פתרון של בעיות תכנון ליניאריות

פתרון אופטימלי יתקבל באחד מקודקודי הפוליגון שנוצר המתאר את תחום הפתרונות האפשריים.

<u>שיטה למציאת פתרון אופטימלי:</u>

1) בעזרת טבלה נרשום את כל קודקודי הפוליגון.

2) נציב לפי פונ המטרה את העמודה של z ע״י הצבה (x_1,x_2) בפונ המטרה.

x_1	x_2	z

← בשיטה זו נשתמש רק עבור פוליגון חסום.

פתרון אופטימלי = הפתרון הכי רלוונטי/טוב לבקשת השאלה min/max.

<u>שיטה נוספת למציאת פתרון אופטימלי בעזרת קוו שוה מחירים:</u> (שיטה זו תשמש אותנו בעיקר כאשר הפוליגון לא יהיה חסום)

1) נבחר z באופן שרירותי (מה שנרצה).

2) נציב בפונ המטרה.

3) נבודד את x_2 ונשרטט את הקו על מערכת צירים.

(הקו מייצג את אוסף של נק בהן פונ המטרה מקבלת את הערך ה-z שבחרנו.)

4) נבחר באופן מקרי עוד z כלשהו ונעשה את שלבים (2,3).

(#) כאשר נבנה קווי שווה הם תמיד יהיו מקבילים בעלי אותו שיפוע.

5) נבחן לפי קווי השווה את המגמה איפה המקסימום ואיפה המינימום ונלך למצוא את הפתרון האופטימלי לפי הבקשה (max/min)

6) נבדוק באיזה קודקוד הקו שווה צריך לצאת ב״יציאתו״ מתחום הפוליגון בכדי למצוא את הקודקוד שיתן את הפתרון האופטימלי.

שיטת הסימפלקס

1) נהפוך את האי שוויונים לשוויונים. ז"א נוסיף לכל משוואה משתנה חוסר מתאים X_i (והינו מספר כלשהו בהתאם למשוואות).
ערכם בפונ' המטרה הוא אפס.

2) משתנה החוסר מקבלים את הערך שאחר השיוויון.

3) נשווה את פונ' המטרה Z לאפס ($0=Z$) ונבצע העברת אגף של C האיברים לצד שמאל.

4) נכתוב את הטבלה המתאימה לנתוני המשוואות ופונ' המטרה.

מקדמי C =	מחירים מקוריים C	x_1 | x_2 | x_3 |		
משתני חוסר =	משתני בסיס	מטריצת מקדמים A של המשוואות	פתרון נוכחי b	מבחן המנה $\frac{b_i}{a_{ik}}$
C prime j =	מחירים נוכחיים C'			

5) מבחן אופטימליות: פתרון אינו אופטימלי אם יש לפחות משתנה אחד שלילי במחירים הנוכחיים

6) נבחר משתנה כדי להכניס למשתני הבסיס והוא יהיה זה שהמקדם שלו המינימלי במחירים הנוכחיים.

בשלב חילוק המנה מחלקים רק בערכים בעמודה בהם הם חיוביים (אם שליליים לא מחשבים).

נבצע כמה איטרציות כאלה כי עבור הפתרון האופטימלי נדרש שבשורה של המחירים נוכחיים C' יהיו שווים או גדולים מאפס.

אלכסון ראשי נקבע ע"י pivot והאיבר אותו נרצה לחשב!

שיטת הסימפלקס

נדון בפתרון בעיות מהצורה $\max\ z=cx$

$Ax\le b \quad x\ge 0$

דרישות: כל האילוצים קטנים שווים, b גדול שווה אפס כל משתני ההחלטה גדולים שווים לאפס.

הסבר: C- וקטור כל המקדמים של משתני ההחלטה

A- מטריצה נתונה כל עמודה זה אילוץ של כל משתנה

b- זהו וקטור הצד השני של A

x- וקטור משתני ההחלטה.

$$A=\begin{pmatrix}1 & 2\\ 2 & 1\end{pmatrix}$$

אילוצים של x_1 אילוצים של x_2

פתרון אופטימלי: עבור פתרון אופטימלי נדרוש שבשורה של המחירים הנוכחיים c' יהיו גדולים שווים לאפס ולא יהיה בה ערך שלילי כלל.

דרך פתרון:

1) נהפוך את האילוצים מאי שוויון למשוואות ע"י הוספת משתני חוסר לכל משוואה.
משתנים אלו הם גדולים שווים לאפס מספר משתני החוסר הם כמספר האילוצים.

2) ניתן לקבל פתרון מורחב ראשון שעבורו כל משתני ההחלטה שווים לאפס ומשתני החוסר הופכים להיות למשתנה הבסיס שמקבל את ערכי מקטור b עבור פתרון זה $z=0$

3) נעביר את כל המשתנים בפונקציית המטרה לצד שמאל
נצייר ונמלא את הטבלה הבאה:

מקדמי C					
משתני חוסר = משתני הבסיס	מטריצת A - מקדמים			וקטור b	מבחן המנה $\frac{b_i}{a_k}$
x_4					
x_5					
x_6					
מקדמי C אחרי העברת אגף ל-z c prime				עבור פתרון מורחב ראשון $z=0$	

דוגמא לשיטת הסימפלקס מציאת פתרון מורחב:

1) לדוגמא נתונה הבעיה: $\max\{z=20x_1+60x_2+8x_3\}$

תחת האילוצים:

$$8x_1+2x_2+3x_3\leq 160$$

$$4x_1+3x_2\leq 100$$

$$2x_1+x_3\leq 50$$

$$x_3\leq 20$$

$$x_1\ \ x_2\ \ x_3\geq 0$$

<u>צעד ראשון:</u>

הפיכת אי שוויונים לשוויונים לשם כך נוסיף משתני חוסר לכל אחד מהאילוצים והם: x_4 x_5 x_6 x_7 בהתאמה ומחיריהם בפונק' המטרה הם אפס.

מערכת המשוואות המתקבלת היא:

$$\max\{z=20x_1+6x_2+8x_3\}$$

תחת האילוצים:

$$8x_1+2x_2+3x_3+x_4=160$$

$$4x_1+3x_2+x_5=100$$

$$2x_1+x_3+x_6=50$$

$$x_3+x_7=20$$

$$x_1\geq 0 \quad i=1,\dots,7$$

ממערכת פתרון זו ניתן מיד לקבל את הפתרון המורחב הראשוני שהוא: $x_7=20$ $x_6=50$ $x_5=100$ $x_4=160$ $x_3=0$ $x_2=0$ $x_1=0$

משתני בסיס: $x_4=160$ $x_5=100$ $x_6=50$ $x_7=20$

משתנים לא בסיסיים: $x_1=0$ $x_2=0$ $x_3=0$

$z=0$ הוא ערך פונ' המטרה לפתרון זה.

נשים לב לכך שהבסיס כולל בדיוק 4 משתנים כמספר האילוצים בבעיה הנתונה.

מקדמי המשתנים הבסיסיים יוצרים את מטריצת היחידה I_4 במטריצת המקדמים של הבעיה.

דוגמא לשיטת הסימפלקס-בעזרת טבלה:

1) לדוגמא נתונה הבעיה: $\max\{z=20x_1+60x_2+8x_3\}$

תחת האילוצים:

$8x_1+2x_2+3x_3\le 160$

$4x_1+3x_2\le 100$

$2x_1+x_3\le 50$

$x_3\le 20$

$x_1,\ x_2\ x_3\ge 0$

__צעד ראשון:__

הפיכת אי שיוויונים לשיוונים לשם כך נוסיף משתנה חוסר לכל אחד מהאילוצים והם: x_4 x_5 x_6 x_7 בהתאמה ומחירהם בפונ המטרה הם אפס.

מערכת המשוואות המתקבלת היא:

$\max\{z=20x_1+6x_2+8x_3\}$

תחת האילוצים:

$8x_1+2x_2+3x_3+x_4=160$

$4x_1+3x_2+x_5=100$

$2x_1+x_3+x_6=50$

$x_3+x_7=20$

$x_i\ge 0\quad i=1,\dots,7$

בנוסף נרשום את פונ המטרה הנוכחית בצורה הבאה: (כאשר כל המשתנים נמצאים באגף אחד) $z-20x_1-6x_2-8x_3=0$

פונ המטרה הנוכחית תבוטא בכל שלב כך בעזרת המשתנים הלא בסיסים.

מקדמי המשתנים בפונ המטרה בצורתה המקורית

אגף ימין של אילוצי השיוויון ערכי משתני הבסיס בפתרון נוכחי.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7		
מקדמי C	20	6	8	0	0	0	0		
משתני חוסר = משתני בסיס	מטריצת A - מקדמים							וקטור b (פתרון נוכחי)	מבחן המנה $\frac{b_i}{a_k}$
x_4	8	2	3	1	0	0	0	160	
x_5	4	3	0	0	1	0	0	100	
x_6	2	0	1	0	0	1	0	50	
x_7	0	0	1	0	0	0	1	20	
C' מחירים נוכחיים	-20	-6	-8	0	0	0	0	עבור פתרון מורחב ראשון $Z=0$	

משתני הבסיס הנוכחי.

מקדמי המשתנים באילוצי השיוויון.

מקדמי המשתנים בפונ המטרה הנוכחית

אגף ימין של פונ המטרה הנוכחית

השלב האיטרטיבי - מבחן האופטימליות

מאחר שהעברנו את כל משתני פונ המטרה הנוכחית לאגף שמאל $z-20x_1-6x_2-8x_3=0$

נבחן את מבחן האופטימליות באופן הבא:

הפתרון הנוכחי הוא פתרון אופטימלי אם ורק אם כל המקדמים המופיעים בפונ המטרה הנוכחית (המחירים הנוכחיים) הם אי שליליים.

איטרציה 1 - צעד 1 - בדיקת מבחן אופטימליות:

משתני בסיס	מטריצת המקדמים A							פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	x_1	x_2	x_3	x_4	x_5	x_6	x_7		
x_4	8	2	3	1	0	0	0	160	
x_5	4	3	0	0	1	0	0	100	
x_6	2	0	1	0	0	1	0	50	
x_7	0	0	1	0	0	0	1	20	
c' מחירים נוכחיים	-20	-6	-8	0	0	0	0	$z=0$	

מבחן האופטימליות: הפתרון אינו אופטימלי מאחר שקיימים מחירים נוכחיים שליליים.

איטרציה 1 - צעד 2 - בחירת המשתנה הנכנס לבסיס:

המשתנה בו נבחר להכניס לבסיס יהיה זה שיגדיל את z בשיעור ההתחלתי הגדול ביותר

הקריטריון לבחירת משתנה הנכנס לבסיס יוגדר כך:

המשתנה הנכנס לבסיס יהיה זה שהמקדם שלו בפונ' המטרה הנוכחית (מחירו הנוכחי) מינימלי.

בחירת המשתנה הנכנס - נרצה להכניס לבסיס את המשתנה הלא בסיסי שיש לו ערך c_j' מינימלי.

בדוגמא הנ"ל זהו x_1 כאשר $c_1'=-20$ ולכן x_1 נבחר כמשתנה נכנס.

המשתנה הנכנס: x_1 עמודה $k=1$ תקרא עמודת הציר.

משתני בסיס	מטריצת המקדמים A							פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	x_1	x_2	x_3	x_4	x_5	x_6	x_7		
x_4	8	2	3	1	0	0	0	160	
x_5	4	3	0	0	1	0	0	100	
x_6	2	0	1	0	0	1	0	50	
x_7	0	0	1	0	0	0	1	20	
c' מחירים נוכחיים	-20	-6	-8	0	0	0	0	$z=0$	

בחירת משתנה נכנס לבסיס. המשתנה שמחירו הנוכחי מינימלי (שלילי).

איטרציה 1 צעד 3 - בחירת המשתנה היוצא מהבסיס

על מנת לבחור את המשתנה היוצא מהבסיס בודקים את ההשפעה של הגדלת המשתנה הנכנס לבסיס ע"י ערכיהם של משתני הבסיס הנוכחי.

המשתנה הראשון שמתאפס עקב הגדלה זו הוא משתנה אותו נבחר להוציא מהבסיס

לצורך כך מחשבים את החסמים העליונים שמשרים האילוצים על המשתנה הנכנס ובוחרים בחסם העליון המינימלי

נסתכל לדוגמא על אילוץ השוויון המכיל את המשתנה הבסיסי x_4: $8x_1+2x_2+3x_3+x_4=160$

מכיוון שמשתנים x_2 ו x_3 הם משתנים לא בסיסיים (ונשארו כך גם באיטרציה הבאה) ערכם שווה ל-0 ולכן: $x_4=160-8x_1$

מאילוץ האי שליליות על x_4 נקבל: $160-8x_1 \geq 0$

$$x_1 \leq \frac{160}{8}$$

האיבר החופשי במשוואה בה מופיע x_4 (b_1)

המקדם של x_1 במשוואה בה מופיע x_4 (a_{11})

באופן כללי: $X_k \le \frac{b_i}{a_{ik}}$ ⟵ נשים לב! חסם זה תקף אך ורק כאשר a_{ik} הוא חיובי!

לקריטריון לבחירת משתנה יוצא מהבסיס נקרא מבחן המנה:

המשתנה שיבחר לצאת מהבסיס הוא המשתנה שבשורה שלו (השורה ה-i) נקבל את המנה המינימלית $\frac{b_i}{a_{ik}}$.

בחירת משתנה היוצא מהבסיס המשתנה שהמנה שלו מינימלית.

משתנה בסיס	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	מטריצת המקדמים A								
x_4	8	2	3	1	0	0	0	160	$\frac{160}{8}=20$
x_5	4	3	0	0	1	0	0	100	$\frac{100}{4}=25$
x_6	2	0	1	0	0	1	0	50	$\frac{50}{2}=25$
x_7	0	0	1	0	0	0	1	20	–
מחירים נוכחיים c'	-20	-6	-8	0	0	0	0	$z=0$	

מבחן המנה: בכל שורה נחשב את המנה של הפתרון הנוכחי והאיבר המתאים בעמודת הציר (בתנאי ש- a_{ik} חיובי)!

