אלגוריתמים 2

הרצאה 1

Δ – האות דלתא, מסמנת את **מספר השכנים המקסימלי** בגרף G.

δ – האות דלתא קטנה, מסמנת את **מספר השכנים המינימלי** בגרף G.

ω – אומגה קטנה, מסמנת את **הקליקה הגדולה ביותר** בגרף G.

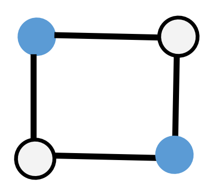
X – האות כי, מסמנת את **המספר הכרומטי הקטן ביותר** של הגרף G.

θ – האות תטא, מסמנת **את המספר המינימלי של קליקות** המכסות את הגרף G.

η - האות אטא, מסמנת את **המספר המינימלי של צמתים** שמכסים את כל הגרף G.

α – האות אלפא, מסמן את **הקבוצה הבלתי תלויה הגדולה ביותר** בגרף G.

הקבוצה הבלתי תלויה – כלומר בחירה של שני צמתים שאינם שכנים בגרף G, למשל בגרף הבא:

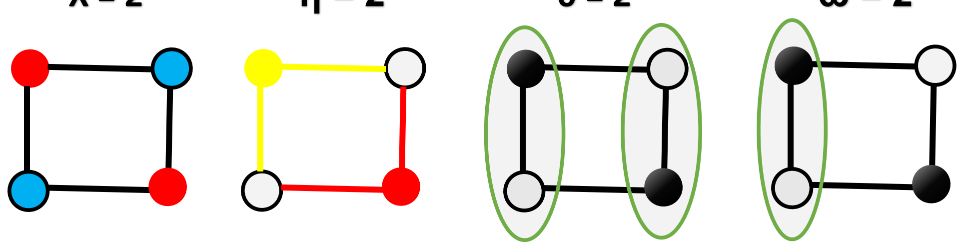


α = 2

מכיוון שניתן לראות שניתן לסמן שני צמתים בגרף כך ששני שכנים לא יסומנו.

- כעת נסתכל שוב על הגרף ונבדוק את כל שאר הפרמטרים:

ω = 2 **θ = 2** **η = 2** **X = 2**



נכתב על ידי אוהד אמסלם, תשפ"ב

מקסימום – כאשר נחפש את הפרמטרים ω או α נשאף תמיד למצוא את **הפתרון הכי טוב** וכאשר נצליח למצוא אותו נגדיר אותו **כמקסימום**.

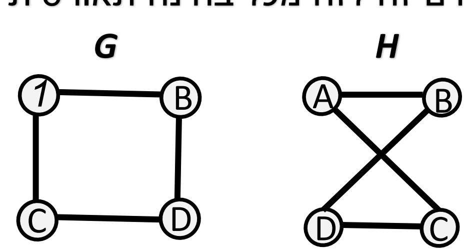
מקסימל – כאשר נחפש את הפרמטרים ω או α וכאשר **לא נצליח למצוא את הפתרון הכי טוב** נקרא לפתרון **מקסימל**.

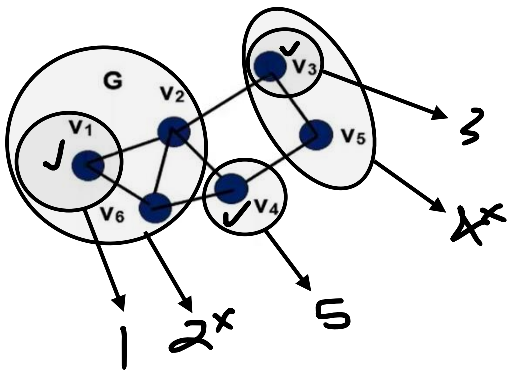
מינימום - כאשר נחפש את הפרמטרים X, θ או η נשאף תמיד למצוא את **הפתרון הכי טוב** וכאשר נצליח למצוא אותו נגדיר אותו **כמינימום**.

מינימל - כאשר נחפש את הפרמטרים X, θ או η וכאשר **לא נצליח למצוא את הפתרון הכי טוב** נגדיר אותו **כמינימל**.

- נוסחאות:
  - אם **$\Delta(G) = \delta(G)$** אז הגרף G הוא **r - רגולרי.**
  - $\alpha(G) = \omega(\bar{G})$
  - $X = \theta(\bar{G})$
  - $|V| = \alpha(G) + \eta(G)$
  - $|V| = \omega(\bar{G}) + \eta(G)$
  - $\omega(G) \le X(G) \le |v|$
  - $\alpha(G) \cdot X(G) \ge |v|$

- גרף **לא מכוון ללא מעגלים** נקרא **יער**.
- גרף **לא מכוון**, **קשיר וללא מעגלים** נקרא **עץ**.
- **כל עץ הוא גרף דו-צדדי**.
- **עץ פורש –** תת גרף קשיר וללא מעגלים (כלומר עץ) המכיל את כל הצמתים בגרף *G*.
- **עץ פורש מינימלי –** עץ פורש שהוא מינימלי בסכום משקלי הקשתות שלו מבין כל העצים הפורשים של הגרף *G*.
- **גרף איזומורפי –** נאמר שהגרף **G** הוא איזומורפי לגרף **H** אם יש התאמה בין הקודקודים של שני גרפים המשרה התאמה בין הקשתות, גרפים איזומורפיים הם זהים זה לזה מכל בחינה תאורטית לדוגמא הגרפים הבאים:





❖ אלגוריתם למציאת קבוצה בלתי תלויה מקסימלית:

G = (V, E) – גרף לא מכוון.

– **Greedy_independent_set (G, S)**

S = ∅ (קבוצה ריקה)

U = V (קבוצת הצמתים)

כל עוד הקבוצה U אינה ריקה (כלומר כל עוד יש בה צמתים):

1. בחר בקודקוד V בעל מספר השכנים הקטן ביותר בקבוצה U, הוסף אותו לקבוצה S.
2. הסר את הקודקוד V ואת כל שכניו מהקבוצה U.

   סוף ה- While.

3. הוסף את כל הקודקודים הנותרים ב- U ל- S.
4. החזר את הקבוצה S.

האלגוריתם שואף למצוא קבוצה בלתי תלויה מקסימום אבל לא בהכרח יצליח למצוא קבוצה כזו, מצד שני **הוא תמיד ימצא קבוצה בלתי תלויה מקסימל**.

לכן זהו אלגוריתם טוב אבל אינו אלגוריתם אופטימלי, כלומר הוא לא בהכרח יצליח למצוא קבוצה בלתי תלויה מקסימום.

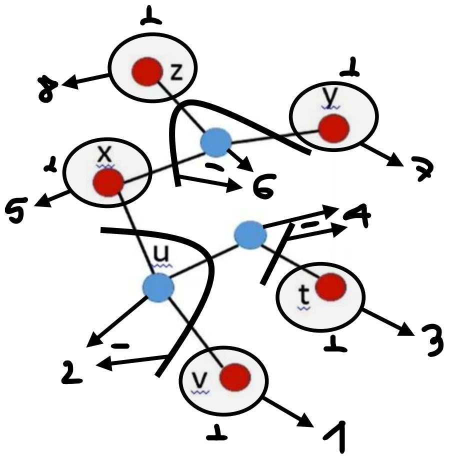
- נוסחה שמתייחסת לאלגוריתם הזה שכדאי להכיר:

$$|S| \geq \frac{|v|}{\Delta(G)+1}$$

לפי הדוגמא למעלה, יכול להתקבל S שונה בכל הרצה של האלגוריתם, תוצאה אפשרית אחת מיני רבות היא: **$S = \{V_1, V_3, V_4\}$** כמו שמסומן בדוגמא שבראש העמוד.

ובהתייחסות לנוסחה:

$$|s| \geq \frac{6}{4+1} = \left\lfloor \frac{6}{5} \right\rfloor = 1$$



❖ **אלגוריתם למציאת קבוצה בלתי תלויה מקסימום ביער:**

F = (V,E) – גרף שהוא יער

Greedy_Independent_Set_Forest (F, S) –

S = ∅ (קבוצה ריקה)

U = E (קבוצת הצלעות)

כל עוד הקבוצה U שונה מקבוצה ריקה (כלומר כל עוד יש בה צלעות):

1. בחר בצלע מהקבוצה U שהיא קצה בגרף כך שהצומת V הוא עלה והוסף אותו לקבוצה S.
2. הסר מהקבוצה U את הצמתים u, v שמרכיבים את הצלע שהוספה ל- S ואת כל הצלעות שיוצאות מהצלע (u, v)

סוף ה- While.

3. הוסף את כל הקודקודים הנותרים ב- U ל- S.
4. החזר את הקבוצה S.

אלגוריתם זה יניב תמיד קבוצה בלתי תלויה מקסימום.

בהתייחסות לדוגמא בראש העמוד האלגוריתם יניב את הקבוצה הבלתי תלויה מקסימום הבאה: **S = {v, t, x, z, y}**

- מכיוון שידוע לנו האלגוריתם הנ"ל הנותן את הקבוצה הבלתי תלויה מקסימום ניתן לקבוע **שהכיסוי מינימום** עבור כל **גרף שהוא יער** הוא **V – S**.

כלומר בדוגמא בראש העמוד כיסוי מינימום הוא הצמתים הצבועים בכחול ו- η = 3.

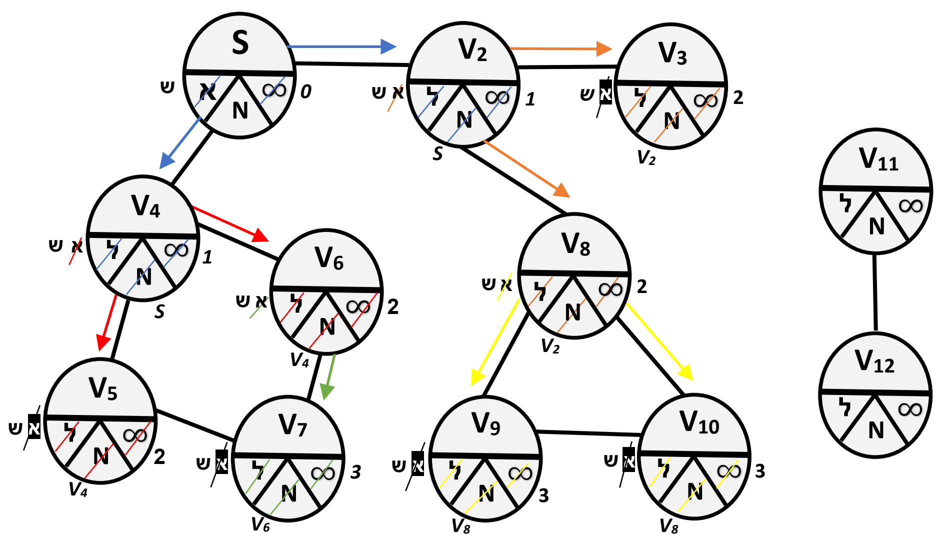
הרצאה 2

❖ אלגוריתם BFS:

S – הקודקוד ממנו האלגוריתם מתחיל להתפשט.
C[v] - מצב הקודקוד V מבחינת צבע (לבן, אפור, שחור), **המצב ההתחלתי הוא לבן**.
π[v] - מסמן מי האבא של הקודקוד V.
d[v] – מסמן מרחק מהצומת ההתחלתית S לקודקוד V_i, מרחק הוא בעצם המסלול הקצר יותר מהקודקוד S לקודקוד V_i, **המצב התחלתי הוא אינסוף**.

האלגוריתם עובד בעזרת תור.

Queue = {~~S~~, ~~V4~~, ~~V2~~, ~~V6~~, ~~V5~~, ~~V3~~, ~~V8~~, ~~V7~~, ~~V9~~, ~~V10~~} = Ø



	S	V_2	V3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}
C[v]	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ל	ל
d[v]	0	1	2	1	2	2	3	2	3	3	∞	∞
π[v]	NILL	S	V_2	S	V_4	V_4	V_6	V_2	V_8	V_8	NILL	NILL

- אלגוריתם זה מחזיר את **המסלול הקצר ביותר**, ניתן להשתמש בו גם כדי לבדוק כמה מסלולים קצרים ביותר יש מ- S לשאר הצמתים.
- האלגוריתם עובד גם על גרף מכוון וגם על גרף לא מכוון.

- התקדמות האלגוריתם הוא תמיד מקודקוד בצבע אפור לקודקוד בצבע לבן.
- ניתן לראות בדוגמא שהאלגוריתם מתחיל מרכיב קשירות אחד ולא ממשיך לרכיב קשירות אחר, מכאן נובע שאם ב- C[v] נשאר צבע לבן בסוף האלגוריתם אז ניתן להסיק כי הגרף G אינו קשיר.

G = (V, E) גרף מכוון או לא מכוון.

S הוא קודקוד בגרף G.

BFS (G, S) –

1. **לכל קודקוד X ששייך לקבוצת הקודקודים V-S בצע:**

   לבן = Color[x], כלומר צבע התחלתי לכל קודקוד פרט ל- S שווה לצבע לבן.

   $d[x] = \infty$, כלומר מרחק התחלתי לכל קודקוד פרט ל- S שווה לאינסוף.

   $\pi[x] = NiLL$, כלומר אבא התחלתי של כל קודקוד פרט ל- S שווה ל- NULL.

2. אפור = Color[S], כלומר צבע את הקודקוד S בצבע אפור.

   d[S] = 0, כלומר קבע כי המרחק של הקודקוד S לעצמו הוא 0.

   $\pi[S] = NiLL$, כלומר קבע כי לקודקוד S אין אבא.

   $Q = \emptyset$, כלומר צור תור ריק.

   Enqueue(Q, S), כלומר הכנס את הקודקוד *S* לתוך התור *Q*.

3. **כל עוד התור *Q* לא ריק:**

   u = Dequeue(Q), כלומר הוצא את הקודקוד הראשון בתור ושים אותו ב- *u*.

