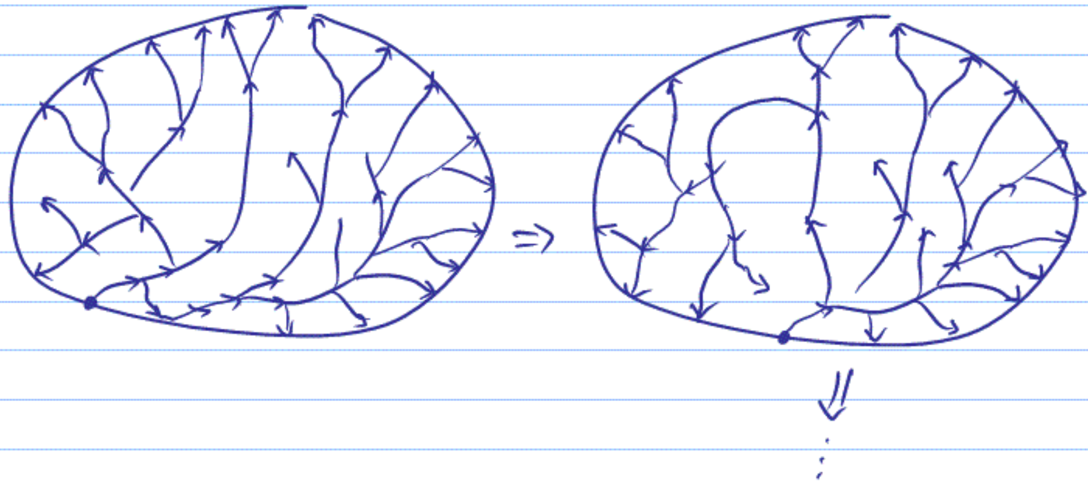
Note Title

Multiple Source Shortest Paths

האלגוריתם שנלמד בשיעור הזה שימושי בפני עצמו ומרכיב באלגוריתמים שנלמד מאוחר יותר.

הבעיה: נתון גרף מישורי מכוון ומשוקלל G, עם אורכים c לחיצים, פאה כלשהי של הגרף f_∞, קדקד v_0 על f_∞ ועץ מרחקים קצרים T_0 מושרש ב-v_0.
רוצים לחשב ייצוג של אוסף עצי המסלולים הקצרים ביותר המושרשים בכל קדקד של f_∞.

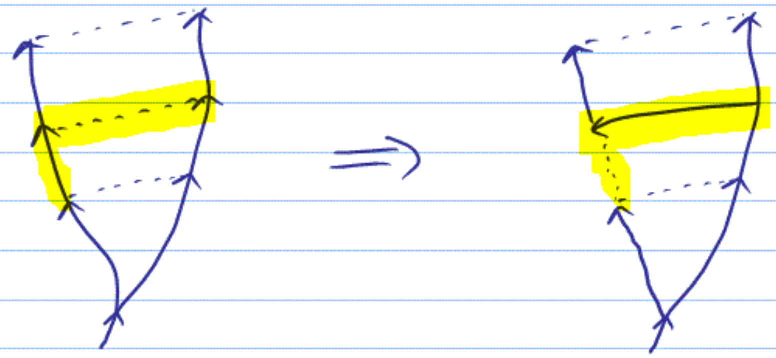
זמן הריצה שנקבל יהיה $O(n\log n)$



לכאורה נראה ש-$O(n\log n)$ זה מעט מדי. בכל עץ יש $n-1$ קשתות, ומספר הקדקודים על f_∞ עלול להיות גדול ($\Theta(n)$), כך שרק לכתוב את כל העצים עלול לקחת $\Omega(n^2)$ זמן.

הייצוג שלנו יהיה של השינויים בלבד. יהיו $v_0, v_1, \dots, v_k$ הקדקודים של f_∞ בסדר ציקלי, ונסמן ב-T_i את עץ המרחקים הקצרים שמושרש ב-v_i. אנו נייצג רק את החיצים שצריך להוציא מ-T_i ולהוסיף ל-T_i כדי לקבל את T_{i+1}, ונראה שמספר החיצים הללו חסום לכל היותר מספר החיצים בגרף.

