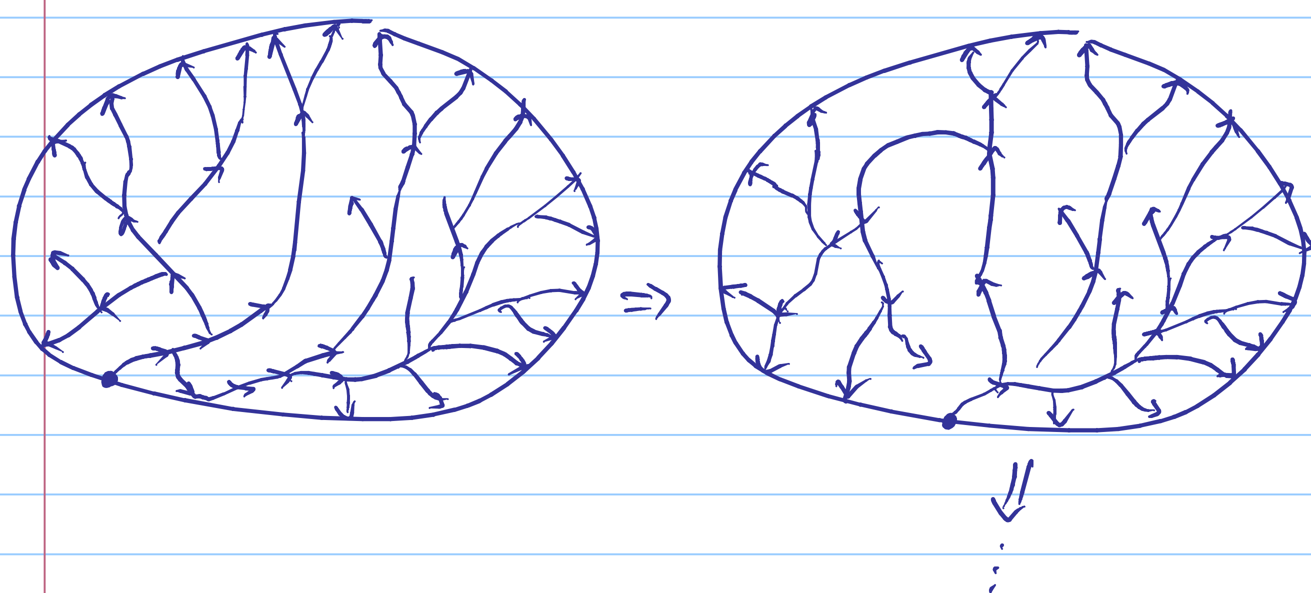
Multiple Source Shortest Paths

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

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

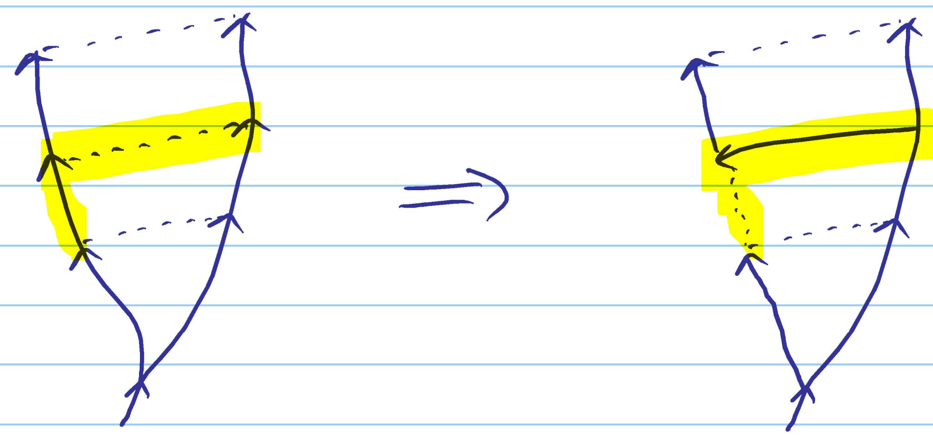
זמן הריצה שנשיג כאן הוא $O(n\log n)$



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 על השינויים בלבד. יהיו $v_0, v_1, \dots, v_k$ הקדקודים על 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 הוא 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 חץ d ל-T ומוציאה את החץ 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 החץ v_i, v_{i+1}, ומוציאים מ-T_i את החץ שנכנס ל-v_{i+1}. כעת v_{i+1}
הנו 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 קדקודים
u, v יש בדיוק מסלול קצר ביותר אחד מ-u ל-v ב-G).