   עבור כל קודקוד *V* שהוא שכן של *u*:

   אם הצבע של *V* הוא לבן אז בצע:

   אפור = Color[v], *כלומר צבע את הקודקוד* v *לצבע אפור.*

   d[v] = d[u] + 1, כלומר קבע את המרחק של הקודקוד *v* מהקודקוד *u*.

   $\pi[v] = u$, כלומר קבע את האבא של הקודקוד *v* להיות הקודקוד *u*.

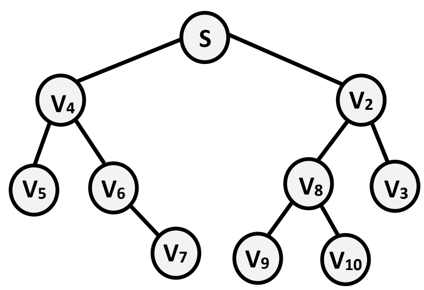
   Enqueue(Q, v), כלומר הכנס את הקודקוד *v* לתור *Q*.

   שחור = Color[u], כלומר אין יותר מה לעשות עם הקודקוד u, אז צבע אותו בשחור.

סיבוכיות – O(V + E)

- $V_{BFS} = v\epsilon V : Color[v] = Black$, כלומר כל הקודקודים הצבועים בשחור.
- $E_{BFS} = \{(\pi[v], v) : v\epsilon V_{BFS}, v \neq S\}$, כלומר כל הצלעות שמורכבות מהקודקודים שנמצאים ב- V_{BFS} והאבות שלהם פרט לקודקוד S.

- לאחר הרצת האלגוריתם **בגרף לא מכוון** G ניתן לומר:
1. אם כל הצמתים של הגרף נצבעו בשחור אז הגרף קשיר.
2. אם איזשהו צומת V נשאר צבוע בלבן אז המרחק שלו הוא ∞ מכאן שאין מסלול בין הקודקוד S לקודקוד V ולכן הגרף G אינו קשיר.
3. $T_{BFS} = (V_{BFS}, E_{BFS})$ הוא עץ פורש של רכיב הקשירות שמכיל את S בגרף G.



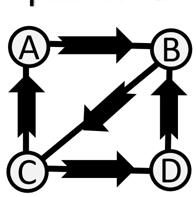
T_{BFS} - לפי הדוגמא בסרטוט בראש הפרק

- לאחר הרצת האלגוריתם **בגרף מכוון** G ניתן לומר:
1. אם כל הצמתים של הגרף נצבעו בשחור אז **הקודקוד S הוא שורש** של G.
2. אם איזשהו צומת V נשאר צבוע בלבן אז המרחק של הצומת V הוא ∞ ומכאן שאין מסלול מכוון בגרף G מהצומת S לצומת V.
3. $T_{BFS} = (V_{BFS}, E_{BFS})$ הוא עץ מכוון בעל שורש ב- S.

- **שימוש באלגוריתם BFS בגרף מכוון:**

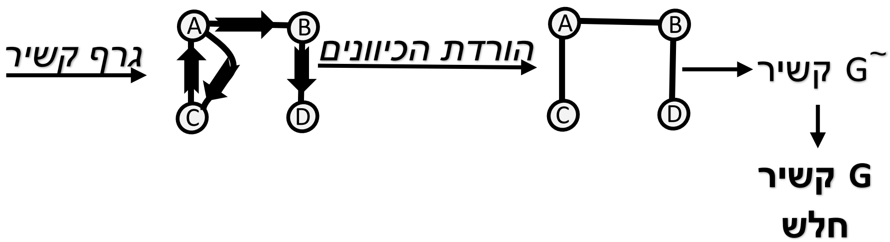
יהיה G = (V, E) גרף מכוון ויהיה $S\epsilon V$.

1. אם יש מסלול מכוון מ- S לכל $v\epsilon V$ אז S הוא שורש של G.
2. $G^T = (V, E^T)$ הוא **הגרף המכוון המוחלף** T= Transpose, כלומר הגרף G^T הוא הגרף המכוון G עם כיוונים הפוכים.
3. אם לכל $u, v\epsilon V$ קיימים מסלולים מ- u ל- v וגם מ- v ל- u אז הגרף G נקרא **גרף קשיר חזק**.

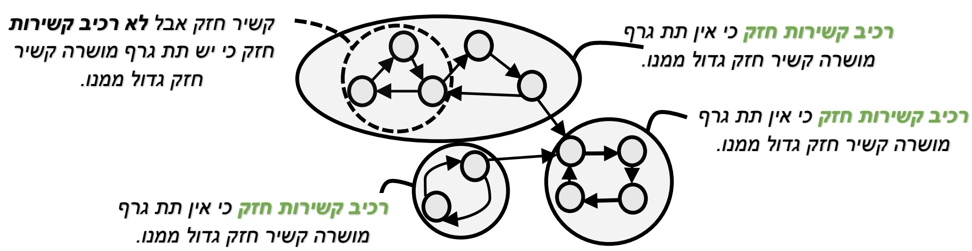


נכתב על ידי אוהד אמסלם, תשפ"ב

4. הגרף הלא מכוון $\tilde{G} = (V, \tilde{E})$ הוא בעצם הגרף המכוון שהורידו לו את הכיוון, כלומר זה הופך להיות G לא מכוון.
5. הגרף G נקרא **קשיר חלש** אם $\tilde{G}$ הוא קשיר.



6. **רכיב קשירות חזק** הוא תת גרף מושרה קשיר חזק בעצמו ולא קיים תת גרף יותר גדול קשיר חזק שכולל אותו בפנים.



- **טענות שנשענות אחת על השנייה:**

יהי G = (V,E) גרף מכוון אז הטענות הבאות שקולות זו לזו –

1. G הוא קשיר חזק.
2. כל צומת של G הוא שורש של G.
3. G^T הוא קשיר חזק.
4. כל צומת של G^T הוא שורש של G^T.
5. קיים $S \epsilon V$ כך שהצומת S הוא שורש של G וגם של G^T.

כלומר אם טענה אחת מתקיימת אז כל הטענות האחרות גם מתקיימות.

מה ניתן לבדוק בעזרת אלגוריתם BFS?

עבור גרף לא מכוון:	**עבור גרף מכוון:**
1. לבדוק האם הגרף G קשיר וכמה רכיבי קשירות יש לו. 2. לבדוק האם הגרף G דו-צדדי, יער או עץ. 3. לבדוק האם בגרף G יש מעגלים. 4. לבדוק האם הקודקוד $v\epsilon V$ הוא צומת הפרדה. 5. לבדוק האם הצלע (x, y) בגרף G היא גשר. 6. לחשב את הקוטר של הגרף G. 7. לחשב את המרחק בין הקודקוד $v\epsilon V$ לשאר הצמתים בגרף G.	1. לבדוק האם הגרף G קשיר חזק או חלש. 2. לבדוק האם הקודקוד $v\epsilon V$ הוא שורש של הגרף G. 3. לחשב את האורך המינימלי של מעגל מכוון דרך הקודקוד $v\epsilon V$.

- **כיצד נדע האם הגרף G קשיר?**

  $\text{If number of NILL } \neq 1$ אז הגרף G אינו קשיר, אחרת הוא קשיר.
- **כיצד נדע האם צלע מסוימת סוגרת מעגל?**

  $\text{If } \pi[v] \neq u \text{ } and \text{ } \pi[u] \neq v$ אז הצלע (v, u) סוגרת מעגל.
- **כיצד נדע האם מעגל בגרף הוא מעגל זוגי או אי זוגי?**

  נוריד את הצלע (x, y) שסוגרת מעגל מהגרף G ונריץ BFS מהקודקוד x, נבדוק $\textbf{If d}[y] = \boldsymbol{even}$ אז המעגל הוא אי זוגי, אחרת המעגל זוגי. (אם לא קיים מעגל אי זוגי בגרף G אז הגרף G הוא דו-צדדי.
- **כיצד נדע האם הצומת V היא צומת הפרדה בגרף קשיר?**

  נריץ BFS על הגרף G-v ונבדוק האם הגרף G-v קשיר, אם הגרף אינו קשיר אז הצומת V אכן צומת הפרדה.
- **כיצד נדע האם הצומת V היא לא צומת הפרדה בגרף קשיר?**

  צומת הפרדה $\neq v \Leftrightarrow$ בגרף $G - v$ כל השכנים של v נשארים באותו רכיב קשירות, כלומר נבדוק האם ניתן להגיע לכל שכן של הצומת v מאחד מהשכנים של v, אם כן אז הצומת v אינה צומת הפרדה.

הרצאה 3

❖ אלגוריתם DFS:

S – הקודקוד ממנו האלגוריתם מתחיל להתפשט.

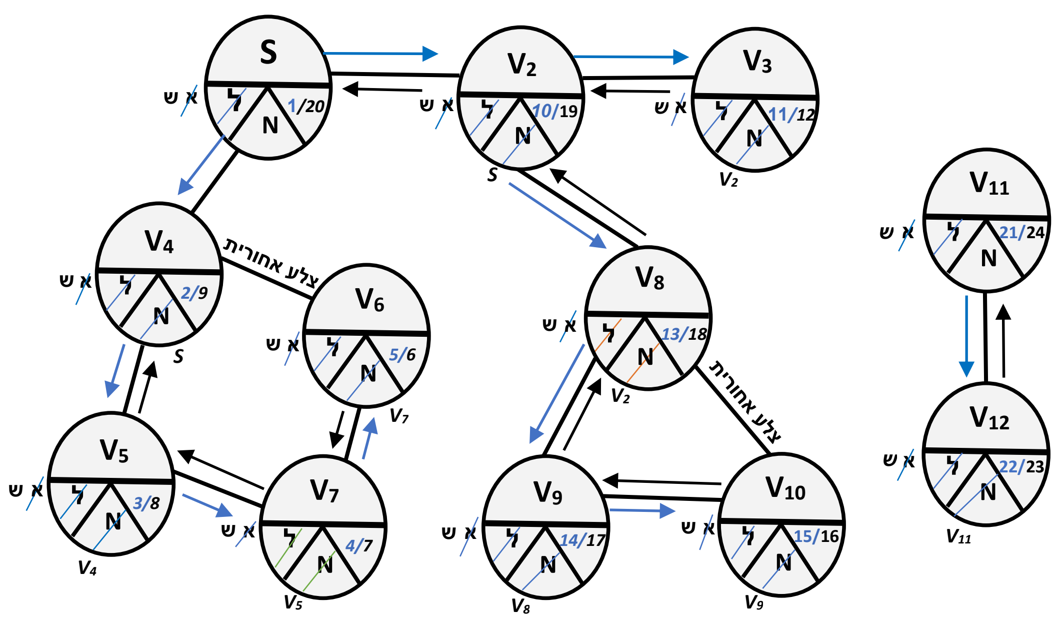
C[v] - מצב הקודקוד V מבחינת צבע (לבן, אפור, שחור), **המצב ההתחלתי הוא לבן**.

π[v] - מסמן מי האבא של הקודקוד V.

b[v] – זמן גילוי, מסמן את **הזמן עד לגילוי** של הצומת V_i.

f[v] – זמן נסיגה, מסמן את הזמן עד ש- DFS **עוזב את הצומת V_i**.

האלגוריתם עובד בעזרת מחסנית.



	S	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}	V_{12}
C[v]	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש	ש
b[v]/f[v]	1/20	10/19	11/12	2/9	3/8	5/6	4/7	13/18	14/17	15/16	21/24	22/23
π[v]	NILL	S	V_2	S	V_4	V_7	V_5	V_2	V_8	V_9	NILL	V_{11}

G = (V, E) גרף מכוון או לא מכוון.

S הוא קודקוד בגרף G.

DFS (G, S) or DFS (G) –

1. עבור כל קודקוד u ששייך לקבוצת הקודקודים V בצע:

Color[u] = White, כלומר צבע את הקודקוד u בצבע לבן.

$\pi[u] = NILL$, כלומר קבע כי אין לאף קודקוד אבא.

Time = 0, כלומר קבע כי הזמן ההתחלתי הוא 0.

End For

2. עבור כל קודקוד u ששייך לקבוצת הקודקודים V:

בצע אם Color[u] = White:

DFS_VISIT(u), כלומר קרא לפונקציה DFS_VISIT עם הקודקוד u.

End For

DFS_VISIT(u)

1. Color[u] = Gray, כלומר צבע את הקודקוד u בצבע אפור.

b[u] = Time + 1, כלומר קבע כי הזמן הגעה לקודקוד u הוא time + 1.

2. עבור כל קודקוד v ששייך לקבוצת השכנים של u:

בצע אם Color[v] = White:

$\pi[v] = u$, כלומר קבע כי האבא של הקודקוד v הוא הקודקוד u.

DFS_VISIT(v), כלומר קרא לפונקציה DFS_VISIT עם הקודקוד v שהוא השכן של u.

End For

3. Color[u] = Black, כלומר צבע את הקודקוד u בצבע שחור (קורה בחזרה של הרקורסיה).

f[u] = time + 1, כלומר קבע כי זמן הנסיגה של הקודקוד u הוא time + 1.

סיבוכיות – O(V + E)

- האלגוריתם עובד על גרף לא מכוון וגם גרף מכוון.
- ניתן לקרוא לפונקציה עם צומת התחלתית או בלי DFS(G,S) או DFS(G).

