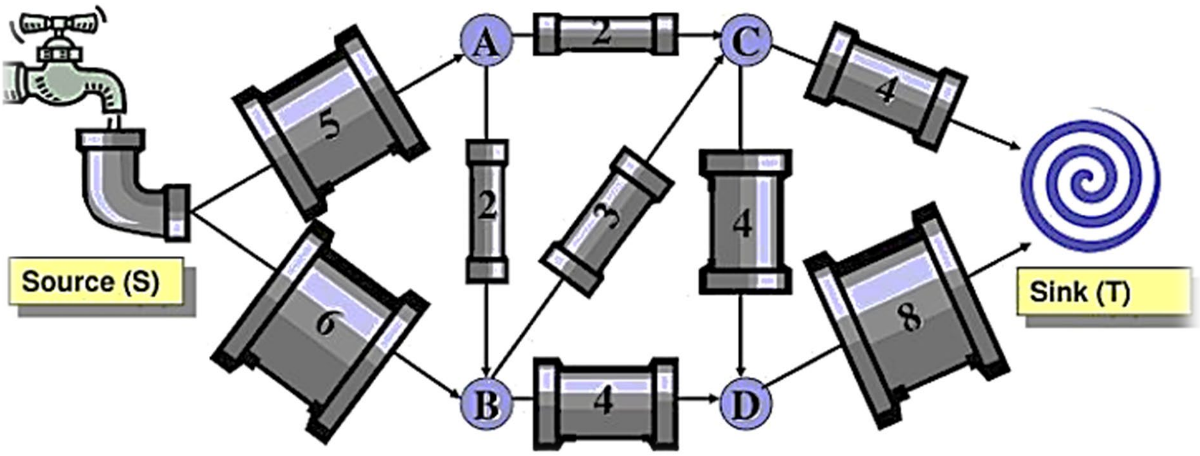
מכללת אורט בראודה
המחלקה להנדסת תכנה
אלגוריתמים - 61753
תשפ״א, סמסטר ב׳

עבודה במסגרת מסלול מצוינות
אלגוריתם Dinitz לזרימה מקסימלית ברשתות



מנחה: ד״ר ילנה קליימן
מגיש: אורי מלכא
ת.ז : 314862996
24.6.2021

תוכן עניינים

תקציר מחקר

בעבודה זו חקרתי את אלגוריתם Dinitz (מוכר גם כ-Dinic's Algorithm) למציאת זרימה מקסימלית ברשתות זרימה. האלגוריתם של Dinitz הוא שיפור לא טריוויאלי של אלגוריתם Ford-Fulkerson למציאת זרימה מקסימלית שנלמד בקורס.

ניתן לחלק עבודת מחקר זו לשני חלקים – בחלק הראשון, הגדרתי את "בעיית הזרימה המקסימלית" ברשת זרימה, אלגוריתם Ford-Fulkerson (הגרסה המקורית והשיפור שהוא: אלגוריתם Edmonds-Karp) ואלגוריתם Dinitz (כולל הוכחת נכונות, ניתוח זמן ריצה והצגת מקרה פרטי כאשר הקיבול 1). בחלק השני, ביצעתי השוואה תיאורטית בין האלגוריתמים, ניסויים הבוחנים את התנהגות האלגוריתמים בפועל (מימוש האלגוריתמים בשפת תכנות ומדידת זמן הריצה של כל אחד מהם על גרפים עם מאפיינים שונים), ניתחתי את תוצאות הניסויים והסקתי מסקנות.

הצגת הבעיה ורקע כללי[1]

רשת זרימה

רשת זרימה היא סוג מיוחד של גרף מכוון, שמשמש למידול בעיות שמערבות מעבר של חומר בין מקומות. כל רשת זרימה מאופיינת על ידי גרף מכוון שמכיל שני צמתים מיוחדים – האחד משמש כ**מקור** – המקום שממנו נובעת הזרימה, והשני משמש כ**בור** – המקום שאליו מתנקזת הזרימה. כמו כן לכל קשת בגרף ישנו **קיבול**, שמתאר את כמות הזרימה שמסוגלת לעבור בה.

מבחינה פורמלית, רשת זרימה מוגדרת כגרף מכוון וממושקל $G(V, E, c)$ בעל שני צמתים מיוחדים – המקור (Source) s והבור (Sink) t (נסמן: $N = (G, c, s, t)$).
פונקציית המשקל - $c : V \times V \to \mathbb{R}^+ \cup \{0, \infty\}$ היא פונקציית קיבול שמתאימה לכל זוג צמתים (u, v) את הכמות הזרימה המקסימלית שיכולה לעבור מ-u ל-v.

בנוסף, מניחים את ההנחות הבאות:

1.1) אם לא קיימת קשת מ-u ל-v אז $c(u, v) = 0$.

1.2) הגרף קשיר, ולכן $|E| \geq |V| - 1$.

1.3) קיים מסלול מ-s ל-t.

זרימה חוקית

זרימה היא מידול מעבר החומר מהמקור אל הבור, דרך שאר צמתי הגרף. כיוון שאנו מעוניינים להתחשב רק במקרים בהם כל הזרימה שיוצאת מן המקור מגיעה אל הבור נגדיר זרימה כזו כ**זרימה חוקית**, המתאפיינת בכך שכל זרימה שנכנסת לצומת דרך הקשתות שנכנסות אליו גם יוצאת ממנו דרך הקשתות היוצאות ממנו (למעט המקור והבור).

[1] מקורות-[3] ו-[4].

מבחינה פורמלית, נתאר זרימה על ידי פונקציית זרימה - $f : V \times V \to \mathbb{R}$ המתאימה לכל זוג צמתים (u, v) את כמות הזרימה מ-u ל-v (עם חשיבות לכיוון), כך שמתקיימים שלושת התנאים הבאים:

1) **אנטי סימטריה** - $f(u, v) = -f(v, u)$
כיוון שזרימה שעוברת מצומת u אל הצומת v מתוארת על ידי מספר חיובי אז ניתן לתאר את הזרימה הזו כזרימה שעוברת מצומת v אל הצומת u (כלומר, בכיוון השני) אך בכמות <u>שלילית</u>. התכונה הזו מאפשרת לדבר על הזרימה בין שני צמתים בשני הכיוונים גם יחד, ולא רק בכיוון אחד, גם אם קיימת קשת ביניהם רק בכיוון אחד.

2) **אילוץ הקיבול** - $f(u, v) \leq c(u, v)$
על אף קשת לא מוזרם יותר מהקיבול שלה.

3) **שימור הזרימה** - $\forall u \neq s, t : \sum_{v \in V} f(u, v) = 0$
כל הזרימה שנכנסה לצומת גם יצאה ממנו. מכיוון שזרימה שנכנסת לצומת מיוצגת על ידי מספר חיובי, וזרמה שיוצאת מהצומת מיוצגת על ידי מספר שלילי, אז סכום הזרימה שווה אפס.

ערך זרימה

הגודל $f(u, v)$ (שיכול להיות חיובי או שלילי) נקרא **הזרימה נטו** מצומת u אל הצומת v.
לכל זרימה מגדירים את **ערך הזרימה** בתור $|f| = \sum_{v \in V} f(s, v)$, כלומר כמות הזרימה נטו שיוצאת מן המקור.

<u>הערה:</u>
נהוג לסמן על קשתות רשת הזרימה את הזרימה והקיבול כך: $f(u, v)/c(u, v)$

בעיית הזרימה המקסימלית

השאלה העיקרית שבה עוסקים כאשר חוקרים רשת זרימה היא מה הכמות הגדולה ביותר של זרימה שניתן להעביר מן המקור ועד לבור, ומה הדרך שבה ניתן לעשות זאת. בעיה זו מכונה "<u>בעיית הזרימה המקסימלית</u>" (מכאן והלאה "הבעיה").

מבחינה פורמלית, נגדיר את הבעיה כך:
בהינתן רשת זרימה N, מהי הזרימה f מ-s ל-t שבה $|f|$ מקסימלי?

רקע תיאורטי

אלגוריתם פורד-פולקרסון (Ford-Fulkerson)

הצגת הרעיון

הרעיון הבסיסי מאחורי האלגוריתם הוא שיפור איטראטיבי של הזרימה: מתחילים עם פונקציית זרימה שנותנת ערך אפס לכל קשת, ולאחר מכן מחפשים דרכים לשפר אותה על ידי בדיקת מסלולים מהמקור אל הבור. אם מתגלה מסלול אשר ניתן להגדיל את הזרימה בו, הפונקציה מתוקנת בהתאם לכך, והתהליך חוזר על עצמו עד שמתקבלת הפונקציה האופטימלית. כלומר, כל עוד יש מסלול שמשפר את הזרימה. תיאור מדויק של השיטה מתבסס על מושג הרשת השיורית, שמתאר את מצב רשת הזרימה ביחס לפונקציית זרימה נתונה.

רשת שיורית

בהינתן זרימה, ניתן לבדוק כיצד ניתן לשפר אותה (כלומר, להגדיל את ערך הזרימה) באמצעות שימוש ברשת זרימה המכונה "הרשת השיורית", עם גרף אשר מייצג את הקשתות שדרכן ניתן להגדיל את הזרימה. כדי להגדיר מהי רשת שיורית בצורה פורמלית, נגדיר קודם כמה מושגים:

קשת רוויה

קשת (u,v) שבה עוברת זרימה מרבית אפשרית – $f(u,v)=c(u,v)$.