ניתן להשיג הנחה זו ע"י פרטורבציה קטנה רנדומית 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 $(1\ 2\ 3\ ...\ k)$

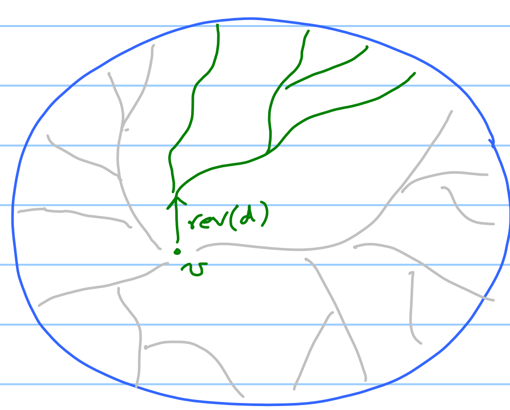
__הוכחה__ – יהי T' עץ מרחקים קצרים מושרש בקדקד v.

כאשר משתמשים בחיצים בכיוון ההפוך (כלומר, $c'[d] = c[rev(d)]$).

לכל קדקד $r \in f_\infty$, אם d הוא החץ האחרון במסלול הקצר ביותר מ-r ל-v, אז החץ הראשון במסלול מ-v ל-r ב-T' הוא $rev(d)$.

כלומר, r הוא צאצא של $tail(d)$ ב-T'.

מכיוון ש-T' אינו חוצה את עצמו, הקבוצה $\{r_i \in f_\infty : r_i \in T'_{tail(d)}\}$ היא סדרה רציפה של קדקדים של הפאה f_∞ $\square$



הצאצאים של $rev(d)$ מהווים סדרה רציפה של קדקדים של f_∞.

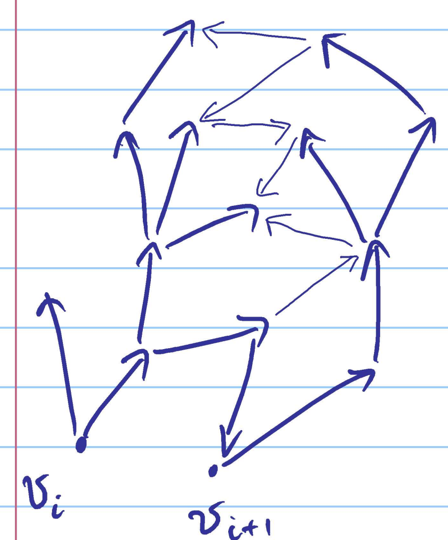
נובע שכל חץ משתתף כחץ שנכנס לעץ בפיבוט אחד בלבד במהלך כל הריצה של האלגוריתם. כלומר, מספר הפיבוטים הוא לכל היותר מספר החיצים ב-G.