נכתב על ידי אוהד אמסלם, תשפ"ב

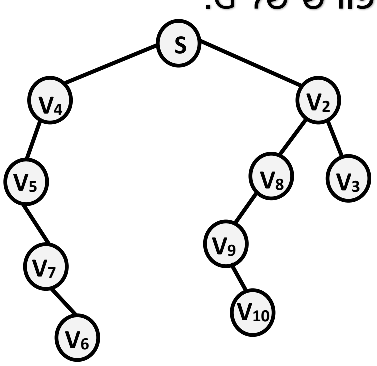
- האלגוריתם DFS שונה מ- BFS בכך שהוא מבצע סריקה לעומק ובנוסף הוא עובר בין רכיבי קשירות.
- לאחר הרצת האלגוריתם מתקבל יער פורש של הגרף G.
- התקדמות האלגוריתם הוא תמיד מאפור ללבן ולכן אף פעם לא סוגרים מעגלים.
- **קשת אחורית** - במהלך התקדמות האלגוריתם **בגרף מכוון ולא מכוון** אם יהיה צלע u, v שהצמתים u ו- v צבועים באפור אז נקרא לצלע הזו קשת אחורית, אפור לאפור = קשת אחורית = צלע שסוגרת מעגל, ניתן גם לבדוק שמתקיים ($b[v]<b[u]<f[u]<f[v]$) ואם כן אז זו קשת אחורית, **מספר הצלעות האחוריות** בגרף G הוא **$|E| - |E_{DFS}|$**.
- **קשת קדמית/ חוצה** – במהלך התקדמות האלגוריתם **בגרף מכוון** אם יהיה צלע u, v שהצומת u צבוע באפור ו- v צבוע בשחור אז נקרא לצלע הזו קשת קדמית/ חוצה, אפור לשחור = קשת קדמית/ חוצה, ניתן גם לבדוק שמתקיים ($b[u]<b[v]<f[v]<f[u]$) ואם כן אז זו קשת קדמית/ חוצה.
- **קשת עץ** – במהלך התקדמות האלגוריתם **בגרף מכוון** אם יהיה צלע u, v שהצומת u צבוע באפור ו- v צבוע בלבן אז נקרא לצלע הזו **קשת עץ**, אפור ללבן = קשת עץ, ניתן גם לבדוק שמתקיים ($b[u]<b[v]<f[v]<f[u]$) ואם כן אז זו קשת עץ.
- **כיצד נבדוק האם קיים מסלול בין שני קודקודים בגרף מכוון ולא מכוון?**
  נבדוק את b[v] ו- b[f], למשל אם יש מסלול בין S ל- V_6 אז החיתוך של הקבוצה צריך להיות שונה מקבוצה ריקה, כלומר הערכים צריכים להיות מוכלים בקבוצה עם הטווח ערכים הגדול יותר לדוגמא -

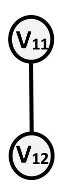
$V_6 = \{5, 6\}, S = \{1, 20\}$

$\{5, 6\} \cap \{1, 20\} = \{5, 6\} \neq \emptyset$

לכן ניתן לקבוע כי קיים מסלול בין S ל- V_6.

- מכל רכיב קשירות נקבל עץ פורש ולכן בסוף האלגוריתם נקבל יער פורש של G.





T_{DFS} - לפי הדוגמא בסרטוט בראש הפרק

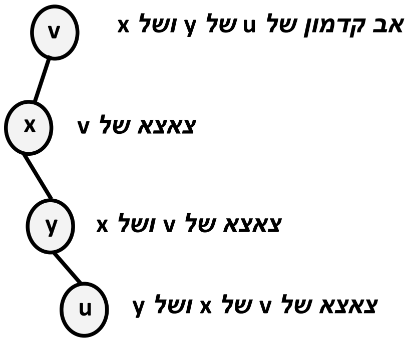
$T_{DFS} = (V_{DFS}, E_{DFS})$

$V_{DFS} = \{v: C[v] = Black\}$, כלומר כל הקודקודים שצבועים בשחור.

$E_{DFS} = \{(\pi[V], v): \pi[v] \neq NILL$, כלומר הצלעות אשר מורכבות מצומת והאבא שלו.

נכתב על ידי אוהד אמסלם, תשפ"ב

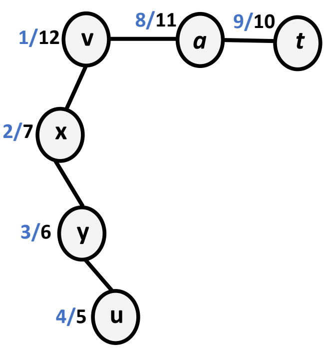
- בגרף לא מכוון מספר הצמתים עם NILL שווה למספר רכיבי הקשירות של הגרף G.
- בגרף לא מכוון אם NILL = 1 אז הגרף G קשיר.
- אם הצומת u מתגלה בזמן שהצומת v צבוע באפור אז הצומת u הוא **צאצא** של v והצומת v הוא **אב קדמון** של u.



משפט הסוגריים:

לכל $u, v \in V$ מתקיים בדיוק אחד מ- 3 המקרים הבאים-

1. $[b[v], f[v]] \cap [b[u], f[u]] = \emptyset$ אז אף אחד משני הצמתים אינו צאצא של השני.
2. $[b[u], f[u]] \subset [b[v], f[v]]$ אז u הוא צאצא של v ב- T_{DFS}.
3. $[b[v], f[v]] \subset [b[u], f[u]]$ אז v הוא צאצא של u ב- T_{DFS}.



דוגמא 1:

$[b[y], f[y]] = \{3, 6\}$

$[b[a], f[a]] = \{8, 11\}$

$[b[a], f[a]] \cap [b[y], f[y]] = \{3, 6\} \cap \{8, 11\} = \emptyset$ לכן אף אחד משני הצמתים אינו צאצא של השני.

דוגמא 2:

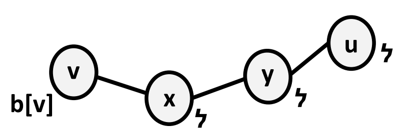
$[b[v], f[v]] = \{1, 12\}$

$[b[u], f[u]] = \{4, 5\}$

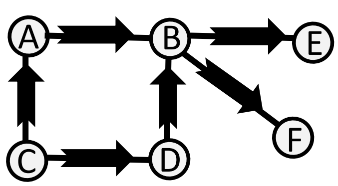
$[b[u], f[u]] \subset [b[v], f[v]] = \{4, 5\} \subset \{1, 12\}$ לכן u הוא צאצא של v.

- **משפט המסלול הלבן:**

צומת u הוא צאצא של צומת v ב- T_{DFS} של הגרף G מכוון או לא מכוון אם ורק אם בזמן b[v] שבו מתגלה v יש מסלול בגרף מ- v ל- u שמורכב כולו מצמתים לבנים.



- גרף מכוון G ללא מעגלים מכוונים נקרא **גרף א-ציקלי** או בלועזית **(DAG)**.

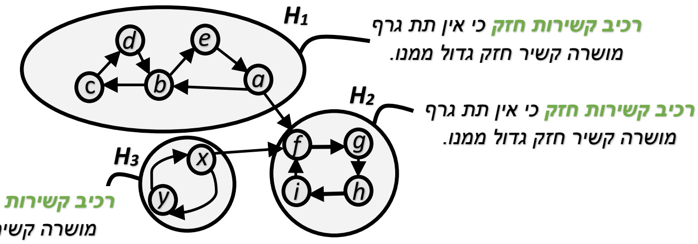


אם לאחר הרצת האלגוריתם DFS מתגלה קשת אחורית אז נקבע כי הגרף G אינו א-ציקלי.

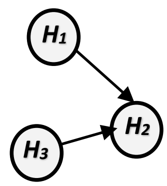
- בגרף $G=(V,E)$ מכוון ולא קשיר חזק, **מספר מחלקות השקילות = מספר רכיבי הקשירות החזקים של G**, בגרף הבא ניתן לראות כי ישנם 3 רכיבי קשירות חזקים ולכן ישנם 3 מחלקות שקילות.

מחלקות השקילות הם:

[a] = {a, b, c, d, e} = **[b] = [c] = [d] = [e]**
[f] = {f, g, h, i} = **[g] = [h] = [i]**
[x] = {x, y} = **[y]**



- **$G^{SCC}=(V^{SCC},E^{SCC})$** – הוא הגרף המצומצם של רכיבי קשירות חזקים.



הגרף G^{SCC} – ניתן לראות כי הגרף הוא א-ציקלי, כלומר זהו גרף מכוון ללא מעגלים מכוונים

יהי $G=(V,E)$ **גרף מכוון ויהיו** $H_1=(V_1,E_1), H_2=(V_2,E_2)$ **רכיבים קשירים חזקים של** G **אז התנאים הבאים מתקיימים:**

1. $\mathbf{V_1 \cap V_2 = \emptyset}$
2. אם קיימת צלע $\mathbf{(u,v) \in E}$ כך ש- $\mathbf{u \in V_1, v \in V_2}$ אז לא קיימת אף צלע $\mathbf{(a,b) \in E}$ כך ש- $\mathbf{b \in V_1, a \in V_2}$.
3. אם קיים מסלול מכוון $\mathbf{P_{u,v}}$ כך ש- $\mathbf{u \in V_1, v \in V_2}$ אז לא קיים אף מסלול מכוון $\mathbf{P_{a,b}}$ כך ש- $\mathbf{b \in V_1, a \in V_2}$.
4. גרף העל $\mathbf{G^{SCC}=(V^{SCC},E^{SCC})}$ הוא א-ציקלי.

מה ניתן לבדוק בעזרת אלגוריתם DFS ?

עבור גרף לא מכוון:	עבור גרף מכוון:
1. לבדוק האם הגרף G קשיר וכמה רכיבי קשירות יש לו. 2. לבדוק האם הגרף G הוא יער או עץ. 3. לבדוק האם בגרף G קיים מסלול שמחבר את $x, y \in V$. 4. לבדוק האם הקודקוד $v \epsilon V$ הוא צומת הפרדה. 5. לבדוק האם הצלע $(x, y) \in E$ בגרף G היא גשר.	4. לבדוק האם הגרף G קשיר חזק או חלש. 5. לבדוק האם הקודקוד $v \epsilon V$ הוא שורש בגרף G. 6. למצוא את כל רכיבי הקשירות החזקים של G. 7. בדיקה האם הגרף G מכיל מעגל מכוון, כלומר לבדוק האם הוא א-ציקלי או לא. 8. מציאת מיון טופולוגי.

- **כיצד לבדוק אילו צמתים נמצאים ברכיב הקשירות החזק של הצומת a בגרף מכוון G ?** נבדוק לאילו צמתים ניתן להגיע גם בגרף **G** וגם בגרף G^T והחיתוך הזה הוא בעצם הצמתים שנמצאים ברכיב הקשירות החזק של הצומת **a**.

F1[] = DFS (G, a)

F2[] = DFS (G^T, a)

A = {a}, כלומר צור קבוצה חדשה והוסף אליה את הצומת **a**.

עבור כל צומת **v** ששייך לקבוצת הצמתים **V**:

If ($F_1[v] < F_1[a]$ AND $F_2[v] < F_2[a]$)

Then A = A ∪ {v}, כלומר הוסף את הצומת **v** לקבוצה **a**.

End for

החזר את הקבוצה **A**.

- **כיצד לבדוק איזה צמתים נמצאים באותו רכיב קשירות של צומת** a **בגרף לא מכוון** G **?**

הקודקוד **a** הוא הקודקוד ממנו אנו מתחילים את האלגוריתם, אז אנו יודעים ש- **f[a]** הוא בעל המספר הגדול ביותר ברכיב הקשירות H_1 לכן נבדוק זאת בצורה הבאה:

Call DFS (G, a)

A = {a}, כלומר צור קבוצה חדשה והוסף אליה את הצומת **a**.

עבור כל קודקוד **v** ששייך לקבוצת הקודקודים **V**:

אם **f[v] < f[a]** אז הצומת **v** נמצא באותו רכיב קשירות של **a**, לכן בצע:

A = A ∪ $\{v\}$

End For

החזר את הקבוצה **A**.

☆ אם נרצה לבדוק האם קבוצה של קודקודים מסוימים למשל **{ b, c, d, e }** נמצאים באותו רכיב קשירות של **a** אז נבדוק האם **max (f[b], f[c], f[d],f[e]) < f[a]** ואם כן אז יהיה ניתן לומר שקבוצת הקודקודים המדוברת אכן באותו רכיב קשירות של הצומת **a**, אם לא אז לא כל הקבוצה באותו רכיב קשירות של **a**.

- **כיצד נדע האם גרף לא מכוון** G **הוא יער?**

נאמר ש- **K = Number of NILL** וגם ש- $K \geq 1$ רכיבי קשירות, אז הגרף **G** הוא יער אם ורק אם **|E| = |V| - K**

ניתן לבדוק את המשפט על הדוגמא בראש הפרק ולבדוק את נכונותה.

- **כיצד נמצא את רכיבי הקשירות החזקים של גרף מכוון** G **?**

נקרא לפונקציה SCC(G) כקופסה שחורה.(SCC – Strongly connected components)

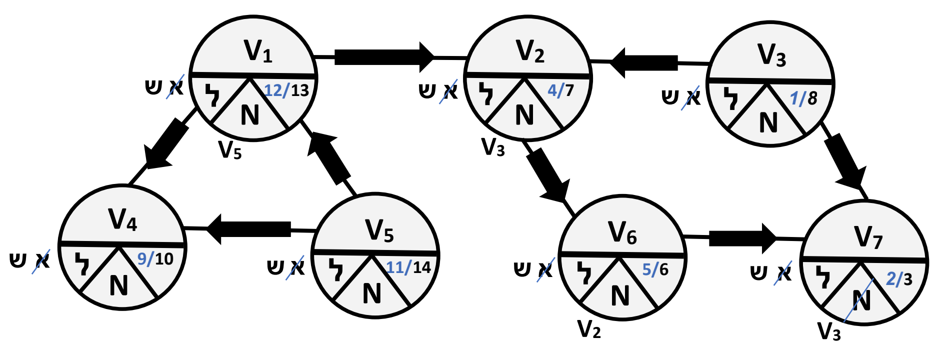
נכתב על ידי אוהד אמסלם, תשפ"ב

הרצאה 4

❖ מיון טופולוגי:

- כאשר מדברים על מיון טופולוגי מתייחסים לגרף מכוון.
- אם לגרף **G** יש מעגל מכוון אז ל- **G** אין מיון טופולוגי.
- אם יש לגרף **G** יש מיון טופולוגי אז ב- **G** אין מעגלים מכוונים.
- מממשים מיון טופולוגי בעזרת רשימה מקושרת.
- נאמר שניתן למיין טופולוגית אם קיימת פונקציה על וחד חד ערכית **$K: V = \{1, 2, \ldots, n\}$** כך ש- **$(u, v) \in E \Leftarrow K[u] < K[v]$**

כלומר מיון טופולוגי של **G** הוא רשימה של כל הצמתים של **G** כך שאם **$(u, v) \in E$** אז **u** מופיע לפני **v** ברשימה, לכן ניתן לצייר את הגרף **G** בצורה ליניארית, כלומר כל הצמתים בקו אופקי והכיוון של כל צלע הוא משמאל לימין.