איטרציה 1 - צעד 3.1 - מבחן המנה:

לבחירת המשתנה היוצא:

נחשב בעמודה האחרונה של הטבלה את היחס $\frac{b_i}{a_{i1}}$

נבחר את המשתנה היוצא כזה שהערך הנ"ל עבורו מינימלי (ראה חישוב בטבלה).

הערך המינימלי שמתקבל הוא $\frac{b_1}{a_{11}}=20$ ולכן x_4 שברשותו התקבל מינימום הוא המשתנה היוצא.

עתה לוקחים את המנה הקטנה המוקפת בעיגול

המשתנה היוצא x_4 השורה שלו תקרא שורת הציר

הערך שנמצא בחיתוך של "שורת הציר" "עמודת הציר" (a_{ik}) ונקרא "ציר ההתמרה".

משתנה בסיס	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	מטריצת המקדמים A								
x_4	8	2	3	1	0	0	0	160	$\frac{160}{8}=20$
x_5	4	3	0	0	1	0	0	100	$\frac{100}{4}=25$
x_6	2	0	1	0	0	1	0	50	$\frac{50}{2}=25$
x_7	0	0	1	0	0	0	1	20	–
מחירים נוכחיים c'	-20	-6	-8	0	0	0	0	$z=0$	

איטרציה 1-צעד 4 - קביעת פיבוט (pivot)

המשתנה x_1 נכנס לבסיס! x_4 יוצא מהבסיס

משתנה בסיס	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	מטריצת המקדמים A								
x_1	8	2	3	1	0	0	0	160	$\frac{160}{8}=20$
x_5	4	3	0	0	1	0	0	100	$\frac{100}{4}=25$
x_6	2	0	1	0	0	1	0	50	$\frac{50}{2}=25$
x_7	0	0	1	0	0	0	1	20	–
מחירים נוכחיים c'	-20	-6	-8	0	0	0	0	$z=0$	

איטרציה 1 צעד 5:

נחלק את איברי שורה של ציר "ההתמרה" בפיבוט ונקבל:

בסיס / משתני	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_1}{a_{1k}}$
	מטריצת המקדמים A								
x_4	1	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{8}$	0	0	0	20	
x_5	4	3	0	0	1	0	0	100	
x_6	2	0	1	0	0	1	0	50	
x_7	0	0	1	0	0	0	1	20	
c' מחירים נוכחיים	-20	-6	-8	0	0	0	0	$z=0$	

איטרציה 1-צעד 6:

נאפס את עמודת הפיבוט ונקבל:

בסיס / משתני	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_1}{a_{1k}}$
	מטריצת המקדמים A								
x_4	1	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{8}$	0	0	0	20	
x_5	0	3	0	0	1	0	0	100	
x_6	0	0	1	0	0	1	0	50	
x_7	0	0	1	0	0	0	1	20	
c' מחירים נוכחיים	0	-6	-8	0	0	0	0	$z=0$	

<u>עתה השלב של המרת מערכת המשוואות - איך עושים זאת?</u>

באופן כללי הערך החדש של האיבר ה- a_{ij} לאחר ביצוע ההתמרה הלינארית יהיה: $a_{ij} \leftarrow a_{ij} - \frac{a_{ik} \cdot a_{rj}}{a_{rk}}$

באופן דומה נחשב את הערכים החדשים של c'_j ושל b_i:

$$b_i \leftarrow b_i - \frac{a_{ik} \cdot b_r}{a_{rk}} \qquad c'_j \leftarrow c'_j - \frac{c'_k \cdot a_{rj}}{a_{rk}}$$

האיבר החדש = האיבר הישן − $\frac{\text{הקדמות של האלכסון הנגדי}}{\text{pivot}}$

נחשב b-ים:

$$b'_4 = b_4 - \frac{a_{41} \cdot b_1}{a_{11}} \longrightarrow b'_4 = 20 - \frac{0 \cdot 160}{8} = 20 \qquad b'_3 = b_3 - \frac{a_{31} \cdot b_1}{a_{11}} \longrightarrow b'_3 = 50 - \frac{2 \cdot 160}{8} = 10 \qquad b'_2 = b_2 - \frac{a_{21} \cdot b_1}{a_{11}} \longrightarrow b'_2 = 100 - \frac{4 \cdot 160}{8} = 20$$

נחשב C-ים: $c'_j = c'_j - \frac{c_1 \cdot a_{ij}}{a_{11}}$:

$$c'_2 = -6 - \frac{-20 \cdot 2}{8} = -6+5 = -1 \qquad c'_3 = -8 - \frac{-20 \cdot 3}{8} = -8+7 \cdot \frac{4}{8} = -\frac{4}{8} \qquad c'_4 = 0 - \frac{-20 \cdot 1}{8} = \frac{20}{8} \qquad c'_5 = 0 - \frac{-20 \cdot 0}{8} = 0$$

חישוב דומה לחישובים שעשינו עד כה נעשה עבור פונ' מטרה:

$$z' = z - \frac{c_1 \cdot b_1}{a_{11}} \longrightarrow z' = 0 - \frac{-20 \cdot 160}{8} = 400$$

<u>איטרציה 1 - צעד 7 המרת מערכת השוואות:</u>

בסוף האיטרציה ה-1 נקבל:

משתני בסיס	x_1	x_2	x_3	x_4	x_5	x_6	x_7	פתרון נוכחי	מבחן המנה $\frac{b_i}{a_{ik}}$
	מטריצת המקדמים A								
x_4	1	$1/4$	$3/8$	$1/8$	0	0	0	20	
x_5	0	2	$-3/2$	$-1/2$	1	0	0	20	
x_6	0	$-1/2$	$1/4$	$-1/4$	0	1	0	10	
x_7	0	0	1	0	0	0	1	20	
מחירים נוכחיים c'	0	-1	$-\frac{1}{2}$	$\frac{5}{2}$	0	0	0	$z=400$	

ניתן לראות כי זהו עדיין לא הפתרון האופטימלי ולכן עלינו להמשיך באיטרציה נוספת על מנת להגיע ששורת המחירים הנוכחיים הינה חיובית או שווה לאפס כולה.

לאחר שמגיעים לפתרון האופטימלי כלומר שהשורה של המחירים נוכחיים c' הינה גדולה מאפס או שווה.

עוזרים ורושמים: הפתרון האופטימלי המצומצם מתקבל בנק' $(x_1=\frac{25}{4}, x_2=25, x_3=20)$ והערך המקסימלי של פונ המטרה $Z=435$

(כל השאר יהיו משתנים אחרים).

כאשר קיימת בעיית מינימום כלומר פונ המטרה היא min שלבי הפתרון הינם אותם שלבי פתרון מקס? שבשביל להגיע לפתרון אופטימלי נדרש להגיע לכך ששורת המחירים הנוכחיים תהיה כולה קטנה או שווה ל-0.

פתרונות מרובים:

שיטת הסימפלקס עוזרת ברגע שהיא מוצאת פתרון אופטימלי ראשון.

יתכן ולבעיה קיימים מספר פתרונות אופטימלים

נזהה מקרים כאלו אם בטבלה האחרונה שבה מצאנו את הפתרון האופטימלי המקדם של המשתנה לא בסיסי מסוים בפונ המטרה C_j' שווה אפס!

במידה וקיימים שניים או יותר פתרונות אופטימלים לבעיה קיימים לה אינסוף פתרונות.

תרגול 3/3/21.

(1

$$\max\{z=3x_1+2x_2\}$$

$$s.t:\ 3x_1+2x_2\le 6$$

$$x_1\ge 0 \quad x_2\ge 0$$

ראשית נעביר את פונ' המטרה: $z-3x_1-2x_2=0$ ונוסיף משתנה חוסר. מאחר ויש אילוץ אחד נוסיף משתנה אחד (x_3)

משתני בסיס	מטריצת המקדמים A x_1	x_2	x_3	פתרון נוכחי b	מבחן המנה
x_3	3	2	1	6	$\frac{6}{3}=2$
C' מחירים נוכחיים	-3	-2	0	$z=0$	

משתנה הכי מינימלי

ניצור טבלה נוספת ובה נאפס את עמודת הציר ואת שורת הציר נחלק ב- pivot המשתנה הנכנס x_1 המשתנה היוצא x_3

משתני בסיס	מטריצת המקדמים A x_1	x_2	x_3	פתרון נוכחי b	מבחן המנה
x_1	1	$\frac{2}{3}$	$\frac{1}{3}$	2	
C' מחירים נוכחיים	0	-2	0	$z=0$	

נחשב לפי הטבלה לפני האיפוס:

$$z=z-\frac{c_1\cdot b_1}{a_{11}}=0-\frac{-3\cdot 6}{3}=6$$

$$C_3=0-\frac{-3\cdot 1}{3}=1$$

$$C_2=-2-\frac{2\cdot(-3)}{3}=0$$

משתני בסיס	מטריצת המקדמים A x_1	x_2	x_3	פתרון נוכחי b	מבחן המנה
x_1	1	$\frac{2}{3}$	$\frac{1}{3}$	2	
C' מחירים נוכחיים	0	0	1	$z=6$	

זהו פתרון אופטימלי, מאחר וכל המחירים הנוכחיים הינם חיוביים או שווים ל-0.

ניתן לראות כי מאחר ו- x_2 הינו משתנה לא בסיס שהמקדם שלו אפס מכאן נובע אינסוף פתרונות.

מערכת שעות סמסטר ב

שני	שלישי	רביעי	חמישי
הנדסת דרישות 10:00-13:50	סטטיסטיקה 08:00-11:50	שפות תכנות 08:00-10:50	תקשורת מחשבים לתוכנה -מעבדה- 09:00-10:50
מערכות הפעלה 14:00-17:50	חלון 12:00-13:00	מודלים חישוביים 11:00-12:50	חלון 11:00-12:00
	תכנון וניתוח אלגוריתמים 13:00-14:50	חלון 13:00-14:00	מודלים חישוביים 12:00-14:50
	תקשורת מחשבים לתוכנה 15:00-18:50	תכנון וניתוח אלגוריתמים 14:00-17:50	

בעיית b-ים שליליים

במקרים רבים צד הימני של האילוץ מופיע כמספר שלילי

יכול לנבוע מצורתו המקורית של האילוץ והן כתוצאה משינוי שבוצע באילוץ על מנת לשנות את כיווני

למשל: $x_1-10x_2\le 5 \iff -x_1+10x_2\ge 5$

הבעיה נוצרת ע"י b-ים שליליים היא שהוספת משתני חוסר אינה מאפשרת קבלת פתרון בסיסי אפשרי היות ואחד או יותר ממשתני בסיס במקרה הזה יקבל ערך שלילי, מאחר וה-b המתאים לו שלילי.

לדוגמא נתונה הבעיה הבאה:

$$\min\{z=8x_1+4x_2+6x_3\}$$

$$\begin{aligned} s.t\quad & 2x_1+x_2+3x_3\ge 2 \qquad x_j\ge 0 \quad j=1,2,3\\ & 5x_1+x_2+2x_3\ge 3\\ & 6x_1+3x_2-x_3\ge 4 \end{aligned}$$

פתרון:

שלב ראשון:

הפיכת אי שוויונים לשוויונים, לשם כך נחסיר מכל משוואה "משתנה עודף" המקביל ל"משתנה חוסר" באי שוויונים עם $\ge$.

מחיר "משתנה העודף" בפונק המטרה יהיה אפס.

המערכת המתקבלת היא:

$$\min\{z=8x_1+4x+6x_3\}$$

$$\begin{aligned} s.t:\quad & 2x_1+x_2+3x_3-x_4=2 \qquad x_j\ge 0 \quad j=1....6\\ & 5x_1+x_2+2x_3-x_5=3\\ & 6x_1+3x_2-x_3-x_6=4 \end{aligned}$$

נעבוד על בעיית מינימום בשיטת הסימפלקס כאשר השינוי היחיד שיתבצע בתהליך הוא הקריטריון לאופטימליות (כפי שלמדנו).

הבעיה במערכת המשוואות המתקבלת היא שלא ניתן לזהות מיד פתרון בסיסי ראשוני.

x_4 x_5 x_6 אינם יכולים להוות פתרון בסיסי ראשוני המקבל את ערכי האילוצים, היות והם יהיו שליליים והפתרון לא יהיה אפשרי.

על מנת להתגבר על בעיה זו נבנה פתרון בסיסי ראשוני באמצעות הוספת "משתנים מלאכותיים" - משתנה אחד לכל משוואה רק ממשתנים אלו נדרוש אי שליליות

מערכת האילוצים שתתקבל היא:

$$\begin{aligned} & 2x_1+x_2+3x_3-x_4+y_1=2 \qquad x_j\ge 0 \quad j=1...6 \qquad y_j\ge 0 \quad j=1,2,3\\ & 5x_1+x_2+2x_3-x_5+y_2=3\\ & 6x_1+3x_2-x_3-x_6+y_3=4 \end{aligned}$$

מערכת האילוצים שהתקבלה אינה זהה למערכת המקורית.

כדי לקבל פתרון שיהיה זהה לפתרון הבעיה המקורי עלינו לדרוש שבפתרון האופטימלי המשתנים המלאכותיים שהוספנו ישוו לאפס

נעשה זאת בשיטת ה-M הגדול

שיטת ה-M הגדול

The big M method.

ניתן למשתנים המלאכותיים שהוספנו מחיר M בפונ' המטרה כאשר M הוא ערך גדול מאוד

עבור בעיית מקסימום ניתן למשתנים המלאכותיים מחיר -M.

במצב כזה הערך האופטימלי של פונ' המטרה מושג כאשר ערכי המשתנים המלאכותיים יגיעו לאפס.

אם זה לא קורה הרי שאין פתרון אפשרי לבעיה

לאחר מתן המחירים הנ"ל אנו ממשיכים בתהליך הסימפלקס הרגיל ומקבלים את הפתרון הדרוש.

במקרה שלנו פונ' המטרה תהיה:

$$\min\{Z=8x_1+4x_2+6x_3+My_1+My_2+My_3\}$$

הפתרון הבסיסי הראשוני הוא:

$$x_1=x_2=x_3=x_4=x_5=x_6=0$$

$$y_1=2M \quad y_2=3M \quad y_3=4M$$

$$Z=9M$$

אין זה פתרון אפשרי לבעיה!