כדי לשמור על כך שבכל עת יש לנו עץ פורש T של הגרף, יחידת השינוי הבסיסית שלנו תהיה כזו שמכניסה קשת d ל-T ומוציאה את הקשת d' עם אותו ראש כמו d



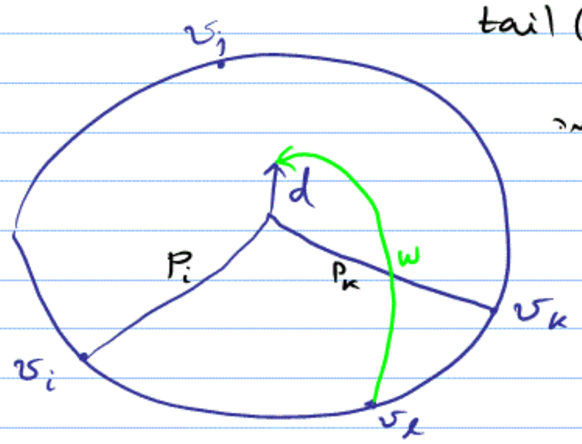
לשינוי כזה קוראים פיבוט (pivot). האלגוריתם מחזיר את סדרת הפיבוטים שהופכת עץ אחד למשנהו.

יש פיבוט מיוחד - כזה שמשנה את השורש של העץ. בצעד זה מכניסים לעץ T_i את הקשת $v_i v_{i+1}$, ומוציאים מ-T_i את הקשת שנכנסת ל-v_{i+1}. כעת v_{i+1} הינו השורש.

לפני שנציג את האלגוריתם, נראה שמספר הפיבוטים הוא לכל היותר מספר החיצים ב-G. לשם כך נניח שמסלולים קצרים ב-G הם יחידים (כלומר, לכל זוג קדקדים u, v יש בדיוק מסלול קצר ביותר אחד מ-u ל-v ב-G).
ניתן להשיג הנחה זו ע"י פרטורבציה קטנה רנדומלית של האורכים, אבל נראה בהמשך שההנחה אינה נחוצה.

משפט- לכל חץ d, קבוצת השורשים של עץ המרחקים הקצרים T_i שמכילים את d הוא תת-סדרה רציפה של המחזור $(1\ 2\ 3 \dots k)$

הוכחה: נניח בשלילה שקבוצת השורשים אינה תת סדרה רציפה. אזי קיימים שורשים v_i, v_j, v_k, v_l בסדר ציקלי זה כך ש $d \in T_i$, $d \notin T_j$, $d \in T_k$, $d \notin T_l$



יהי P_α המסלול מ-v_α ל-$tail(d)$ ב-T_α.
בה"כ המצב כמו המתואר בדיאגרמה (המקרה השני דומה).
יהי C המעגל המוגדר ע"י P_k, $rev(P_i)$, והקשתות של f_∞ בין v_i ו v_k שעוברות דרך v_l.
החץ d אינו מוקף ע"י C.

יהי Q המסלול מ-v_l ל-$head(d)$ ב-T_l. החץ הראשון של Q מוקף ע"י C נובע ש-Q חייב לחצות את C בקדקד כלשהו w של P_k או P_i. בה"כ $w \in P_k$.
נכתוב את P_k כ-$P_k^1 \circ P_k^2$ כאשר P_k^1 מסתיים ב-w ו- P_k^2 מתחיל ב-w.
נכתוב את Q כ-$Q^1 \circ Q^2$ 〃 Q^1 〃 〃 ו- Q^2 〃 〃

המסלול $P_k^2 \circ d$ הוא מסלול קצר ביותר מ-w ל-$head(d)$, וכך גם המסלול Q^2, בסתירה לדרישת היחידות של מסלולים קצרים ביותר $\square$