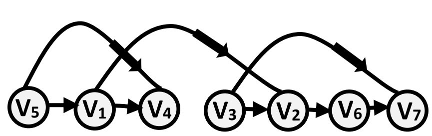
$L = \{V_5, V_1, V_4, V_3, V_2, V_6, V_7\}$

נשים לב שמכניסים כל פעם לראש הרשימה.

לאחר שנקבל את הרשימה נמספר אותה לפי סדר מהראשון לאחרון.

$\{V_5 - 1, V_1 - 2, V_4 - 3, V_3 - 4, V_2 - 5, V_6 - 6, V_7 - 7\}$

צורה ליניארית של G



$f[v_5] > f[v_1] > f[v_4] > f[v_3] > f[v_2] > f[v_6] > f[v_7]$

$14 > 13 > 10 > 8 > 7 > 6 > 3$

G- גרף מכוון ללא מעגלים מכוונים (א-ציקלי).

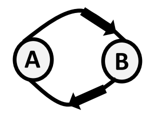
L- רשימה מקושרת ריקה.

– Topological_Sort (G)

1. **Call DFS (G)** כדי לחשב את **f[u]** עבור כל קודקוד **u**.
2. כאשר קודקוד **u** נסוג אז ה- **f[u]** שלו מחושב, כאשר זה קורה הכנס אותו לתחילת הרשימה **L**.
3. החזר את הרשימה המקושרת של הקודקודים.

סיבוכיות - O(E + V)

- אם **$|V| \in G^{SCC} < |V| \in G$** אז יש מעגל מכוון בגרף **G**. (כי **G^{SCC}** נותן את רכיבי הקשירות החזקים) ואם כל קודקוד אינו רכיב קשירות חזק בפני עצמו סימן שיש לפחות רכיב קשירות חזק אחד שמורכב מיותר מקודקוד אחד
 למשל הרכיב הבא:

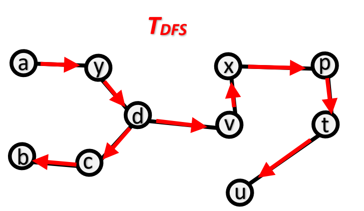


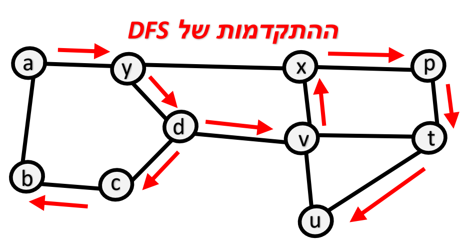
- הגרף **$G^* = (V, E^*)$** הוא הגרף **G** לא מכוון שהוסיפו לו כיוון כך שיהיה קשיר חזק.
- לכל צלע של **G** נקרא **מכוונות** של **G**.
- לכל צלע ב- **G** אפשר לתת אחד משני כיוונים ולכן ל- **G^*** יש **$2^{|E|}$** אופציות לגרפים.
- משפט רובינס - לגרף **G** לא מכוון **יש G^* קשיר חזק** אם ורק אם **G קשיר וללא גשרים**.

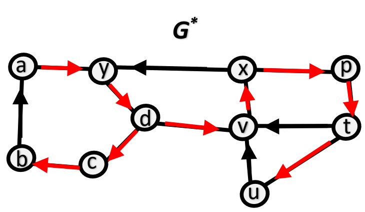
G = (V, E) – גרף לא מכוון, קשיר ללא גשרים.
$G^* = (V, E^*)$ – הגרף שהאלגוריתם הבא מפיק - גרף מכוון קשיר חזק.

Strong (G, G^*) –

1. קרא לאלגוריתם **DFS (G)** כך שיתקבל עץ פורש **$T_{DFS} = (V, E_{DFS})$**.
2. **עבור כל צלע** e = (u, v) **של** G **בצע:**
 אם הצלע $e \in E_{DFS}$ **אז:**
 תן לצלע e את הכיוון שבכיוון ההתקדמות של DFS.
 אם הצלע $e \notin E_{DFS}$ **אז:**
 תן לצלע e את הכיוון מהצאצא לאב הקדמון.







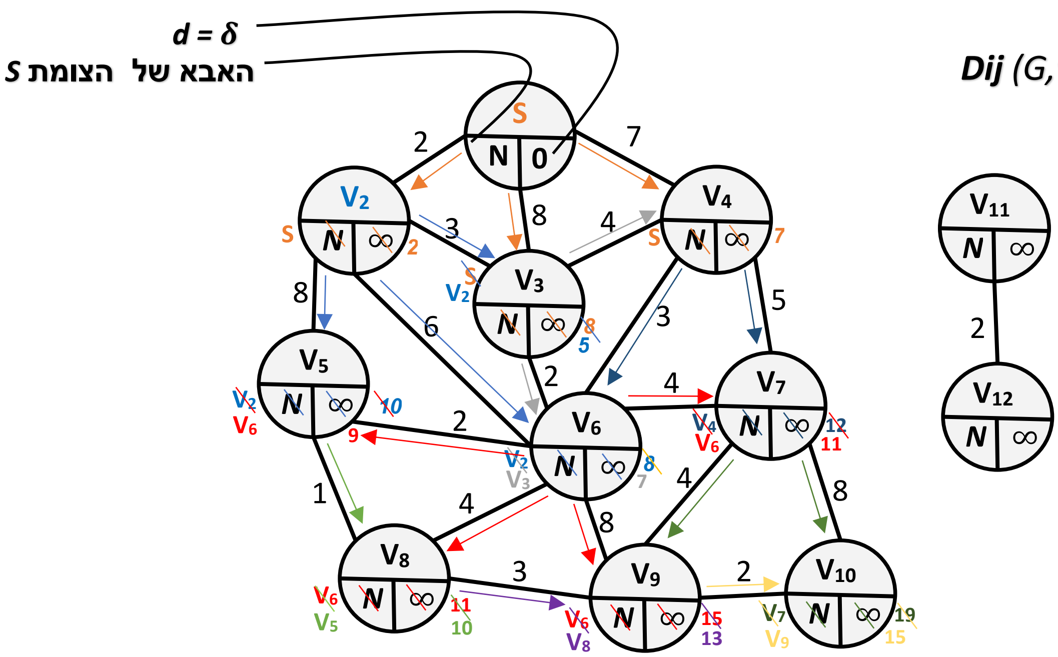
- כיצד נמצא את כל הגשרים בגרף **G = (V, E)** לא מכוון?
 Call Strong (G, G^*)
 Call SCC (G^*) – נותן רכיבי קשירות חזקים. **$H_i = (V_i, E_i), i = 1, 2.., k , k \geq 1$**
 $A = E - E_1 \cup E_2 \cup ... \cup E_k$ = קבוצת הגשרים.

הרצאה 5

❖ אלגוריתם Dijkstra:

- גרף משוקלל הוא גרף מכוון או לא מכוון **G = (V, E, w)** כאשר **w** הוא פונקציית משקל שיכול לקבל את הערכים **[0, ∞)** לכל צלע **e = (u, v) ∈ E**
  אז **w(u, v) = w(e)** = משקל הצלע **(u, v)**.
- המשקל הכולל של המסלול $P_{x,y}$ מוגדר כסכום המשקלים של הצלעות.
- המרחק מהצומת **x** לצומת **y** מוגדר כ- $dist(x, y)$ או $\delta(x, y)$ והוא שווה למשקל המסלול בעל המשקל המינימלי מבין כל המסלולים בין הצומת **x** לצומת **y**.
- מטרת האלגוריתם הוא למצוא מסלולים קלים ביותר.
- האלגוריתם עובד בעזרת שתי רשימות:
  Q – רשימת כל הקודקודים בגרף **G**.
  L – רשימה ריקה, מכניסים אליה צמתים שלא ניתן להתקדם מהם יותר.

Dij (G,w,S)



$Q = \{\cancel{S}, \cancel{V_2}, \cancel{V_3}, \cancel{V_4}, \cancel{V_5}, \cancel{V_6}, \cancel{V_7}, \cancel{V_8}, \cancel{V_9}, \cancel{V_{10}}, V_{11}, V_{12}\}$

$L = \{S, V_2, V_3, V_4, V_6, V_5, V_8, V_7, V_9, V_{10}\}$

נכתב על ידי אוהד אמסלם, תשפ"ב

G = (V, E, w) גרף ממושקל.
S – קודקוד ששייך לקבוצת הקודקודים של **G**.

- Dijkstra (G, w, S)

1. $\boldsymbol{\pi[S] = NILL}$, כלומר קבע כי לצומת **S** אין אבא.
   d[s] = 0, כלומר קבע כי המשקל מהצומת **S** לעצמו הוא 0.
   Q = V, כלומר צור רשימה **Q** והכנס אליה את כל הצמתים ששייכים לקבוצת הצמתים של **G**.
   L = ∅, כלומר צור רשימה ריקה בשם **L**.

2. **עבור כל קודקוד** v∈ V – {s} **בצע:**

   $\boldsymbol{\pi[v] = NiLL}$, כלומר קבע כי לכל הצמתים פרט לצומת **S** אין אבא.
   d[v] = ∞, כלומר קבע כי המשקל של כל הצמתים הוא אינסוף.

3. **כל עוד הקבוצה** Q **אינה ריקה:**
   מצא קודקוד **u∈ Q** כך שהמשקל **d[u] = min{d[v] : v∈ Q}**, כלומר בחר בקודקוד בעל המשקל הנמוך ביותר בקבוצה **Q**.
   אם **d[u] = ∞** שבור את הלולאה.
   L = L ∪ {u}, כלומר הכנס את הקודקוד **u** לקבוצה **L**.
   Q = Q – {u}, כלומר הסר את הקודקוד **u** מהקבוצה **Q**.

   עבור כל קודקוד v **ששייך לקבוצת השכנים של הקודקוד** Q ∩ u **(במידה והגרף** G **אינו מכוון) או קבוצת השכנים** $N^{+}(u) \cap Q$ **(במידה והגרף** G **מכוון) בצע:** כלומר השכנים של **u** שעדיין נמצאים ברשימה **Q**.
   אם d[u] + w((u, v)) < d[v] **אז בצע:** כלומר אם המשקל של הקודקוד **u** ועוד המשקל של הצלע המחברת בין הקודקוד **u** לקודקוד **v** קטן מהמשקל של הקודקוד **v** אז:
   d[v] = d[u] + w((u, v)), כלומר קבע כי המשקל של הקודקוד **v** שווה למשקל של הקודקוד **u** ועוד המשקל של הצלע שמחברת בין הקודקוד **u** לקודקוד **v** (המשקל הנמוך יותר).
   $\boldsymbol{\pi[v]}$ **= u**, כלומר קבע שהאבא של הקודקוד **v** הוא הקודקוד **u** (כלומר הגענו לצומת **v** דרך הצומת **u**).

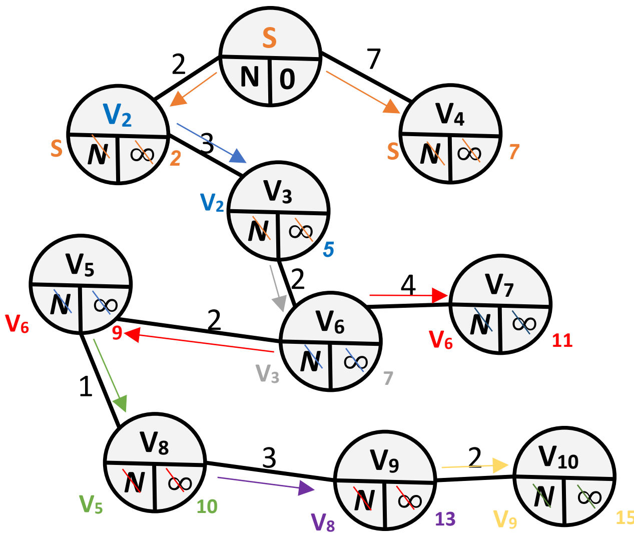
   End For
End While
החזר את d[v] **ואת** $\pi[v]$ **לכל צומת** $v \in V$.

סיבוכיות - **$O(|V|^2)$**

- האלגוריתם בוחר צומת בעל **d** מינימלי מהרשימה **Q**, מחשב את המשקל לצמתים שהוא יכול להתקדם אליהם, אם המשקל שחושב נמוך מהמשקל שהיה לפני כן אז הוא מעדכן את המשקל למשקל הנמוך יותר, ברגע שאין יותר צמתים מהרשימה **Q** שהוא יכול להתקדם אליהם אז הצומת מוסר מהרשימה **Q** ונכנס לרשימה **L** (לכן הוא לא חוזר לצמתים שכבר נבדקו), לאחר מכן שוב נבחר צומת מהרשימה **Q** בעל ה- **d** המינימלי וכן הלאה..
- האלגוריתם עוצר כאשר הרשימה **Q** ריקה או כאשר הצומת בעל המשקל המינימלי בקבוצה **Q** שווה לאינסוף.