רק בשלב שבו כל ה-y יצאו מהבסיס ויקבלו את הערך 0 נקבל פתרון בסיסי אפשרי לבעיה המקורית

כעת נציג את הבעיה שהתחלנו לפתור בטבלה ונריץ את הסימפלקס:

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
	בסיס	$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_1}{a_{1k}}$
1	y_1	2	1	3	-1	0	0	1	0	0	2	
2	y_2	5	1	2	0	-1	0	0	1	0	3	
3	y_3	6	3	-1	0	0	-1	0	0	1	4	
	C'_j										9M	= Z

נחשב C'_j-ים:

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
	בסיס	$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_1}{a_{1k}}$
1	y_1	2	1	3	-1	0	0	1	0	0	2	
2	y_2	5	1	2	0	-1	0	0	1	0	3	
3	y_3	6	3	-1	0	0	-1	0	0	1	4	
	C'_j	13M-8	5M-4	4M-6	-M	-M	-M	0	0	0	9M	= Z

הפתרון אינו אופטימלי כי ישנם C'_j-ים חיוביים (עבור x_1, x_2, x_3) והמקסימלי זהו x_1 ולכן זהו המועמד לבסיס הבא

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_1}{a_{1k}}$
1	y_1	2	1	3	-1	0	0	1	0	0	2	1
2	y_2	5	1	2	0	-1	0	0	1	0	3	$\frac{3}{5}$
3	y_3	6	3	-1	0	0	-1	0	0	1	4	$\frac{2}{3}$
	c'_j	$13M-8$	$5M-4$	$4M-6$	$-M$	$-M$	$-M$	0	0	0	$9M$	$= Z$

מהחישוב של הערכים $\frac{b_i}{a_{ik}}$ מתקבל כי המשתנה שיעזוב את הבסיס הוא y_2 נבצע את ההתמרה לפי ציר ההתמרה 5 ונקבל הטבלה הבאה:

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_1}{a_{1k}}$
1	y_1	0	$\frac{3}{5}$	$\frac{11}{5}$	-1	$\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	$\frac{4}{5}$	
2	x_1	1	$\frac{1}{5}$	$\frac{2}{5}$	0	$-\frac{1}{5}$	0	0	$\frac{1}{5}$	0	$\frac{3}{5}$	
3	y_3	0	$\frac{9}{5}$	$\frac{-17}{5}$	0	$\frac{6}{5}$	-1	0	$\frac{-6}{5}$	1	$\frac{2}{5}$	
	c'_j	0	$\frac{12}{5}M-\frac{12}{5}$	$-\frac{6}{5}M-\frac{14}{5}$	$-M$	$\frac{8}{5}M-\frac{8}{5}$	$-M$	0	$\frac{-13}{5}M+\frac{8}{5}$	0	$\frac{6}{5}M+\frac{24}{5}$	$= Z$

עדיין הפתרון לא אופטימלי כי קיימים c_j חיוביים המקסימלי ביניהם מתקבל עבור x_2

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_1}{a_{1k}}$
1	y_1	0	$\frac{3}{5}$	$\frac{11}{5}$	-1	$\frac{2}{5}$	0	1	$-\frac{2}{5}$	0	$\frac{4}{5}$	$\frac{4}{3}$
2	y_2	1	$\frac{1}{5}$	$\frac{2}{5}$	0	$-\frac{1}{5}$	0	0	$\frac{1}{5}$	0	$\frac{3}{5}$	3
3	y_3	0	$\frac{9}{5}$	$\frac{-17}{5}$	0	$\frac{6}{5}$	-1	0	$\frac{-6}{5}$	1	$\frac{2}{5}$	$\frac{2}{9}$
	c'_j	0	$\frac{12}{5}M-\frac{12}{5}$	$-\frac{6}{5}M-\frac{14}{5}$	$-M$	$\frac{8}{5}M-\frac{8}{5}$	$-M$	0	$\frac{-13}{5}M+\frac{8}{5}$	0	$\frac{6}{5}M+\frac{24}{5}$	$= Z$

המשתנה היוצא מהבסיס הוא y_3 ציר ההתמרה הוא $\frac{9}{5}$

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_i}{a_{ik}}$
1	y_1	0	0	$\frac{10}{3}$	-1	0	$\frac{1}{3}$	1	0	$-\frac{1}{3}$	$\frac{2}{3}$	
2	x_1	1	0	$\frac{7}{9}$	0	$-\frac{1}{3}$	$\frac{1}{9}$	0	$\frac{1}{3}$	$-\frac{1}{9}$	$\frac{5}{9}$	
3	x_2	0	1	$-\frac{17}{9}$	0	$\frac{2}{3}$	$-\frac{5}{9}$	0	$-\frac{2}{3}$	$\frac{5}{9}$	$\frac{2}{9}$	
	c'_j	0	0	$\frac{10}{3}M-\frac{22}{3}$	$-M$	0	$\frac{1}{3}M-\frac{4}{3}$	0	$-M$	$-\frac{4}{3}M+\frac{4}{3}$	$\frac{2}{3}M+\frac{16}{3}$	$=Z$

עדין הפתרון לא אופטימלי כי קיימים C_j-ים חיוביים המקסימלי בינהם מתקבל עבור X_3

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3		
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_i}{a_{ik}}$
1	y_1	0	0	$\frac{10}{3}$	-1	0	$\frac{1}{3}$	1	0	$-\frac{1}{3}$	$\frac{2}{3}$	
2	x_1	1	0	$\frac{7}{9}$	0	$-\frac{1}{3}$	$\frac{1}{9}$	0	$\frac{1}{3}$	$-\frac{1}{9}$	$\frac{5}{9}$	
3	x_2	0	1	$-\frac{17}{9}$	0	$\frac{2}{3}$	$-\frac{5}{9}$	0	$-\frac{2}{3}$	$\frac{5}{9}$	$\frac{2}{9}$	
	c'_j	0	0	$\frac{10}{3}M-\frac{22}{3}$	$-M$	0	$\frac{1}{3}M-\frac{4}{3}$	0	$-M$	$-\frac{4}{3}M+\frac{4}{3}$	$\frac{2}{3}M+\frac{16}{3}$	$=Z$

המשתנה היוצא מהבסיס הוא y_1. ציר ההתמרה הוא $\frac{10}{3}$.

		x_1	x_2	x_3	x_4	x_5	x_6	y_1	y_2	y_3	פתרון	
	מחירים	8	4	6	0	0	0	M	M	M		
		$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_3$	$\bar{a}_4$	$\bar{a}_5$	$\bar{a}_6$	$\bar{a}_7$	$\bar{a}_8$	$\bar{a}_9$	$\bar{b}$	$\frac{b_i}{a_{ik}}$
1	x_3	0	0	1	$-\frac{3}{10}$	0	$\frac{1}{10}$	$\frac{3}{10}$	0	$-\frac{1}{10}$	$\frac{1}{5}$	
2	x_1	1	0	0	$\frac{7}{30}$	$-\frac{1}{3}$	$\frac{1}{30}$	$-\frac{7}{30}$	$\frac{1}{3}$	$-\frac{1}{30}$	$\frac{2}{5}$	
3	x_2	0	1	0	$-\frac{17}{30}$	$\frac{2}{3}$	$-\frac{11}{30}$	$\frac{17}{30}$	$-\frac{2}{3}$	$\frac{11}{30}$	$\frac{3}{5}$	
	c'_j	0	0	0	$-\frac{11}{5}$	0	$-\frac{3}{5}$	$-M+\frac{11}{5}$	$-M$	$-M+\frac{3}{5}$	$\frac{34}{5}$	$=Z$

הפעם זהו הפתרון האופטימלי כי לא נותרו C_j-ים חיוביים.

הפתרון האופטימלי שקיבלנו: $x_1=\frac{2}{5}$ $x_2=\frac{3}{5}$ $x_3=\frac{1}{5}$ (המשתנים הבסיסים).

כל שאר המשתנים הלא בסיסיים הינם 0.

ערך פונ' המטרה הינו $z=\frac{34}{5}$

הפתרון מורכב כולו מכל המשתנים המקוריים וכל משתני העודף שווים ל-0 כלומר האילוצים מנוצלים במלואם!

בנוסף שהמשתנים המלאכותיים אינם מופיעים בפתרון האופטימלי אחרת זה לא היה פתרון חוקי.

מסקנות:

משתני הבסיס לפי הסדר הם: $\{x_3, x_1, x_2\}$ לכן:

$$B=\begin{array}{c} \begin{array}{ccc} x_3 & x_1 & x_2 \end{array} \\ \begin{pmatrix} 3 & 2 & 1 \\ 2 & 5 & 1 \\ 1 & 6 & 2 \end{pmatrix} \end{array} \qquad B^{-1}=\begin{array}{c} \begin{array}{ccc} y_1 & y_2 & y_3 \end{array} \\ \begin{pmatrix} 3/10 & 0 & -1/10 \\ -7/30 & 1/3 & -1/30 \\ 17/30 & -2/3 & 11/30 \end{pmatrix} \end{array}$$

לכן:

$$X_B=B^{-1}\cdot b=\begin{pmatrix} 3/10 & 0 & -1/10 \\ -7/30 & 1/3 & -1/30 \\ 17/30 & -2/3 & 11/30 \end{pmatrix}\cdot\begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}=\begin{pmatrix} 1/5 \\ 2/5 \\ 3/5 \end{pmatrix}=\begin{pmatrix} x_3 \\ x_1 \\ x_2 \end{pmatrix}$$

סיכום השימוש בשיטת ה-M הגדול לפתרון בעיות עם b-ים שליליים

עבור בעיית max:

1) הפוך את כל מרכיבי b לאי שליליים ע"י הכפלת אי שוויונים בהם b-ים שליליים ב 1-.

2) עבור אילוצים המופיעים כשוויון, הוסף משתנה מלאכותי עם מקדם 1+

3) עבור אי שוויונים שהיו הפוכים בכיוונם הוסף משתנה עודף עם מקדם 1- ומשתנה מלאכותי עם מקדם 1+.

4) הוסף כל משתנה מלאכותי (בעל מקדם 1+) לפונ' המטרה עם מחיר M- כאשר M מספר חיובי גדול מאוד.

5) לכל אי שוויון בכיוון הנכון הוסף משתנה חוסר עם מקדם 1+.

6) פתור את הבעיה בשיטת הסימפלקס

7) לבעיה המקורית קיים פתרון אפשרי רק אם המשתנים המלאכותיים עם מקדם M- בפונ' המטרה אינם בבסיס האופטימלי

עבור בעיית מינימום (min) יש לתת למשתנים המלאכותיים מחיר M+ בפונ' המטרה.

גרפים

גרף מורכב מקבוצה סופית לא ריקה של קודקודים (vertices) ומקבוצת צלעות (edges).

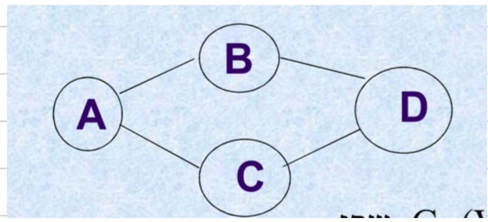
סימון של גרף: $G=(V,E)$ כאשר:

G - מציין גרף.

V- מציין קבוצת קודקודים.

E- מציין קבוצת צלעות.

<u>התרשים הבא:</u>



מתאר גרף שבו $G=(V,E)$:

$V=\{A,B,C,D\}$

$E=\{(A,B),(A,C),(B,D),(C,D)\}$

נשים לב שכל קשת הקו המקשר בין זוג קודקודים בגרף מצויינת ע"י זוג קודקודים.

הזוג (A,B) מציין קשת המחברת את הקודקודים A ו B

הזוג (A,C) מציין קשת המחברת את הקודקודים A ו C

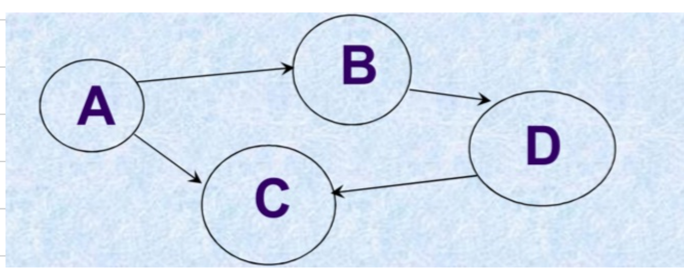
הזוג (B,D) מציין קשת המחברת את הקודקודים B ו D

הזוג (C,D) מציין קשת המחברת את הקודקודים C ו D

נשים לב כי הזוגות אינם סדורים לומר במקום הזוג (A,B) יכולים לציין (B,A)

<u>גרף מכוון - directed graph</u>: זהו גרף שבו הקשתות מכוונות, המכונה גם קבוצה של זוגות סדורים.

<u>התרשים הבא מייצג גרף מכוון:</u>

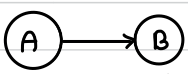


גרף $G=(V,E)$ שבו

$V=\{A,B,C,D\}$

$E=\{(A,B),(A,C),(B,D),(D,C)\}$

נשים לב לזוג סדור אשר מייצג קשת מכוונת, נתבונן בקשת



הקשת זו יוצאת מקודקוד A ונכנסת לקודקוד B מתוארת ע"י זוג סדור (A,B).

1. <u>הקודקוד הראשון</u> בזוג הקודקודים הסדור מציין את מקור הקשת (מאיזה קודקוד יוצאת הקשת).

2. <u>הקודקוד השני</u> בזוג הקודקודים הסדור מציין את היעד של הקשת (לאיזה קודקוד נכנסת הקשת).

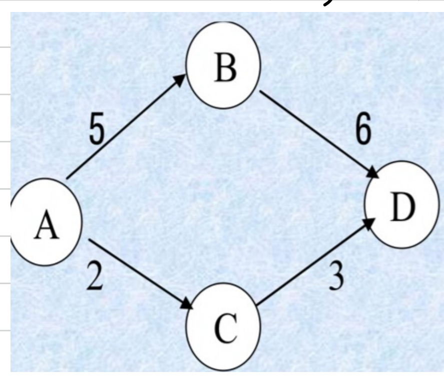
כל עץ הינו גרף אך לא כל גרף הינו עץ.