קיבול שיורי

הקיבול השיורי של קשת (u,v) – $c_f(u,v)=c(u,v)-f(u,v)$, מייצג את כמות הזרימה נטו שניתן להעביר מהצומת u אל הצומת v.

קשת משפרת

קשת (u,v) שאפשר להזרים לאורכה עוד זרימה – $f(u,v)<c(u,v)$.

באופן שקול קשת (u,v) היא משפרת אם $c_f(u,v)>0$.

מסלול משפר

מסלול משפר הוא מסלול מ-s ל-t שמורכב כולו מקשתות משפרות. קיום של מסלול משפר שכזה פירושו שברשת המקורית ניתן להעביר זרימה נוספת דרך אותו מסלול.

<u>הערה:</u>

הגרף המקורי הוא מכוון, אך במסלול משפר מותר ללכת "<u>נגד הכיוון</u>".

קיבול שיורי של מסלול

הקיבול השיורי של מסלול P מוגדר להיות הקיבול השיורי המינימלי לאורכו, $c_f(P)=\min_{(u,v)\in P} c_f(u,v)$. כלומר, משקל הקשת המינימלית שמרכיבה אותו.

הרשת השיורית $N_f = (G_f, c_f, s, t)$ היא רשת זרימה עם **גרף שיורי** G_f מכוון וממושקל שצמתיו הם צומתי גרף רשת הזרימה המקורית G, וקשתותיו הן **קשתות משפרות** שמשקלן הוא **הקיבול השיורי** שלהן.
כלומר, הגרף השיורי מייצג את כל המקומות שבהם ניתן להעביר זרימה נוספת ברשת המקורית. כמובן כי כל מסלול מ-s ל-t ב-G_f הוא מסלול משפר ב-G ולהפך.

Ford-Fulkerson's Algorithm [2]

<u>קלט</u>: רשת זרימה N.
<u>פלט</u>: זרימה f ב-N כך ש-$|f|$ מקסימלית.

<u>תיאור הסכימה</u>:

1. אתחול - $\forall u, v \in V: \; f(u,v) = 0$ (יכול להיות כל ערך זרימה חוקית).
2. בנה את הגרף השיורי G_f - $\forall u, v \in V: \; c_f(u,v) = c(u,v)$.
3. כל עוד יש מסלול מ-s ל-t ב-G_f, בצע:
   א. נמצא מסלול P ב-G_f ונחשב את קיבולו השיורי $c_f(P)$.
   ב. נגדיל את הזרימה ב-G לאורך המסלול P ב-$c_f(P)$:
   $$\forall (u,v) \in P \text{ נעדכן } - \begin{cases} f(u,v) = f(u,v) + c_f(P) \\ f(v,u) = -f(u,v) \end{cases}$$
   ג. נעדכן את G_f לאורך המסלול P:
   $$\forall (u,v) \in P \text{ נעדכן } - \begin{cases} c_f(u,v) = c(u,v) - f(u,v) \\ c_f(v,u) = c(v,u) - f(v,u) \end{cases}$$
4. החזר את f.

<u>זמן ריצה</u>: $O(|E|F)$, כאשר F ערך הזרימה המרבית (אם כל הקיבולים שלמים).

Edmonds-Karp's Algorithm [3]

אם בוחרים מסלול משפר בעזרת BFS[4], כלומר מוצאים מסלול <u>קצר ביותר</u>, אז מספר האיטרציות הוא $O(|V||E|)$ ולכן זמן הריצה הכולל הוא $O(|V||E|^2)$.

מכאן נובע כי אם הקיבולים שלמים ובוחרים מסלולים משפרים על ידי BFS זמן הריצה של אלגוריתם Ford-Fulkerson הוא: $O(\min\{|V||E|^2, |E|F\})$.

---

[2] נכונות האלגוריתם וניתוח זמן הריצה נלמדו בקורס ואינן מוצגות.
[3] משפט זה ניתן בקורס ללא הוכחת נכונות, מקור-[3].
[4] אלגוריתם Breadth-First Search שנלמד בקורס, זמן ריצה - $O(|E|)$.

אלגוריתם דיניץ (Dinitz)

הצגת הרעיון

הרעיון הבסיסי באלגוריתם הוא זה של פורד-פולקרסון – מציאת מסלול שיפור, ושיפור הזרימה לאורכו, אך האלגוריתם של דיניץ מאפשר למצוא מספר מסלולי שיפור "בבת אחת", ובצורה יעילה יותר מאשר אלגוריתם אדמונדס-קרפ.

האלגוריתם מבוסס על מושג נוסף לזה של הרשת השיורית – **רשת השכבות**. בהינתן רשת שכבות, ניתן למצוא בקלות יחסית זרימה שתנצל אותו "לחלוטין". זרימה שכזו מכונה **זרימה חוסמת**. האלגוריתם של דיניץ מוצא זרימה חוסמת ברשת השכבות ומשפרת באמצעותה את הזרימה הקיימת ברשת המקורית. הוא חוזר על התהליך שוב ושוב עד אשר ברשת השכבות אין מסלול המחבר בין המקור לבור.

רשת שכבות

בהינתן רשת זרימה ורשת שיורית עבורה, **רשת השכבות** (או גרף שכבות) בנויה על הרשת השיורית, ומחולקת ל"שכבות" על פי מרחקי הצמתים כשבכל שכבה מצומת המקור s. כלומר, בשכבה ה-k נמצאים כל הצמתים שמרחקם מ-s הוא בדיוק k. הקשתות היחידות שברשת השכבות הן אלו שמחברות בין צמתים השייכים לשתי שכבות סמוכות (כלומר, כל קשת שייכת למסלול קצר ביותר מ-s אל איזושהי צומת).

מבחינה פורמלית, רשת השכבות $L_f(V_L, E_L)$ של הרשת השיורית G_f הינה גרף שכבות המכיל את כל המסלולים הקצרים ביותר מ-s ל-t. כאשר:

$$\begin{cases} V_L = \bigcup_{i=0}^{|V|-1} V_i \quad E_L = \bigcup_{i=0}^{|V|-2} E_i \\ V_i = \{v \in V | \delta(s,v) = i\} \quad E_i = \{(u,v) \in E_f | u \in V_i, v \in V_{i+1}\} \end{cases}$$

<u>הערות:</u>

1. השכבה V_i מכילה את הצמתים שמרחקם מ-s הוא i.
2. E_i אלו כל הקשתות בין השכבה i לשכבה $i+1$.
3. נסמן: $\delta(s,t) = L$.
4. $\delta(s,v)$ הוא המרחק המינימאלי (בקשתות) מ-s **ל-**v ב-G_f.

אלגוריתם לבניית רשת שכבות[5] (מורחב BFS)

<u>קלט</u>: רשת שיורית N_f.
<u>פלט</u>: רשת שכבות L_f.

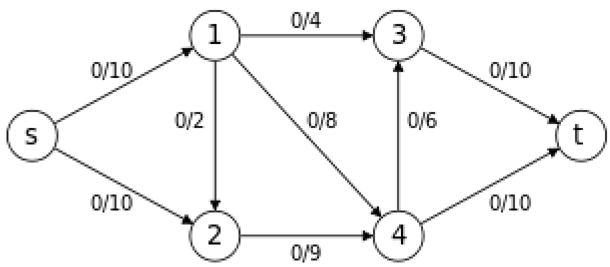
<u>תיאור הסכימה</u>:

1. אתחול – הכנס את s ל-V_0, $i \leftarrow 0$.
2. כל עוד V_i אינו ריק ו-t לא שייך ל- V_i, בצע:
   א. עבור כל צומת $u \in V_i$ בצע:
      עבור כל שכן v של u אם v לא נמצא באף $V_{j\le i}$, בצע:
      a. הוסף את v ל-$V_{j\le i}$ (אם הוא עדיין לא שם).
      b. הוסף את (u, v) ל-E_f.
   ב. $i \leftarrow i + 1$
3. אם t לא שייך ל-V_i, אין מסלול בין s ל-t.
   אחרת (כאשר t שייך ל-V_i) תסיים.

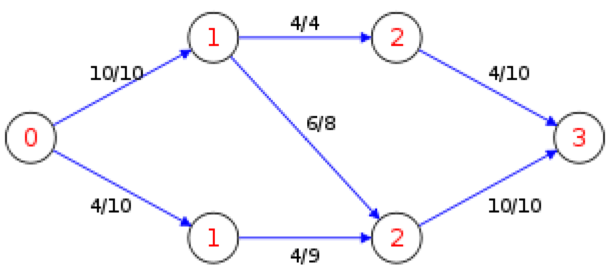
<u>זמן ריצה</u>: למדנו בהרצאה כי זמן הריצה של BFS הוא ליניארי בגודל הגרף, וכיוון שהגרף השיורי G_f קשיר אז זמן הריצה של אלגוריתם זה הוא: $O(|E|)$.

הדוגמה 2 **- בניית גרף שכבות על גבי גרף שיורי**[6]

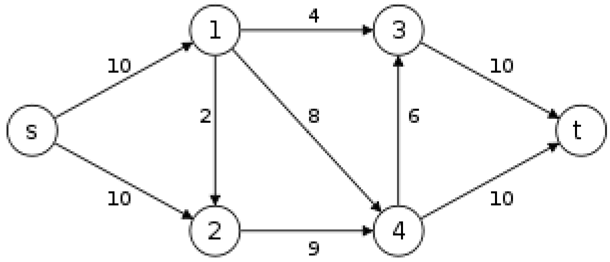
איור 2 - הגרף G של רשת הזרימה.