- **$T_{Dij} = (V_{Dij}, E_{Dij})$** - הוא עץ פורש של רכיב הקשירות של **G** שכולל את הצומת **S** עבור גרף לא מכוון. אם הגרף מכוון אז הוא עץ מכוון בעל שורש בצומת **S**.

$$V_{Dij} = \{v : d[v] \neq \infty\}$$
$$E_{Dij} = \{(\pi[v], v) : \pi[v] \neq NILL\}$$



T_{Dij} - על סמך הדוגמא בראש הפרק

- **לאחר ביצוע האלגוריתם בגרף מכוון ניתן לומר:**
1. אם נשאר בקבוצת הקודקודים **V** קודקוד **v** שהמשקל שלו הוא אינסוף אז אבא של הקודקוד **v** שווה ל- **NILL**.
בנוסף ניתן לומר שאין מסלול מכוון בין הקודקוד **S** לקודקוד **v** בגרף **G**.
וגם שהקודקוד **v** שונה מהקודקוד ההתחלתי **S**.
2. אם המשקל של כל צומת **v** ששייך לקבוצת הצמתים **V** קטן מאינסוף אז הצומת ההתחלתית **S** היא שורש של **G**.

- **לאחר ביצוע האלגוריתם בגרף לא מכוון ניתן לומר:**
1. אם נשאר צומת **v** ששייך לקבוצת הצמתים **V** כך שהמשקל שלו שווה לאינסוף אז אין מסלול בין הצומת **S** לצומת **v** וגם **G** אינו קשיר.

נכתב על ידי אוהד אמסלם, תשפ"ב

- בגרף לא מכוון $d[v] = \delta(S, v)$, כלומר המשקל של הקודקוד **v** שווה למרחק בין הצומת **S** עד לצומת **v** , כמובן שזה בהתאם למסלול שבחר האלגוריתם.

מה ניתן לבדוק בעזרת אלגוריתם Dijkstra ?

עבור גרף משוקלל לא מכוון :G = (V, E, w)	עבור גרף משוקלל מכוון :G = (V, E, w)
1. לבדוק האם הגרף G קשיר וכמה רכיבי קשירות יש לו. **2.** לבדוק האם הגרף G הוא דו-צדדי, יער או עץ. **3.** לבדוק האם בגרף G קיים מסלול שמחבר את $x, y \in V$. **4.** לבדוק האם הקודקוד $v\epsilon V$ הוא צומת הפרדה. **5.** לבדוק האם הצלע $(x, y) \in E$ בגרף G היא גשר. **6.** לבדוק האם ב- G יש מעגלים זוגיים/ אי זוגיים. **7.** לחשב את הקוטר של הגרף G.	1. לבדוק האם הגרף G קשיר חזק או חלש. 2. לבדוק האם הקודקוד $v\epsilon V$ הוא שורש בגרף G. 3. לבדוק האם קיים מסלול מכוון שמחבר את $x, y \in V$.

באופן כללי ניתן לעשות בעזרת אלגוריתם **Dijkstra** את כל מה שניתן לעשות בעזרת אלגוריתם **BFS**.

- **כיצד למצוא מעגל מכוון קל ביותר שעובר דרך הצלע (x, y) בגרף G מכוון משוקלל עם פונקציית משקל חיובית?**

  Select y$\in V$

  d[] = Dij(G, w, y)

  אם $d[x] = \infty$ החזר אין מעגל מכוון דרך הצלע (x, y).

  אחרת החזר d[x] + w(x, y) = משקל כולל של מעגל מכוון קל ביותר שעובר דרך (x, y).

- **כיצד נמצא מסלולים קצרים ביותר מצומת התחלתית S לכל הצמתים של G בגרף מכוון ללא מעגלים מכוונים (גרף א-ציקלי) בצורה יותר יעילה מ- Dijkstra ?**

נשתמש באלגוריתם הבא:

- DAG_Shortest_Paths (G, w, S)

1. נריץ את האלגוריתם Topological_Sort(G)
2. **עבור כל צומת v ששייך לקבוצת הקודקודים V בצע:**
   $d[v] = \infty$, כלומר קבע כי המשקל של כל המצתים הוא אינסוף.
   $\pi[v] = NILL$, כלומר קבע כי לכל הצמתים אין אבא.
3. $d[S] = 0$, כלומר קבע כי המשקל של הצומת ההתחלתית S הוא 0.
4. **עבור כל צומת u בקבוצת הצמתים הממוינת טופולוגית בצע:**
   עבור כל צומת v שהוא שכן $N^{+}(u)$ בצע:
   אם $d[u] + w(u,v) < d[v]$ **אז בצע:**
   $d[v] = d[u] + w(u, v)$, כלומר קבע כי המשקל של הצומת v הוא המשקל של הצומת u ועוד המשקל של הצלע המחברת בין הצומת u לצומת v.
   $\pi[v] = u$, כלומר קבע כי האבא של הצומת v הוא הצומת u.

סיבוכיות – O(E + V)

- **תזכורת –**

יהי $G = (V, E)$ גרף לא מכוון ו- C מעגל בגרף G.

1. C נקרא **מעגל אוילר** אם **C עובר דרך כל הצמתים וכל הצלעות** של G **אך על כל צלע פעם אחת בדיוק.**
2. G נקרא **גרף אוילר** אם ב- G קיים מעגל אוילר.
3. G הוא גרף אוילר אם ורק אם G **קשיר וגם זוגי (כלומר לכל הצמתים יש דרגה זוגית).**

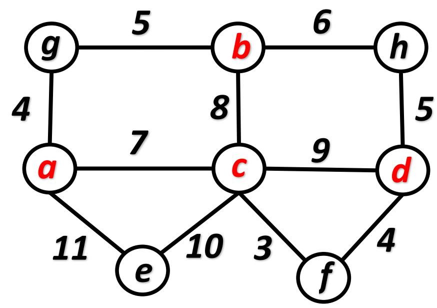
אם הגרף G הוא קשיר וזוגי אז ניתן למצוא C מעגל אוילר על ידי האלגוריתם של Hierholzer.

הרצאה 6

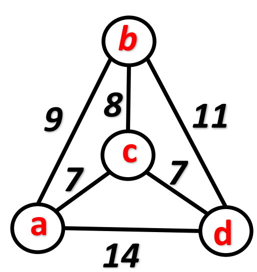
❖ **אלגוריתם CPP** - הבעיה הסינית של הדוור

נראה איך מוצאים מעגל אוילר בגרף בעל 4 צמתים בעלי דרגה אי זוגית-

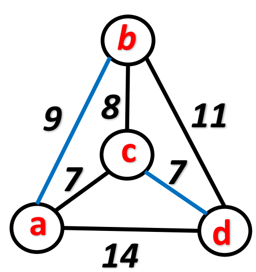
הגרף G -



1. מצא את כל הצמתים בעלי הדרגה האי זוגית.
2. נריץ Dijkstra מכל אחד מהצמתים בעלי הדרגה האי-זוגית ונקבל את המסלולים מהצמתים האלה לכל שאר הצמתים.
3. נבנה גרף קטן בעל המסלולים הקצרים מהצמתים בעלי הדרגה האי-זוגית לצמתים האחרים בעלי הדרגה האי-זוגית.

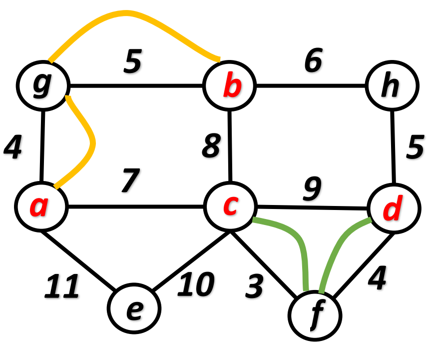


4. נבחר צלעות ללא צומת משותף בעל סכום מינימאלי של משקלים.



5. נשכפל את הצלעות במסלול מ- a ל- b ומ- c ל- d בגרף G. (כלומר המסלולים בין הצמתים בעלי הדרגה האי-זוגית בשני המסלולים בעלי המשקל המינימאלי שמצאנו בשלב 4).

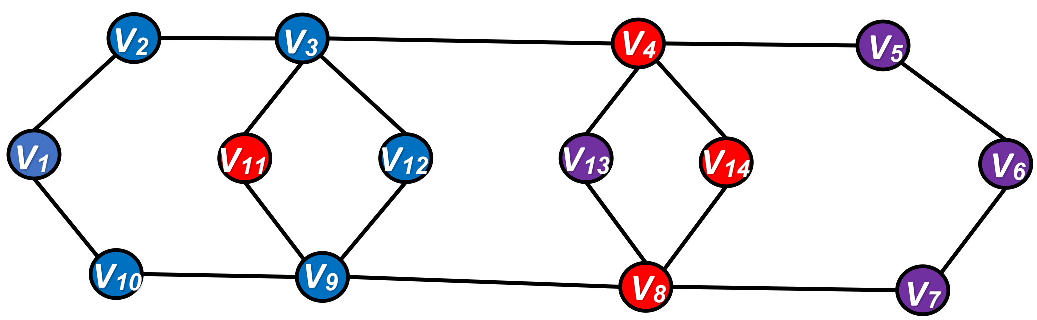
הגרף G -



כעת אנו יכולים לראות שלגרף G אין צמתים בעלי דרגה אי- זוגית, כלומר הגרף הוא זוגי ולכן ניתן למצוא מעגל אוילר בגרף G החדש.

6. נריץ את אלגוריתם Hierholzer(G) כדי לקבל מעגל אוילר C.

❖ אלגוריתם Hierholzer –



C_1 :{ V_1, V_2, V_3, V_{12}, V_9, V_{10}, V_1 } - ניתן לראות כי יש צומת שממנו יוצאות צלעות שלא היו באחד המעגלים - V_3.

C_2 :{ V_3, V_4, V_{14}, V_8, V_9, V_{11}, V_3 } -

C_3 :{ V_1, V_2, V_3, V_4, V_{14}, V_8, V_9, V_{11}, V_3, V_{12}, V_9, V_{10}, V_1} - ניתן לראות כי יש צומת שממנו יוצאות צלעות שלא היו באחד המעגלים –הצומת V_4.

C_4 :{ V_4, V_5, V_6, V_7, V_8, V_{13}, V_4 }

C_5 :{ V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8, V_{13}, V_4 V_{14}, V_8, V_9, V_{11}, V_3, V_{12}, V_9, V_{10}, V_1}

C_4

C_3

מה שקורה בשרטוט זה הדבר הבא: מוצאים שני מעגלים, ואז ומבצעים שילוב ביניהם (**C_3**).

לאחר מכן מוצאים מעגל נוסף ושוב מבצעים שילוב – הפעם בין המעגל החדש שמצאנו לבין **C_3**.

עושים זאת עד שמתקבל מעגל אחד של כל הגרף (מעגל אוילר).

G = (V, E) – **גרף מכוון ללא מעגלים מכוונים** (DAG)**.**

❖ אלגוריתם למציאת המסלול הארוך ביותר בגרף G-

מערך בגודל מספר הקודקודים של הגרף = **Length_to[]**

1. **Length_to[] = 0** עבור כל קודקוד **v**$\in V$
2. קרא לאלגוריתם **Topological_Sort(G)**
3. הנח כי הקודקודים **V** ממוינים טופולוגית-
   עבור כל $(v, u) \in E$**:**
   אם Length_to[u] < 1 + Length_to[v] אז בצע:
   Length_to[u] = 1 + Length_to[v]
4. החזר **max { Length_to[v] : v** $\in V$ **}**

סיבוכיות – O(E + V)

G = (V, E, w) – **גרף מכוון ללא מעגלים מכוונים** (DAG) **בעל פונקציית משקל.**

❖ אלגוריתם למציאת מסלול קריטי בגרף G –

1. **Finish(x) = 0** עבור כל **x**$\in V$
2. קרא לאלגוריתם **Topological_Sort(G)**
3. עבור כל **(u, v)** $\in E$ בצע:
   finish(v) = max{finish(v), finish(u) + w(v)}
   End For

החזר את המערך finish(x).

❖ אלגוריתם Bellman-Ford:

- אלגוריתם בלמן פורד הוא אלגוריתם שפועל על גרף מכוון וממושקל ומשמש למציאת המסלול הקל ביותר בין צומת אחד מסוים אל כל אחד משאר הצמתים בגרף.
- אלגוריתם זה שונה מאלגוריתם **Dijkstra** בכך שהוא עובד גם כאשר הגרף מכיל קשתות בעלות משקל שלילי.
- אם הגרף מכיל מעגל שסכום משקלי קשתותיו שלילי האלגוריתם מסוגל לזהות את זה ולהתריע על כך.
- האלגוריתם עובד גם לגרף לא מכוון בעזרת טריק של שכפול כל צלע בגרף והוספת כיוון אבל הטריק הזה עובד רק בתנאי שהמשקלים של כל הצלעות הם חיוביים. אם תנאי זה לא מתקיים אז **Bellman-Ford** לא יעבוד על הגרף הזה.

```
                    Step 1: initialize graph ---------------- O(|V|)
for each vertex v∈V
        d[v] = ∞ and π[v] = NIL
End For
d[s] = 0

                // Step 2 ................................ relax edges repeatedly
for i from 1 to |V|-1:
        for each edge (u, v) ∈ E ......................... O(|V|*|E|)
                if d[u] + w(u,v) < d[v]:
                        d[v] = d[u] + w(u,v)
                        π[v] = u

            // Step 3 ............................... check for negative-weight cycles
for each edge (u, v) ∈ E
        if d[u] + w(u,v) < d[v]: then return error "Graph contains a negative-weight cycle"
        return {d[v], π[v] : v∈V}
```

Complexity : O(|V|*|E|)

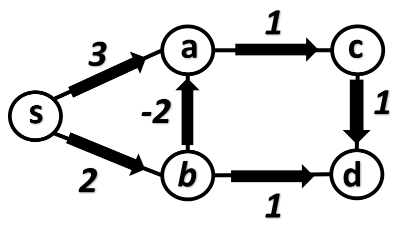
האלגוריתם רץ |V| פעמים כאשר **באיטרציה |V-1| נקבל את המסלולים הקצרים ביותר**, אך **אם** האלגוריתם **מוצא שיפור גם באיטרציה ה-|V| אז הוא משגר הודעת שגיאה כי יש מעגל שלילי בגרף G** (כי אם זה קורה אז בהכרח זה אומר שיש מעגל שלילי בגרף G).