סימונים עבור גרף $G=(V,E)$:

1. מספר הצמתים בגרף מסומן $|V|$
2. מספר הקשתות בגרף מסומן $|E|$

רשת (network): הינה גרף (מכוון/לא מכוון) שבו לכל קשת מיוחס מספר/מספרים.
המספרים המיוחסים לקשת נקראים משקל/ות הקשת (weight)
רשת מכונה גם כגרף משוקלל (weight graph).

התרשים הנ"ל מתאר רשת:



לכל קשת מיוחסת מספר, אשר יכול לייצג מחיר, מרחק.
מקסימום כמות הזרם שניתן להזרים מקודקוד אחד לאחר וכו'.

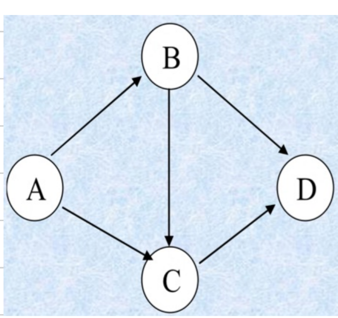
צומת a סמוך (או עוקב) adjacent: לקודקוד b אם יש קשת לא מכוונת בין קודקוד a לקודקוד b
או אם יש קשת מכוונת מקודקוד b לקודקוד a.
בתרשים האחרון (המציג מהי רשת): קודקוד B סמוך לצומת A מאחר שיש קשת
מכוונת מקודקוד A ל-B.
קודקוד C סמוך לצומת A מאחר שיש קשת מכוונת מקודקוד A לקודקוד C וכו'.

דרגת הכניסה (indegree) של קודקוד- מספר הקשתות הנכנסות לקודקוד

דרגת היציאה (outdegree) של קודקוד- מספר הקשתות היוצאות מקודקוד

דרגה (degree) של קודקוד- מספר הקשתות הנוגעות בו.

בתרשים הנ"ל:



עבור קודקוד D:
הדרגה שלו- 2, דרגת הכניסה שלו- 2
דרגת יציאה- 0.

עבור קודקוד C:
הדרגה שלו- 3, דרגת הכניסה שלו- 2
דרגת יציאה- 1

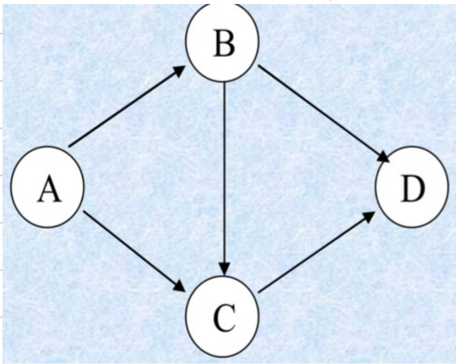
מולטי גרף (multigraph): הינו גרף (מכוון/לא מכוון) בו אנו מרשים יותר מקשת אחת בין שני קודקודים.

לולאה (loop): הינה קשת מצומת כלשהי V לאותה הצומת כלומר קשת מהצורה (V,V) נקראת לולאה.

גרף פשוט (simple graph): הינו גרף ללא לולאות ושאינו מולטי גרף.

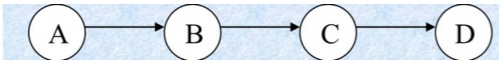
מסלול (path) באורך k: מקודקוד a לקודקוד b הינה סדרה של k+1 קודקודים בגרף $n_1\ n_2\ n_3\ n_4\ n_5 \ldots\ n_i\ n_{i+1} \ldots n_k\ n_{k+1}$
כך שלכל $1\le i\le k$ קודקוד n_{i+1} סמוך לקודקוד n_i בגרף ו $n_1=a$ $n_{k+1}=b$.

בתרשים הנ"ל:



קיים מסלול באורך 3 מקודקוד A לקודקוד D
מאחר שישנה סדרה בגרף של קודקודים באורך 4 A,B,C,D
ישנה קשת מקודקוד A ל B
ישנה קשת מקודקוד B ל C
ישנה קשת מקודקוד C ל D

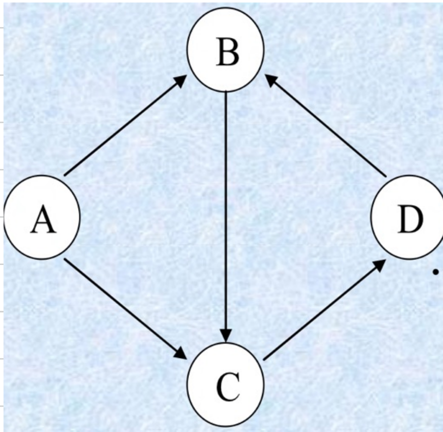
תיאור המסלול:



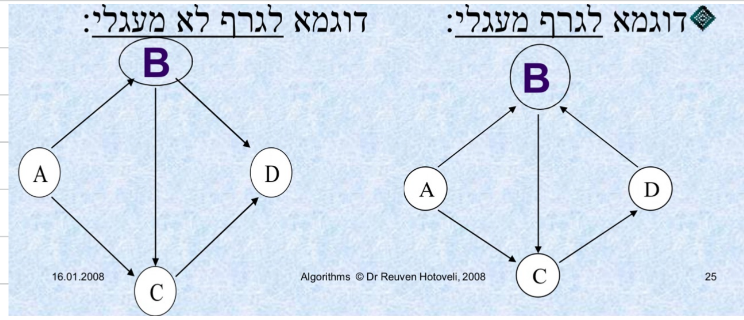
קודקוד נשיג: קודקוד b יקרא נשיג (להשיג) מקודקוד a בגרף אם קיים מסלול באורך כלשהו בגרף מקודקוד a לקודקוד b.
בתרשים האחרון, קודקוד D נשיג מקודקוד A מאחר ויש מסלול באורך 3 מקודקוד A לקודקוד D

מעגל (cycle) - מסלול בגרף מקודקוד כלשהו לעצמו.
בתרשים הנ"ל:

מעגל מקודקוד B ל B
המסלול הינו $B\to C\to D\to B$



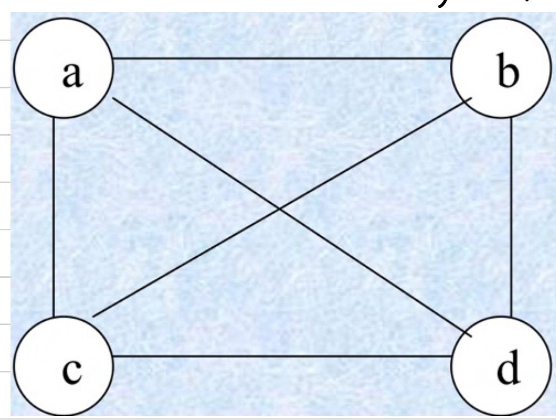
גרף מעגלי (cycle graph) - גרף שבו לפחות מעגל אחד, אחרת הגרף נקרא לא מעגלי (acycle).



מסלול פשוט (simple path) - הינו מסלול בו אף קשת של הגרף לא מופיעה יותר מפעם אחת.
באופן אנלוגי כך ניתן גם להגדיר מעגל פשוט.

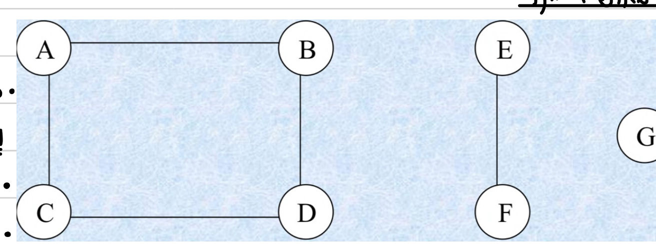
גרף שלם (או מלא) (complete graph) - הינו גרף שבו כל קודקוד מחובר לשאר קודקודי הגרף.
סימון: גרף מלא שבו n קודקודים יסומן ב- K_n.

בתרשים הנ"ל K_4:



רכיב קשיר (connected Complete) - זהו קבוצה מקסימלית של קודקודים בגרף לא מכוון שבה יש מסלול פשוט בין כל שני קודקודים בגרף.
קודקוד בודד ללא שכנים יקרא גם כן רכיב קשר

בתרשים הנ"ל:

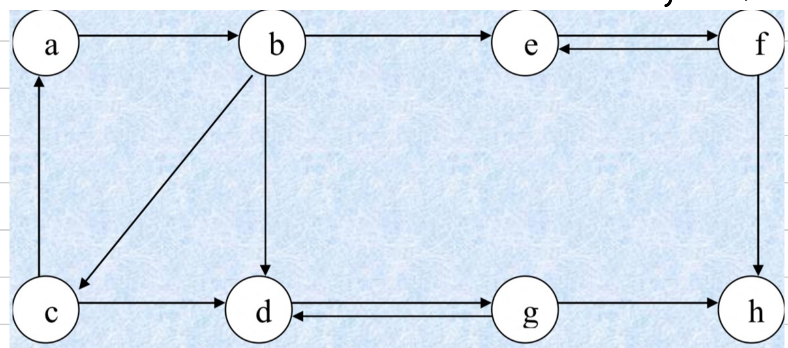


קיימים 3 רכיבים קשירים:
- רכיב קשיר אחד - הינו קבוצה מקסימלית של קודקודים בגרף והיא {A,B,C,D} מאחר והקבוצה זו יש מסלול פשוט בין כל שני קודקודים בגרף.
- רכיב קשיר שני - קבוצה מקסימלית של קודקודים בגרף והיא {E,F}
- רכיב קשיר שלישי - {G} מאחר וקודקוד בודד יכול להיקרא גם כן רכיב קשיר.

גרף קשיר (connected graph) - מכונה גם "גרף מחובר" הינו גרף לא מכוון בעל רכיב קשיר אחד בלבד
כלומר קיים מסלול פשוט בין כל שני קודקודים בגרף.

רכיב קשיר מכוון בחוזקה (רקמ"ח) (strongly connected directed graph) - הינה קבוצה מקסימלית של קודקודים בגרף מכוון שבו יש מסלול פשוט ומכוון בין כל שני קודקודים בגרף.
קודקוד בודד ללא שכנים יקרא גם רכיב קשיר מכוון בחוזקה

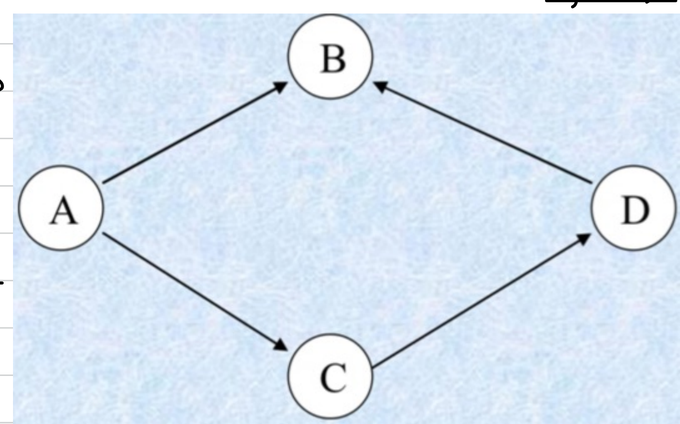
התרשים הנ"ל:



מתאר גרף מכוון שבו יש 4 רכיבי קשירות חוזקה.
הרכיבים $\{a,b,c\}$, $\{h\}$ $\{e,f\}$ $\{d,g\}$

גרף דו צדדי (bipartite graph) - הינו גרף שבו קבוצת הצמתים מתחלקת לשתי קבוצות זרות $V=V_1\cup V_2$.
כך ש- V_1 ו- V_2 קבוצות זרות באופן שאם $x,y\in V_1$ או $x,y\in V_2$
אז לא קיימת קשת בין x לy נהוג לסמן גרף דו צדדי ע"י $G=(V_1,V_2,E)$

התרשים הנ"ל:

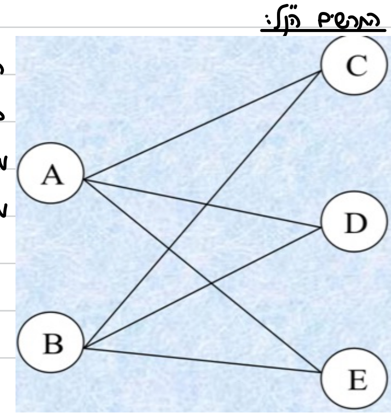


התרשים מתאר גרף דו צדדי בגרף זה $V_1=\{A,D\}$ $V_2=\{B,C\}$
קשתות בגרף זה יוצאות מקודקוד כלשהו השייך ל- V_1 אל קודקוד אחר השייך ל- V_2 או להיפך.
אין קשתות מקודקוד אחד השייך ל- V_1 לקודקוד אחר השייך ל- V_1 כנ"ל לגבי V_2.

הערה: גרף דו צדדי יכול להיות גרף מכוון או לא מכוון

גרף דו צדדי מלא - הינו גרף דו צדדי $G=(V_1,V_2,E)$ בו לכל $x\in V_1$ ולכל $y\in V_2$ קיימת קשת $(x,y)\in E$
גרף זה יסומן ע"י K_{nm} כאשר $|V_2|=m$ $|V_1|=n$ כדי לדעת כמה קודקודים סה"כ נחשב $m\cdot n$

התרשים הנ"ל:

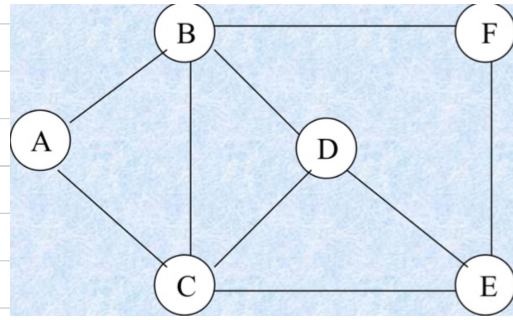


התרשים מתאר את K_{23}
בגרף זה: $V_1=\{A,B\}$ $V_2=\{C,D,E\}$
מקודקוד A ישנה קשת לכל קודקוד אחר השייך ל- V_2 ומכל קודקוד השייך ל- V_2 ישנה קשת לקודקוד A.
מקודקוד B ישנה קשת בינו לבין כל קודקוד אחר השייך ל- V_2.