איור 4 – גרף השכבות G_L של רשת השכבות L_f אחרי הזרמה (באדום ערכי $\delta(s,v)$)



איור 3 – הגרף השיורי G_f של הרשת השיורית N_f.



[5] אלגוריתם זה מוכר כ-BFS מורחב במאמר המקורי של דיניץ, מקור-[6] עמ' 1.
[6] דוגמא טריוויאלית לבניית גרף שכבות, מקור-[4].

זרימה חוסמת

זרימה g תיקרא **חוסמת** ב-N אם בכל מסלול P מ-s ל-t קיימת קשת אחת לפחות המקיימת - $c(u,v) = g(u,v)$ (קשת רוויה).

אלגוריתם למציאת זרימה חוסמת[7]

<u>קלט</u>: רשת שכבות L_f.
<u>פלט</u>: זרימה חוסמת g.

<u>תיאור הסכימה</u>:

1. אתחול - $\forall u,v \in V{:}\ \ g(u,v) = 0$.
2. כל עוד[8] $\deg_{in}(t) > 0$, בצע:
   א. מצא מסלול P מ-s ל-t על ידי הליכה אחורה* מ-t ל-s.
   ב. חשב את $c_g(P)$.
   ג. חשב זרימה f_p** והוסיפה ל-g.
   ד. ניקוי רשת השכבות - $\forall(u,v) \in P$, בצע:
   אם $c_g(u,v) = f_p(u,v)$, אז:
      a. מחק את (u,v).
      b. אם $\deg_{in}(v) = 0$, בצע: $clean(v)$**.
3. החזר את g.

*מציאת מסלול ברשת השכבות מתבצעת על ידי סריקה אחורה מ-t עד ל-s. כיוון שדואגים לנקות את הרשת בכל איטרציה, מובטח כי סדרת הצעדים בהכרח לא תעצור לפני הגעה ל-s, כאשר כל קשת היא בין שכבות עוקבות. כלומר, P יהיה מסלול קצר ביותר מ-s ל-t. הזמן הדרוש למציאת מסלול P ברשת השכבות הוא לינארי במספר השכבות, $O(|P|)$ (נוכיח בהמשך).

** $$f_p(u,v) = \begin{cases} c_g(P) & (u,v) \in P \\ -c_g(P) & (v,u) \in P \\ 0 & else \end{cases}$$

***
$$\begin{aligned}
&clean(v){:}\\
&\quad for\ each(v,w){:}\\
&\qquad remove(v,w)\\
&\qquad if(\deg_{in}(w) = 0)\\
&\qquad\quad clean(w)
\end{aligned}$$

---

[7] מקור-[2] עמ' 73.
[8] תכונה נשמרת ב-L_f, כיוון ש-s הוא הצומת היחיד עם $\deg_{in}(s) = 0$.

<u>ניתוח זמן ריצה:</u>

1. אתחול - $O(|E|)$.
2. כל עוד $\deg_{in}(t) > 0$[9] - בכל איטרציה של 2 נמחקת לפחות קשת אחת ולכן לא ייתכנו יותר מ-$|E|$ איטרציות של 2.
   א. מציאת מסלול P – מ-* נובע $O(|P|)$. כיוון שP הוא מסלול קצר ביותר (מסלול פשוט) בין s ל-t ב-L_f שבה יש V צמתים נובע כי במקרה הגרוע מציאת מסלול P תתבצע ב-$O(|V|)$.
   ב. חישוב $c_g(P)$ - $O(|P|) = O(|V|)$ במקרה הגרוע.
   ג. חישוב f_p - $O(|P|) = O(|V|)$ במקרה הגרוע.
   ד. שלב הניקוי נועד לתחזוקה של רשת השכבות ולוקח בממוצע* $O(1)$ בכל איטרציה של 2.
3. החזר את g - $O(1)$.

עד כאן (ללא עלות התחזוקה הכללית), קיבלנו: $O(|V||E|)$.

*אם נתבונן על העלות של ד' באיטרציה כולה נראה כי כל קשת וצומת נמחקת בפרוצדורה $clean(v)$ פעם אחת לכל היותר וטיפול בקשת שנמחקת מתבצע ב$O(1)$ ולכן במקרה הגרוע הטיפול בקשתות עולה: $O(|V| + |E|) = O(|E|)$.

<u>זמן ריצה:</u> קיבלנו כי מציאת זרימה חוסמת ברשת השכבות עולה:

$$O(|E|) + O(|V||E|) + O(|E|) = O(|V||E|)$$

הדגמה 3 - מציאת זרימה חוסמת[10]

נתבונן שוב בגרף השיורי G_f שראינו קודם (***איור 3***).
ניתן לראות כי יש שלושה מסלולים (בכחול ב***איור 4***) מהמקור לבור והם:

P_1 - $\{s, 1,3, t\}$ קיבול מינימלי של 4 לאורך המסלול.
P_2 - $\{s, 1,4, t\}$ קיבול מינימלי של 6 לאורך המסלול.
P_3 - $\{s, 2,4, t\}$ קיבול מינימלי של 4 לאורך המסלול.

לכן הזרימה החוסמת היא בגודל 14. כלומר, כעת $|f| = 14$ (נשים לב כי בכל מסלול משפר בזרימה החוסמת יש שלוש קשתות).

כדי למצוא את הזרימה המקסימלית, נצטרך לעבוד לפי אלגוריתם דיניץ. אציג את האלגוריתם, נמצא את הזרימה המקסימלית (כלומר נמשיך את ההדגמות) ואז נוכיח את נכונותו וננתח את זמן הריצה.

---

[9] תכונה נשמרת ב-L_f, כיוון ש-s הוא הצומת היחיד עם $\deg_{in}(s) = 0$.
[10] דוגמא טריוויאלית למציאת זרימה חוסמת, מקור-[4].

Dinitz's Algorithm [11]

<u>קלט</u>: רשת זרימה N.
<u>פלט</u>: זרימה f ב-N כך ש-$|f|$ מקסימלית.

<u>תיאור הסכימה</u>:

1. אתחול - $\forall u,v \in V:\ f(u,v) = 0$
2. בנה את הרשת השיורית N_f.
3. כל עוד יש מסלול מ-s ל-t ב-G_f, בצע:
   א. בנה את רשת השכבות L_f של N_f.
   ב. מצא זרימה חוסמת g ב-L_f.
   ג. עדכן ב-N: $f \leftarrow f + g$
   ד. בנה מחדש את N_f.
4. החזר את f.

(לנוחות מאילך והלאה לכל איטרציה של 3 בסכמה נקרא "שלב" של האלגוריתם).

***הדגמה 4* – מציאת הזרימה מקסימלית**[12]

1. אתחול - ***איור 2*** ($|f| = 0$)
2. בניית הרשת השיורית N_f - ***איור 3***
3. כל עוד יש מסלול מ-s ל-t ב-G_f, בצע:
   איטרציה I:
   א. בניית רשת השכבות L_f - ***איור 4***
   ב. מציאת זרימה חוסמת g - ***הדגמה – 3***
   ג. עדכון f - $|f| = 0 + 14 = 14$
   ד. בנייה מחדש של N_f ועדכון N ***איור 5-6***

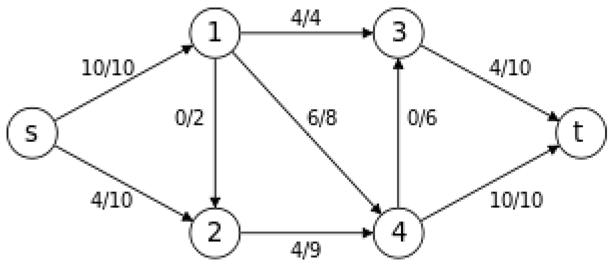
   איטרציה II:
   א. בניית רשת השכבות L_f - ***איור 7***
   ב. מציאת זרימה חוסמת g -
   ניתן לראות כי נשאר מסלול אחד (בכחול ב***איור 6***) מהמקור לבור: P_4 - $\{s,2,4,3t\}$ קיבול מינימלי של 5 לאורך המסלול.
   כמובן שמכאן נובע כי הזרימה החוסמת היא $g = 5$.
   ג. עדכון f - $|f| = 14 + 5 = 19$
   ד. בנייה מחדש של N_f ועדכון N ***איור 8-9***

---
[11] מקור-[2] עמ' 69.
[12] דוגמא טריוויאלית לבניית גרף שכבות, מקור-[4].

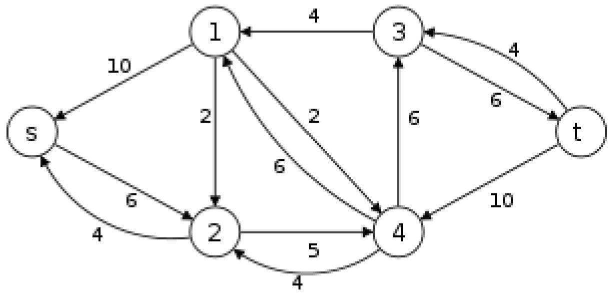
4. כפי שניתן לראות ב*איור 9*, כיוון ש-t לא ישיג מ-s ב-G_f, האלגוריתם מסיים ומחזיר את הזרימה המקסימלית $|f| = 19$.
בנוסף, ניתן לראות ב*איור :* את גרף השכבות המעודכן (למרות שהאלגוריתם לא דרש לבנותו).