דרך ריצת האלגוריתם (בניית הטבלה לפי העין) –

1. נסדר את הצלעות ברשימה לפי הצלעות היוצאות מכל צומת:
(s, a), (s, b), (a, c), (b, a), (b, d), (c, d)
2. ניצור טבלה שעוקבת אחרי המשקל הנמוך ביותר מבין כל המסלולים שנמצאים במהלך הריצה של האלגוריתם, כל פעם שנמצא מסלול קל יותר מצומת אחת לאחרת אז נעדכן את הטבלה.
3. נרוץ על הצלעות לפי הסדר שסידרנו אותם בשלב 1 וכאשר נסיים לרוץ על כולם נגדיר זאת כאיטרציה אחת של האלגוריתם – ככה |V| פעמים.
4. אם הטבלה תתעדכן גם באיטרציה האחרונה (האיטרציה ה- |V|) אז נשגר הודעה שקיים מעגל שלילי בגרף G, אחרת נחזיר את **d[v], π[v]** שמייצגים את המשקל של המסלול הקל ביותר מצומת אחת לכל שאר הצמתים ואת האבא של כל אחד מהצמתים.

סדר הצלעות – (s, a), (s, b), (a, c), (b, a), (b, d), (c, d)
נשים לב שמה שמסומן בצבע **אדום** זה עדכון של המשקל הקל ביותר במהלך ריצה על הצלעות שלנו לפי הסדר באותה איטרציה – אם היה עדכון אז סימן שמצאנו מסלול קל יותר מהמסלול הקודם ולכן אנו מתייחסים אליו כמשקל הקל ביותר באיטרציה הנוכחית.
דבר נוסף – לדוגמא באיטרציה הראשונה כאשר אנו בודקים את הצלע (c, d) אנו רואים שאפשר כבר להגיע לצומת d לפי מסלול בעל משקל 3 ולכן לא נעדכן את המשקל לצומת d למשקל 5. כלומר מצומת C לצומת d זה לא המסלול הקל ביותר שלנו באיטרציה הראשונה.

שלבים	d	c	b	a	s
אתחול	∞	∞	∞	∞	0
איטרציה 1	3	4	2	3, **0**	0
איטרציה 2	3, **2**	1	2	0	0
איטרציה 3	2	1	2	0	0
איטרציה 4	2	1	2	0	0
איטרציה 5	2	1	2	0	0



נכתב על ידי אוהד אמסלם, תשפ"ב

❖ אלגוריתם Floyd-Warshall:

- מטרת האלגוריתם היא למצוא את המסלולים הקצרים ביותר בין כל שתי צמתים בגרף G מכוון וממושקל.
- עובד גם עם משקלים שליליים.
- האלגוריתם לא עובד כאשר יש מעגלים שליליים בגרף G.
- האלגוריתם **מזהה אם קיים מעגל שלילי בגרף** אם אחד או יותר מערכי האלכסון במטריצות משתנה מהספרה 0 לספרה אחרת.
- **זמן הריצה הוא $O(V^3)$**

כיציד עובד האלגוריתם –

נבנה $|V+1|$ טבלאות כאשר הטבלה הראשונה נקראת D^0 **והיא מייצגת את הגרף המכוון והממושקל G**, את שאר הטבלאות נבנה בעזרת הנוסחה הבאה כאשר:

$$D^k_{ij} = \min(D^{(k-1)}_{ij}, D^{(k-1)}_{ik} + D^{(k-1)}_{kj}$$

כלומר

$$D^k_{ij} = \min(D^{(k-1)}_{ij}, D^{(k-1)}_{ik} + D^{(k-1)}_{kj})$$

i = צומת המקור.

j = צומת היעד.

k = מספר הטבלה (כלומר באיזה קודקוד אנחנו נעזרים כדי להגיע לצומת היעד שלנו).

מרחק מצומת לעצמו = 0 לכן האלכסון תמיד נשאר 0 – ולכן הוא מסומן **באפור** בכל הטבלאות.

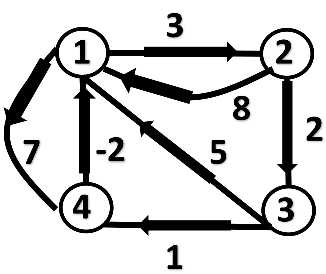
בנוסף כל מה שמסומן **בכתום** בטבלאות הם בעצם האיטרציות שנשארות זהות למה שהיה בטבלה הקודמת – למשל ב- D^1 ברור שהשורה הראשונה והעמודה הראשונה לא ישתנו ויהיו בדיוק כמו שהיו ב- D^0 כי הרי זה ברור שכולם (התאים בשורה 1 ועמודה 1) עוברים דרך הצומת 1. גם ברור שב- D^2 השורה הראשונה והעמודה הראשונה לא ישתנו ויהיו בדיוק כמו שהיו ב-

D^1 כי זה ברור שכולם (התאים בשורה 2 ועמודה 2) עוברים דרך הצומת 2.
וכן הלאה..

מה שמסומן בירוק מיועד לשחזור המסלול ונתייחס אליו מאוחר יותר.

D^0 – מייצגת את הגרף המכוון והממושקל G
D^1 – המסלול הקצר ביותר שעובר דרך צומת מספר 1.
D^2 – המסלול הקצר ביותר שעובר דרך צומת מספר 2.
D^3 – המסלול הקצר ביותר שעובר דרך צומת מספר 3.
D^4 – המסלול הקצר ביותר שעובר דרך צומת מספר 4.

הגרף G -



Destination = j

Source = i

D^0	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	-2	∞	∞	0

D^1	1	2	3	4
1	0	3	∞	7
2	8	0	2	15
3	5	8	0	1
4	-2	1	∞	0

$D^1_{23} = \min(D^0_{23}, D^0_{21} + D^0_{13}) = \min(2, 8 + \infty) = 2$
$D^1_{24} = \min(D^0_{24}, D^0_{21} + D^0_{14}) = \min(\infty, 8 + 7) = 15$
$D^1_{32} = \min(D^0_{32}, D^0_{31} + D^0_{12}) = \min(\infty, 5 + 3) = 8$
$D^1_{34} = \min(D^0_{34}, D^0_{31} + D^0_{14}) = \min(1, 5 + 7) = 1$
$D^1_{42} = \min(D^0_{42}, D^0_{41} + D^0_{12}) = \min(\infty, - + 3) = 1$
$D^1_{43} = \min(D^0_{43}, D^0_{41} + D^0_{13}) = \min(\infty, -2 + \infty) = \infty$

D^2	1	2	3	4
1	0	3	5	7
2	8	0	2	15
3	5	8	0	1
4	-2	1	3	0

$D^2{}_{13} = \min(D^1{}_{13}, D^1{}_{12} + D^1{}_{23}) = \min(\infty, 3 + 2) = 5$

$D^2{}_{14} = \min(D^1{}_{14}, D^1{}_{12} + D^1{}_{24}) = \min(7, 3 + 15) = 7$

$D^2{}_{31} = \min(D^1{}_{31}, D^1{}_{32} + D^1{}_{21}) = \min(5, 8 + 8) = 5$

$D^2{}_{34} = \min(D^1{}_{34}, D^1{}_{32} + D^1{}_{24}) = \min(1, 8 + 15) = 1$

$D^2{}_{41} = \min(D^1{}_{41}, D^1{}_{42} + D^1{}_{21}) = \min(-2, 1 + 8) = -2$

$D^2{}_{43} = \min(D^1{}_{43}, D^1{}_{42} + D^1{}_{23}) = \min(\infty, 1 + 2) = 3$

D^3	1	2	3	4
1	0	3	5	6
2	7	0	2	3
3	5	8	0	1
4	-2	1	3	0

$D^3{}_{12} = \min(D^2{}_{12}, D^2{}_{13} + D^2{}_{32}) = \min(3, 5 + 8) = 3$

$D^3{}_{14} = \min(D^2{}_{14}, D^2{}_{13} + D^2{}_{34}) = \min(7, 5 + 1) = 6$

$D^3{}_{21} = \min(D^2{}_{21}, D^2{}_{23} + D^2{}_{31}) = \min(8, 2 + 5) = 7$

$D^3{}_{24} = \min(D^2{}_{24}, D^2{}_{23} + D^2{}_{34}) = \min(15, 2 + 1) = 3$

$D^3{}_{41} = \min(D^2{}_{41}, D^2{}_{43} + D^2{}_{31}) = \min(-2, 3 + 5) = -2$

$D^3{}_{42} = \min(D^2{}_{42}, D^2{}_{43} + D^2{}_{32}) = \min(1, 3 + 8) = 1$

D^4	1	2	3	4
1	0	3	5	6
2	1	0	2	3
3	-1	2	0	1
4	-2	1	3	0

$D^4{}_{12} = \min(D^3{}_{12}, D^3{}_{14} + D^3{}_{42}) = \min(3, 6 + 1) = 3$

$D^4{}_{13} = \min(D^3{}_{13}, D^3{}_{14} + D^3{}_{43}) = \min(5, 6 + 3) = 5$

$D^4{}_{21} = \min(D^3{}_{21}, D^3{}_{24} + D^3{}_{41}) = \min(7, 3 + -2) = 1$

$D^4{}_{23} = \min(D^3{}_{23}, D^3{}_{24} + D^3{}_{43}) = \min(2, 3 + 3) = 2$

$D^4{}_{31} = \min(D^3{}_{31}, D^3{}_{34} + D^3{}_{41}) = \min(5, 1 + -2) = -1$

$D^4{}_{32} = \min(D^3{}_{32}, D^3{}_{34} + D^3{}_{42}) = \min(8, 1 + 1) = 2$

כעת לאחר שהבנו כיצד לבנות את הטבלאות (הטבלה האחרונה היא בעצם המרחקים הקצרים ביותר בין צומת אחת לשנייה) נרצה לשאול את השאלה הבאה:

כיצד לשחזר את **המסלול** הקצר ביותר?

כדי לשחזר את המסלול הקצר ביותר נצטרך לבנות **מטריצת אבות** לכל טבלה וכאשר נקבל את המטריצת אבות **האחרונה** אז **נשתמש בה** כדי לשחזר את המסלול שנבחר בין צומת אחת לאחרת. נבנה כל מטריצת אבות במקביל לבניית המטריצה של המסלולים הקצרים ביותר (D^0, D^1..).

נבנה את הטבלה π^0 (כלומר טבלת האבות בין שני צמתים שיש להם דרך ישירה בגרף G) ולאחר מכן זה הופך להיות פשוט אפילו יותר – נסתכל על הטבלה D^0 ועל הטבלה D^1 **ונבדוק האם יש תאים שהשתנו** (ניתן לראות שהתאים שמסומנים בירוק הם התאים שהשתנו ולכן נסמן בתאים האלו את מספר הטבלה שאנו עובדים עליה (במקרה של הטבלה π^1 נרשום את המספר 1 שמייצג את האבא החדש של צומת היעד, במקרה של π^2 נמלא את התאים שמסומנים בירוק (אלה שהשתנו בין π^2 ל- π^1) בספרה 2 שמייצג שהאבא החדש של צומת היעד הוא צומת 2 וכן הלאה..) **וכל שאר התאים יישארו זהים למה שהיו בטבלה הקודמת.**

π^0	1	2	3	4
1	NIL	1	NIL	1
2	2	NIL	2	NIL
3	3	NIL	NIL	3
4	4	NIL	NIL	NIL

π^1	1	2	3	4
1	NIL	1	NIL	1
2	2	NIL	2	1
3	3	1	NIL	3
4	4	1	NIL	NIL

π^2	1	2	3	4
1	NIL	1	2	1
2	2	NIL	2	1
3	3	1	NIL	3
4	4	1	2	NIL

π^3	1	2	3	4
1	NIL	1	2	3
2	3	NIL	2	3
3	3	1	NIL	3
4	4	1	2	NIL

π^4	1	2	3	4
1	NIL	1	2	3
2	4	NIL	2	3
3	4	4	NIL	3
4	4	1	2	NIL

כעת כשקיבלנו את הטבלה האחרונה π^4 נשתמש בה כדי לשחזר את המסלול בין שני צמתים שנבחר -
למשל אם נרצה לשחזר את המסלול הקל ביותר בין צומת 1 לצומת 3 אז נלך לתא [1][3] ב- π^4 ונראה מה הערך בתא הזה (נאמר X) – ניקח את הערך הזה ונבדוק את התא הבא בעזרתו, התא הבא יהיה [1][X] ונמשיך ככה עד שנגיע לערך NIL.
המסלול יהיה מהסוף להתחלה.