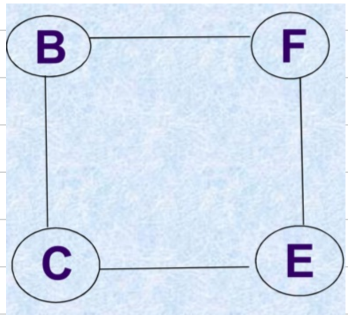
תת גרף (Sub-graph) - של $G=(V,E)$ הינו גרף $G=(V',E')$ כך ש: $V'\subseteq V$ ו $E'\subseteq E$

התרשים הנ"ל מתאר:

בהינתן הגרף הבא:



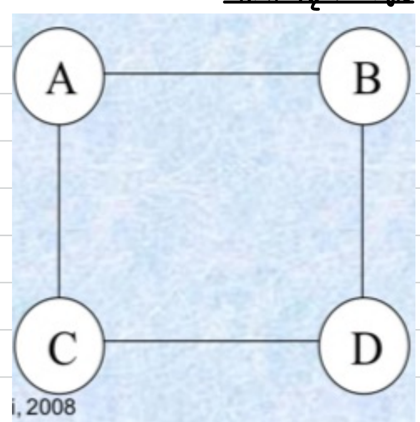
התרשים הבא מתאר תת הגרף של הגרף הנתון:



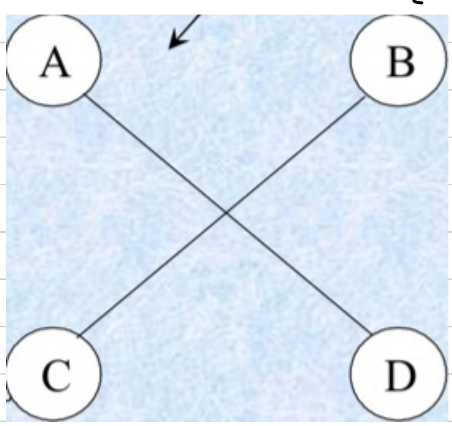
גרף משלים - נתון גרף פשוט $G=(V,E)$ חסר לולאות וצלעות מרובות. גרף משלים של G מסומן כ- $\bar{G}$ ומוגדר כדלהלן. $\bar{G}=(V,\bar{E})$ כאשר $\bar{E}$ הינה אוסף הצלעות החסרות ב-G בין כל שני קודקודים כלשהם בגרף.

התרשים הנ"ל:

עבור הגרף הבא:



הגרף המשלים:



משפטים הקשורים לגרפים

1. נתון גרף קשיר לא מכוון $G=(V,E)$ אם $|V|=n$ ו $|E|\geq n$ אזי בגרף יש מעגל.

2. התנאים הבאים שקולים:
   א) הגרף G הינו עץ
   ב) G קשיר ובעל $n-1$ קשתות
   ג) G חסר מעגלים ובעל $n-1$ קשתות

3. בכל גרף פשוט $G=(V,E)$ מתקיים $\sum d(v)=2|E|$ כאשר $d(v)$ מציין דרגת של צומת v בגרף.

<u>נימוק:</u> כל צלע בגרף G מחברת בדיוק שני קודקודים כלומר כל צלע יוצאת בדיוק משני קודקודים ולכן כל צלע ב-G נמנית בדיוק פעמיים בחשוב הדרגה של הקודקודים.

פעולות בסיסיות על גרפים

1. <u>הוספת קשת join</u> - הפעולה join(x,y,g) או "הוסף קשת (x,y,g)" מוסיפה קשת מצומת x לצומת y בגרף g.

2. <u>הוספת קשת משוקללת (קשת שעליה משקל) joinw</u> - הפעולה joinw(x,y,w,g) או "הוסף קשת משוקללת (x,y,w,g)" מוסיפה קשת משוקללת מצומת x לצומת y ברשת (גרף משוקלל).

3. <u>הסרת קשת remv</u> - הפעולה remv(x,y,g) או הסר קשת(x,y,g)" מסירה קשת מצומת x לצומת y בגרף g.

4. <u>הסרת קשת משוקללת (x,y,w,g)</u> - מסירה קשת מצומת x לצומת y ברשת g ומציבה ב-w את משקל הקשת המוסרת.

5. <u>בדיקת שכנות (סמיכות) של קודקודים adjacent</u> - הפעולה adjacent(x,y,g) או סמוכים? (x,y,g) מחזירה ערך "אמת"(true) אם קודקוד y שכן (סמוך) לקודקוד x אחרת, מחזירה ערך "שקר"(false).

<u>בהתאם למימוש הגרף (אותו נראה בהמשך) לא תמיד ניתן מראש את מספר הצמתים (קודקודים) בגרף, במקרים אלו להלן מספר פעולות בסיסיות על גרפים:</u>

1. <u>בדיקת גרף ריק graph_is_empty</u> - הפעולה graph_is_empty(G) או "גרף ריק(G)" מחזירה ערך אמת (true) אם גרף G ריק אחרת מחזירה ערך שקר (false).

2. <u>בדיקת שייכות אלמנט בקבוצת הצמתים של הגרף findelement</u> - הפעולה findelement(G,X) או "קיים אלמנט?(G,X)" היא פעולה המקבלת גרף ואלמנט X ומחזירה מצביע (מיקום) לצומת המידע עם המידע X אחרת הפעולה תחזיר מצביע (מיקום) לכלום.

3. <u>הוספת צומת - addnode</u> הפעולה addnode(G,X) או "הוסף צומת (G,X)" מוסיפה את פריט המידע X לגרף G

4. <u>הסרת צומת - remvnode</u> הפעולה remvnode(G,p) או "הסר צומת (G,p)" מקבלת גרף G ומיקום p ומסירה אלמנט (פריט מידע-קודקוד בגרף) שבמיקום p בסדרת הקודקודים של הגרף.
יש לציין שפעולה זו לא מוגדרת על גרף ריק.

ייצוג של גרפים

בהינתן מפרט אפשר לבנות עבורו מימושים שונים.

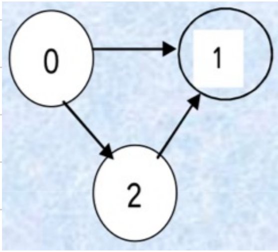
נבחין בין שני מקרים:

1. מספר הצמתים (קודקודים) בגרף ידוע מראש.

2. מספר הקודקודים בגרף לא ידוע מראש.

<u>השיטה הראשונה</u> לייצוג גרף משתמשת ב"מטריצת סמיכות" (שכנות/קישוריות).
מאחר ומספר הקודקודים בגרף ידוע מראש (נניח n) אזי כל קודקוד בגרף מיוצג ע"י מספר שלם בין 0 ל- n-1.
בשלב זה נניח שאין מידע המיוחס לקודקודים כמו כן נרצה לדעת רק על קיום קשתות הגרף ולא במידע המיוחס להן.
לאור האמור הדרך הטבעית להצגה של גרף הוא מערך דו מימדי מסדר nxn נסמן אותו ב- g.

עבור הגרף המכוון הנ"ל נייצג את הגרף בעזרת המטריצה הריבועית הבאה:



g	0	1	2
0	0	1	1
1	0	0	0
2	0	1	0

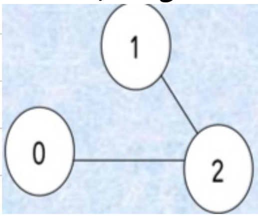
השורות הן הקודקודים מהן יוצאות הקשתות
העמודות הן הקודקודים בהם נכנסות הקשתות
במטריצה זו נרשום ערך 1 (ערך true) או 0 (ערך false)
g[i][j] שווה ל-1 כאשר הקודקוד j סמוך לקודקוד i (קיימת קשת).
g[i][j] שווה ל-0 כאשר הקודקוד j לא סמוך לקודקוד i (לא קיימת קשת).

<u>עבור גרף לא מכוון</u> g המערך הדו מימדי יוגדר כך:

$$g[i][j] = \begin{cases} 1 & \text{אם קיימת קשת לא מכוונת } (i,j) \\ 0 & i=j \text{ אם} \\ 0 & \text{אחרת} \end{cases}$$

<u>ניתן לשים לב-</u> שבעבור גרף לא מכוון g המטריצה הריבועית המייצגת אותו היא מטריצה סימטרית ביחס לאלכסון הראשי

<u>עבור הגרף הנ"ל:</u> <u>נייצג את הגרף בעזרת המטריצה הריבועית הבאה:</u>



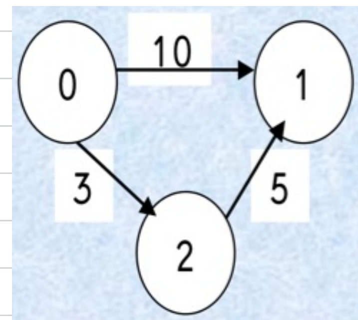
g	0	1	2
0	0	0	1
1	0	0	1
2	1	1	0

g[0][2]=g[2][0]=1 מאחר וקיימת קשת לא מכוונת בין הקודקודים 0 ו2.

<u>השיטה השנייה</u> לייצוג גרף משוקלל (משקל בעל קשת) משתמשת במטריצה הריבועית הבאה:
מאחר שמספר הקודקודים בגרף ידוע מראש (נניח n) אזי כל קודקוד בגרף מיוצג ע"י מספר שלם בין 0 ל- n-1
בשלב זה נניח שאין כל מידע המיוחס לקודקודים כמו כן נרצה לדעת על קיום קשתות הגרף וגם במידע המיוחס להן.
לאור זאת הדרך הטבעית להצגה של גרף היא מערך דו מימדי מסדר nxn נכנה אותה ב- מטריצת המשקולות ונסמנה ב-g
מערך דו ממדי מייצג כל זוג סדור אפשרי של קודקודים.
לגבי כל אלמנט במטריצה זו נחזיק מספר מאפיינים:

- האם קיימת קשת בין זוג סדור של קודקודים.
- במידה וקיימת קשת בין זוג סדור של קודקודים נרצה לדעת מהו המידע המיוחס לה.

<u>בתרשים הנ"ל:</u>

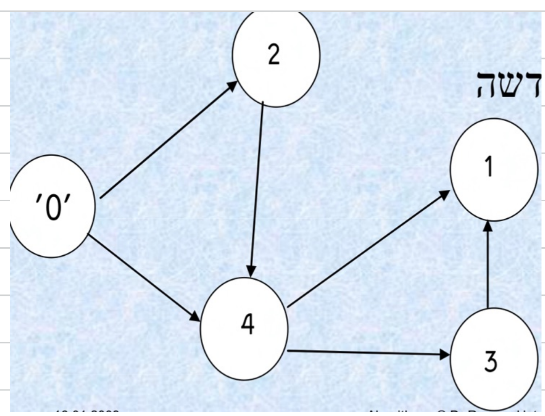


<u>המטריצה המיוחסת לה:</u>

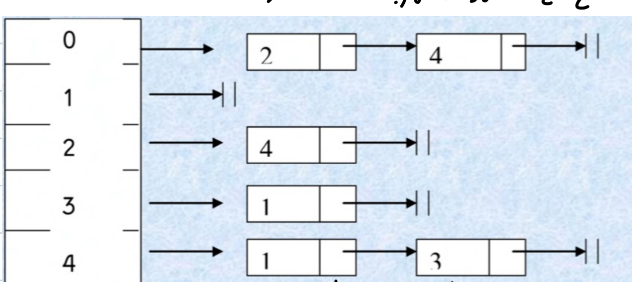
g	0	1	2
0	0	1 / 10	1 / 3
1	0	0	0
2	0	1 / 5	0

המערך הדו מימדי g מייצג כל זוג סדור אפשרי של צמתים
אם g[i][j].adj שווה ל-1 אזי ניתן לומר שקיימת קשת <i,j>
והמידע המיוחס לה נמצא ב- g[i][j].mishkal

<u>השיטה השלישית לייצוג גרף באמצעות מבנה מקושר- רשימת סמיכות</u>- בשיטה זו אני עדיין מניחים שמספר הקודקודים בגרף ידוע מראש (נניח n) וכל צומת בגרף מיוצג ע"י מספר שלם בין 0 ל- n-1. בשלב זה אנו עדיין לא מייחסים מידע לקודקודים אך כן מאפשרים לייחס מידע לקשתות עתה נציג את הגרף הנ"ל בעזרת מערך של רשימות כאשר גודל המערך הינו n



"ייצוג הגרף בעזרת מערך של רשימות



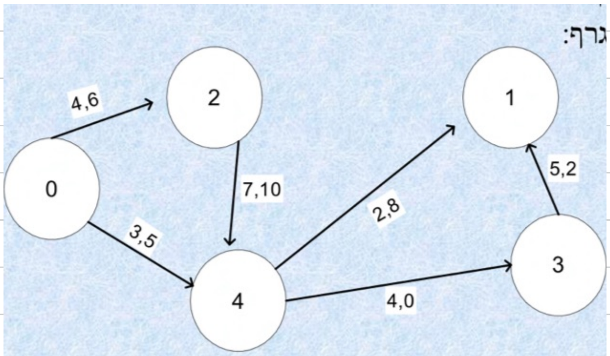
בין הקודקודים 1-3 לאו בהכרח קיימת קשת <4,3><4,1>

הקודקודים של הגרף מיוצגים ע"י מערך כל אינדקס של המערך מייצג צומת (קודקוד) בגרף
הקשתות של הגרף מיוצגות ע"י רשימה הנקראת רשימת סמיכות (שכנות/קשירות)
כל צומת ברשימת הסמיכות מייצג קשת בגרף.
כל כניסה במערך אשר מייצגת קודקוד כלשהו x של הגרף מצביע לרשימת הסמכויות של צמתים המייצגת את הקשתות היוצאות מקודקוד x-זה

ברשימת סמיכות של קודקוד x אין חשיבות לסדר הקשתות היוצאות מ-x.