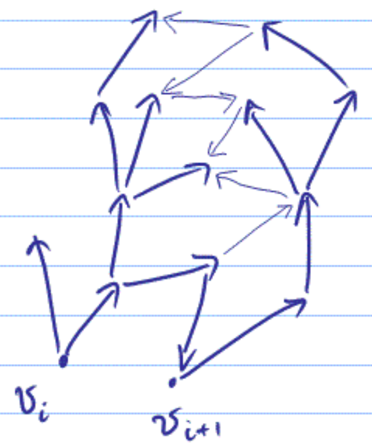
נובע שכל חץ משתנה כחץ שנכנס לעץ בפיבוט אחד בלבד במהלך כל הריצה של האלגוריתם. כלומר, מספר הפיבוטים הוא לכל היותר מספר החיצים ב-G.

בהמשך נציג את האלגוריתם בפירוט ונראה שבאמצעות מבני נתונים מתוחכמים ניתן למצוא את הפיבוטים הקרובים ולבצע אותם בזמן amortized $O(\log n)$ לפיבוט.

העץ שנחזיק במהלך האלגוריתם יהיה בכל עת עץ מרחקים קצרים.
נעבור בצעדים מ-T_i, העץ המושרש ב-v_i, ל-T_{i+1} העץ המושרש ב-v_{i+1}. האלגוריתם יבצע מעבר כזה, החל ב-T_0 וכלה ב-T_k.

נתבונן ב-T_i. זהו עץ מרחקים קצרים שורשו v_i. נסמן את המרחק מ-v_i ל-v_{i+1} ב-c_0.
כדי להפוך את T_i לעץ המושרש ב-v_{i+1}, נכניס לעץ את החץ $d = v_{i+1}v_i$ ונוציא את החץ של T_i שראשו v_{i+1}.
העץ המתקבל T אינו עץ מרחקים קצרים ביותר. אך אם נשנה את ארכו של החץ d להיות $-c_0$, הוא כן עץ מרחקים קצרים ביותר (הוכחה בהמשך).

T_i :



T :

דמיינו שמגדילים בזמן רציף את האורך של החץ d כדי להחזירו לערכו האמיתי. כשעושים זאת, אורכם של המסלולים בעץ T אל קדקדים בתת העץ של v_i (מסומנים בירוק) גדל. אורכם של מסלולים שאינם משתמשים בחץ d, אינו משתנה.

נתמקד בקדקד ירוק u. בהתחלה, כאשר ארכו של d הוא $-c_0$, המסלול הירוק הוא המסלול הקצר ביותר ל-u. כשמגדילים את ארכו של d, המסלול הירוק מתארך, ובאיזשהו שלב יעשה ארוך כמו המסלול הצהוב שעובר דרך החץ a (a אינו שייך לעץ T). ברגע זה כדאי להכניס את a ל-T.

זהו פיבוט - a נכנס ל-T והחץ הירוק שראשו $head(a)$ יוצא מ-T.

איך מגלים אילו חיצים צריכים להכנס ל-T ומתי?

נמדוד לכל חץ uv שאינו ב-T, בכמה המסלול $uv \circ T[u]$ ארוך מהמסלול $T[v]$ ($T[w]$ מסמל את המסלול משורש ל-w בעץ T).

נקרא לגודל הזה העודף (slack) של uv ביחס ל-T ונסמנו $c_\rho(uv)$ כאשר נסמן ב-$\rho(w)$ את אורכו של המסלול $T[w]$

$$c_\rho(uv) = c(uv) + \rho(u) - \rho(v)$$

מהו העודף של חץ ששייך ל-T? מהו העודף של חץ ביחס לעץ מרחקים קודם?
כיצד משתנה $c_T(uv)$ כאשר מאריכים את החץ v_i, v_{i+1} בשיעור Δ?