<u>השערה 1:</u>
נשים לב כי כאשר מצאנו את הזרימה החוסמת באיטרציה הראשונה, כל מסלול משפר היה באורך שלוש קשתות ובאיטרציה השנייה אורך המסלול המשפר היה ארבע קשתות. ההשערה היא שבכל איטרציה של אלגוריתם דיניץ אורך המסלול המשפר בין המקור לבור גדל לפחות בקשת אחת וזה בעצם מה שגורם לעצירתו.

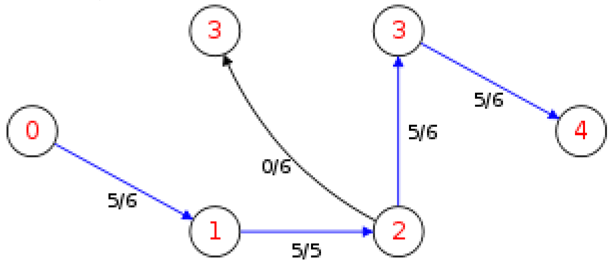
איור 5 - הגרף G של רשת הזרימה N מעודכן בסוף איטרציה II.



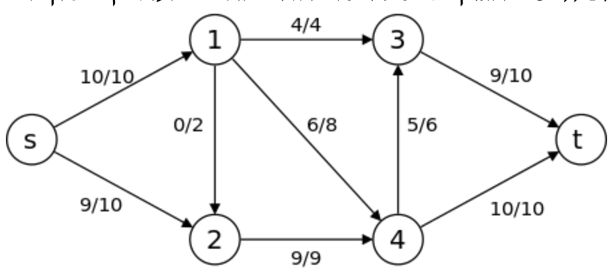
איור 6 - הגרף G_f של השרת השיורית N_f אחרי בנייה מחדש באיטרציה II.



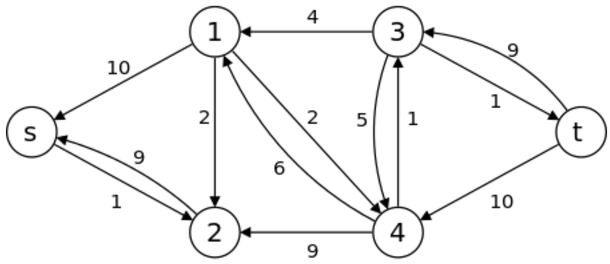
איור 7 - גרף השכבות G_L של רשת השכבות L_f אחרי ההזרמה השנייה (באדום ערכי $\delta(s,v)$)



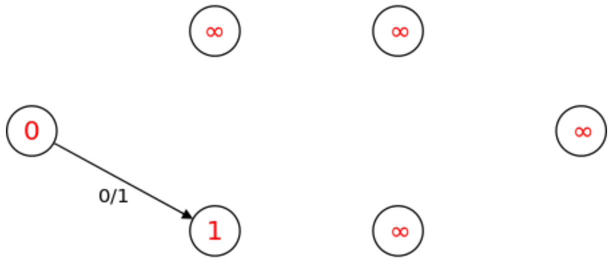
איור 8 - הגרף G של רשת הזרימה N מעודכן בסוף איטרציה III.



איור 9 - הגרף G_f של השרת השיורית N_f אחרי בנייה מחדש באיטרציה III.



איור : - גרף השכבות G_L של רשת השכבות L_f אחרי ההזרמה השלישית (באדום ערכי $\delta(s,v)$)



הוכחת נכונות

אלגוריתם Dinitz מחזיר זרימה חוקית

בכל שלב מוצאים זרימה חוסמת g ב-L_f. כיוון ש-L_f רשת המוכלת ברשת N_f, g חוקית ב-N_f. חוקיות הסכום $g + f$ ב-N נובעת מהלמה[13] שהוכחנו עבור הרשת השיורית בהרצאה.

אלגוריתם Dinitz מחזיר זרימה מקסימלית

מתנאי העצירה של האלגוריתם (כל עוד יש מסלול מ-s ל-t ב-G_f) אנו מקבלים כי f מקסימלית על פי משפט: "Max Flow-Min Cut[14]" שהוכחנו בהרצאה.

אלגוריתם Dinitz עוצר אחרי $|V|-1$ שלבים לכל היותר

הקדמה

נוכיח כי מרחקו של הבור t מהמקור s גדל ממש בכל שלב (השערה 1).
נניח כי בשלב הנוכחי בונים את L_f כאשר t בשכבה ה-k. כלומר $\delta(s,t) = k$ ב-N_f.
אם נסמן ב-f' את הזרימה המעודכנת בסוף השלב הנוכחי אז $N_{f'}$ היא הרשת השיורית בסוף השלב הנוכחי ובתחילת השלב הבא.
לכן כדי להוכיח את השערה 1 מספיק להראות כי $\delta(s,t) \geq k+1$ ב-$N_{f'}$.

---

[13] מצגת – רשתות זרימה עמ' 6.
[14] מצגת – רשתות זרימה עמ' 12 סעיף 2.

הוכחה

תהי $N_{f'}$ הרשת השיורית בסוף השלב הנוכחי ותהי קשת $(u,v)\in N_{f'}$.
לכל $v\in V$ נגדיר את הרמה (Level) $L(v)$ – כמספר השכבה של v ב-L_f של N_f.

<u>טענת עזר 1</u>: לכל קשת $(u,v)\in N_{f'}$, $L(v)\le L(u)+1$.

נניח כי (u,v) שייכת גם ל-N_f (קשת ישנה):
אם נניח כי $L(u)$ השכבה של u – אז יש ל$L(v)$ שלוש אפשרויות:

1. v ו-u באותה שכבה - $L(v)=L(u)$ (אס"ם $(u,v)\in L_f$).
2. v בשכבה הקודמת ל-u - $L(v)=L(u)-1$.
3. v בשכבה העוקבת ל-u - $L(v)=L(u)+1$.

נניח כי (u,v) חדשה ב-$N_{f'}$ (שלא הייתה ב-N_f):
אז דרך (v,u) זרמה זרימה כלשהי במהלך השלב הנוכחי בו מצאנו זרימה חוסמת.
לכן (v,u) השתייכה ל-L_f (כי רק בה מוצאים זרימה חוסמת). מכאן נובע כי:

$$L(u)=L(v)+1\Longrightarrow L(v)=L(u)-1\Longrightarrow L(v)<L(u)+1$$

לסיכום, כל קשת (u,v) (חדשה או ישנה) ב-$N_{f'}$ מקיימת: $L(v)\le L(u)+1$.

יהי P מסלול קצר ביותר מ-s ל-t ב-$N_{f'}$ כאשר $L=\delta(s,t)=|P|$ ב-$N_{f'}$.
נסמן: $P=\langle s=v_0,v_1,\dots,v_l=t\rangle$.

<u>טענת עזר 2</u>: $|P|=L\ge k$

אם נתבונן בסדר מספרי השכבות המתאימה לצומתי P:

$$P=\langle s=\underbrace{v_0}_{L(v_0)=0},v_1,\dots,\underbrace{v_l}_{L(v_l)=k}=t\rangle$$

נראה כי מספר השכבה לא גדל ביותר מ-1. לכן כדי להגיע מ-0, מספר השכבה של המקור s, אל k, מספר השכבה של t, יש לבצע לפחות k צעדים (בהם מספר השכבה גדל ב-1 לכל היותר). לכן אורך המסלול הוא לכל הפחות k, כלומר $L\ge k$.

<u>טענת עזר 3</u>: $|P|=L\ge k+1$

מהוכחת טענת עזר 1, ראינו כי קשת (u,v) מקיימת $L(v)=L(u)+1$ רק אם $(u,v)\in L_f$. מכאן נובע כי אורכו של P הוא בדיוק k אם כל קשת מגדילה את L בסדרת מספרי שכבה ב-1 בדיוק. מצב זה אפשרי רק אם כל קשת ב-P שייכת ל-L_f.
כלומר $P\subseteq L_f$ וזה לא יכול להיות כיוון ש-g חסמה את כל המסלולים ב-L_f במהלך השלב. לכן $|P|=\delta(s,t)=L\ge k+1$ מ.ש.ל.

ניתוח זמן ריצה

1. אתחול - $O(|E|)$.
   2. בנה את הרשת השיורית N_f - $O(|E|)$.
   3. כל עוד יש מסלול מ-s ל-t ב-G_f - $O(|V|)$ איטרציות (הוכח קודם).
      א. בנה את רשת השכבות L_f של N_f - $O(|E|)$ (הוכח קודם).
      ב. מצא זרימה חוסמת g ב-L_f - $O(|V||E|)$ (הוכח קודם).
      ג. עדכן ב-N: $f \leftarrow f + g$ - $O(|E|)$ לכל היותר.
      ד. בנה מחדש את N_f - $O(|E|)$.
   4. החזר את f - $O(1)$.

מסקנה 2; אלגוריתם דיניץ מתבצע ב$O(|V|^2|E|)$.

רשת זרימה עם קיבול 1 [15](מקרה פרטי)

הקדמה

חישוב החסם של אלגוריתם דיניץ שראינו קודם נכון עבור רוב המקרים שבהם רשת הזרימה מורכבת מגרף כללי עם קיבולים כלשהן. ישנן רשתות עבורן ניתן למצוא חסמים טובים יותר. אחת מהן היא רשת זרימה בעלת קשתות עם קיבול 1 בלבד. מקרה פרטי זה אינו משפיע על מבנה האלגוריתם אלא רק על זמן הריצה.

