Kiril Solovey

אלגוריתמים – סיכום

ייצוג של גרפים:

1. **ע"י רשימת שכנות**: הצמתים מיוצגים איכשהו ולכל צומת יש מצביע לרשימה מקושרת של שכניו.
   יתרון: חסכוני במקום $O(|V|+|E|)$. חסרון: יקר להגיע לקשת ספציפית.
2. **ע"י מטריצת שכנות**: A יש בה $|V|$ שורות ו-$|V|$ עמודות.
   $A_{ij}=1$ אם קיימת קשת המחברת צומת מס' i לצומת j. 0 אחרת. צורה זו בזבזנית בזכרון. בעקר אם הגרף קליל $\binom{n}{2} \ll |E|$.

מעגל \ מסלול אוילר

<u>משפט</u>: בגרף יש מעגל אוילר אמ"מ דרגת כל קודקוד זוגית.
<u>משפט</u>: בגרף יש מסלול אוילר אמ"מ דרגת כל קודקוד היא זוגית מלבד 2 קודקודים.
<u>טענה</u>: ניתן להפוך גרף שיש בו מסלול אוילר לגרף בעל מעגל אוילר ע"י יצירת קשת בין 2 הקודקודים עם הדרגות האי זוגיות.
<u>טענה</u>: בגרף בעל גשר (קשת שהסרתה תהפוך את הגרף ללא קשיר) אין מעגל אוילר.

BFS

<u>מטרה:</u> נתון גרף $G=(V,E)$ מכוון או לא וצומת התחלה s. רוצים לבקר בצורה יעילה בכל הצמתים הנגישים מ-s וגם לחשב לכל צומת כזה u את אורך המסלול הקצר ביותר מs לu. (אורך=מס' הקשתות)

כל צומת u מחזיק שני ערכים: $\pi[u]$ – מצביע לצומת הקודם במסלול הקצר ביותר מ-s ל-u שהתגלה עד כה;
$d[u]$ - אורך המסלול הקצר ביותר מ-s ל-u שהתגלה עד כה.
כמו כן האלגוריתם משתמש בתור Q. $\delta(s,u)$ – אורך המסלול הקצר ביותר מ-s ל-v או ∞ אם אין מסלול כזה.

<u>תאור האלגוריתם:</u>
במהלך ריצת האלגוריתם הצמתים נצבעים בשלושה **צבעים**:
- לבן – הצומת טרם טופל ועדיין לא ביקרו בו או שהוא לא מחובר לצומת שכבר ביקרו בו.
- אפור – הצומת הוא שכן של צומת אחר שכבר הוצא מן התור.
- שחור – הצומת כבר הוצע מן התור וכבר נבדקו שכניו.

ניתן לחלק את האלגוריתם למספר **שלבים**:
1. איתחול (שורות 1-8) – כל הצמתים נצבעים בלבן ומתעדכנים מרחקי ה-d וה-π שלהם. S הוא הצומת היחיד המוכנס לתור.
2. לולאת ה-$while$ (שורות 9-18): מוציאים צמות u מראש התור. עוברים על כל שכניו ואם מגלים צומת שטרם ביקרנו בו (צבע לבן) אז מעדכנים את הצבע שלו, מרחקו מ-s (הוספת 1 למרחק של u מ-s) ועידכון המסלול (הצומת הקודם במסלול הוא u).
   כל שכן כזה מכניסים לתור ולבסוף צובעים את u בשחור.

<u>טענות לגבי ריצת האלגוריתם:</u>
- אם u נגיש מ-s אז האלגוריתם מבקר בו.
- בכל רגע נמצאים ב-Q צמתים עם לכל היותר 2 ערכי δ עוקבים.
- לכל צומת $d[u]=\delta(s,u)$(יכול להיות גם ∞) בתום ריצת האלגוריתם.
- מצביעי π יוצרים עץ מכוון ששורשו s.

<u>סיבוכיות</u>: $O(|V|+|E|)$.

<u>שימושים</u>:
- נגישות – בדיקה האם צומת כלשהו נגיש מצומת אחר.
- מציאת מרחק קצר ביותר מ-s.
- חלוקה של הגרף לשכבות.
- בדיקה האם הגרף דו צדדי: גרף מכוון הוא דו צדדי $\Leftrightarrow$ אין בו מעגל באורך אי זוגי. נסדר את הגרף לפי שכבות ונבדוק שאין קשתות בין 2 צמתים באותה שכבה.

DFS

האלגוריתם מבקר בכל צומת בגרף, ובניגוד ל-BFS הוא לא סורק את הגרף לפי רמות אלא מנסה להגיע לעומק.
ע"י ערכי ה-π האלגוריתם מגדיר גרף מכוון חסר מעגלים אשר מהווה יער.

<u>תאור האלגוריתם</u>:
צבעים: לבן – טרם בקרנו ב-u, אפור – נתקלנו ב-u אך טרם סיימנו, שחור – סיימנו את הביקור ב-u.

קיים מונה $time$, בכל פעימה שלו נתקלים בצומת חדש או מסיימים טיפול בצומת.
שדות לכל צומת:
- $d[u]$ = הזמן בו נתקלנו ב-u לראשונה.
- $f[u]$ = הזמן בו סיימנו טיפול ב-u.
- $\pi[u]$= הצומת הקודם שגילה את u.