אם u ! v צאצאים של v_i ב-T — אינו משתנה

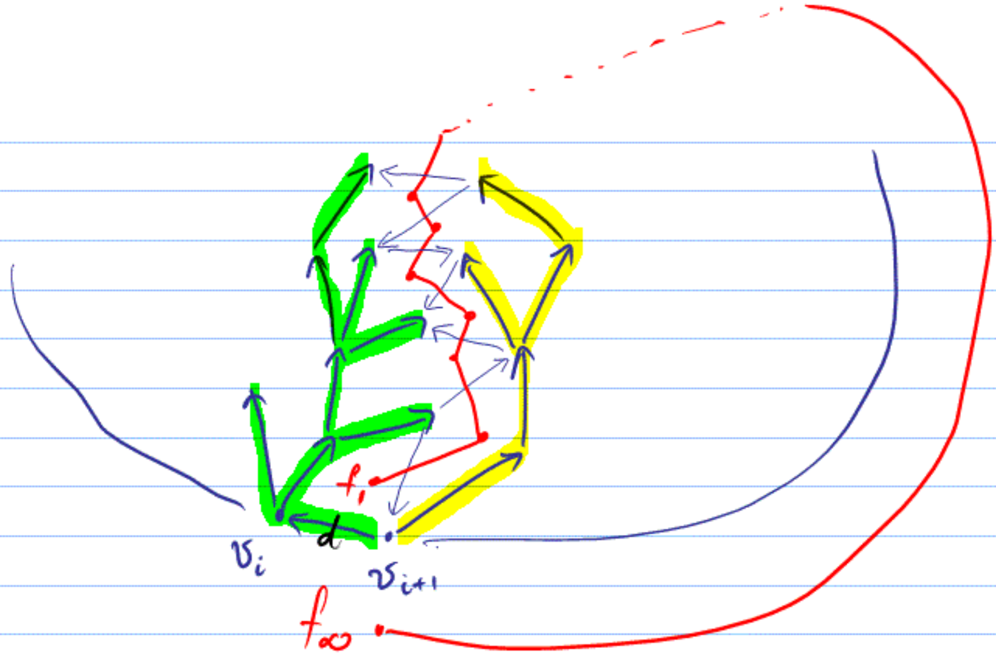
אם u ! v אינם צאצאים של v_i ב-T — אינו משתנה

אם u צאצא של v_i ! v לא — גדל ב-Δ

אם u לא צאצא של v_i ! v צאצא של v_i — קטן ב-Δ

מתי צריך להכניס את uv לעץ? כאשר העודף שלו נעשה אפס.
מכיוון ש-T עץ קצר ביותר, העודף של כל קשת שאינה ב-T אינו שלילי.
לכן, מספיק לעקוב אחרי החיצים שזנבם בתת העץ של v_{i+1} וראשם בתת העץ של v_i. אלו בדיוק החיצים במעגל הבסיסי של $d=v_iv_{i+1}$ ביחס ל-T^*!

או במילים אחרות, אלו החיצים במסלול מפאה $f_1 = head_{G^*}(d)$ לפאה $f_\infty = tail_{G^*}(d)$.



כל שעלינו לעשות, אם כן, הוא לעקוב אחרי העודפים של החיצים במסלול מ-f_1 ל-f_∞ ב-G^*. כל החיצים הללו אינם ב-T, כלומר הם ב-T^*.

העודף של כל החיצים הללו יורד ב-Δ כאשר מגדילים את האורך של d ב-Δ. לכן, החץ הראשון שהעודף שלו יגיע לאפס הוא זה בעל העודף המינימלי. כלומר כדי לבצע פיבוט צריך:

1) מצא את החץ בעל עודף מינימלי במסלול P מ-f_1 ל-f_∞ ב-T^*. יהי $\hat{d}$ החץ, ו-Δ העודף שלו

2) הורד Δ מהעודף של כל החיצים ב-P
הוסף Δ לעודף של ה-rev של כל חץ ב-P.
(הוסף Δ לאורכו של החץ d)

3) הוצא את החץ של T שראשו כ-$head(\hat{d})$ מ-T
והכנס את $\hat{d}$ ל-T.

עד מתי? כל עוד d לא חזר לאורך המקורי שלו.

<u>למה</u> - תמיד T הוא עץ מרחקים קצרים ביותר לאורך כל ריצת האלגוריתם.