שיפור החסם על עלות שלב

קודם ראינו כי כל שלב באלגוריתם דיניץ עולה $O(|V||E|)$ וזה נובע מכך שניתן למצוא לכל היותר $O(|E|)$ מסלולים משפרים ב-$O(|V|)$.
נתבונן ברשת בתחילת שלב כלשהו t, ונסמן את מספר השכבות ברשת זו כ-$k + 1$.
ברשת זו, אורך כל מסלול שיפור הינו k. נשים לב כי עלות מציאת כל מסלול שיפור היא $O(k)$.

כמה מסלולי שיפור יכול האלגוריתם למצוא בכל שלב במקרה זה?

ברשת השכבות, לכל קשת קיבול 1 או 2 (אם הזרמנו 1 על הקשת ההפוכה) ולכן סך הקיבולות של הגרף קטן או שווה ל-$2|E|$. בכל פעם שמוצאים מסלול שיפור בגרף יש בו k קשתות ומזרימים בו לפחות 1, ולכן סך הקיבולות בגרף יורד לפחות ב-k.
מכאן שמספר מסלולי השיפור שניתן למצוא ברשת השכבות הוא לכל היותר $\frac{2|E|}{k}$.
במילים פשוטות כל קשת בגרף השכבות תהיה רוויה אחרי השתתפות במסלול שיפור אחד או שניים. אם עלות מציאת כל מסלול שיפור היא $O(k)$ אז עלות

---
[15] מקור-[1] עמ' 15.

מציאת כל המסלולים עם העדכונים הדרושים היא: $\frac{2|E|}{k}O(k) = O(|E|)$.
קודם ראינו גם כי עלות שאר פעולות התחזוקה והבנייה בכל שלב היא: $O(|E|)$ לכן נובע כי במקרה הנזכר לעיל עלות כל שלב חסומה על ידי $O(|E|)$. כלומר, נכון לעכשיו הוכחנו כי אלגוריתם דיניץ רץ בסיבוכיות של $O(|V||E|)$ על רשת זרימה עם קיבול 1.

שיפור החסם של כמות השלבים

הקדמה

עבור חתך $(S, V\backslash S)$, <u>קשת חוצה</u> היא קשת שעוברת מS ל-$V\backslash S$, <u>וקיבולת החתך</u> היא סכום קיבולות הקשתות החוצות אותו.
נתבונן שוב בשלב בו המרחק מ-s ל-t שווה k. נסמן ב-f את הזרימה בתחילת השלב. נגדיר חתכים $(S_i, V\backslash S_i)$ <u>ברשת השיורית</u> N_f, כאשר S_i היא קבוצת כל הצמתים במרחק עד i מ-s. נסמן: $V_i = \{v \in V | \delta(s, v) = i\}$. כאשר V_i היא "השכבה ה-i" של G_f.

אבחנה

לפי התכונות של BFS לא ייתכן שישנן קשתות משכבה i לשכבה גדולה מ- $i + 1$ במקרה הנראה לעיל. מכאן נובע כי הקשתות מV_i ל-V_{i+1} הן היחידות החוצות את החתך מ-S_i ל-$V\backslash S_i$ והמסקנה שהחתכים: $(S_i, V\backslash S_i)$ ברשת השיורית זרים בקשתות החוצות.

כיוון שאנו מתבוננים בשלב בו המרחק מ-s ל-t שווה k, אז יש לכל היותר k חתכים כאלו עם סך הכל $|E|$ קשתות בהם לכל היותר. מכאן נובע כי ממוצע הקשתות לכל חתך כזה הוא לכל היותר $\frac{|E|}{k}$. לפיכך, קיים ביניהם חתך אשר מספר הקשתות בו הוא לכל היותר $\frac{|E|}{k}$. כיוון שקיבולי כל הקשתות הם לכל היותר 2, קיבולו של חתך זה חסום על ידי $\frac{2|E|}{k}$.
במשפט: "Max Flow-Min Cut[16]" שהוכחנו בהרצאה ראינו כי "גודל זרימה מקסימלית שווה לקיבולת החתך המינימלי". כלומר, גודל זרימת המקסימום במקרה הפרטי הנזכר לעיל חסום על ידי $\frac{2|E|}{k}$. בגלל שכל מסלול שיפור מגדיל את הזרימה ב-1 לכל הפחות, אז $\frac{2|E|}{k}$ הינו חסם על מספר מסלולי השיפור שהאלגוריתם ימצא החל משלב זה ועד סוף ביצועו.

נתבונן בשלב בביצוע האלגוריתם שבו לראשונה $\delta(s,t) \geq \sqrt{2|E|}$. מהטענה

[16] מצגת – רשתות זרימה עמ' 12 סעיף 2.

שהוכחה קודם כי מרחקו של t מ-s גדל ממש בכל שלב, נסיק כי לפניו היו לכל היותר $\sqrt{2|E|}$ שלבים. לכן מן השלב הנוכחי ועד תום ריצת האלגוריתם יתבצעו לכל היותר $\frac{2|E|}{k}$ איטרציות, ולכן לכל היותר $\frac{2|E|}{k}$ שלבים. מכיוון ש-$k \geq \sqrt{2|E|}$, נסיק כי יישארו לכל היותר $\frac{2|E|}{\sqrt{2|E|}} \leq \sqrt{2|E|}$שלבים. כלומר, קיבלנו כי בסך הכל מספר השלבים שמבצע האלגוריתם במהלך ריצתו הינו לכל היותר $\sqrt{2|E|}$ שלבים שבוצעו ועוד $\sqrt{2|E|}$ שיבוצעו. לכן במקרה הנזכר לעיל החסם על מספר השלבים הוא: $O(\sqrt{|E|})$.

מסקנה 3; עבור רשת זרימה עם קיבול 1 זמן הריצה של אלגוריתם דיניץ חסום על ידי $O(|E|^{\frac{3}{2}})$.

שיטות

נרצה לבדוק האם התיאוריה תואמת את הפרקטיקה.
נשווה בין האלגוריתמים[17] באופן תיאורטי ולאחר מכן נממש אותם בשפת תכנות.
ניצור כמה סוגים שונים של רשתות זרימה (אפרט בהמשך) ונבדוק באופן ניסויי כיצד מתנהגים האלגוריתמים בכל אחד מהניסויים והאם ההשערה התיאורטית שנובעת מההשוואה ביניהם תואמת לתוצאות הניסוי (הפרקטיקה במקרה הזה).

הכנות לניסוי

תיאור משאבים וכלים

רשתות זרימה

ניצור ארבעה גרפים שונים. לכל גרף יהיו שני העתקים, הראשון – קיבולים שלמים גדולים והשני – עם קיבולים שלמים קטנים. בנוסף, לגרפים מרובי הקשתות ניצור עותק שלישי שלו יהיו קיבולים 1.

גרף 1 (גדול רב קשתות) – $|V| = 10^5$, $|E| = O(|V|^2) \approx 27M$.
גרף 2 (גדול דל קשתות) – $|V| = 10^5$, $|E| = O(|V|) \approx 400K$.
גרף 3 (קטן רב קשתות) – $|V| = 10^3$, $|E| = O(|V|^2) \approx 400K$.
גרף 4 (קטן דל קשתות) – $|V| = 10^3$, $|E| = O(|V|) \approx 7K$.

קיבול גדול – ערך שלם בטווח: $(1, 2^{15} - 1)$.
קיבול קטן – ערך שלם בטווח: $(1, 2^{10} - 1)$.
קיבול 1 – 1.

אלגוריתמים

אלגוריתם 1 – $Ford - Fulkerson$
אלגוריתם 2 – $Edmonds - Karp$
אלגוריתם 3 – $Dinitz$

כמו כן, אממש גם אלגוריתמי עזר דרושים כמו BFS, DFS, $Queue$ וכו'.

[17] אלגוריתם פורד-פולקרסון בגרסה המקורית, אלגוריתם אדמונדס-קרפ ואלגוריתם דיניץ.

יצירת המשאבים והכלים

בניית הגרפים

לבניית הגרפים השתמשתי בכלים הבאים:

1. סביבת עבודה – Eclipse.
2. שפת תכנות – Java 11.
3. ספריות מיוחדות – Graph stream (ספרייה למימוש בסיסי של גרף).

מבנה הנתונים שהשתמשתי לתיאור הגרף הוא AdjacencyListGraph. שזה בעצם מימוש של גרף כרשימות שכנויות. בחרתי את המימוש הנזכר לעיל כיוון שהוא עובד בצורה מקבילית ונחשב לחסכוני בזיכרון (במיוחד בגרפים גדולים ורבי קשתות).

נרצה שהניסוי יתבצע על גרפים שמתארים רשת מציאותית ככל הניתן. כיום משערים שרוב הרשתות הנפוצות כמו: רשתות חברתיות, האינטרנט וכיוצא בזאת הן רשתות "Scale-Free" – בהן קיימת היררכיה של חשיבות בין הצמתים כך - שככל שדרגת הצומת גדולה יותר עולה הסיכוי שיהיו לו יותר קשתות. כלומר, צמתים "חשובים" הם בעלי דרגה גדולה יותר ולכן הגיוני שיהיו להם יותר קשתות. המודל של Barabási-Albert[18] עוזר לנו לבנות רשת כזו (לא מכוונת). וקיים מימוש שיוצר גרף לפי המודל שלו בספריה המוזכרת לעיל.