האלגוריתם מתחלק ל-2 תוכניות: DFS התוכנית הראשית שבה מתבצע האתחול ומעבר על כל הצמתים שטרם טופלו (לבן) והפעלת $DFS\text{-}VISIT$ עליהם.

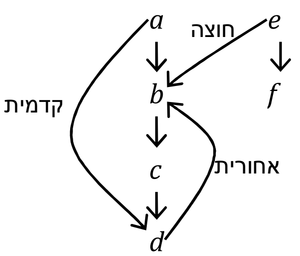
ב-$DFS\text{-}VISIT$ מתעדכן צבע הצומת לאפור (בטיפול) ועל כל אחד משכניו הלבנים מפעילים שוב את $DFS\text{-}VISIT$ רקורסיבית.

בתחילת ריצת ה-$DFS\text{-}VISIT$ מתעדכן ערך ה-d של הצומת ובסוף מתעדכן ערך ה-f.

<u>סיבוכיות</u>: $O(|V|+|E|)$.

<u>עקרון הקינון</u>
לכל צומת u מתאים אינטרוול זמן שבו הוא פעיל $[d[u] \dots f[u]]$. אם ל-2 צמתים אינטרוולים זרים אז אז אחד מהקודקודים צאצא של האחר. במקרה שהאינטרוולים זרים אז אין קשר של צאצא בין 2 הקודקודים.

<u>סיווג קשתות הגרף ע"י ה-DFS</u>
ה-DFS עובר על כל קשתות הגרף. בחלק מהן הוא משתמש להגדרת ה-π לבניית עץ ה-DFS, ובשאר לא.
- קשת קדמית מחברת אב קדמון לצאצא
- קשת אחורית מחברת צאצא לאב קדמון
- קשת חוצה מחברת 2 צמתים שאינם צאצא \ אב קדמון



<u>עקרון המסלול הלבן</u>: נניח שב-G יש מסלול $v_1 \to v_2 \to \cdots v_k$ ונניח שהקדקד הראשון במסלול שה-DFS מגלה הוא v_1. אז כל שאר הצמתים יהיו צאצאים של v_1 ב-DFS.

שימושים:

1. מציאת רכיבי קשירות בגרף לא מכוון (כל עץ הוא רכיב קשירות)
2. האם שני צמתים נמצאים באותו רכיב קשירות? (בדיקה האם שניהם נמצאים באותו עץ)
3. בדיקת קיום מעגל בגרף מכוון \ לא מכוון (בדיקת קשת אחורית)
4. מיון טופולוגי של גרף מכוון **אציקלי** (חסר מעגלים): רוצים לסדר את צמתי הגרף G בסדר כלשהו. מריצים DFS והסדר של הצמתים יהיה סדר $f(\cdot)$ שלהם במהופך, כלומר v_1 יהיה זה עם ערך מקסימלי.
5. מציאת גשרים, קדקודים מנתקים ורכיבי דו קשירות.

מציאת רק"חים:

1. נריץ DFS על G ונסדר את הצמתים לפי סדר $f(\cdot)$ יורד, נקרא לרשימה L.
2. נבנה את הגרף ההפוך G^T.
3. נריץ DFS על G^T כאשר הפרוצדורה הראשית משתמשת ב-L בתור רשימת הצמתים.
4. כל עץ ב-DFS השני הוא רק"ח.

טענה: יהיו c_1, c_2 רק"חים עם קשת-על $c_1 \rightarrow c_2$. אז מבין הצמתים של $c_1 \cup c_2$, זה עם ערך f מקסימלי (מה-DFS הראשון) נמצא ב-c_1.

אלגוריתם למציאת מסלולים קצרים (אורך מסלול = מס' הקשתות) ביותר בין קודקוד s ל-t:

מוצאים את גרף המסלולים הקצרים ביותר מ-s ‎$G`$ (באמצעות BFS).

הופכים את כיווני קשתות הגרף החדש והפעם יוצרים גרף $G``$ שהוא גרף המסלולים הקצרים ביותר מ-t.

שוב נהפוך את כיווני הקשתות של $G``$ והגרף שקיבלנו הוא גרף המסלולים הקצרים ביותר מ-s ל-t.

עצים פורסים מינימליים

הקלט הוא גרף לא מכוון וקשיר $G = (V, E)$ עם משקל על הקשתות: לכל קשת e יש משקל $w(e)$.

עץ פורש: תת גרף של G שמכיל את כל הצמתים הוא עץ המורכב מ- $|V| - 1$ קשתות של E.

תמיד קיים עץ כזה, כי G קשיר והוא לא בהכרח יחיד.

גישה כללית

תת קבוצה A של E נקראת קבוצה מבטיחה אם קיים עפ"מ T המכיל את A.

אם A קב' מבטיחה, ו-e קשת שאינה ב-A, נקרא ל-e קשת בטוחה עבור A אם $A \cup \{e\}$ עדיין מבטיחה.

"אלגוריתם"

$A \coloneqq \emptyset$. כל עוד מס' הקשתות ב-A $< |V| - 1$, מצא קשת e בטוחה עבור A. $A \coloneqq A \cup \{e\}$. החזר את A.

