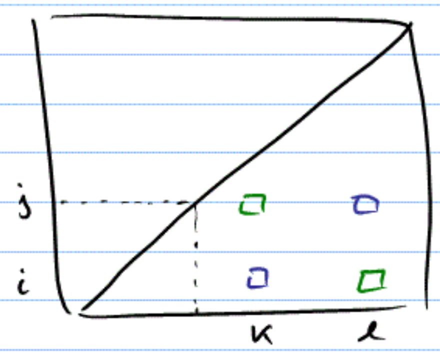
אם (j,k) ה-BP הראשון שנרד, צריך למצוא את העמודה בה
שורה i מפסיקה להיות המינימום ושורה j מתחילה להיות המינימום.
ניתן לעשות זאת ע"י חיפוש בינארי.

<u>אנליזת זמן:</u>

בכל פעם שמוסיפים שורה מוסיפים לכל היותר BP אחד.
לכן, מספר ה-BP הכולל שנוסיף לכל היותר m
כל השוואה שנעשית, פרט לאחרונה בכל איטרציה, מורידה BP אחד.
לכן מספר ההשוואות הכולל הוא לכל היותר $2m$.
בנוסף, בכל איטרציה עושים חיפוש בינארי על לכל היותר n איברים,
כלומר בזמן $O(\log n)$.
סה"כ $O(m \log n)$ למציאת ה-BP של מטריצה עם m שורות ו-n עמודות

מה הקשר לבעיה שלנו ?

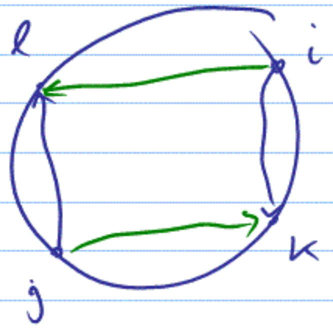


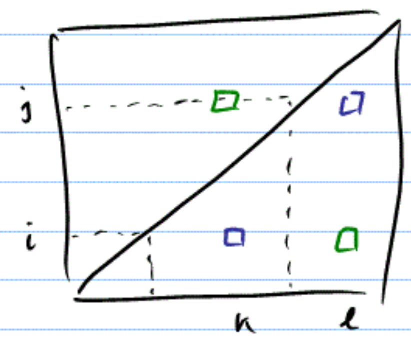


$i < j < k < l$

$$\begin{array}{ccccccc} A(i,k) & + & A(j,l) & \geq & A(i,l) & + & A(j,k) \\ +B(i) & & +B(j) & & +B(i) & & +B(j) \end{array}$$

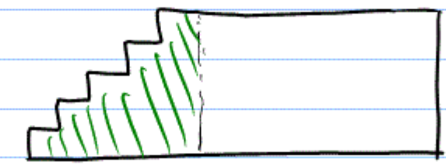
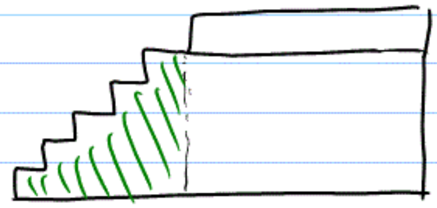
$$A'(i,k) + A'(j,l) \overset{?}{\geq} A'(i,l) + A'(j,k)$$





???

המטריצה שלנו אינה Monge , אבל כל מטריצה משולשת
הינה Monge .

לשורה ה $(i+1)$ אין כל השפעה על ה- BPs במשולש הריק.
שאר המטריצה ריבועית, ולהוסיף שורה למטריצה ריבועית אנו כבר יודעים.

כלומר, אנחנו יודעים למצוא את ה- BPs של מטריצת Monge משולשת ב $O(n\log n)$ זמן.

$\Leftarrow$ יודעים למצוא את האיבר המינימלי בכל עמודה של מטריצת Monge משולשת ב- $O(n\log n)$ זמן

$\Leftarrow$ יודעים למצוא את המינימום בכל עמודה של A'_i ע"י חלוקה לשתי מטריצות משולשות, והשוואת שני האיברים המינימליים בכל עמודה ב $O(n\log n)$ זמן.

כאן n הוא מספר השורות/עמודות במטריצה. כלומר, גודל הספרטור C כלומר $O(\sqrt{n})$.

מציאת כל המינימות בעמודות שקולה לאיטרציה אחת של BF.
מספר האיטרציות הוא כמספר הקדקודים (שהם $\sqrt{n}$), כלומר סה"כ זמן הריצה של BF הוא

$$\sqrt{n}\cdot\sqrt{n}\log(\sqrt{n}) = O(n\log n)$$

וזמן הריצה של האלגוריתם כולו:

$$T(n) = T(n_1) + T(n_2) + O(n\log n) = n\log^2 n$$

מהו צוואר הבקבוק האמיתי של האלגוריתם?