את המימוש המקורי התאמתי כדי שיצור גרפים מכוונים כך שאם בגרף הלא מכוון המקורי הקשת (u, v) לא קיימת אז יש סיכוי שתיווצר קשת בכיוון ההפוך - (v, u) (בגרפים רבי קשתות הסיכוי הוא 1 ל2 ובדלי קשתות 1 ל-n). התאמה זו נועדה לאזן את הרשת כיוון שהמודל המקורי יוצר גרף בו ככל שמתקרבים לצומת הבור יש יותר קשתות בין הצמתים ולכן הקשתות שיוצאות מהמקור יהפכו לרוויות מהר יותר. התאמה זו יוצרת גרף "מעניין" עם יותר מסלולים לא טריוויאליים בין המקור לבור ומנסה לדמה רשתות אמיתיות ונפוצות היום. בנוסף, הגדרתי, מטעמי נוחות, כי הצומת הראשונה – תהיה המקור והצומת האחרונה היא הבור (בייצוג של הגרף כמטריצת שכנויות).

השליטה על כמות הקשתות בכל גרף תלויה בפרמטר seed שה-Generator מקבל. בדומה לאלגוריתמים רנדומליים, ה-seed שולט על ממוצע הקשתות היוצאות מכל צומת (תוספת של קשתות נגדיות מתבצעת על ידי ההתאמה שהזכרתי קודם). ביצרת הגרפים רבי הקשתות ה-seed היה $O(|V|)$ ובדלי קשתות $O(\sqrt{|V|})$.

הגרפים מיוצאים כמטריצת שכנויות לקובץ טקסט שבו נעשה שימוש בהמשך. כל קובץ טקסט מכיל גרף (כלומר סך הכל ישנם ארבעה קבצים: G#.txt) ובנוי כך שבשורה הראשונה רשום את כמות הצמתים וכמות הקשתות (משמאל לימין) ומהשורה השנייה והלאה רשומה מטריצת שכנויות המתארת את הגרף.

קבצי המקור בהם ניתן לראות את מימוש המודל (לאחר ההתאמה) ודוגמא ליצירת גרף מצורפים בקובץ ה-ZIP של ההגשה.

[18] מקור - [7].

בניית רשתות הזרימה

לבניית רשתות הזרימה מהגרפים שיצרנו קודם השתמשתי בכלים הבאים:
1. סביבת עבודה – Visual Studio.
2. שפת תכנות – C.

נממש רשת זרימה כמטריצת שכנויות משודרגת:

$$A[i][j] = \begin{cases} (i,j) \in E : c(i,j) \\ \quad else : 0 \end{cases}$$

יהיו לנו בסך הכל עשר רשתות שונות המבוססות על הגרפים שיצרנו וכל רשת תשמר בקובץ text נפרד כפי שתיארתי קודם. ההקצאה הרנדומלית של הקיבולים ושמירתם מתבצעת בעזרת פונקציית עזר בשם void CapacityBuild().

בשלב זה יש בידנו קבצי טקסט המכילים מידע שאותו נצטרך לעבד לרשת זרימה בתוכנית בה ממומשים האלגוריתמים. כדי לייצג קשת ברשת הזרימה יצרתי מבנה:

```
typedef struct FlowEdge {
    long capacity;
    long flow;
}FlowEdge;
```

מבנה קשת זרימה

מכיל: קיבולת מקסימלית וזרימה נוכחית.

כך שרשת הזרימה שלנו היא: FlowEdge FlowNetwork[V][V] - מערך דו-ממדי של מבני קשתות זרימה. אתחול מבנה הנתונים הזה מתבצע בעזרת פונקציית עזר בשם void GraphInit() המעתיקה את המידע שבקובץ הטקסט לרשת הזרימה.

הערות:
1. חשוב לציין כי המימוש הזה טוב לגרפים רבי קשתות כאשר $|E| = O(|V|^2)$.
2. קבצי המקור בהם ניתן לראות את המימושים ודוגמאות מצורפים בקובץ ה-ZIP של ההגשה.

מימוש האלגוריתמים (ממליץ לקרוא חלק זה כאשר צופים בקוד המקור עצמו)

למימוש האלגוריתמים אשתמש בכלים הבאים:
1. סביבת עבודה – Visual Studio.
2. שפת תכנות – C.

int Ford_Fulkerson()

מוצאים את המסלול המשפר באמצעות int FF_DFS(int location, int flow) – שזה מימוש של אלגוריתם DFS עם תוספת קטנה שעוזרת לנו למצוא את הקיבול השיורי המינימלי. כל עוד יש מסלול משפר בין המקור לבור סוכמים את הקיבול השיורי שלו ומעדכנים את ערך הזרימה המקסימלית. כאשר לא נותרים מסלולים משפרים מחזירים את ערך הזרימה המקסימלית.

int Edmonds_Karp()

מוצאים את המסלול המשפר באמצעות Boolean EK_BFS() – מימוש של אלגוריתם BFS שמוצא מסלול משפר קצר ביותר בין המקור לבור. המסלול המשפר נשמר במערך הגלובלי - int p[V] עליו עוברים בפונקציה עצמה כדי למצוא את הקיבול השיורי המינימלי. כמו כן, לאחר שערך הקיבול השיורי המינימלי נמצא מעדכנים את הזרימה המקסימלית ואת הזרימה לאורך כל הקשתות שבמסלול שמצאנו. כאשר לא נותרים מסלולים משפרים מחזירים את ערך הזרימה המקסימלית.

הערה:
כדי לממש את Boolean EK_BFS() מימשתי תור בצורה של מערך גלובלי כפי שנלמד בקורס מת"מ. לתור - int q[V] יש את הרוטינות הטריוויאליות: void Enqueue(int x) ו-int Dequeue() (הכנסה והוצאה).
בנוסף, כדי לממש את הרעיון של הצבעים שלמדנו בהרצאה (בשביל שלא נעבור על אותה צומת פעמיים בריצת ה-BFS) יצרתי מערך גלובלי - int color[V] ו-enum Color (לבן, אדום ושחור).

int Dinitz()

המימוש של אלגוריתם זה שומר על הרעיון התיאורטי אבל לא מתחזק את כל מבני הנתונים והרוטינות שנמצאות בתיאור האלגוריתם המקורי. זאת מכיוון שניתן לממש את אותו הרעיון באופן יעיל יותר בצורה מעשית.

המימוש שבחרתי מתבסס על הרעיון[19] למימוש שהציעו פרופ' שמעון אבן ופרופ' אלון איתי בו מטפלים בdead-ends (מבוי סתום) on-the-fly. כלומר, הרוטינה clean שראינו קודם ממומשת תוך כדי חיפוש הזרימה החוסמת ולא בנפרד.

יצרתי מערך לוקאלי - next[V] בו next[i] הוא האינדקס של הקשת הלא משומשת הבאה במערך השכניות של הצומת i.

כדי לממש את הרעיון של <u>גרף השכבות</u> יצרתי מערך גלובלי - level[V] בו נשמור את מספר השכבה של כל צומת. כלומר מספר השכבה של הצומת ה-i ברשת הזרימה שלנו היא level[i]. החישוב של השכבות מתבצע באמצעות Boolean D_BFS() – מימוש של BFS הדומה לזה שתיארתי קודם רק שבנוסף למציאת מסלול <u>משפר</u> קצר ביותר בין המקור לבור - הוא גם מחשב את מספר השכבה של כל צומת בדיוק כמו החישוב של d[v] שלמדנו בהרצאה. חישוב <u>הזרימה החוסמת</u> מתבצע באמצעות int D_DFS(int location, int * next, int flow) - מימוש ל-DFS שמוצא מסלולים משפרים בין המקור לבור בגרף השכבות, מחשב ומחזיר את הזרימה החוסמת. האלגוריתם רץ כל עוד יש מסלול משפר בין המקור לבור בגרף השכבות.

---
[19] מקור – [8]

השערת מחקר

מבחינה תיאורטית הוכחנו כי אלגוריתם דיניץ הוא היעיל מבין האלגוריתמים ולכן נצפה לראות שזמן הריצה שלו (מעשית) נמוך יותר בהשוואה לשני האלגוריתמים האחרים. במיוחד בגרפים רבי קשתות שם הוא אמור להיות עדיף בהשוואה לאלגוריתם אדמונדס-קרפ שזמן הריצה שלו (תיאורטית) תלוי בכמות הקשתות יותר מאלגוריתם דיניץ.

במקרה שהקיבולים הם 1 נצפה לראות שיפור משמעותי בזמן הריצה של כל האלגוריתמים ושל דיניץ בפרט. בנוסף, כיוון שזמן הריצה של אלגוריתם פורד-פולקרסון הוא פסודו-פולינומיאלי התלוי בגודל הזרימה המקסימלית אני משער כי ברשתות עם קיבול 1 זמן הריצה שלו יהיה דומה ואפילו טוב יותר משל אדמונדס-קרפ ומצד שני זמן הריצה שלו יהיה ארוך מאוד ברשתות עם הקיבולים הגדולים.

תאוריה של הניסוי

1. מערכת הפעלה – Windows 10 64b.
2. סביבת עבודה – Visual Studio.
3. ספריות –

< stdio.h > - ספרייה סטנדרטית לפעולות I/O.
< stdlib.h > - בשביל int rand().
< time.h > - כדי למדוד זמן באמצעות השעון.