מספר הגדרות:

- חתך בגרף $G = (V, E)$ זהו פרוק של הקודקודים V לשתי קבוצות זרות ולא ריקות V_1, V_2. $V_1 \cup V_2 = V$.
- קשת e חוצה את החתך, אם היא מחברת צומת ב-V_1 לצומת ב-V_2.
- אם A קב' מבטיחה החתך מכבד את A אם אין ב-A קשת חוצה.
- בהנתן חתך, קשת e נקראת חוצה קלה אם משקלה מינימלי מבין כל הקשתות החוצות.

טענה: תהא A קב' מבטיחה ויהא (V_1, V_2) חתך שמכבד את A, ותהא e קשת חוצה קלה של החתך, אז e קשת בטוחה עבור A.

אלגוריתם *Kruskal*

הרעיון: בכל שלב, נחפש קשת שמחברת 2 רכיבי קשירות שונים של A במשקל מינימלי ונוסיף אותה ל-A.

בכדי לממש זאת, האלגוריתם ממיין את הקשתות לפי סדר משקל עולה (חלש) ועובר עליהם בסדר זה.
לכל קשת e נבדוק אם היא מחברת שני רכיבי קשירות שונים. אם לא נתעלם ממנה, ואם כן נוסיף אותה ל-A ונאחד את הרכיבים לרכיב קשירות משותף.

רכיבי הקשירות של A מתוחזקים באמצעות מבנה נתונים המתחזק אוסף קבוצות זרות של V, והפעולות שלו הן *find(u), union(u,v), make-set(u)*.

הפלט של האלגוריתם תלוי אך ורק בסדר המיון של הקשתות.

<u>סיבוכיות</u>: מיון הקשתות – $O(|E|\, log|E|) = O(|E|\, log|V|)$

$$|E| \le |V|^2$$
$$log|E| \le 2log\ \ |V|$$

שאר הפעולות $O(|E|+|V|)$ + זמן מימוש *Union-find*: $Find\ \ 2|E|,\ \ Union\ \ |V|-1$.

<u>תכונות עפ"מים הנובעות מהאלגוריתם</u>

- האלג' מסוגל לייצר כל עפ"מ אפשרי (אם המשקלים שונים אז יש עפ"מ יחיד)
- תהא $f:\mathbb{R}\to\mathbb{R}$ פונ' עולה ממש: $x<y \Leftrightarrow f(x)<f(y)$ ונתון G עם משקלות w על הצלעות. נגדיר משקלות חדשים $w^*(e)=f\big(w(e)\big)$. אזי T עפ"מ לפי $w \Leftrightarrow T$ עפ"מ לפי w^*.
- אם T_1, T_2 עפ"מים לפי (G,w) ואם נסדר את משקלות הקשתות של T_1 בסדר עולה אז: $x_1 \le x_2 \le \cdots \le x_{|V|-1}$ וכנ"ל לגבי $y_1 \le y_2 \le \cdots \le y_{|V|-1}$. אז $x_1=y_1,\dots,x_{|V|-1}=y_{|V|-1}$.

אלגוריתם *Prim*

<u>הרעיון</u>: הפעם אין מספר רכיבי קשירות אלה רכיב קשירות אחד C, ומוסיפים אליו בכל פעם צומת היושב על הקשת הקלה ביותר המחברת אותו ל-C ואז מעדכנים את שכניו.
האלגוריתם משתמש בתור עדיפות $Q=V\setminus C$, כשלכל צומת מפתח המייצג את משקל הקשת הקלה ביותר המחברת אותו ל-C.

<u>סיבוכיות</u>: מבצעים הוצאת מינימום $|V|$ פעמים – $O(|V|\, log|V|)$, הפחתת מפתח * $|E|$ -
$\underbrace{O(|E|)}_{\text{פיבונצ'י}}$ או $\underbrace{O(|E|\, log|V|)}_{\text{ערמה רגילה}}$. לכן **סך הכל**, בערמה רגילה $O(E\cdot logV)$ או בפיבונצ'י $O(V\cdot log\,V+E)$.
במערך לא ממויין העלות היא $O(|V|^2+|E|)$, ואם $|E|=\Theta(|V|^2)$ אז $O(|E|)$.

מסלולים קצרים ביותר

<u>תאור הבעיה:</u>
נתון גרף G (מכוון או לא). המשקלות על הקשתות הם מספרים ממשיים שיכולים להיות גם שליליים.
לזוג צמתים u,v נסמן ב- $\delta(u,v)$ את המשקל המינימלי של מסלול שמחבר את u ל-v.אם אין מסלול בין u ל-v אז $\delta(u,v)=+\infty$.
נתון צומת התחלה מסוים s ונרצה לחשב את כל הגדלים $\delta(s,u)$ לכל צומת u.
צריכים להניח שאין ב-G מעגלים שליליים (מעגל שסכום משקלותיו <0) הנגישים מ-s. מעגלים במשקל אפס אפשריים.

<u>תכונות מק"בים</u>

1. תת מסלול של מק"ב הוא מק"ב.

2. אי שיוויון המשולש: אם $(u,v)\in E$ אז $\delta(s,v)\le\delta(s,u)+w(u,v)$.
3. אם u צומת אחד לפני v על מק"ב ב-s ל-v אז $\delta(s,v)=\delta(s,u)+w(u,v)$.
4. אם כל המשקלות = 1 אז משקל מסלול = מס' הקשתות והבעיה נפתרה ע"י BFS.