<u>הוכחה</u> - נראה הטענה בכל שלב בין שני סיבובים מוחדים
באינדוקציה על הצעדים של האלגוריתם. לבסיס אנו לראות ש-T הוא עץ מרחקים
קצרים מיד אחרי הסיבוב הראשון שמכניס ל-T_i את הצלע $d = v_{i+1}v_i$ ואחרי
שמשנים את אורכו של d להיות $-c_0$ (להזכירכם, c_0 הוא המרחק מ-v_i ל-v_{i+1})

נניח בשלילה כי קיים קדקד u עבורו $dist(v_{i+1},u)$ קטן מהמרחק $dist_T(v_{i+1},u)$
u אינו צאצא של v_{i+1} ב-T_i כי אחרת T_i אינו עץ מרחקים קצרים.
לכן המסלול מ-v_{i+1} ל-u ב-T מתחיל בצלע d וממשיך במסלול מ-v_i ל-u ב-T_i.
כלומר:
$$dist_T(v_{i+1},u) = length(d) + dist(v_i,u)$$
אבל הנחנו
$$dist(v_{i+1},u) < dist_T(v_{i+1},u) = -dist(v_i,v_{i+1}) + dist(v_i,u)$$
כלומר אי שוויון המשולש למרחקים קצרים מופר
$$dist(v_i,v_{i+1}) + dist(v_{i+1},u) < dist(v_i,u)$$
בסתירה.

צעד האינדוקציה: נניח ש-T עץ מרחקים קצרים, ונראה שהוא כזה גם אחרי סיבוב.
מכיון ש-T עץ מרחקים קצרים, העודפים של כל הצלעות שאינן ב-T אינם שליליים.
האלגוריתם מגדיל את אורכו של d עד שהעודף של אחת כלשהי d' הופך להיות אפס.
כעת סיבוב שמכניס את d' ל-T לא ישנה את היותו של T עץ מרחקים קצרים משום
שהעודף של d' הוא אפס.

□

מימוש יעיל

מהן הפעולות בהן הייצוג שלנו צריך לתמוך?

הפעולה היחידה שמבצעים על העץ הפרימאלי T היא הוצאה והכנסה של חיצים.

לכן מספיק לייצג את T על ידי טבלה שאומרת, לכל קדקד v, מיהו ההורה של v ב-T.

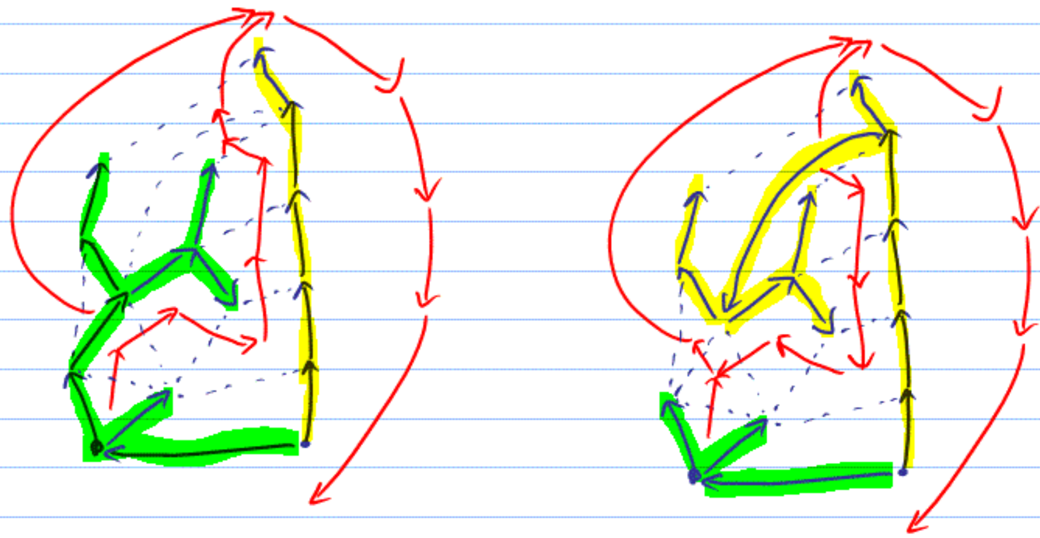
ב-T^* המצב מסובך יותר. מלבד פיבוטים, אנו צריכים לנהל גם את העומקים של הקשתות ב-T^*. הפעולות הנדרשות:

- מצא את הצלע בעלת עומק מינימלי במסלול מקדקד (f_1) לשורש (f_∞)
- הוסף קבוע לכל החיצים במסלול מקדקד לשורש
- הוסף קבוע לכל החיצים במסלול משורש לקדקד

כדי לבצע פיבוט צריך:

- לחתוך עץ לשני עצים (להוציא את $\hat{d}$ מ-T^*)
- לשנות את השורש של עץ
- לחבר שורש של עץ אחד כבן של קדקד של עץ אחר (להכניס חץ ל-T^*)

dynamic trees תומכים בכל הפעולות הללו ב-$O(\log n)$ זמן amortized.



<u>טיפול במרחקים מנוונים</u> -

נגדיר קריטריון לבחירת החץ עם זנב מינימלי במקרה שיש יותר מאחד.
בחר את החץ הקרוב ביותר לעלה של T^* (leafmost).

<u>למה</u> - לכל שורש v_i ולכל קדקד u קיימים לאורך ריצת
האלגוריתם לכל היותר שני מסלולים שונים מ-v_i ל-u.
אם קיימים שניים, המוקדם מביניהם מתחיל בחץ $a=v_iv_{i-1}$
והמאוחר אינו עובר ב-v_{i-1}.

הוכחה - נתרכז בקבוצת העצים שמושרשים ב-v_i. יהי T עץ כזה.

האלגוריתם מעניק ל-T חץ $\hat{a}$ שראשו צאצא של v_{i-1}
ב-T וזנבו לא. קבוצת הקדקודים A שהמסלול אליהם משתנה
בעקבות הסיבוב שמעניק את $\hat{a}$ ל-T היא בדיוק הצאצאים
של $head(\hat{a})$ ב-T. לפני הסיבוב כולם צאצאים של
v_{i-1} ואחרי הסיבוב אינם. מרגע זה המסלול לאף קדקד ב-A
לא ישתנה* משום שרק מסלולים לקדקודים שהם צאצאים של v_{i-1}
ב-T משתנים. □

* כל עוד v_i הוא השורש.

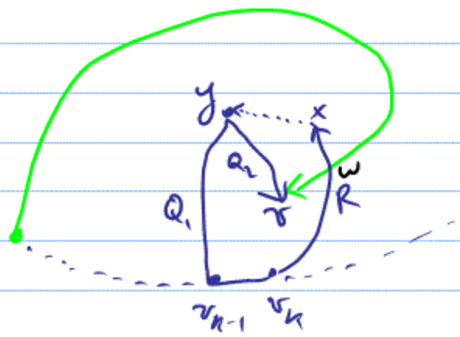
מהמשפט הבא נובע שכל חץ נכנס לעץ T פעם אחת לכל היותר.

<u>משפט</u>: יהי P_i^v המסלול ב-T מהשורש ל-v אחרי i סיבובים. המסלולים $P_0^v, P_1^v, P_2^v, \ldots$ אינם חותכים זה את זה.

<u>הוכחה</u>: נניח בשלילה כי הטענה שגויה. יהי j הטבעי המינימלי כך ש-P_j^v חותך את P_i^v עבור קדקד כלשהו v ועבור $j<i$.
P_j^v מתקבל מ-P_{j-1}^v ע"י סיבוב. הסיבוב אינו מיוחד כי סיבוב מיוחד לא יוצר חיתוכים. נסמן ב-T את העץ לפני הסיבוב (P_{j-1}^v מסלול ב-T).
נסמן ב-xy את החץ שנכנס ל-T בסיבוב. נכתוב $P_{j-1}^v = Q_1 \circ Q_2$
כאשר Q_1 נגמר ב-y ו-Q_2 מתחיל ב-y.
נסמן ב-R את המסלול ב-T ל-x. כלומר $P_j^v = R \circ xy \circ Q_2$

מהמינימליות של j, P_i^v, Q_1, Q_2 אינם נחתכים.
מכאן ש-P_i^v חייב לחתוך את R.