דוגמא 2:

$P_{1,4}$ =

p = [1][4] = 3

p = [1][3] = 2

p = [1][2] = 1

p = [1][1] = NIL

$p_{1,4}$ = 1, 2, 3, 4

דוגמא 1:

$P_{1,3}$ =

P = [1][3] = 2

p = [1][2] = 1

p = [1][1] = Nil

$p_{1,3}$ = 1, 2, 3

- **סגור טרנזיטיבי**

סגור טרנזיטיבי הוא בעצם גרף שבו אם ניתן להגיע מנקודה a לנקודה b ומנקודה b לנקודה c אז ניתן להגיע מנקודה a לקודה c ולכן בגרף של הסגור טרנזיטיבי יש צלע ישירה בין הצומת a לצומת c.
ניתן למצוא את הסגור הטרנזיטיבי באמצעות הטבלה האחרונה שמחזיר האלגוריתם של Floyd-Warshall (D^4 במקרה של הדוגמא הקודמת). אם אין אינסוף באחד מהתאים אז ניתן להגיע מכל צומת לכל צומת אך אם יש אינסוף בתא מסוים אז לא ניתן להגיע בשום דרך מהצומת מקור לצומת יעד ולכן הוא לא יהיה בסגור הטרנזיטיבי.

נכתב על ידי אוהד אמסלם, תשפ"ב

❖ רשתות זרימה –

רשת זרימה היא סוג מיוחד של **גרף מכוון**, שמשמש למידול בעיות שמערבות מעבר של חומר בין מקומות.
ניתן להשתמש ברשתות זרימה כדי למדל זרימה של מים בצינורות או מעבר של מידע ברשתות תקשורת, מעבר של תנועה בכביש ועוד.

- כל רשת זרימה מאופיינת על ידי גרף מכוון שמכיל שני צמתים מיוחדים:
  1. **S** משמש **כמקור**.
  2. **t** שנקרא "**בור**" (המקום אליו מתנקזת הזרימה).
- לכל קשת בגרף יש **קיבול** (כמות הזרימה שאפשר להעביר בקשת) המסומן באות **c**, **c(e)** זה בעצם הזרימה המקסימלית שניתן להעביר דרך קשת $\boldsymbol{E}$ ∋**e**.
- הקצב המקסימלי שבו אפשר להזרים חומר ברשת מצומת המקור **s** לבור **t** = לחתך המינימלי. **כלומר (זרימה מקסימלית = חתך מינימלי).**
- בחתך מינימלי כל הקשתות בחיתוך חייבות להיות קשתות רוויה (**קשת רוויה** זו קשת שעוברת **בה זרימה מרבית אפשרית**).
- **אילוץ הקיבול** - על אף קשת לא מוזרם יותר מהקיבול שלה.
- **אנטי-סימטריה –** f(u, v) = -f(v, u) כלומר אם יש לי זרימה 5 מצומת (u, v) אז ניתן לחשוב שהזרימה מהכיוון ההפוך (v, u) היא -5.
- **חוק שימור הזרימה –** אומר שכל זרימה שנכנסה לצומת גם יצאה ממנו, מכיוון שזרימה שנכנסת לצומת מיוצגת על ידי מספר חיובי וזרימה שיוצאת מצומת מיוצגת על ידי מספר שלילי אז סכום הזרימות יהיה שווה ל- 0.

- **שיטת Ford-Fulkerson לפתרון בעיית הזרימה המקסימלית –**

  - **צלע רוויה** – קשת שבה עוברת זרימה מרבי אפשרית, כלומר f(u, v) = c(u, v)
  - **צלע משפרת –** קשת שאפשר להזרים לאורכה עוד זרימה, כלומר לא קשת רוויה.
  - **מסלול שיפור –** מסלול מהמקור s לבור t שמורכב כולו מצלעות משפרות (כלומר צלעות שכולן אינן רוויות).

האלגוריתם הבסיסי –

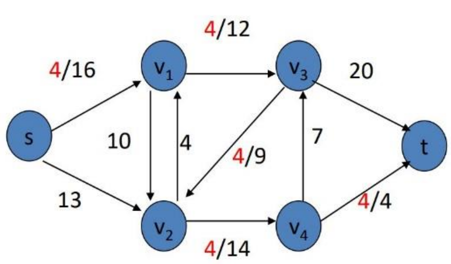
1. אתחול f(u, v) לכל **u, v**.
2. כל עוד יש מסלול שיפור **P** נגדיל את הזרימה לאורך **P** במידה מרבית (תוך כדי שמירת אילוץ הקיבול).
3. זמן הריצה הוא **O(Ef*)** אם כל הקיבולים שלמים, ערך הזרימה המקסימלית הוא **f***. זמן ריצה שיכול להגיע ל- 2^n בגלל ש- **f*** יכול להיות מיוצג על ידי **log(f*)** ביטים.

רשתות שיוריות –

רשת שבאמצעותה עובד האלגוריתם ומסומנת ב- G_f.

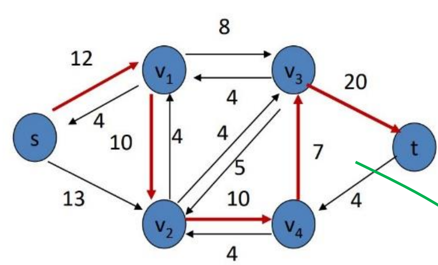
- **קיבול שיורי** c_f – כמות הזרימה שאפשר עדיין להזרים לאורך קשת: $c_f(u, v) = c(u, v) - f(u, v)$ כלומר קיבול פחות הזרימה בקשת.
- **הקיבול השיורי של מסלול P = הקיבול המינימלי לאורך P.**
בכתיב מתמטי – $c_f(P) = \min c_f(u, v) : (u, v) \in P$
- כל מסלול מצומת המקור **S** לבור **t** בגרף השיורי G_f הוא מסלול שיפור בגרף **G** ולהפך.

הגרף G



מסומן באדום זה הזרימה ב- G

הגרף G_f



מסומן באדום זה מסלול שיפור ב- G_f

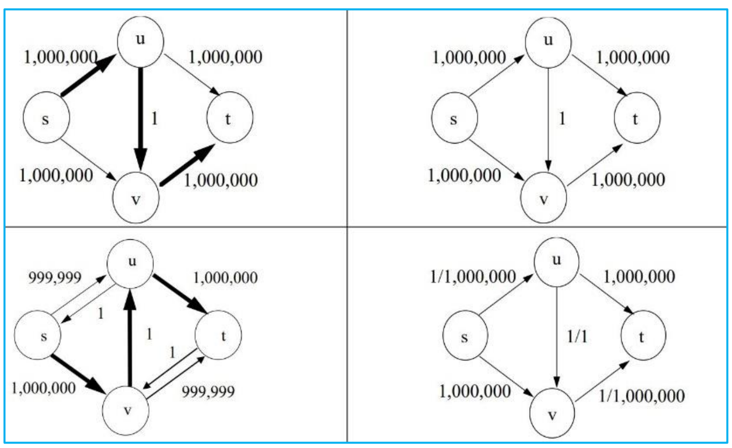
אנו רואים שהקיבול השיורי של המסלול שמצאנו הוא 7 וזה בגלל שהקשת בעל הקיבול המינימלי במסלול הזה היא בעלת קיבול 7. **כלומר מה שזה אומר זה שניתן להזרים במסלול השיפור הזה מקסימום עוד 7. אז $c_f(P) = 7$**

נכתב על ידי אוהד אמסלם, תשפ"ב

בנוסף ניתן לראות כי נוספו קשתות בצד הנגדי לזרימה בגרף **G_f** בכל קשת בה הייתה זרימה – זו מסמנת את הזרימה שאנו יכולים "להתחרט" עליה.
למשל הזרמנו בגרף **G** מהצומת **s** לצומת **t** 4 ואנחנו יכולים להזרים עוד 12 כי הקיבול של הקשת הוא 16, לכן בגרף השיורי ניתן "להתחרט" על ה-4 שכבר הזרמנו (קשת לכיוון ההפוך מהזרימה) ולהזרים עוד 12 (בקשת שבכיוון הזרימה).

- **דוגמא** לרשת שבה ניתן לבחור מסלולים משפרים כך שיהיו ***f** איטרציות וזמן ריצה **$O(2^n)$**:

גרף G **גרף G_f**



אנו רואים שבגרף השיורי אנו כל פעם מוצאים מסלול שיפור שבו ניתן להזרים רק 1 בכל איטרציה – פעם אחת על במסלול שיפור העליון ובפעם השנייה במסלול שיפור התחתון (שהוא בעצם מה שניתן להתחרט עליו בצוואר בקבוק) וכך בעצם זה ממשיך עד ***f** איטרציות שהוא בעצם ערך הזרימה המקסימלית.

- **שיטת Edmonds-Karp לפתרון בעיית הזרימה המקסימלית –**

שיטה שאומרת שאם בוחרים מסלול שיפור בעזרת אלגוריתם **BFS**, כלומר מוצאים מסלול קצר ביותר אז מספר האיטרציות הוא **VE** ולכן זמן הריצה הכולל יהיה **$O(VE^2)$** ומשפט זה נכון תמיד, כלומר גם אם הקיבולים אינם שלמים.

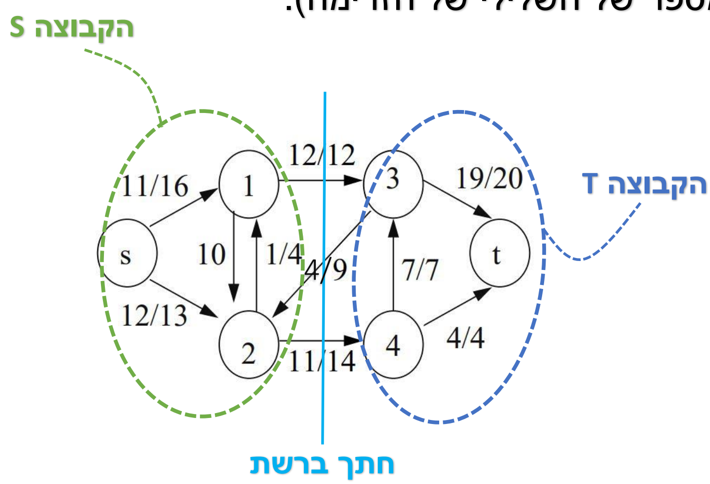
מזה אנו מסיקים שאם הקיבולים שלמים ובוחרים מסלולים על ידי **BFS** אז זמן הריצה יהיה **$O(\min(VE^2, Ef^*))$**.

כעת נדבר על חתך יותר לעומק:

חתך ברשת: חתך **(S, T)** של רשת זרימה הוא חלוקה של הצמתים לשתי קבוצות זרות כאשר צומת המקור $s \in S$ והבור $t \in T$.

קיבול חתך - סכום הקיבולים שחוצים את החתך בכיוון הזרימה (כאשר זה נגד כיוון הזרימה נגיד שסכום הקיבול של אותה הקשת הוא 0).

זרימה בחתך – סכום הזרימות שחוצות את החתך (אם זה נגד כיוון הזרימה אז נוסיף את המספר של השלילי של הזרימה).



הקיבול בחתך = **c(S, T) = c(1, 3) + c(2, 3) + c(2, 4) = 12 + 0 + 14 = 26**

כי (3,2) הולך נגד כיוון הזרימה.

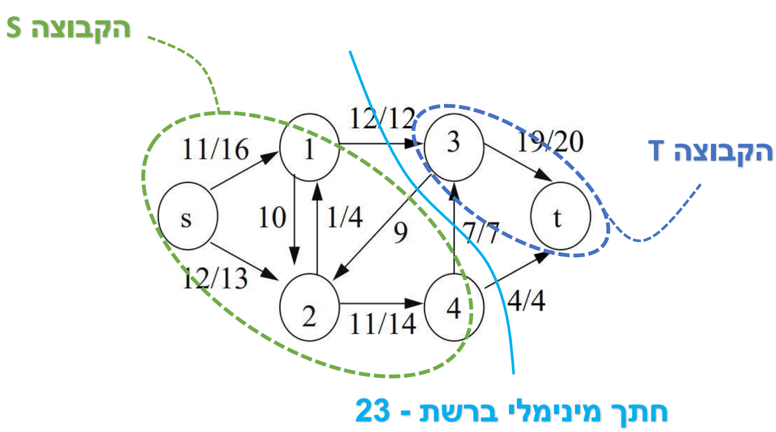
הזרימה בחתך **= f(S,T) = f(1,3) + f(2,4) + f(2,3) = 12 + 11 + (–4) = 19**

(2,3) כי אנחנו רוצים את הערך שהולך עם כיוון הזרימה.

החתך המינימלי –

דיברנו בתחילת הפרק על שתי משפטים בנוגע לחתך מינימלי – נזכיר אותם שוב.

1. בחתך מינימלי כל הקשתות בחיתוך חייבות להיות קשתות רוויה (**קשת רוויה** זו קשת שעוברת **בה זרימה מרבית אפשרית**).
2. הקצב המקסימלי שבו אפשר להזרים חומר ברשת מצומת המקור **s** לבור **t** = לחתך המינימלי. **כלומר (זרימה מקסימלית = חתך מינימלי).**
3. ברשת יכול להיות יותר מחתך אחד וכדי למצוא את הזרימה המקסימלית נצטרך לבדוק את כל החתכים בעלי קשתות רוויות ולבחור בחתך המינימלי מבניהם.



כעת מה שאנחנו מזהים בגרף העליון הוא שהאלגוריתם של **Ford-Fulkerson** לא ימצא מסלול שיפור כי יש חתך מינימלי בגרף (לא ניתן למצוא מסלול שיפור מ- s ל- t כי הקשתות שמובילות ל- t רוויות) ולכן הוא יעצור ויחזיר את הזרימה המקסימלית.

מסקנות משני השיטות:

1. אם הקיבולים רציונליים אז האלגוריתם של **Ford-Fulkerson** יגיע לעצירה באיזשהו שלב, אחרת הוא יכול לא לעצור – אבל אם הוא עוצר אז הוא מחזיר את הזרימה המקסימלית.
2. השיטה של **Edmonds-Karp** עובדת על ידי מציאת מסלולי שיפור בעזרת BFS והיא תמיד תעצור ותחזיר את הזרימה המקסימלית.