האלגוריתמים יתחזקו לכל צומת u שני שדות:
- $d[u]$ = משקל המסלול הקל ביותר מ-s ל-u שהתגלה עד עתה.
- $\pi[u]$ = מצביע לצומת הקודם על המסלול הטוב ביותר שהתגלה עד עתה.

סכמת אלגוריתם:
- איתחול
- מעבר על כל הקשתות בסדר כלשהו
- כל עוד קיימת קשת שפעולת ה-*Relax* עליה משפרת, נבצע את ה-*Relax*. (פעולת ה-*Relax* מקבלת 2 קודקודים *u,v* ובודקת האם ניתן לשפר את ערך ה-*d* של הצומת *v* ע"י השמה של $d[u]+w(u,v)$).
- עוצרים כאשר לא קיים עוד *Relax* משפר.

טענה: בכל רגע שהוא, מצביעי π יוצרים עץ מכוון כלפי השורש (s).

טענה: לאחר ריצת האלגוריתם, נסתכל על העץ G_π שקשתותיו הן מצביעי π, זהו עץ מכוון ששורשו s, וגם:
a. הצמתים שלו הם בדיוק אלה הנגישים מ-*s*.
b. לכל צומת u בעץ, המסלול מ-*s* אליו הוא מק"ב.

סיבוכיות מקום: בניגוד ל-*BFS* הדורש $O(|V|^2)$ מקום, עץ המק"בים דורש רק $O(|V|)$.

חישוב מק"בים בגרף מכוון אציקלי

נבצע מיון טופולוגי של G = סדור כלשהו של הצמתים כך שכל קשת ב-*G* מכוונת מצומת קטן לצומת גדול.
$O(|E|+|V|)$ - עלות המיון הטופ'.
נוכל להתעלם מכל הצמתים הקודמים ל-*s*.
האלגוריתם מבצע אתחול ועובר על הצמתים לפי סדר המיון הטופולוגי ומכל צומת מבצע *RELAX* על הקשתות שיוצאות ממנו.

אלגוריתם *Ford*

זהו אלגוריתם כללי בסה"כ ולא באמת השתמשנו בו אלא בשידרוג שלו (*bellman-ford*).

תאור שלבי האלגוריתם
- אתחול – הקלט הוא גרף כללי, משקלות אולי שליליים אך בהכרח אין מעגלים שליליים.
- לולאה – כל עוד קיימת קשת (u,v) עם *Relax* משפר נבצע אותו. אם האלג' עוצר נקבל $d(u)=\delta(s,u)$ לכל *u*.

אלגוריתם *Bellman – Ford*

תאור האלגוריתם: אלגוריתם זה מוצא מסלולים קצרים ביותר ובנוסף מגלה אם יש מעגלים שליליים בגרף (במידה וקיימים כאלה).

תחילה האלגוריתם מבצע איתחול. לאחר מכן הוא מבצע *Relax* על כל קשתות הגרף (שורות 2-4), כך שעבור כל קשת מתבצע ה-*Relax* $|V|-1$ פעמים. לאחר מכן הוא שוב רץ על כל קשתות הגרף (5-7) ובמידה ומוצא קשת שיש לה *Relax* משפר, הוא מחזיר שיש מעגל שלילי בגרף.

<u>זמן ריצה</u>: $O(|V|\cdot|E|)$

אלגוריתם *Dijkstra*

מניח שכל המשקלות אי-שליליים, הרבה יותר יעיל מבלמן פורד, לכן כדאי להשתמש בו אם ידוע כי המשקלות אי שליליים.

האלגורתים משתמש בתור עדיפות בשם Q, S היא קב' הצמתים שכבר הוצאנו מהתור ושעבורם כבר חשבנו את $\delta(s,v)$. התור מנוהל לפי $d(\,)$ המפתח.

אחרי שמוציאים מ-Q את הצומת המינימלי u מבצעים $RELAX$ על כל שכניו v:

1. אם $v \notin Q$ אין צורך ב-$RELAX$ ואין לו השפעה.
2. אם $v \in Q$ זה גורר פעולת $Decrease\_key(v)$.

<u>טענה</u>: כאשר מוציאים צומת u מהתור מתקיים $d(u)=\delta(s,u)$.

<u>סיבוכיות</u>: **בערימה רגילה** זמן הריצה יהיה $O\big((|E|+|V|)\,log|V|\big)$, **בערימת פיבונצ'י** $O(|V|\,log|V|+|E|)$.

שיטות נוספות

במקרה ונתון גרף מכוון חסר מעגלים, ניתן לחסוך זמן ריצה ולבצע את חישוב המרחקים באמצעות מיון טופולוגי, כך שזמן הריצה יהיה לינארי $O(|E|+|V|)$.

מק"בים בין כל זוגות הצמתים – *All Pairs Shortest Path*

<u>ייצוג</u>: הגרף ייוצג ע"י מטריצת שכנויות W המוגדרת כך: $W_{ij}=\begin{cases}0 & i=j\\ \infty & (i,j)\notin E\\ W(i,j) & (i,j)\in E\end{cases}$. (לצמתים מספרים סידוריים). כמו כן מגדירים מטריצה D (המאותחלת ל-W), כך ש-D_{ij} מייצג את אורך המסלול מ-i ל-j הקצר ביותר שהתגלה עד כה. בנוסף מוגדרת מטריצת מצביעים Π כך Π_{ij} מצביע לצומת הקודם ל-j במסלול הטוב ביותר שהתגלה עד כה מ-i. מטריצה זו מאותחלת כך: $\Pi_{ij}=\begin{cases}NIL & i=j\\ NIL & W_{ij}=\infty\\ i & W_{ij}<\infty\end{cases}$.