נניח תחילה כי P_i^v חותך את R מספר אי-זוגי של פעמים.



מקרה א': P_i^v מתחיל מחוץ למעגל
$R \circ xy \circ rev(Q_1)$

נכתוב את P_i^v כ $S_1 \circ S_2$, כאשר S_2 מתחיל בקדקד האחרון של P_i^v ששייך גם ל-R. לפחות חץ אחד d' של S_2 אינו ב-T, אחרת קיים מעגל ב-T.
בנוסף, S_2 הוא מסלול קצר ביותר, כלומר d' בעל עודף אפס יקרוב יותר לעלה של T^* מ-xy, בסתירה לבחירה של xy.

<u>מקרה ב'</u> - P_i^v מתחיל בתוך המעגל Roxyorev(Q_1)

P_i^v ו- P_{j-1}^v שני מסלולים ב-T

המושרשים ב- v_i והמובילים ל- v, אבל אז

P_j^v מסלול שאינו כזה, בסתירה ללמה הקודמת.

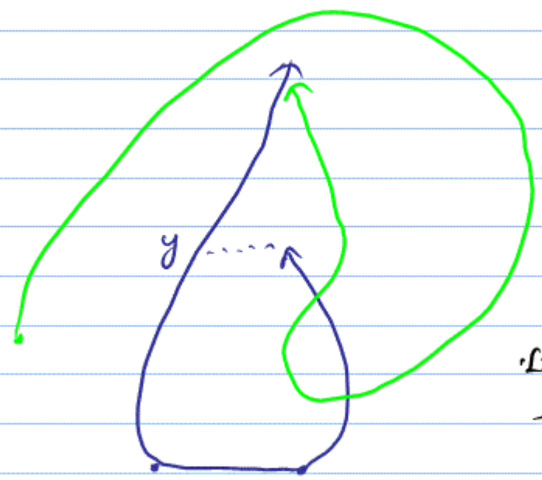
P_i^v מוקדם מ- P_{j-1}^v. מהלמה הקודמת, חייב

לעבור לאורך הפאה האינסופית עד v_{k-2},

בסתירה להנחה שמתחיל בתוך המעגל ואינו חוצה

את P_{j-1}^v.

<u>מקרה ג'</u>: P_i^v חוצה את R מספר
זוגי של פעמים.



נכתוב את R כ- $R_1 \circ R_2$, כאשר
R_1 מסתיים בקודקוד האחרון של P_i^v
ששייך גם ל- R.
אחד החיתוכים הקודמים של P_i^v עם
R חייב להיות בקודקוד מוקדם יותר של
R_1 (אחרת P_i^v לא יוכל להגיע ל- v בלי
לחצות את עצמו), בסתירה למינימליות
של j.

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, החיצים של v מחלקים את הקדקדים של f_∞ לאינטרוולים רציפים.
נשמור לכל קדקד v ייצוג של האינטרוולים הללו. בהינתן קדקד v ושורש r, הייצוג מחזיר את החץ d של v שהאינטרוול שלו מכיל את r.
כלומר, אנו יודעים מיהו 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 האינטרוולים הללו, כל שאילתא היתה לוקחת $O(\log|f_\infty|) = O(\log n)$. מכיוון שהאינטרוולים הם של שלמים שגדולים לכל היותר $|f_\infty|<n$, ניתן לעשות זאת בזמן $O(\log\log n)$
[Van Emde Boas 77, Melhorn & Naher 90]
הזמן לשחזר 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 הצלעות של קדקד, ושאילתות 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 $O(k \log n)$ לטיפול ב k שאילתות לעץ הדינאמי של T.

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

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).