4. סימונים –

<u>בקוד :</u>
V – מספר הצמתים ברשת הזרימה.
s – האינדקס של צומת המקור (מוגדר להיות 0).
t – האינדקס של צומת הבור (מוגדר להיות $V - 1$).
MAX_CAP – הקיבולת המקסימלית ברשת הזרימה (משתמשים בבנייה של רשתות הזרימה בפונקציה void CapacityBuild() שהזכרתי קודם).
inf – אינסוף, מוגדר להיות $2^{31} - 1$.

<u>בספר הפרויקט :</u>
G# – גרף # (כפי שתואר בשלב ההכנות לניסוי).
G#.1C - רשת זרימה שמורכבת מגרף G# עם קיבול 1.
G#.SC - רשת זרימה שמורכבת מגרף G# עם קיבול קטן.
G#.BC - רשת זרימה שמורכבת מגרף G# עם קיבול גדול.

5. מדידת זמנים – נקצה שני משתנים מטיפוס clock_t (start ו- end) ונשתמש בפונקציה clock() כדי לשמור בהם את זמן ההתחלה והסיום של ריצת כל אלגוריתם. ההפרש ביניהם הוא הזמן שעבר ואותו נחלק בCLOCKS_PER_SEC כדי לחלץ את הזמן שעבר <u>בשניות</u>. (הערה : <u>זמן האתחול</u> של הרשת לא נכלל בחישוב הזמן אלא נטו ריצת האלגוריתמים).

מהלך הניסוי

יצרתי שלוש פונקציות בדיקה (אחת לכל אלגוריתם) -void FF_Test(char []),
void EK_Test(char[]) ו- void D_Test(char[]) המקבלות שם של קובץ טקסט המכיל רשת זרימה. כל אחת מן הפונקציות מאתחלות את רשת הזרימה על ידי קריאה לGraphInit(name). ברגע שהאתחול הסתיים נמדוד את הזמן (start), נקרא לאלגוריתם (אחד מהשלושה) ונמדוד שוב את הזמן (end). נדפיס את שם האלגוריתם, הזרימה המקסימלית שהוחזרה (למען בקרה) ומשך הזמן שלקח לאלגוריתם לרוץ כפי שהסברתי קודם.

הניסוי הכולל מורכב מעשרה שלבים (נפרט על כל אחד בנפרד בהמשך) המודדים את זמן ריצת האלגוריתמים על כל אחת מעשר רשתות הזרימה שהגדרנו קודם. על חמשת רשתות הזרימה הגדולות לא נריץ את פורד-פולקרסון כיוון שזמן הריצה שלו יהיה ארוך (מאוד) ומספיק להריץ אותו על רשתות הזרימה הקטנות כדי להסיק את המסקנות הדרושות.

שלב 1

הרצת EK_Test ו-D_Test על **G1. 1C**.

שלב 2

הרצת EK_Test ו-D_Test על **G1. SC**.

שלב 3

הרצת EK_Test ו-D_Test על **G1. BC**.

שלב 4

הרצת EK_Test ו-D_Test על **G2. SC**.

שלב 5

הרצת EK_Test ו-D_Test על **G2. BC**.

שלב 6

הרצת FF_Test, EK_Test ו-D_Test על **G3. 1C**.

שלב 7

הרצת FF_Test, EK_Test ו-D_Test על **G3. SC**.

שלב 8

הרצת FF_Test, EK_Test ו-D_Test על **G3. BC**.

שלב 9

הרצת FF_Test, EK_Test ו-D_Test על **G4. SC**.

שלב 10

הרצת FF_Test, EK_Test ו-D_Test על **G4. BC**.

הערה:

בשלבים 1-5 $V = 1000$, ובשלבים 6-10 $V = 10000$.

השערות הניסוי

שלב 1

ראינו שבמקרה הפרטי בו רשת הזרימה עם קיבול 1 זמן הריצה של אלגוריתם דיניץ חסום (תיאורטית) על ידי: $O\left(|E|^{\frac{3}{2}}\right)$. לכן אני משער כי זמן הריצה של אלגוריתם דיניץ (מבחינה מעשית) יהיה הרבה יותר מהיר מזה של אלגוריתם אדמונדס-קרפ.

שלבים 2-3

ראינו קודם כי תיאורטית זמן הריצה של אלגוריתם דיניץ חסום על ידי: $O(|V|^2|E|)$
וזמן הריצה של אלגוריתם אדמונדס-קרפ חסום על ידי: $O(|V||E|^2)$.
תיאורטית, זמן הריצה של אלגוריתמים אלו לא תלוי בגודל הקיבול ולכן ההשערה שלי תהיה זהה עבור שני השלבים (כך גם בנוגע להמשך במקרים דומים).
נשים לב כי רשת הזרימה בשלבים אלו היא מרובת קשתות. כלומר, $|E| = O(|V|^2)$ ולכן אני משער שמבחינה מעשית אלגוריתם דיניץ יהיה עדיף על אלגוריתם אדמונדס-קרפ.

שלבים 4-5

נשים לב כי בשלבים אלו רשת הזרימה היא דלת קשתות. ולכן אני משער כי מבחינה מעשית זמן הריצה של אלגוריתם אדמונדס-קרפ יהיה מהיר יותר בהשוואה לזמן הריצה שלו בשלבים 2-3 (השערה זו נובעת מכך שמבחינה תיאורטית כמות הקשתות אמורה להשפיע יותר על זמן הריצה המעשי של אלגוריתם אדמונדס-קרפ). בנוסף, אם המימוש של הגרף היה כרשימת שכנויות ולא כמטריצת שכנויות הייתי מצפה לראות מבחינה מעשית גם שיפור בזמן הריצה של אלגוריתם דיניץ בהשוואה לזמן הריצה שלו בשלבים 2-3 שכאמור תיאורטית זמן הריצה שלו מושפע יותר מכמות הצמתים ופחות מכמות הקשתות.

בנוגע לזמן הריצה (מבחינה מעשית) של אלגוריתם דיניץ מול זמן הריצה של אלגוריתם אדמונדס-קרפ אני משער שכמו בשלבים הקודמים הוא אמור להיות עדיף.

שלב 6

בדומה לשלב 1 אני משער כי מבחינה מעשית זמן הריצה של אלגוריתם דיניץ יהיה מהיר יותר מזה של אלגוריתם אדמונדס-קרפ ומזה של אלגוריתם פורד-פולקרסון. כמו כן, אני משער כי זמן הריצה המעשי של אלגוריתם פורד-פולקרסון יהיה מהיר יותר בהשוואה לזה של אלגוריתם אדמונדס-קרפ וזאת מכיוון שמבחינה תיאורטי הזרימה המקסימלית צריכה להיות קטנה (יחסית) כאשר רשת הזרימה בעלת קיבול 1.

שלבים 7-8

בדומה לשלבים 2-3 אני משער כי מבחינה מעשית זמן הריצה של אלגוריתם דיניץ יהיה מהיר יותר מזה של אלגוריתם אדמונדס-קרפ ומזה של אלגוריתם פורד-פולקרסון. בנוסף, כיוון שעכשיו ,מבחינה תיאורטית, הזרימה המקסימלית אמורה להיות גדולה אני משער כי זמן הריצה המעשי של אלגוריתם פורד-פולקרסון יהיה איטי (מאוד) בהשוואה לזה של אלגוריתם אדמונדס-קרפ וזה של אלגוריתם דיניץ (במיוחד בשלב 8 - בו הקיבולים אמורים להיות תיאורטית גדולים עוד יותר).

שלבים 9-10

אני משער כי מבחינה מעשית זמני הריצה אלגוריתם דיניץ ואלגוריתם אדמונדס-קרפ יהיו דומים. זאת מכיוון שבשלבים אלו הרשת קטנה ודלת קשתות וכפי שראינו קודם מבחינה תיאורטית ההבדל בין האלגוריתם אמור להיות זניח במצב הנזכר לעיל. בנוסף, אני משער כי מבחינה מעשית, זמן הריצה של אלגוריתם אדמונדס-קרפ וזמן הריצה של אלגוריתם פורד-פולקרסון יהיו מהירים יותר בהשוואה לזמני הריצה שלהם בשלבים 7-8. כמובן שמבחינה תיאורטית זמן הריצה של אלגוריתם פורד-פולקרסון אמור להיות איטי מזה של שני האלגוריתמים האחרים ולכן אני משער כי מבחינה מעשית זמן הריצה של אלגוריתם פורד-פולקרסון יהיה איטי (מאוד) בהשוואה לזמני הריצה המעשיים של שני האלגוריתמים האחרים.

תוצאות הניסוי

Algorithm / שלב	Ford-Fulkerson		Edmonds-Karp		Dinitz	
	זרימה מקסימלית	זמן בשניות	זרימה מקסימלית	זמן בשניות	זרימה מקסימלית	זמן בשניות
1			299	56.8	299	4.4
2			152274	135	152274	4.5
3			4732325	127.8	4732325	4.5
4			25311	25.1	25311	4.5
5			737898	25.8	737898	4.7
6	633	0.7	633	1.3	633	0.09
7	351774	345.5	351774	2.6	351774	0.06
8	11786632	2h+	11786632	3.08	11786632	0.07
9	2816	7.53	2816	0.04	2816	0.04
10	85104	143.59	85104	0.05	85104	0.06

ניתוח תוצאות ומסקנות

ניתוח תוצאות

שלב 1

ההשערה כי זמן הריצה של אלגוריתם דיניץ (מבחינה מעשית) יהיה הרבה יותר מהיר מזה של אלגוריתם אדמונדס-קרפ תואמת לתוצאות הניסוי וניתן לראות כי זמן הריצה של אלגוריתם אדמונדס-קרפ בשלב זה הוא 56.8 שניות מול 4.4 שניות של אלגוריתם דיניץ.