אלגוריתם ראשון (חיקוי של בלמן פורד באופן מטריציאלי)

<u>תאור האלגוריתם</u>:

עבור כל i,j נעדכן את D_{ij} באופן הבא: $D_{ij}` \leftarrow min_{1\le k\le n}\{D_{ik}+W_{kj}\}$ (זהו בעצם שלב אחד של בלמן פורד).

נבצע $|V|-1$ איטרציות כאלה במידה ואין מעגלים שליליים.

את חישוב D_{ij} ניתן גם להציג בתור $D_{ij}` \leftarrow \sum_{k=1}^{n}\{D_{ik}*W_{kj}\}$ שזה בעצם כפל מטריצות D ו-W (לא באמת מתבצע כפל מטריצות אלה הצורה שבה עוברים על הנתונים דומה).כלומר לאחר כל פעם שעידכנו את D נכפול אותה שוב ב-W ונקבל D אחרת, וכך הלאה. בעצם נאלץ לכפול *n-1* פעמים.

ניתן לצמצם את מספר המכפלות הנדרשות ע"י שינוי סדר הכפל והכנסת סוגריים.

<u>זמן ריצה</u>: $O(|V|^3\,log|V|)$ במקרה של הכפל המשופר.

<u>הערות</u>:

1. מעגלים שליליים ניתן לגלות ע"י בדיקה של איברים שליליים באלכסון בחזקה ה-$1-|V|$.
2. עדכון המטריצה Π מתבצע ע"י בחירת ה-k שנתן לנו את הערך המינימלי החדש ל-D_{ij} והשמתו ב-Π_{ij}.
3. אין צורך לשמור עותקים של כל המטריצות ואפשר להסתדר רק עם 2 או אפילו מטריצה אחת.

אלגוריתם *Floyd Warshall*

<u>תאור האלגוריתם</u>:
ממספרים את הקודקודים באופן שרירותי וקוראים להם $1, 2, 3, \ldots, n = |V|$. נגדיר את גובה המק"ב כאינדקס המקסימלי בצמתים הפנימיים.
נבנה מק"ב לפי גובה עולה – לכל *i,j,k* נחשב את אורך המסלול הקצר ביותר מ-*i* ל-*j* בגובה לכל היותר *k*.
זה נעשה על פי החישוב $D_{ij}^{(k)} \leftarrow min\left\{D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\right\}$ כאשר $D_{ij}^{(m)}$ הוא אורך המסלול הקצר ביותר מ-*i* ל-*j* בגובה m.

<u>הערות</u>:
1. מבחינת זיכרון ניתן להסתפק במטריצה אחת בלבד.
2. האלגוריתם מחשב את אורכי המסלולים. אם נרצה גם את המסלול עצמו, מצביע $\Pi_{ij}^{(k)}$ יקבל את הערך $\Pi_{ij}^{(k-1)}$ אם אורך המסלול לא השתנה, אחרת יקבל את הערך $\Pi_{kj}^{(k-1)}$.

<u>סיבוכיות</u>: $O(|V|^3)$

אלגוריתם *Johnson*

<u>תאור האלגוריתם</u>: אילו כל המשקלות היו חיוביים היינו יכולים להריץ מכל צומת את *Dijkstra* ובסך הכל הסיבוכיות הייתה $O(|V|^2 log|V| + |V| \cdot |E|)$.
לכן אלגוריתם זה תחילה הופך את המשקולות לאי שליליים (מפורט בהמשך) ומריץ *Dijkstra* $|V|$ פעמים.

<u>טענה</u>: תהא h פונ' כלשהי על הצמתים $h: V \rightarrow \mathbb{R}$. נגדיר משקלות חדשים $w^*(u,v) = w(u,v) + h(u) - h(v)$ לכל קשת (u,v). אז מק"ב תחת w $\Leftrightarrow$ מק"ב תחת w^*.

<u>בניית פונ' h כך ש-$w^* \geq 0$</u>: נוסיף לגרף צומת חדש s כצומת התחלה ונוסיף את כל הקשתות מ-s לכל צומת ב-V ולכולן משקל 0. נריץ בלמן פורד על הגרף החדש וניקח $h(v) = \delta(s,v)$.

<u>סיבוכיות</u>: כאמור $O(|V|^2 log|V| + |V| \cdot |E|)$.

רשתות זרימה

ברשתות זרימה מניחים כי הגרף קשיר ולכן $|E| \geq |V| - 1$.