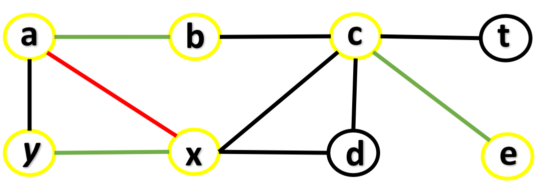
❖ זיווגים –

זיווג מוגדר באופן הבא – אם $M \subseteq E$ ולכל $(u, v), (x, y) \in M$ אין צומת משותף, אז M נקראת זיווג (או שידוך).

זיווג מקסימום - הוא זיווג בגודל מקסימאלי והוא מסומן $\mu(G)$.
זיווג מושלם – הוא זיווג **שמכסה** את **כל הצמתים** של G.

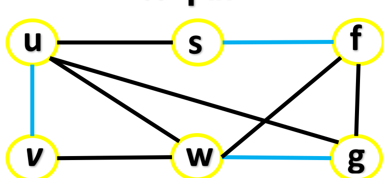
1. **כל זיווג מושלם הוא גם זיווג מקסימום.**
2. **אם ב- G יש זיווג מושלם אז |V| הוא מספר זוגי.**

גרף G

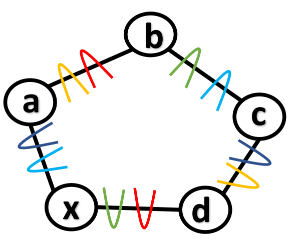


$M_0 = \{(a, b), (a, x)\}$ – **לא זיווג**.
$M_1 = \{(a, b)\}$ - **זיווג**.
$M_2 = \{(a, b), (x, y)\}$ – **זיווג**.
$M_3 = \{(a, b), (x, y), (c, e)\}$ – **זיווג מקסימום** לכן
נרשום $\boldsymbol{\mu(G) = |M_3| = 3}$.

גרף H



$\boldsymbol{\mu(H)}=3$ = $M = \{(u, v), (s, f), (g, w)\}$ הוא **זיווג מושלם** כי הוא מכסה את כל הצמתים.



- **במעגל אי-זוגי** $\mu(C_{2k+1}) = \boldsymbol{K} = \left\lfloor\frac{(2K+1)}{2}\right\rfloor$
  ל- C_{2k+1} **אין זיווגים מושלמים** ו- $\left\lfloor\frac{(2K+1)}{2}\right\rfloor$ זיווגי מקסימום.

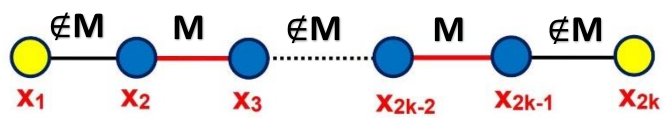
- **במעגל זוגי** $\mu(C_{2k}) = \boldsymbol{K} = \frac{2K}{2}$, יש לו **2 זיווגים מושלמים** (ו-2 זיווגי מקסימום).

- יהי $G = (A, B, E)$ **גרף לא מכוון, דו-צדדי** אז מתקיימים :
1. **משפט Hall** אומר שקיים זיווג שמכסה את A אם ורק אם $|X| \leq |N(X)|$ לכל $X \subseteq A$.
2. **משפט החתונה** אומר שקיים זיווג מושלם אם ורק אם מתקיים $|A| = |B|$ וגם $|X| \leq |N(X)|$ לכל $X \subseteq A$.
3. **משפט Konig – Egervary** אומר ש- $\alpha(G) + \mu(G) = |A| + |B|$ או $\mu(G) = \eta(G)$

מסלול הרחבה

נגדיר **מסלול הרחבה** ביחס ל- M אם מתקיימים **שתי תנאים**:

$P_{X1,\ X2K}$ -

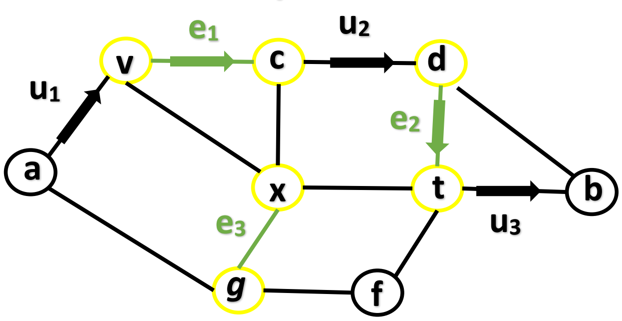


1. X_{2k} וגם X_1 אינם מכוסים על ידי הצלעות של M.
2. $(X_2, X_3), (X_4, X_5), (X_{2K-2}, X_{2K-1}) \in M$.

נשים לב שהמסלול $P_{X1,\ X2K}$ כולל **2K צמתים** ו- **2K – 1 צלעות** כך ש- **K – 1 צלעות שייכות ל- M** ושאר K הצלעות שייכות ל- E – M.

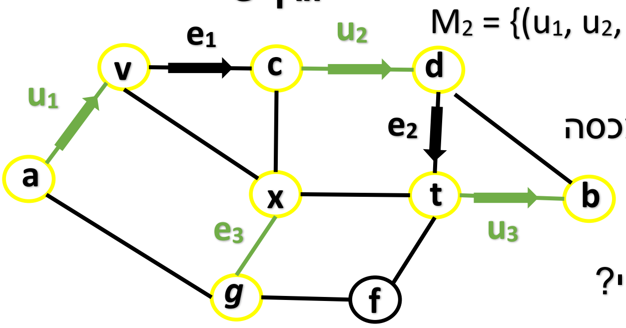
אז ניתן לראות שיש פה מסלול שיפור שגודלו הוא M + 1 (המסלול שמתחיל מהצלע (X_1, X_2)) כלומר כל הצמתים שלא שייכים ל- M.

גרף G



בגרף הבא אנחנו רואים זיווג $M_1 = \{e_1, e_2, e_3\}$ שמכסה את 6 הצמתים הצבועים בצהוב.

גרף G

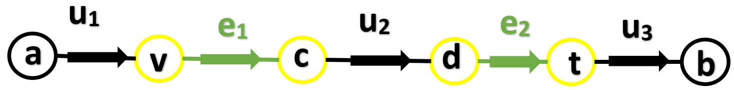


אך אנו רואים שניתן לבנות זיווג חדש $M_2 = \{(u_1, u_2, u_3, e_3)\}$ בעל M + 1 צלעות.

לכן $\mu(M_2) = 4$ הוא **זיווג מקסימום** כי הוא מכסה 8 צמתים מתוך 9 ולכן הוא **לא זיווג מושלם**.

כיצד נבנה את הזיווג החדש M_2 בכתיב מתמטי?

$P_1 = P_{a,\ b}$ -



$M_2 = M_1 \Delta E(P_1) = (M_1 \cup E(P_1)) - (M_1 \cap E(P_1)) =$ **{u1, u2, u3,e3}**

נראה את החישוב:

$(\{e_1, e_2, e_3\} \cup \{u_1, e_1, u_2, e_2, u_3\}) - (\{e_1, e_2, e_3\} \cap \{u_1, e_1, u_2, e_2, u_3\} =$

$\{u_1, e_1, u_2, e_2, u_3, e_3\} - \{e_1, e_2\} = \mathbf{\{u1, u2, u3, e3\}}$

משפט Berge - אומר שיהי גרף לא מכוון כלשהו ויהי M זיווג ב- G אז M הוא זיווג מקסימום **אם ורק אם** **לא קיים מסלול הרחבה** ביחס ל- M.

קלט - $G = (A, B, E)$ גרף דו צדדי.

פלט - M_{max}, כלומר מחזיר זיווג מקסימום.

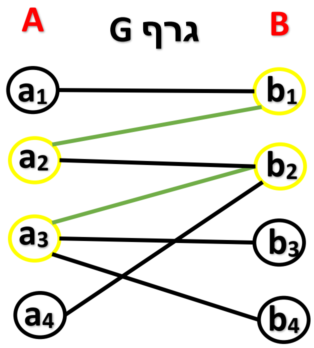
אלגוריתם לבניית זיווג מקסימום בגרפים דו-צדדיים (Bipartite graphs):

1. $M = \emptyset$.
2. **בצע עד שאין מסלול הרחבה (כלומר $P = \emptyset$):**
   P = **Augmenting_Path(G, M)**
   $M = M \Delta P$
   End While
3. תחזיר את $M = M_{max}$.

Augmenting_Path(G, M) (אלגוריתם למציאת מסלול הרחבה):

1. **נגדיר גרף חדש** – לכל הצלעות **שלא** שייכות לזיווג הנוכחי **נוסיף כיוון מ- A ל- B**, ולכל הצלעות בגרף G **השייכות** לזיווג הנוכחי **נוסיף כיוון מ- B ל- A**.
2. נוסיף שתי צמתים s, t ונחבר אותם בצורה הבאה –
   נחבר את הצומת s לכל הצמתים **ב-A** שאינם מכוסים על ידי הזיווג הנוכחי.
   נחבר את כל הצמתים **ב-B** שאינם מכוסים על ידי הזיווג הנוכחי לצומת t.
3. הרץ את אלגוריתם BFS מהצומת s לצומת t והחזר את המסלול P הקצר ביותר מהצומת s לצומת t.
4. החזר את המסלול $P - \{s, t\}$.

סיבוכיות - O(|V|*|E|)



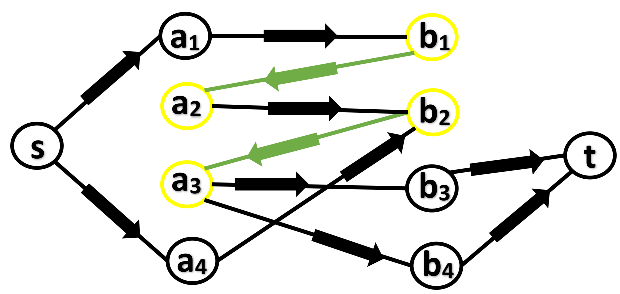
דוגמא להרצת האלגוריתם:

נניח שהזיווג הנוכחי ב- G הוא

$M_1 = \{(a_2, b_1), (a_3,b_2)\}$, כלומר הצלעות המסומנות בירוק.

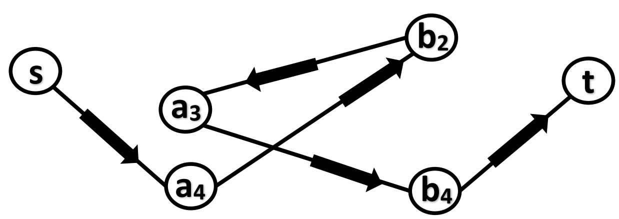
כעת נרצה לחפש מסלול הרחבה אז נבנה גרף חדש וניתן לצלעות כיוונים לפי מה שהגדרנו באלגוריתם **Augmenting_Path**.

גרף G’



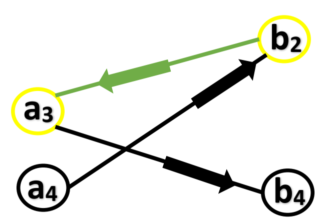
בשלב הבא נריץ את אלגוריתם BFS מהצומת s לצומת t ונחזיר את המסלול הקצר ביותר.

מסלול קצר ביותר t ל- s



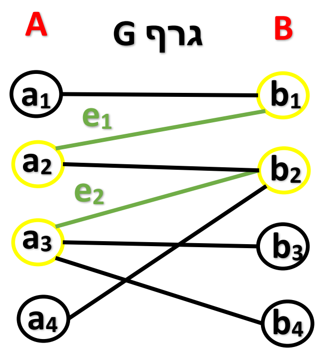
כעת נוריד את הצמתים s ו- t ונקבל מסלול הרחבה:

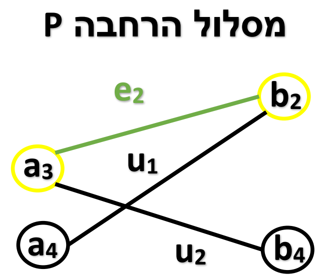
מסלול הרחבה P



זה מסלול הרחבה כי זה עונה על הדרישות שציינו בעמוד 40 (הצמתים a_4 ו- b_4 אינם מכוסים על ידי הצלעות של M הנוכחי וגם הצלע (b_2, a_3) נמצאת בזיווג הנוכחי). **את המסלול הזה מחזירים לאלגוריתם הראשי.**

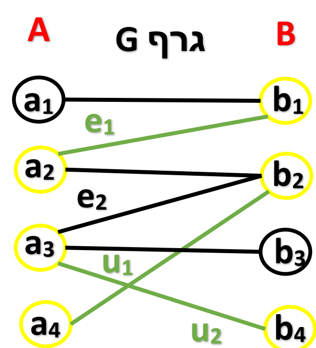
כל שנותר הוא להחליף בין הצלעות בעזרת הפעולה המתמטית שראינו $M \Delta P = (M \cup P) - (M \cap P)$ ונקבל את הזיווג החדש:





$$M \Delta P = (M \cup P) - (M \cap P) = \{e_1, e_2, u_1, u_2\} - \{e_2\} = \boldsymbol{\{e_1, u_1, u_2\}}$$

וסה"כ נקבל את הזיווג הבא:



כך האלגוריתם ממשיך עד שהאלגוריתם לא מצליח למצוא מסלול הרחבה.

בשלב האחרון על פי **משפט Berge** (אם לא נמצא מסלול הרחבה אז M הוא זיווג מקסימום) אנו **נחזיר את M** שהוא בעצם $\boldsymbol{M_{max}}$.

בנוסף ניתן לומר לפי משפט Hall כי לא קיים זיווג שמכסה את A כי
$|N(X)| \leq \neg\ |X|$ לכל $X \subseteq A$.
כי אם נבחר את הצמתים a_1, a_2, a_4 אז הגודל של השכנים שלהם b_1, b_4 קטן מהם.

נכתב על ידי אוהד אמסלם, תשפ"ב