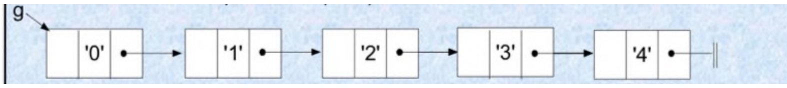
<u>השיטה הרביעית לייצוג גרף באמצעות מבנה רב מקושר</u>- בשיטה זו נניח שאיננו יודעים מראש את מספר הקודקודים בגרף וגם מספר הקשתות בגרף.
לאור זאת עלינו להשתמש במבנה מקושר והקצאת הצמתים ושחרורם תתבצע לפי הצורך. ועלינו לבנות מבנה רב מקושר
מאחר ואנו לא יודעים מראש את מספר הקודקודים בגרף נמיר את המערך אשר היה מייצג את קודקודי הגרף לרשימה מקושרת של קודקוד הגרף

<u>בתרשים הנ"ל:</u>

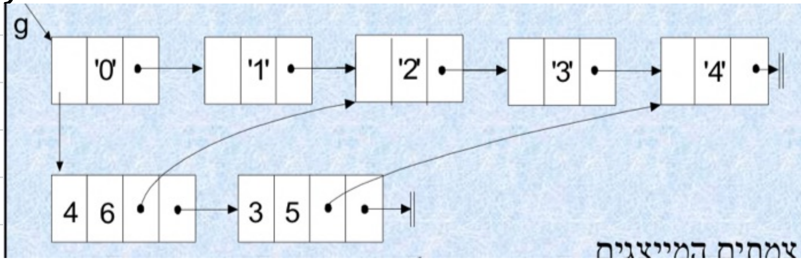


רק בחלופה זו לא מייחסים מידע לקודקודים אך כן מאפשרים לייחס משקלות לקשתות באיור רואים שלכל קשת מייחסים שני משקלות (בשלב זה לא חשוב מה משמעותן).

<u>הרשימה הבאה מייצגת את קודקודי הגרף:</u>



מאחר ומקודקוד 0 יוצאות 2 קשתות <0,4> <0,2> אז המבנה המקושר הינו:

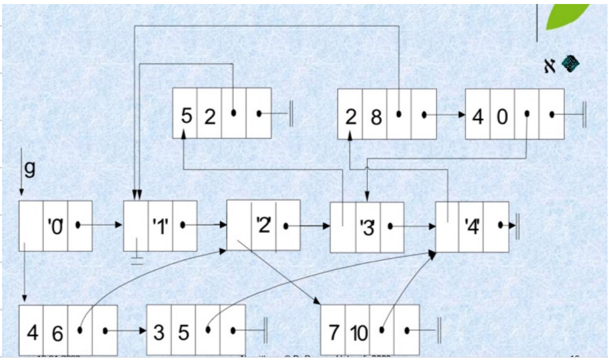


6 צומת המייצג קשת <x,y> מכיל את.

- המידע המיוחס לקשת
- מצביע לצומת המייצג צומת y בגרף בו נכנסת הקשת.
- מצביע לצומת המייצג את הקשת הבאה היוצאת מצומת x.

באיור האחרון מקודקוד 0 יוצאות 2 קשתות-אחת מהן נכנסת לקודקוד 2 והשניה לקודקוד 4

<u>לבסוף המבנה המקושר של הגרף יראה כך:</u>



בשיטה חלופית זו קבוצת צמתי הגרף מיוצגת ע"י רשימה ליניארית מקושרת וכל צומת ברשימה מקושרת זו מייצג צומת בגרף.
קבוצת הקשתות מיוצגת ע"י רשימת-שכנות (סמיכות/קשירות) וכל צומת ברשימת סמיכות מייצג קשת בגרף.

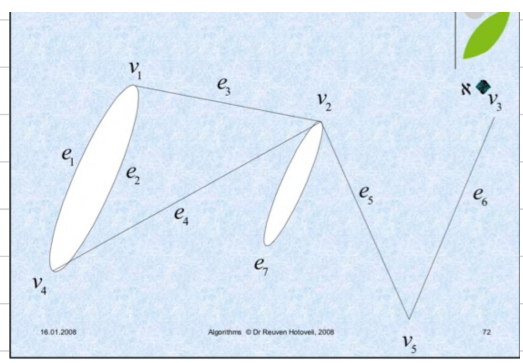
במבנה נתונים זה לכל צומת X המייצג קודקוד בגרף שלושה שדות:
- info - שדה מידע
- Verticelink - שדה מצביע לצומת המייצג קודקוד בגרף אם יש קודקוד כזה בגרף.
- Arcpoint - שדה מצביע לרשימת צמתים המייצגת את הקשתות היוצאות מקודקוד מסוים X בגרף

לכל <u>צומת המייצג קשת</u> כלשהי <x,y> בגרף שלושה שדות:
- Arcinfo - שדה המשקלות המיוחסת לקשת (שדה זה הינו רשות).
- Verticepoint - שדה המצביע לצומת המייצג קודקוד y בגרף.
- Arclink - מצביע לצומת המייצג את הקשת הבאה היוצאת מקודקוד X בגרף אם בכלל יש קשת כזו.

יש לשים לב ששדה המידע arcinfo של קשת לא בהכרח זהה למבנה של השדה המכיל info של צומת בגרף
מכאן ניתן להסיק שבמבנה רב מקושר יש לנו שני סוגי צמתים:
- סוג אחד של צומת מייצג קודקוד בגרף
- סוג שני של צומת שמייצג קשת בגרף

<u>השיטה החמישית</u> לייצוג גרף משתמשת במטריצת קשתות. בשיטה זו מספר הקודקודים בגרף ידוע מראש (נניח n) וכל הצמתים בגרף מיוצגים ע"י מספר שלם בין 0 ל- n-1.
נניח ששונן כל מידע המיוחס לקודקודים ולקשתות כלומר נרצה לדעת רק על קיום קשתות הגרף ולא במידע המיוחס להן. להצגת הגרף נחזיק מערך דו מימדי מסדר $n \times m$
כאשר $|E|=m$ $|V|=n$ מערך דו מימדי מייצג את כל הקשתות הנוגעות לכל צומת.

<u>נבהיר את השיטה בעזרת הגרף הבא:</u>



<u>נייצג בעזרת המטריצה הבאה:</u>

$$\begin{array}{c|ccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ \hline v_1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ v_3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ v_4 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ v_5 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}$$

הקודקוד v_1 נוגע לקשתות e_1 e_2 ו e_3
ולכן בשורה v_1 במטריצה מופיעים "1" לעמודות הקשתות e_1 e_2 e_3.

השורות במטריצה הן הקודקודים מהן יוצאות הקשתות
העמודות במטריצה הן קשתות הנוגעות לקודקוד בגרף

אם $g[i][j]=1$ אזי ניתן לומר שקשת שמספרה j נוגעת לקודקוד שמספרו i ואחרת הקשת j אינה נוגעת לקודקוד שמספרו i.
אם בגרף n קודקודים אזי קיימות בו n^2 קשתות.
למרות שאנו יודעים מראש את מספר הצמתים בגרף איננו יודעים מראש דבר על מספר הקשתות.

בניית גרף

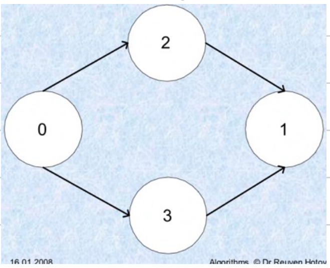
עד כה ראינו 5 שיטות חלופיות לייצוג גרף, השאלה המרכזית כיצד יראה הקלט לבניית גרף?

<u>שיטה ראשונה</u>- משתמשת במטריצת סמיכויות

- בשיטה זו מספר הצמתים בגרף ידוע מראש (נניח n).
- הקלט חייב לכלול אך ורק קשתות שבגרף כל קשת תיוצג בקלט ע"י זוג סדור או זוג של קודקודים.

<u>דוגמא:</u>

<u>ידוע מראש שבגרף 4 צמתים עבור הגרף הבא:</u>



<u>מבנה הקלט הינו:</u>

0	2
0	3
2	1
3	1

קודם מקצים מטריצה ריבועית של אפסים מסדר 4x4
עם קריאת הקלט בכל פעם שנקרא בקלט זוג מספרים <a,b> מוכנס הערך 1 באיבר המתאים במטריצה בשורה ה-a ובעמודה ה-b של המטריצה).
בתום קריאת הקלט המטריצה המייצגת את הגרף:

	0	1	2	3
0	0	0	1	1
1	0	0	0	0
2	0	1	0	0
3	0	1	0	0

<u>אלגוריתם לבניית הגרף:</u>

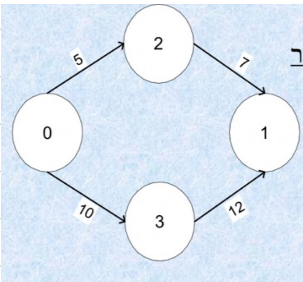
- צור מטריצה של אפסים, מסדר $n \times n$.
- כל עוד קיים קלט בצע:
  - קלוט זוג מספרים (a,b) המייצגים קשת מסויימת
  - בצע: $g[a][b]=1$
- סוף הלולאה.

<u>שיטה שנייה לייצוג גרף משוקלל</u>- שיטה זו משתמשת במטריצת המשקלות ונסמנה ב-g.

- בשיטה זו מספר הצמתים בגרף ידוע מראש (נניח n).
- הקלט חייב לכלול אך ורק קשתות ומשקלות המיוחסות אליהם בגרף.

<u>דוגמא:</u>

<u>נתון גרף שבו 4 קודקודים</u>



- מבנה הקלט הינו -
- <u>משקל</u> <u>צומת יעד</u> <u>צומת מקור</u>

צומת מקור	צומת יעד	משקל
0	2	5
0	3	10
2	1	7
3	1	12

קודם מקצים מטריצה ריבועית של אפסים מסדר 4x4 עם קריאת הקלט בכל פעם שנקרא בקלט שלושה מספרים (a,b,w) מוכנס ערך 1 והמשקל באיבר המתאים במטריצה (בשורה ה-a ובעמודה b של המטריצה)
בתום קריאת הקלט המטריצה המייצגת את הגרף הינה:

	0	1	2	3
0	0	0	1 5	1 10
1	0	0	0	0
2	0	1 7	0	0
3	0	1 12	0	0

אלגוריתם לבניית הגרף:

- צור מטריצה של רשומות, כך שלכל איבר במטריצה המשתנה g[i][j] . adj יהיה בעל ערך 0.
- כל עוד קיים קלט:
- קלוט שלישיה (מסודרת) (a , b, w).
- בצע השמה

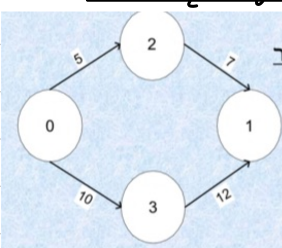
```
g [a][b] . mishkal = w;
g [a][b] . adj = 1 ;
```

שיטה שלישית - ייצוג גרף באמצעות מבנה מקושר:

- מספר הצמתים בגרף ידוע מראש (נניח n).
- הקלט חייב לכלול אך ורק קשתות ומשקלות המיוחסות להן בגרף.

דוגמא:

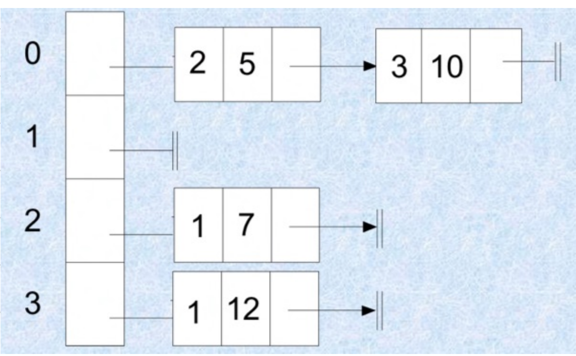
ידוע מראש שבגרף 4 צמתים עבור הגרף הבא:



מבנה הקלט הינו -

צומת מקור	צומת יעד	משקל
0	2	5
0	3	10
2	1	7
3	1	12

קודם מקדם מערך שגודלו n מאחר והצמתים של הגרף מיוצגים ע"י מערך.
עם קריאת הקלט בכל פעם שנקרא בקלט שלושה מספרים (a,b,w) נדרש צומת חדשה לצבור הקשת החדשה <a,b> לרשימת הסמיכות של צומת a
בתום קריאת הקלט המבנה המקושר יראה כך:



אלגוריתם לבניית המבנה המקושר:

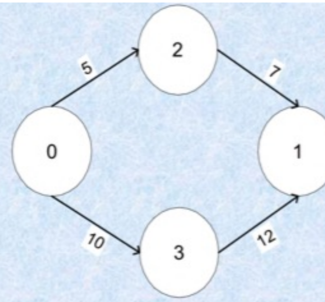
- צור מערך g (שגודלו n), כאשר כל תא במערך $g[i]$ יקבל את הערך NULL.
- כל עוד קיים קלט בצע:
- קלוט שלישיה (מסודרת) (a , b , w)
- בצע: joinw (a , b , w , g)
- סוף הלולאה

<u>שיטה רביעית</u> לייצוג גרף באמצעות מבנה רב מקושר- בשיטה זו איננו יודעים מראש את מספר הצמתים והקשתות בגרף.

• הקלט חייב לכלול אך ורק קשתות של הגרף וכאמור כל קשת תיוצג בקלט ע"י זוג סדור.

<u>דוגמא:</u>

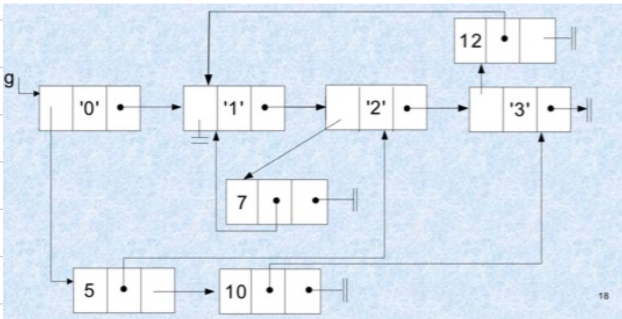
עבור הגרף הבא:



- מבנה הקלט הינו -
- <u>משקל</u> <u>צומת יעד</u> <u>צומת מקור</u>

משקל	צומת יעד	צומת מקור
5	2	0
10	3	0
7	1	2
12	1	3

בתום קריאת הקלט המבנה הרב מקושר יראה כך:



<u>אלגוריתם לבניית הגרף בייצוג הרב מקושר:</u>

- 1. /* צור גרף ריק */ empty-graph (&g) ;
- 2. כל עוד קיים קלט בצע:
- 2.1 קרא קלט (x , y , wt)
- 2.2 /* אתר צומת בגרף שמספרו x */
- p = findelement (g ,x)
- /* האם קיים צומת בגרף שמספרו x ? אם לא תוסיף אותו */
- if (p = = NULL) p = addnode (&g , x)
- 2.4 /* אתר צומת בגרף שמספרו y */
- q = findelement (g , y)
- 2.5 /* האם קיים צומת בגרף שמספרו y ? אם לא תוסיף אותו*/
- if (q = = NULL) q = addnode (&g , y)
- 2.6 /* צור קשת מ- p ל- q */
- joinw (p , q , w , g) ;
- סוף הלולאה.