<u>זרימה</u>: פונ' על הקשתות $f(u,v)$. $f: V \times V \rightarrow \mathbb{R}$ מוגדרת לכל זוג צמתים.
<u>זרימה חוקית מקיימת</u>:
- אלוצי קיבול $f(u,v) \leq c(u,v)$ לכל (u,v)
- אנטי סימטריה $f(u,v) = -f(v,u)$ לכל (u,v)
- שימור הזרימה: לכל צומת u $\sum_v f(u,v) = 0$

<u>ערך זרימה</u>: סך כל הזרימה הנכנסת ל-t - $|f| = \sum_{u \in V} f(u,t)$.
<u>רשת שיורית</u>: נתונה רשת זרימה G עם פונ' קיבול s,t,c ונתונה זרימה חוקית f. לכל קשת (u,v) נגדיר קיבול שיורי $c_f(u,v) = c(u,v) - f(u,v)$ ונגדיר את G_f כרשת השיורית כאשר הקיבולים הם הקיבול השיורי החדש.

<u>סימונים</u>

- $(A, B \subseteq V)$ $f(A,B) = \sum_{u \in A, v \in B} f(u,v)$
- אם *X,Y* זרות אז:
  - $f(X \cup Y, Z) = f(X,Z) + f(Y,Z)$

- $f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$

אלגוריתם *Ford Fulkerson*

<u>תאור האלגוריתם:</u>

איתחול הזרימה הנוכחית f ל-0 עבור כל הקשתות.

כל עוד קיים מסלול משפר כלשהו אז הגדל את הזרימה לאורך מסלול זה.

האלג' לא מציין כיצד מוצאים מסלול מ-s ל-t אך ניתן למשל לעשות את זה באמצעות BFS.

האלגוריתם עלול לרוץ בצורה מאוד לא יעילה על רשתות זרימה מסוימות ובמקרה של קיבולים אי רציונליים הוא עלול לא לעצור לעולם.

<u>מסלול משפר</u>: מסלול מ-s ל-t ב-G_f. קשתות עם קיבול 0 לא מופיעות ב-G_f.

אם p מסלול משפר, נסמן ב- $c_f(p)$ את הקיבול המינימלי של קשת של p = הקיבול השיורי של p.

אם p מסלול משפר שקבולו השיורי $c_f(p)$, נזרים דרך p זרימה נוספת בכמות $c_f(p)$.

נגדיר זרימה חדשה: $f(x) = \begin{cases} f(u,v), & (u,v) \notin p \\ f(u,v) + c_f(p), & (u,v) \in p \end{cases}$.

<u>טענה</u>: $|f^*| = |f| + c_p > |f|$ זרימה חוקית.

משפט *Min Cut – Max Flow*

נגדיר חתך – חלוקה של V לשתי קבוצות זרות (S,T) כך ש-$s \in S, t \in T$.

הקיבול של חתך (S,T) הוא $c(S,T)$ והזרימה דרך החתך (S,T) היא $f(S,T)$.

<u>טענה</u>: לכל זרימה f ולכל חתך (S,T) מתקיים $|f| = f(S,T)$.

<u>טענה</u>: לכל זרימה f ולכל חתך (S,T) מתקיים $f(S,T) \le c(S,T)$.

<u>משפט (*Min Cut – Max Flow*):</u>

התנאים הבאים שקולים:

1. f זרימה מקסימלית.
2. הרשת השיורית G_f לא מכילה מסלול מ-s ל-t.
3. $|f| = c(S,T)$ לאיזשהו חתך (S,T).

<u>סיבוכיות:</u>

מחפשים מסלול משפר קצר ביותר (מספר קשתות) על הרשת השיורית באמצעות BFS.

כמו כן מספר האיטרציות של מסלולים משפרים הוא $O(|E| \cdot |V|)$.

לכן הסיבוכיות הכוללת היא $O(|E|^2 \cdot |V|) \underset{\text{הגרף קשיר}}{=} O\big((|E| + |V|) \cdot |E| \cdot |V|\big)$

אלגוריתם דיניץ

<u>תאור</u>

בניגוד לאלגוריתם FF הבונה רשת שכבתית מוצא מסלול משפר אחד, ואז בונה רשת שכבתית אחרת, דיניץ בונה רשת שכבתית ומחפש בה מספר מסלולים משפרים. רק לאחר ש-t לא נגיש יותר מ-s בונים רשת שיורית חדשה ועליה רשת שכבתית חדשה. מתעלמים מכל הקשתות ההפוכות ברשת השכבתית.

במקום להריץ BFS על הרשת השכבתית, מריצים DFS עם מספר שינויים:
<u>$Advance(u)$</u>: אם אין קשת יוצאת מ-u, לך ל-$Retreat(u)$.
אם יש קשת כזאת (u,v), $v \neq t$ אז נלך ל-v ונמשיך ברקורסיה ל-$Advance(v)$.
אם $v = t$ נבצע $Augment$.
<u>$Retreat(u)$</u>: אם $u = s$ זהו סוף הטיפול ברשת השכבתית הנוכחית. זהו סוף <u>הפאזה</u> הנוכחית.
אחרת נמחק את u ואת כל הקשתות הנכנסות אליו, נחזור לאב v שקרא ל-u ונבצע $Advance(v)$.
<u>$Augment$: (מסלול משפר)</u> נדחוף זרימה נוספת לאורך המסלול. נמחק קשתות רוויות, נקטין קיבול של שאר הקשתות ונחזור לצומת ההתחלה u של הקשת הרוויה הכי קרובה ל-s במסלול. נבצע $Advance(u)$.