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

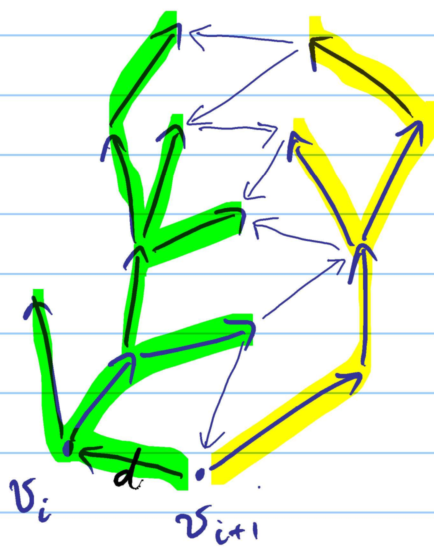
העץ שנחזיק במהלך האלגוריתם יהיה בכל עת עץ של מרחקים קצרים.
נראה כיצד עוברים מ-T_i, העץ המושרש ב-v_i, ל-T_{i+1} העץ המושרש ב-v_{i+1}. האלגוריתם יבצע מעבר כזה, החל ב-T_0 וכלה ב-T_k.

נתבונן ב-T_i. זהו עץ מרחקים קצרים שמושרש ב-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 הקשת $d = v_{i+1}v_i$ ונוציא את הקשת של T_i שראשה v_{i+1}.
העץ המתקבל T אינו עץ מרחקים קצרים ביותר. אך אם נשנה את אורכה של הקשת d להיות $-c_0$, הוא כן עץ מרחקים קצרים ביותר (הוכחה בהמשך).

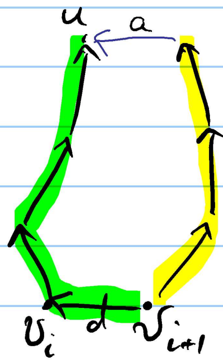
T_i :



T :



דמיינו שמגדילים בהדרגה את האורך של הקשת d כדי להחזירו לערכו האמיתי.
כשעושים זאת, אורכם של המסלולים בעץ T אל קדקודים בתת העץ של v_i (מסומנים בירוק) גדל. אורכם של מסלולים שאינם משתמשים בקשת d, אינו משתנה.



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

זהו פבול - 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 $T[u]\circ uv$ 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 $c_p(uv)$ כאשר נסמן ב-$\rho(w)$ את אורכו של המסלול $T[w]$

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

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

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

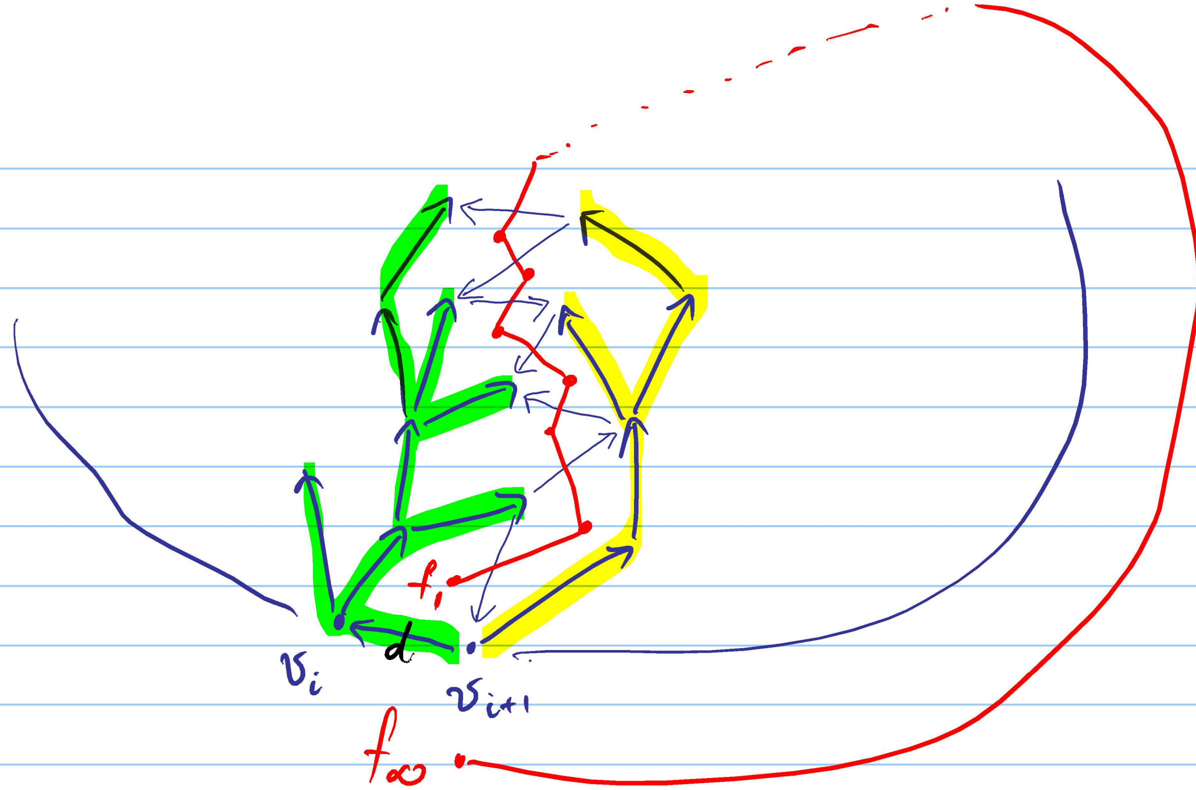
אם u ו-v אינם צאצאים של v_i ב-T — אינו משתנה