פעולה חשובה ושכיחה בגרף היא סריקה של קודקודי הגרף.
בעת הסריקה נעבור על פני הגרף כך שנבקר בכל קודקוד פעם אחת בלבד ונבצע בו עיבוד בדרך כלשהי.
קיימות שתי שיטות סריקה שונות הנבדלות זו מזו בסדר שבו מתבצעת סריקת קודקודי הגרף
השיטות הן:

- סריקה לרוחב- Breadth First Search (BFS)
- סריקה לעומק- Depth First Search (DFS)

סריקה לרוחב BFS

<u>תיאור שיטת BFS:</u>

- נתון גרף G=(V,E)
- נתון קודקוד S
- בשיטה זו מתחילים את הסריקה מאחד הקודקודים בגרף הנקרא קודקוד מקור ונסמנו ב-S.

המטרה לטייל על פני הקשתות מקודקוד לקודקוד של גרף כך שנבקר בכל קודקוד של גרף ונבצע בו עיבוד כלשהו פעם אחת.
להשגת המטרה שיטת סריקה זו תתבצע כדלהלן:

- מחלקים את קודקודי הגרף לשכבות באופן הבא:

א) בשכבה הראשונה יהיו הקודקודים שניתן להגיע אליהם מקודקוד מקור S בעזרת מסלול שאורכו 1
ב) בשכבה השניה יהיו קודקודים שניתן להגיע אליהם מקודקוד מקור S בעזרת מסלול שאורכו 2.

באופן כללי- בשכבה ה-i לכל i יהיו קודקודים כך שניתן להגיע אליהם מהקודקוד מקור S בעזרת מסלול שאורכו i

בשלב הראשון של האלגוריתם הנדון מבקרים בכל הקודקודים השייכים לשכבה הראשונה
בשלב השני של האלגוריתם מבקרים בכל הקודקודים השייכים לשכבה השניה
כך נמשך תהליך הביקור בקודקודי הגרף עד שנבקר בכל קודקודי הגרף פעם אחת בלבד.
נדגיש כי האלגוריתם הנדון מבקר בכל קודקודי הגרף השייכים לשכבה ה-k לפני שמבקרים (מגלים) קודקוד כלשהו השייך לשכבה ה-k+1

האלגוריתם פועל על גרפים מכוונים ולא מכוונים כאחד!

<u>מבני נתונים:</u>
המטרה לבקר בכל קודקוד אחד ורק פעם אחת!
לכן נשתמש במערך בוליאני used כך ש- used[v] יציין אם ביקרנו בקודקוד v או לא
בתחילת האלגוריתם עדיין לא ביקרנו את קודקודי הגרף
לכן עבור כל קודקוד v בגרף נביא ל- used[v] את הערך false
בנוסף נשמור עבור כל קודקוד מידע על זהות הקודקוד הקודם לו בעת הסריקה, לכן נשתמש במערך p כך ש- p[v] יציין הקודקוד בגרף שממנו הגענו ל-V
בעת הסריקה.
בנוסף נשמור עבור כל קודקוד u מידע על המרחק מקודקוד S לקודקוד u
לכן נשתמש במערך dist כך ש- dist[u] יציין את המרחק מקודקוד המקור S לקודקוד u.

בנוסף נשתמש במבנה נתונים תור (Queue) אשר ינהל את השכבות.
כמו כן נשתמש בפעולות בסיסיות המוגדרות על תור:

- Insert(Q,w) - המוסיפה את האיבר w לתור Q .
- Delete(Q,v) - המסירה את האיבר שבחזית התור Q ושמה אותו ב - v .
- Is_empty(Q) - המחזירה ערך "אמת" (TRUE) אם התור Q ריק,אחרת היא מחזירה ערך "שקר" (FALSE).
- Head(Q) - מחזירה את האיבר שבחזית התור Q.

<u>אלגוריתם לשיטת סריקה לרוחב:</u>

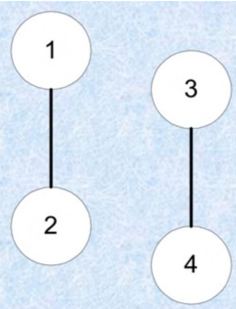
- <u>צעד 1.</u> לכל קודקוד $v \in V$, פרט לקודקוד מקור בצע:
- 1.1 $dist[v] \leftarrow \infty$ (בתחילת האלגוריתם אורך המסלול מקודקוד מקור S לקודקוד כלשהו v הינו ∞).
- 1.2 $P[v] \leftarrow NIL$ (בתחילת האלגוריתם לקודקוד v אין אבות).
- 1.3 $used\ [v] \leftarrow\ FALSE$
- <u>צעד 2.</u> 2.1 $dist[S] \leftarrow 0$ (אורך המסלול מהמקור S ל-S עצמו הוא 0).
- 2.2 $P[S] \leftarrow NIL$ (לקודקוד מקור S אין אב).
- 2.3 $used\,[S] \leftarrow TRUE$ (בתחילת אלגוריתם מבקרים רק בצומת המקור S).
- <u>צעד 3.</u> $Q \leftarrow \{S\}$ (אתחול התור).
- <u>צעד 4.</u> כל עוד התור Q לא ריק , בצע:
- 4.1 $v \leftarrow Head(Q)$
- 4.2 לכל קודקוד w בגרף,שהינו שכן של v ,בצע:
- 4.2.1 אם $used[w] = FALSE$?
- אז בצע:4.2.1.1 $used[w] \leftarrow TRUE$
- 4.2.1.2 $dist[w] = dist[v] + 1$
- 4.2.1.3 $P[w] \leftarrow v$
- 4.2.1.4 $Insert\ (Q, w)$
- 4.3 Delete(Q,v)

- <u>צעד 5.</u> סוף האלגוריתם.

<u>עץ פורש (BFS)</u> - תוך סריקת קודקודי הגרף $G=(V,E)$ נוצר תת גרף $G'=(V,E')$ כאשר E' זוהי קבוצת הקשתות שעברנו דרכן בכדי לסרוק את קודקודי הגרף בשיטת BFS.
G' הוא עץ המכיל את כל הקודקודים הנגישים מקודקוד מקור S
נראה בהמשך, שלכל צומת V שנגיש מ-S אורך המסלול הקצר ביותר מ-S ל-V נמצא בעץ זה

<u>טענה:</u>
תת גרף $G'=(V,E')$ מוגדר עץ T אשר יקרא עץ פורש של שיטת סריקה לרוחב (בקיצור T- עץ פורש BFS או עץ רוחב).
זו טענה טריוויאלית מאחר וקל לראות כי $|E'|=|V|-1$ וגרף G' הוא גרף קשיר ולכן הגרף G' הינו עץ.

סיבוכיות זמן ריצה של אלגוריתם סריקה לרוחב הינו $O(|V|+|E|)$



- ***הערה:*** נתבונן על הגרף הבא:
- ונניח שקודקוד מקור הנו S=1.
- קל לראות שמקודקוד מקור S לא ניתן להגיע לקודקודים שמספרם 3 ו 4 .

לכן ברור כי הערך של dist[3] הוא ∞, מכיוון שלא קיים מסלול מקודקוד 1 לקודקוד 3.

- לאור זאת נחליט שבתחילת האלגוריתם לכל קודקוד v, פרט למקור, נבצע את ההצבה הבאה:

$$dist\,[v] \leftarrow \infty$$

- מאחר שלא ידוע מראש אם קיים מסלול באורך כלשהו מקודקוד מקור S לקודקוד כלשהו v .
- במהלך האלגוריתם נרצה לשפר את אורך המסלול מקודקוד מקור – S לקודקוד v , לכל $v \in V$.

<u>מסלולים קצרים:</u>
<u>הגדרה:</u> נתון גרף $G=(V,E)$ אורך המסלול הקצר מקודקוד מקור לקודקוד כלשהו v בגרף מסומן כ- $L(S,v)$ ומוגדר כמספר הקשתות המינימלי שדרכן עובר המסלול בכדי להגיע מקודקוד מקור לקודקוד v.
הערה: אם אין מסלול מקודקוד מקור לקודקוד כלשהו v בגרף אזי $L(S,v)\leftarrow\infty$

<u>טענה:</u> נתון גרף $G=(V,E)$ (יכול להיות מכוון או לא מכוון). לכל קשת $(v,w)\in E$ מתקיים: $L(s,w)\leq L(s,v)+1$

<u>טענה:</u> נתון גרף $G=(V,E)$ ויהי S צומת מקור נריץ את האלגוריתם BFS(G) לכל קודקוד v שבגרף מתקיים: $dist[v]\geq L(s,v)$

__משפט:__ נתון גרף $G=(V,E)$ ויהי s צומת מקור נריץ את האלגוריתם BFS(G) לכל קודקוד v בגרף מתקיים $dist[v]=L(s,v)$
עבור כל קודקוד v שניתן להגיע אליו מ-s אחד המסלולים הקצרים ביותר מ-s ל-v הוא המסלול הקצר ביותר מ-s ל"הורה" של v $(P[v])$ ואחריו הקשת $(P[v],v)$.

__השגרה הרקורסיבית הבאה מדפיסה את הקודקודים לאורכו של המסלול הקצר ביותר מ-s ל-v:__

- ***הדפס מסלול (G,s,v)***
- אם $v==s$ אז הדפס את הקודקוד s
- אחרת אם $P[v]=nil$
- אז הדפס שלא קיים מסלול ועצור!
- אחרת בצע:
- א. **הדפס_מסלול**($G,s,P[v]$) (*קריאה רקורסיבית*)
- **ב.** הדפס את הקודקוד v .

סריקה לעומק (DFS)

נתון גרף $G=(V,E)$ בשיטה זו מתחילים את הסריקה מאחד הקודקודים בגרף ונטייל על פני הקשתות מקודקוד לקודקוד של גרף כך שנבקר בכל קודקוד של הגרף ונבצע בו עבוד כשהו פעם אחת בלבד.
להשגת המטרה שיטת הסריקה תתבצע כדלהלן:
שיטת סריקה זו מבצעת סריקה לעומק בגרף ככל שניתן.
בשיטת סריקה DFS לא מבקרים מקודקוד v בכל קודקודי הגרף שהינם שכנים של v ושעדיין לא נתגלו (לא נסרקו).
בשיטה זו בוחרים באופן שרירותי אחד הקודקודים w שהוא שכן של v ושעדיין לא סומן, כלומר עד כה עדין לא ביקרו בו בעת הסריקה.
תהליך הסריקה ממשיך מ-w לעומק.
בשלב מסוים של הסריקה כאשר מגיעים לקודקוד כלשהו v בגרף שכל שכניו מסומנים כלומר עד כה סרקנו את כל השכנים של v אנו נסוגים להורה של קודקוד v וממנו נמשיך את תהליך הסריקה.
תהליך הסריקה נמשך עד שנגלו (נסרקו) כל הקודקודים שהינם נשיגים מ-s שהוא קודקוד מקור.

__מבני נתונים:__
נשתמש במערך בוליאני used כך ש-used[v] יציין אם ביקרנו בקודקוד v או לא.
בתחילת האלגוריתם used[v] לכל קודקוד v בגרף נציב את הערך False מכיון שבתחילת האלגוריתם עדין לא ביקרנו בקודקודי הגרף.
לגבי כל קודקוד חייב האלגוריתם לשמור מידע על זהות הקודקוד הקודם לו בעת הסריקה
לכן, נשתמש במערך P כך ש-$P[v]$ ייצג קודקוד שממנו הגענו ל-v בעת הסריקה.
כמו כן נרצה לגבי כל קודקוד נרצה (אם כי לא הכרחי) למספר את קודקודי הגרף לפי סדר הביקורים (התיוג) בקודקודי הגרף
לקודקוד הראשון שנבקר - נצמיד "תג 1" לקודקוד השני - "תג 2" וכו'.
לכן נשתמש באלגוריתם במערך C כך ש $C[v]$ יציין את המספור שנקבע לקודקוד v בגרף בעת הסריקה.

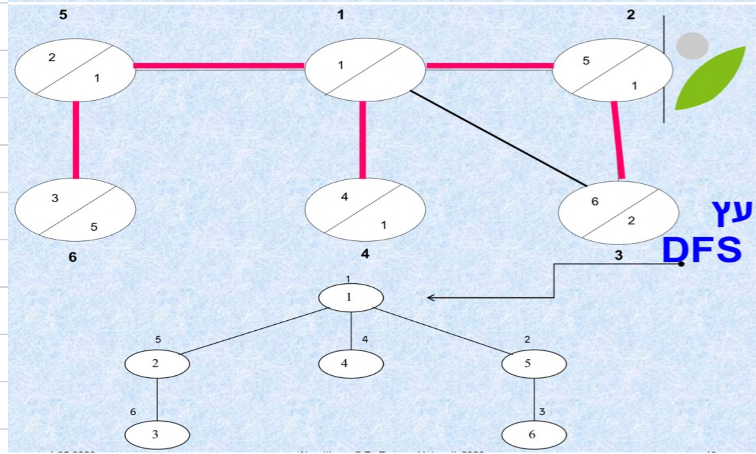
מהלך האלגוריתם:

- צעד 0: אפס את המשתנה i
- $s \longrightarrow v$ (v מכיל קדקוד מקור)
- צעד 1: לכל קדקוד a בגרף פרט לקדקוד המקור
  - $Used[a] \leftarrow FALSE$, (* בתחילת האלגוריתם עדיין לא ביקרנו בקדקודי הגרף *)
  - $Used[v] \leftarrow TRUE$, (* מאחר שהסריקה מתחילה מקדקוד מקור ובתחילת האלגוריתם מבקרים בו *).
- צעד 2: $i \leftarrow i+1$ ו- $C[v] \leftarrow i$ (* הקניית מספור*)
- צעד 3: אם לקדקוד v אין קדקודים שכנים שעדיין לא
- ביקרנו בהם אז לך לצעד 5.
- צעד 4: (* עתה יש לפחות קדקוד אחד u שהינו "שכן" של v ושעדיין לא ביקרנו בו. לכן נבצע את הצעדים הבאים:
- 4.1 $P[u] \longleftarrow v$ ו $v \leftarrow u$ ו $Used[u] \longleftarrow TRUE$
- 4.2 לך לצעד 2.
- צעד 5: (* אם v קדקוד מקור ואין אף קדקוד שכן שלו בגרף שעדיין לא ביקרנו בו אז צריך להפסיק את האלגוריתם *). כלומר, אם $C[v]=1$ אזי עצור!
- צעד 6: (* אם הקדקוד v אינו קדקוד מקור ואין אף שכן של הקדקוד v שעדיין לא ביקרנו בו, אז יש מקום לחזור ל"הורה" של v ומה"הורה" של v יש להמשיך את הסריקה *).