<u>סיבוכיות</u>
עלות כל פאזה $O(|E| \cdot |V|)$ ויש $O(|V|)$ פאזות ולכן הסיבוכיות היא $O(|E| \cdot |V|^2)$ לעומת $O(|E|^2 \cdot |V|)$ באדמונדס קארפ (FF).

רשתות 1\0

רשת שקיבול כל קשת בה הוא 1\0. האלג' של דיניץ יעיל מאוד במקרה זה:

- ברשת 1\0 רגילה מס' הפאזות הוא $O\left(\sqrt{|E|}\right)$, לכן סיבוכיות כוללת $O\left(|E|^{\frac{3}{2}}\right)$.
- ברשת 1\0 מטיפוס 1 (רשת ללא קשתות מקבילות ואנטי מקבילות) יש $O\left(|V|^{\frac{2}{3}}\right)$ פאזות, לכן סיבוכיות $O\left(|E| \cdot |V|^{\frac{2}{3}}\right)$.
- ברשת $0/1$ מטיפוס 2 (לכל צומת דרגת יציאה או כניסה 1) יש $O\left(|V|^{\frac{1}{2}}\right)$ פאזות, לכן הסיבוכיות $O\left(|E| \cdot |V|^{\frac{1}{2}}\right)$.

<u>הערה</u>: טיפוס הרשת נשמר גם ברשתות השיוריות.

<u>טענה</u>: נניח שהזרימה הנוכחית $\equiv 0$ ושערך הזרימה המקסימלית M אז מס' השכבות ברשת השכבתית לרשת מטיפוס 2 הוא $k \leq \frac{|V|-2}{M} + 1$, ולרשת כללית $k \leq \frac{|E|}{M}$.

שימושים של זרימה

<u>משפט $Hall$:</u>
יהא G גרף דו צדדי $V = X \underset{\text{זר}}{\cup} Y$, $E \leq X \times Y$. $|X| = |Y|$.
לכל $A \subseteq X$ נסמן ב-$\Gamma(A)$ את קבוצת השכנים של צמתים של A ב-Y.
אז ל-G יש <u>זיווג מושלם</u>, זיווג בגודל $|Y| = |X|$ אמ"מ לכל $A \subseteq X$, $|A| \leq |\Gamma(A)|$.

<u>מציאת מסלולים זרים בקשתות</u>: רוצים למצוא את מספר המסלולים הזרים בקשתות מ-s ל-t. הופכים את הגרף לרשת זרימה עם קיבול 1 על הקשתות. גודל הזרימה המקסימלית יהיה מספר המסלולים הזרים בקשתות.

<u>משפט מנגר (לקשתות)</u>: $G = (V, E)$ גרף מכוון. $s, t \in V$. המס' המקסימלי של מסלולים זרים בקשתות, k, מ-s ל-t שווה למס' המינימלי של קשתות למחוק כדי לנתק את s מ-t.

<u>טענה</u>: אם מנתקים את החתך המינימלי, אז לא ניתן להשיג max-flow ברשת הזרימה.

תכנות דינמי

תחליף לגישה רקורסיבית לפתרון בעיות כאשר הפתרון הרקורסיבי יקר מידי, כי הוא נתקל שוב ושוב באותן תתי-בעיות (מבלי להרגיש) והוא שוב ושוב פותר אותן.

<u>מספר דוגמאות:</u>

חישוב מספר פיבונצ'י – במקום לחשב רקורסיבית את הנוסחה, מחשבים כל מקרה העלול להופיע בחישוב הנוסחה הכללית עבור הקלט הנוכחי, ואז לגשת למקרים הללו בדרך כלשהי.

כפל מטריצות יעיל - נתונה סדרה של מטריצות (לאו דווקא ריבועיות) $A_1, A_2, A_3, \ldots, A_n$. רוצים לחשב $A_1 \cdot A_2 \cdot A_3 \cdot \ldots \cdot A_n$. במקום לחשב כפל של מטריצות אחת אחר השניה בצורה רגילה (שעלולה להיות מאוד לא יעילה) ניתן למצוא סדר הכפלה (בלי לשנות את סדר האיברים) כך שהעלות תהיה מינימלית.
למשל כפל מטריצה $p*q$ במטריצה $q*r$ הוא pqr.
נסמן $C[i,j]$ = המחיר המינימלי של כפל המטריצות $A_iA_{i+1}A_{i+2} \cdot \ldots \cdot A_{j-1}A_j$ ונגדיר $C[i,j] = min_{i\le k<j}\{C[i,k] + C[k+1,j] + p_{i-1}p_kp_j\}$ וכן $C[i,i] = 0$.

עלות של ביצוע רקורסיבי עלולה לקחת 3^m כאשר m הוא מספר המטריצות.

נשים לב שבסך הכל יש $O(2^n)$ תתי בעיות, כך שאם נחשב כל אחת בדיוק פעם אחת ונשמור אותן בצורה כשהי, נוכל לכתוב אלגוריתן העובד בזמן $O(n^3)$.

חישוב תת סידרה משותפת מקסימלית – מקבלים 2 מחרוזות Y,X ורוצים למצוא סדרה המופיעה באותו סדר אך לא בהכרח באותו רצף בשתי המחרוזות באורך מקסימלי.

נסמן ב- $X^{(i)}$ את הרישא $prefix$ של i האיברים הראשונים של x.