שלבים 2-3

ההשערה כי מבחינה מעשית אלגוריתם דיניץ יהיה עדיף על אלגוריתם אדמונדס-קרפ תואמת לתוצאות הניסוי. ניתן לראות כי זמן הריצה של אלגוריתם אדמונדס-קרפ אכן איטי מזה של אלגוריתם דיניץ (135 ו-127.8 שניות מול 4.5 ו-4.5 שניות).

שלבים 4-5

ההשערה כי בשלבים אלו (כמו בשלבים הקודמים) זמן הריצה של אלגוריתם דיניץ מהיר יותר בהשוואה לזה של אלגוריתם אדמונדס-קרפ (מבחינה מעשית) תואמת לתוצאות הניסוי. גם ההשערה כי מבחינה מעשית זמן הריצה של אלגוריתם אדמונדס-קרפ יהיה מהיר יותר בהשוואה לזמן הריצה שלו בשלבים 2-3 תואמת לתוצאות הניסוי (2-3 : 135 ו-127.8 שניות מול 4-5 : 25.1 ו-25.8 שניות).

אפשר גם לראות כי זמן הריצה של אלגוריתם דיניץ לא השתפר ואפילו נגרע בהשוואה לשלבים 2-3 למרות הירידה החדה בכמות הקשתות וזאת כפי ששיערתי נובע מכך שמבחינה תיאורטית זמן הריצה המעשי של אלגוריתם דיניץ אמור להיות מושפע פחות מכמות הקשתות ויותר מכמות הצמתים.

שלב 6

ההשערה כי מבחינה מעשית זמן הריצה של אלגוריתם דיניץ יהיה מהיר יותר מזה של אלגוריתם אדמונדס-קרפ ומזה של אלגוריתם פורד-פולקרסון (בדומה לשלב 1) תואמת לתוצאות הניסוי (קל לראות). כמו כן, גם ההשערה כי זמן הריצה המעשי של אלגוריתם פורד-פולקרסון יהיה מהיר יותר בהשוואה לזה של אלגוריתם אדמונדס-קרפ תואמת לתוצאות (0.7 שניות מול 1.3 שניות).

שלבים 7-8

ההשערה כי מבחינה מעשית זמן הריצה של אלגוריתם דיניץ יהיה מהיר יותר מזה של אלגוריתם אדמונדס-קרפ ומזה של אלגוריתם פורד-פולקרסון (בדומה לשלבים 2-3) תואמת לתוצאות (קל לראות). וגם ההשערה כי זמן הריצה המעשי של אלגוריתם פורד-פולקרסון יהיה איטי (מאוד) בהשוואה לזה של אלגוריתם אדמונדס-קרפ וזה של אלגוריתם דיניץ תואמת לתוצאות (7-8 : 345.5 שניות ושעתיים (!) של פורד-פולקרסון מול 2.6 ו-3.08 של אדמונדס-קרפ ו-0.06 ו-0.07 של דיניץ).

שלבים 9-10

ההשערה כי מבחינה מעשית זמני הריצה אלגוריתם דיניץ ואלגוריתם אדמונדס-קרפ יהיו דומים תואמת לתוצאות הניסוי (9-10 : 0.04 ו-0.05 שניות של אדמונדס-קרפ מול 0.04 ו-0.06 של דיניץ). גם ההשערה כי מבחינה מעשית, זמן הריצה של אלגוריתם אדמונדס-קרפ וזמן הריצה של אלגוריתם פורד-פולקרסון יהיו מהירים יותר בהשוואה לזמני הריצה שלהם בשלבים 7-8 תואמת לתוצאות (יותר מורגש באדמונדס-קרפ: 7-8 : 2.6 ו-3.08 שניות מול 9-10 : 0.04 ו-0.05 שניות). בנוסף, גם ההשערה כי מבחינה מעשית זמן הריצה של אלגוריתם פורד-פולקרסון יהיה איטי (מאוד) בהשוואה לזמני הריצה המעשיים של שני האלגוריתמים האחרים תואמת לתוצאות (9-10 : 7.53 ו-143.59 מול הזמנים שהוצגו לעיל).

אבחנה

ניתן לראות כי הזרימה המקסימלית בשלב 3 (גרף גדול עם הרבה קשתות) קטנה יותר מזו שבשלב 8 (גרף קטן עם הרבה קשתות). אינטואיטיבית מצפים כי בגרף גדול יותר עם נתוני קיבול דומים הזרימה המקסימלית תגדל. כפי שלמדנו בהרצאות הזרימה המקסימלית בכלל תלויה בכמות המסלולים המשפרים הזרים בין המקור לבור. לכן השערה הגיונית היא שכנראה אבחנה זו נובעת מהבדלים טכניים בין הגרפים (יש יותר מסלולים משפרים בגרף 3 מאשר בגרף 1).

כדי לבדוק מעשית השערה זו כתבתי פונקציה בשם - `int augmentingPathNum()` המוצאת את מספר המסלולים המשפרים הזרים בגרף.

הרצתי את הפונקציה הנזכרת לעיל על שני הגרפים וקיבלתי כי בגרף 1 יש 165 מסלולים משפרים (שלב 3) ובגרף 3 יש 633 מסלולים משפרים (שלב 8). תוצאה זו אכן מאוששת את ההשערה כי מקור ההבדל בגדלי הזרימות המקסימליות נובע מהבדלים טכניים בין הגרפים.

השערה נוספת היא שהבדלי הגודל בין הזרימות המקסימליות יכולים לנבוע מכך שיש הבדל באורכי המסלולים המשפרים בין שני הגרפים. כלומר, ברגע שמסלול משפר מכיל יותר קשתות כך הסיכוי שקיבולו השיורי יהיה נמוך יותר. לצערי כיוון שאין ברשותי את הידע הסטטיסטי הדרוש להוכחת השערה זו נצטרך להסתפק בזה שככל הנראה מקור הסטייה הוא אכן בגלל השוני בכמות המסלולים. בנוסף גם אם השערה זו נכונה היא כנראה תהיה בעלת השפעה רק במקרים בהם משווים בין שני גרפים שוני גודל אך עם אותו סדר גודל של כמות מסלולים משפרים זרים.

<u>הערה:</u>
לכך שהגרפים שונים מבחינה טכנית בכמות המסלולים המשפרים הזרים אין השפעה על הניסוי הכולל כיוון שמטרתו היא לבדוק מה הם זמני הריצה של האלגוריתמים השונים על הרשתות השונות ולהשוות בין האלגוריתמים ולא בין הזרימות של הגרפים. כלומר, אם לדוגמא בשלב 8 יש יותר מסלולים משפרים אז זמן הריצה של כל האלגוריתמים מבחינה מעשית יהיה ארוך יותר אבל ההשוואה ביניהם לא נפגעת כיוון שערך זה משפיע על כולם באותו מידה (כולם רצים כל עוד יש מסלול משפר..).

מסקנות

- ✓ בממוצע זמן הריצה המעשי של אלגוריתם דיניץ נמוך יותר בהשוואה לשני האחרים.
- ✓ כמות הקשתות ברשת הזרימה משפיעה על זמן הריצה המעשי של אלגוריתם אדמונדס-קרפ יותר מאשר על זמן הריצה המעשי של אלגוריתם דיניץ. לדוגמא: זמן הריצה של אלגוריתם אדמונדס-קרפ בשלבים 4-5 הוא: 25.1 ו-25.8 שניות שזה שיפור משמעותי בהשוואה לזמן הריצה שבשלבים 2-3: 135 ו-127.8 שניות. בעוד שבשלבים אלו ההבדל בין זמן הריצה של דיניץ זניח.
- ✓ ברשתות זרימה עם קיבול 1 (שלבים 1 ו-6) יש שיפור של זמן הריצה המעשי של כל האלגוריתמים ושל דיניץ בפרט (כמו ההשערה התיאורטית).
- ✓ ברשתות זרימה עם קיבול 1 (שלב 6) זמן הריצה של אלגוריתם פורד-פולקרסון טוב יותר מזה של אלגוריתם אדמונדס-קרפ (תואם לחסמי זמני הריצה התיאורטיים).
- ✓ ברשתות זרימה עם קיבולים שלמים (גדולים וקטנים) זמן הריצה המעשי של אלגוריתם פורד-פולקרסון ארוך בהשוואה לשני האחרים.
- ✓ כמות המסלולים המשפרים הזרים ברשת הזרימה משפיע על גודל ערך הזרימה המקסימלית.

מקורות

[1] – Dinic, E.A (1970). "Algorithm for solution for a problem of maximum flow in a network with power estimation". *Soviet Math. Doklady* (Doklady) 11: 1277-1280.
[2] – Barami, T. (2018). "Algorithms Design lectures". Ben-*Gurion University*.
[3] – Kleiman, E. (2020). "Algorithms lectures". *ORT Braude College of Engineering*.
[4] – Dinic's algorithm. (2021, May.1), *Wikipedia*.
[5] – Kleinberg-Tardos. "Flow-Networks E.7".
[6] – Design of Algorithms – Practice 9.
[7] - Barabási–Albert model - . (2021, Jan.4), *Wikipedia*.
[8] - Dinic, E.A (2006). "Dinitz's Algorithm: The Original Version and Even's Version".