- MSSP $O(n \log n)$
- BF $O(n \log n)$ $(n\alpha(n)$ אפשר$)$
- Dijkstra $O(n)$

כלומר, MSSP הוא צוואר הבקבוק.

הזכרנו כבר של - MSSP יש חסם תחתון של $O(n\log n)$.
האם אין תקווה לשפר?

1 $\Leftarrow$ אם הצמצום של MSSP ו- BF אינו מאוזן, יכולים להרשות להריץ יותר BF.

במקום לחלק לשניים, משתמשים ב- r-division עם
$r = \frac{n}{\log n}$ (במפריד רגיל $r = \frac{3n}{4}$)

כל איטרציה עדיין לוקחת $n \log n$ (עכשיו MSSP ו BF מאוזנים)
אבל עומק הרקורסיה הוא רק

$$\log_{\log n}(n) = \frac{\log n}{\log\log n}$$

וזמן הריצה יהיה

$$O(n \log^2 n / \log\log n)$$

2) החסם התחתון של MSSP אינו תופס לחישוב המרחקים
רק בין הקדקודים של הפאה האינסופית

?

חסם תחתון חזק יותר

אלגוריתם לינארי לבעיה

Dijkstra על ה- Dense Distance Graph

Note Title

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.

$$G' = \bigcup_i G_i'$$

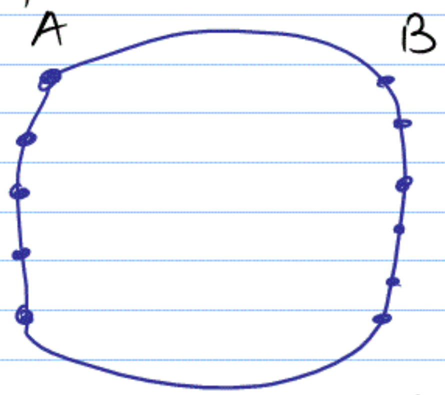
ראינו בשיעור הקודם איך עושים Bellman-Ford על G'.
אם ל-G' N קדקודים, מימוש נאיבי של BF לוקח $O(N^3)$
זמן. ראינו איך משתמשים בתכונת Monge כדי להשיג
זמן ריצה של $O(N^2 \log N)$.

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 לקרוא את כל הקשתות של G'!)

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



מסלולים מכוונים מ-A ל-B.

כלומר: Activate תמיד פועל

על קדקד ב-A, ! ExtractMin

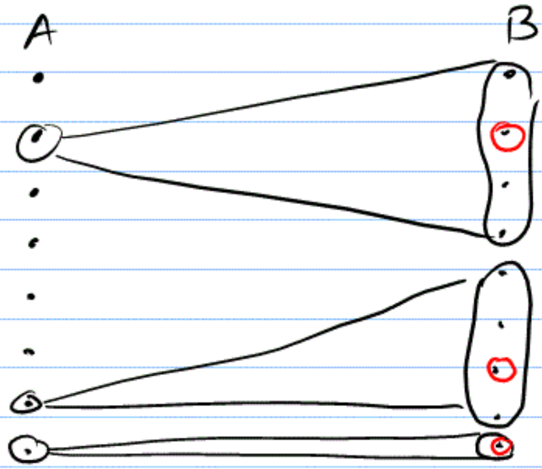
תמיד מחזיר קדקד ב-B.

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'.

הרעיון: לא חייבים לעשות רלקסציה לכל הקשתות של ע.

מספיק לדעת מיהו הקדקד ב-B שהמרחק אליו מינימלי.



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

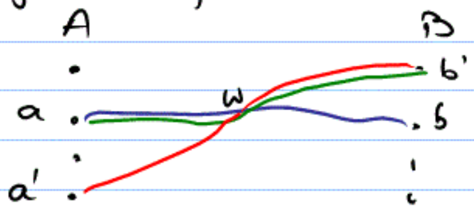
$$a = \operatorname*{argmin}_{a' \in A} d(a') + c(a'b)$$

נשמור לכל $a \in A$ שעשו 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 בשעור הקודם



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

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

מ- $d(a')$ + אורך המסלול האדום.

כלומר a הורה של b'. $\square$

<u>תולדה</u> - קבוצת הילדים 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 ניצג באמצעות שלשות (a,b_1,b_2)

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