אם u צאצא של v_i ו-v לא — גדל ב-Δ

אם u לא צאצא של v_i ו-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 $d=v_{i+1}v_i$ ביחס ל-T^*!

או במילים אחרות, אלו החיצים במסלול מהפאה $f_1 = head_{G^*}(d)$ 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 ב-$\text{head}(\hat{d})$ מ-T והכנס את $\hat{d}$ ל-T.

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

למה - העץ T הוא עץ מרחקים קצרים ביותר לאורך כל ריצת האלגוריתם.

הוכחה - נראה הטענה בכל שלב בין שני פיבוטים מיוחדים באינדוקציה על הצעדים של האלגוריתם. לבסיס עלינו להראות ש-T הוא עץ מרחקים קצרים מיד אחרי הפיבוט המיוחד שמכניס ל-T_i את הקשת $d = v_{i+1}v_i$ 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 בקשת 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 פיבוט.
מכיוון ש-T עץ מרחקים קצרים, המחירים של כל הקשתות שאינם ב-T אינם שליליים.
האלגוריתם מגדיל את אורכו של d עד שהעודף של קשת כלשהו d' הופך להיות אפס.
כעת פיבוט 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.

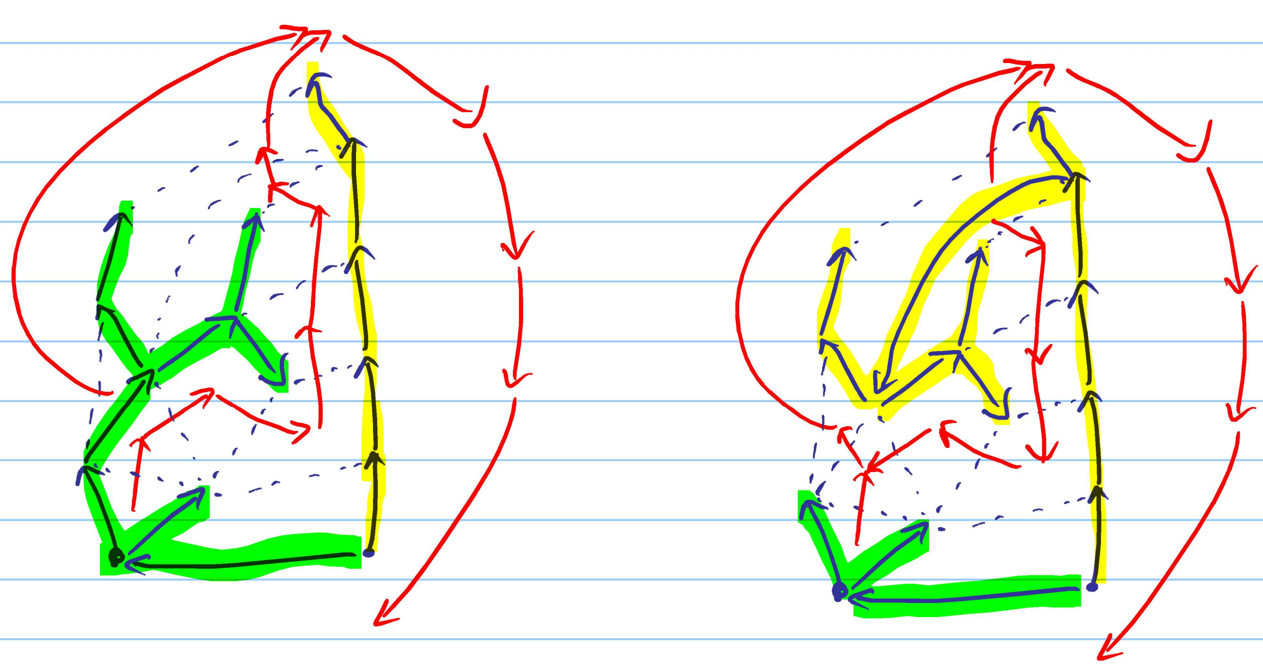
לכן מספיק לייצג את T ע"י טבלה שאומרת, לכל קדקד v, מיהו ההורה של v ב-T.