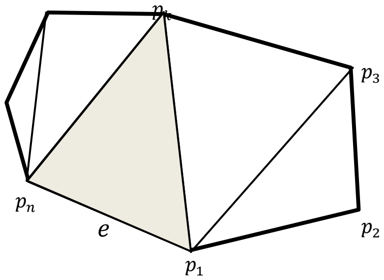
כמובן שנוכל להציע את האלגוריתם הבא:

$$LCS[X,Y]:\ if\ X_m = Y_n\ then\ return\ LCS\left(X^{(m-1)}, Y^{(n-1)}\right) \overset{\text{שרשור}}{\|} (X_m)$$
$$else\ \ return\ the\ longest\ of\ LCS\left(X^{(m-1)}, Y\right), LCM\left(X, Y^{(n-1)}\right)$$

אך הוא לא יעיל, שכן הוא מחשב את אותן תתי בעיות שוב ושוב. בסך הכל יש $m \cdot n$ בעיות כאלה, לכן נוכל לחשב אותן מראש ולשים את הערכים בתוך מטריצה ונחזיר את הערך m,n במטריצה (אורך מקסימלי). סה"כ הסיבוכיות היא $O(m \cdot n)$.

בעיית מספרים - נתונים מספרים טבעיים $b\ \ a_1, a_2, \ldots, a_n$ כאשר $a_i > 0, b \ge 0$.
רוצים לבדוק אם קיים פתרון, או מספר פתרונות ל- $\sum_{i=1}^{n} a_ix_i = b$ כאשר כל x_i הוא 0 או 1.
זמן הריצה של הפיתרון שנציע תלוי ב-n וב-$M = \sum_{i=1}^{n} a_i$, אחרת אין פיתרון בזמן P.

טריאנגולציה עם משקל מינימלי - נתון מצולע קמור p עם n קודקודים במישור. ע"י העברת מיתרים שלא חוצים זה את זה נפרק את p למשולשים. לכל מצולע כזה יש $\binom{n}{3}$ משולשים פוטנציאליים.



משקל הטריאנג' = סכום משקלות המשולשים שלה (למשל הקף המשלוש).
מטרה: לבנות טריאנג' עם משקל מינימלי.

באופן רקורסיבי ניתן להגדיר את הבעיה כך:

$$\min_{2\le k\le n-1}\left[w(p_1p_kp_n) + w(p_1 \text{ טריאנגולציה מינימלית של}) + w(p_2 \text{ טריאנגולציה מינימלית של})\right]$$

בשיטה יותר יעילה נוכל לחשב את כל תתי המצולעים (יש בסה"כ $O(n^2)$ כאלה)

התאמת מחרוזות - *String Matching / Pattern matching*

הקלט: מחרוזת טקסט T (הארי פוטר), ועוד מחרוזת תבנית $Pattern$ (הארי) P.
המטרה: למצוא את כל המקומות ב-T בהם מופיע P.
$|T| = n$ אורך הטקסט. $|P| = m$.

אלגוריתם אוטומט

<u>תאור האלגוריתם:</u>
האלגוריתם מקבל את המחרוזת T, פונקציית המעברים δ ואורך המחרוזת m.
הלולאה שרצה מ-1 עד n מדמה אוטומט אשר כל קלט של תו של T הוא לולאה אחת באלגוריתם.
כאשר מתקבל תו מ-T, a,מחושב $\delta[q,a] = k$ שהוא גודל הרישא המקסימלית של P שהיא סיפא של $P[1 \dots q]a$.
k במקרה שלנו הוא גם מספר המצב הבא. אם מקבלים k=m אז מודיעים כי מצאנו הופעה אחת של P ב-T,
וממשיכים בקלט כרגיל.

<u>חישוב פונקציית המעברים:</u>
הפונקציה מחשבת מהו אורך הרישא המקסימלית של P שהיא גם סיפא של $P[1 \dots q]a$ כאשר q מייצג מצב כלשהו. התוכנית רצה על כל השילובים האפשריים של q ו-a.

<u>סיבוכיות:</u>
תוכנית האוטומט רצה בזמן של $O(n)$ ואילו חישוב פונקציית המעברים נעשה בזמן של $O(m^3)$.

KMP

חישוב פונקציית המעברים גוזלת זמן רב במקרה של האלגוריתם הקודם, וזאת בשל ריבוי המקרים שהתוכנית בודקת מאחר והיא צריכה להתאים עצמה לכל תווי הקלט.

<u>תאור האלגוריתם</u>
באלגוריתם הנוכחי פונקציית המעבר אינה תלויה בקלט האות הבאה אלה במחרוזת P בלבד.
ברגע שאנו נתקלים בתו של T שאינו תואם לספירת ה-P הנוכחית, פונקציית המעבר מקבלת את המצב הנוכחי q (מספר התוים של P שהצלחנו להתאים ל-T עד המקום הנוכחי) ומחזירה את המצב הבא (או את מספר התוים המקסימלי שהתאמנו לאחר שהזזנו את המחרוזת P בצורה כזאת שהרישא של P הנוכחית תהיה הסיפא של P בצעד הבא).

<u>סיבוכיות</u>
חישוב פונקציית המעברים נעשית בזמן $O(m)$. לפיכך התוכנית הראשית רצה בזמן $O(m+n)$.