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

- מצא חץ בעל משקל 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 $\hat{a}$ 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.



כיצד משתמשים ב- MSSP

- מסלולים קצרים ביותר.

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

כלומר, החיצים של v מחלקים את הקודקודים של f_∞ לאינטרוולים רציפים. נשמור לכל קודקוד v ייצוג של האינטרוולים הללו. בהינתן קודקוד v ושורש r, הייצוג מחזיר את החץ d של v שהאינטרוול 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 האינטרוולים הללו, כל שאילתה היתה לוקחת $O(\log|f_\infty|) = O(\log n)$. מכיוון שהאינטרוולים הם של שלמים שגדולים לכל היותר n, $|f_\infty|<n$, ניתן לעשות זאת בזמן $O(\log\log n)$
[Van Emde Boas 77, Melhorn & Naher 90]
הזמן לשחזור מסלול באורך l יהיה $O(l \log\log n)$.

- מרחקים

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 לשמור גם את העץ T בעץ דינאמי שתומך ב Link, Cut, הוספה קבוע לכל הצאצאים של קודקוד, ושאילתת משקל של קודקוד.
כאשר מקבלים את T_i, שואלים מהם המרחקים לכל הקודקודים בזוגות בהם r_j הוא השורש של T_i.
הזמן: $O(n\log n)$ להרצת MSSP ועוד $O(k\log n)$ עבור k שאילתות לעץ הדינאמי של T.

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

כדי לשמור את ההיסטוריה של האלגוריתם בצורה "נגישה".
סיבוכיות המקום עולה ל- $O(n \log n)$, אבל אז אפשר לגשת לכל
בהיסטוריה של האלגוריתם ב- $O(\log n)$ זמן. כך שבאופן מעשי
ניתן לקבל כל מרחק משורש לכל קדקד ב- $O(\log n)$ זמן (amortized).

הערות:

- ניתן לממש האלגוריתם ללא dynamic trees, בזמן פרופורציוני לסכום המשקולות בגרף.
  למשל - בגרף לא ממושקל, זמן הריצה לינארי.
- קיים חסם תחתון של $\Omega(n \log n)$ ל- MSSP

טיפול במרחקים מנוונים -

נגדיר קריטריון לבחירת הקשת עם עוגן מינימאלי במקרה שיש יותר מאחד.
בחר את הקשת הקרוב ביותר לעלה של T^* (leafmost).

ניתן להוכיח ששימוש בקריטריון זה מבטיח שכל קשת נכנס לעץ
T לכל היותר פעם אחת במהלך הרצת האלגוריתם.