  $v \longleftarrow P[v]$ ולך לצעד 3. <u>סוף</u>

הערה:

- תת גרף $G'=(V,E')$ עבור גרף לא מכוון G מגדיר עץ T אשר יקרא עץ פורש של שיטת סריקה לעומק
- העץ הפורש של שיטת סריקה לעומק(בקיצור: עץ פורש DFS) של הגרף שראינו בהרחבה בדוגמה, הינו:



בסמוך: לכל צומת מופיע מספרו של קודקוד ובכל קודקוד רשומים מספרים המציינים את סדר התיוג של הקודקודים בשיטת DFS.

<u>הערות חשובות:</u>

1. נשים לב שבצעד 4 יכולנו לבחור כל קודקוד u מבין השכנים של קודקוד v שעדיין לא ביקרנו בו.
2. הבחירה היא שרירותית לחלוטין לכן סדר התיוג (או הסריקה) לא בהכרח יחיד.

<u>להלן אלגוריתם לשיטת סריקה DFS של קודקודי הגרף G כאשר הגרף G הוא לא בהכרח קשיר ויכול להיות גרף מכוון או לא מכוון:</u>

<u>DFS(G)</u>

1. $T=\emptyset$ (*קבוצה ריקה*) .2 $i \leftarrow 1$
3. לכל קדקוד v בצע: $Used[v] \leftarrow FALSE$
4. סוף הלולאה.
5. כל עוד קיים קדקוד $v\in V$, שעבורו $Used[v]=FALSE$ אזי בצע:

   קרא לשיגרה Search_DFS(v).
6. סוף הלולאה.

<u>להלן אלגוריתם המנסה למצוא עץ פורש DFS (נסמנו ב-T) תוך כדי סריקת קודקודי הגרף וקביעת סדר התיוג של הקודקודים.</u>

<u>Search_DFS(v).</u>

1. $Used[v] \leftarrow TRUE$
2. $C[v] \leftarrow i$ .3 $i \leftarrow i+1$
4. לכל קדקוד w שהינו שכן של קדקוד v
   בצע: 4.1 אם $Used(w)=FALSE$? אז בצע:
   - הוסף את הקשת (v,w) לקבוצת הקשתות המהוות עץ פורש T.
   - 4.1.2 $P[w] \leftarrow v$
   - 4.1.3 Search_DFS(w)
5. סוף הלולאה

סיבוכיות זמן הריצה של האלגוריתם DFS הינו: $O(|V|+|E|)$.

<u>הערה:</u> כאשר מפעילים DFS או BFS על גרף מקבלים אותו עץ פורש, פרט לסדר הביקור בקודקודי העץ.

<u>אלגוריתם נוסף לחיפוש לעומק:</u>

באלגוריתם זה הקודקודים נצבעים במטרה לציין את מצבם.

כל קודקוד נצבע באחד מבין הצבעים הבאים: לבן שחור ואפור.

בתחילת האלגוריתם כל קודקוד יהיה צבוע בצבע לבן.

כאשר קודקוד מתגלה, כלומר מגיעים אליו תוך כדי הסריקה הוא נצבע בצבע אפור.

כאשר אנו מסיימים את הטיפול בקודקוד (עוברים אותו ולא חוזרים אליו לטפל) אז הקודקוד יצבע בצבע שחור.

<u>נגדיר:</u>

- d[u] - מציין את מועד גילוי של קודקוד u בעת הסריקה.
- f[u] - מציין את מועד סיום הטיפול בקודקוד u בעת הסריקה

ברור כי: $d(u)<f(u)$.

כמו כן, ברור כי הקודקוד u לכל קודקוד $u\in V$:

יהיה בצבע לבן לפני הזמן d[u]

בצבע אפור בין הזמן d[u] ל-f[u]

בצבע שחור אחרי הזמן f[u]

<u>מהלך האלגוריתם:</u>

<u>DFS(G)</u>

1. עבור כל קודקוד $u\in V$ **בצע:**
   - 1.1 color[u] ⟵ white
   - 1.2 p[u] ⟵ NULL

/* בהתחלה כל הקודקודים צבועים בצבע לבן ולאף */

/* ולאף אחד מהם אין "הורה". */

2. time ⟵ 0

/* השעון מתחיל לתקתק */

3. עבור כל קודקוד $u\in V$ בצע :

   אם color[u] הוא צבע לבן

   אז בצע :DFS_Visit(u)

<u>DFS_Visit(u)</u>

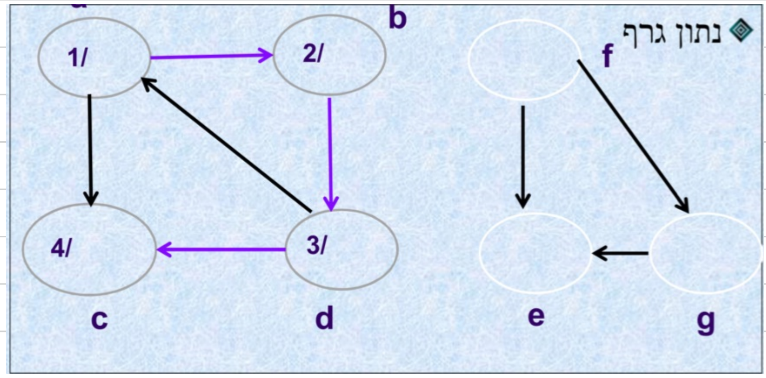
1. color[u] ⟵ אפור
2. time ⟵ time+1
3. d[u] ⟵ time
4. עבור כל קודקוד v שהינו קודקוד סמוך ל- u בצע :

   אם color[v] הוא צבע לבן

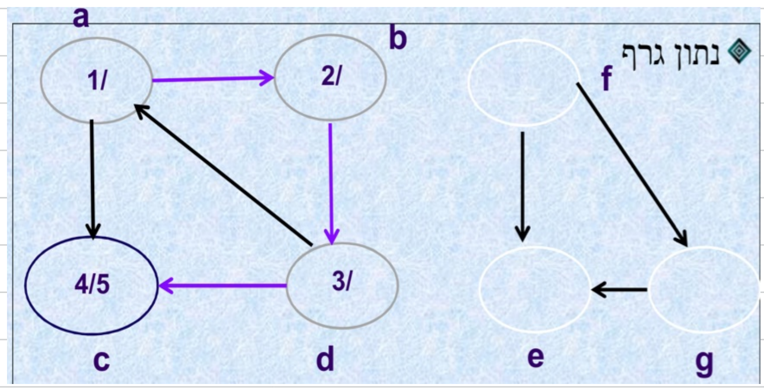
   אז בצע : 4.1 p[v] ⟵ u

   4.2 DFS_Visit(v)
5. color[u] ⟵ שחור
6. time ⟵ time+1
7. f[u] ⟵ time

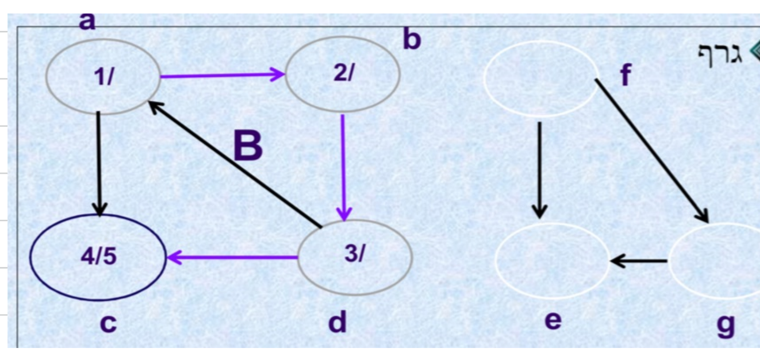
סורקים את הקודקודים a,b,c,d רושמים בהם את זמני הלידה
והם נהפכים לצבע אפור הקשתות שדרכן עברנו (הן בצבע סגול)
נקראות- קשתות עץ.



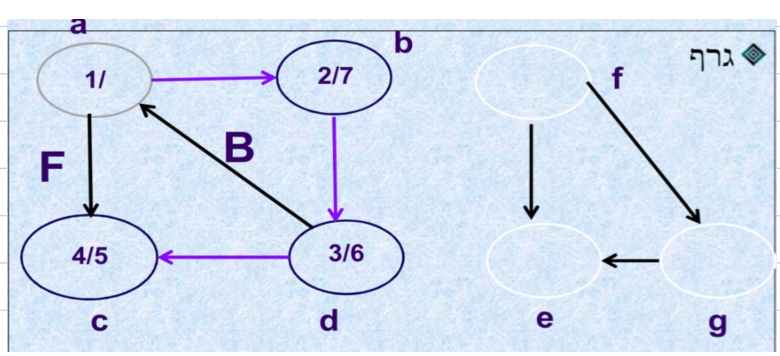
עתה אין אפשרות להתקדם מ-C לכן - C "מת" ביחידת זמן 5 והוא
נהפך לשחור ונסוג להורה d.



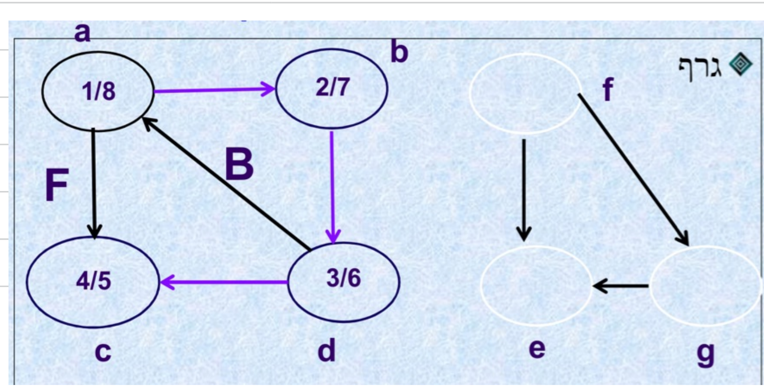
עתה בוחנים את הקשת מ-d ל-a אי אפשר לבקר ב-a כי ביקרנו בו
(הוא אפור) במקרה כזה הקשת היא מצומת אפור לאפור קשת כזו נקראת
"<u>קשת אחורית</u>" ותסומן ב-B <u>קשת כזו קיימת מצאצא לאב קדמון</u>.



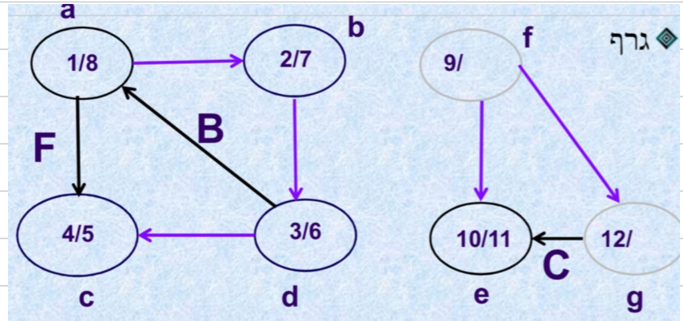
עתה בוחנים את הקשת מ-a ל-c אי אפשר לבקר ב-c כי טיפלנו בו
(הוא שחור) במקרה זה הקשת היא מצומת אפור לשחור קשת כזו נקראת
"<u>קשת קידמית</u>" מסומן ב-F <u>קשת כזו קיימת מאב קדמון אל צאצא</u>.



עתה אין אפשרות להתקדם מ-a לכן a "מת" ביחידת זמן 8 והוא נהפך
לצבע שחור ונסוג להורה Null כי הוא המקור.



עתה השגרה DFS_Visit הפסיקה לעבוד וחוזרים לשגרה DFS - אך עדיין יש קודקודים לבנים לכן תמשיך.



עתה בוחנים את הקשת מ-g ל-e אי אפשר לבקר ב-e
כי כבר טיפלנו בו (הוא שחור) במקרה זה הקשת היא מצומת אפור לשחור
אך קשת זו היא לא "מאב קדמון" אל "צאצא"
קשת כזו נקראת "קשת חוצה" ותסומן ב-C.

ברגע שאין קודקודים לבנים תהליך הסריקה יסתיים
בסוף הפעלת האלגוריתם כל קודקודי הגרף צבועים בצבע שחור.
ניתן לשכתב את האלגוריתם DFS כך שהוא יוכל לסווג את הקשתות שהוא נתקל בהן.

כל קשתות הגרף המכוון תסווגנה כך:
- חלקן קשתות אחוריות, נסמן ב-B
- חלקן קשתות קדמיות, נסמן ב-F
- חלקן קשתות חוצות נסמן ב-C
- כל היתר הינן קשתות שמהוות עץ פורש DFS.

טענה: עבור גרף בלתי מכוון G בתום האלגוריתם DFS כל קשת היא קשת עץ פורש (DFS) או קשת אחורה.

להלן אלגוריתם DFS שמסווג כך את הקשתות:

◆ ***להלן השגרה המשוכתבת DFS_Visit(u)***

◆

1. color[u] ⟵ אפור
2. time ⟵ time+1
3. d[u] ⟵ time
4. עבור כל קדקוד v שהינו קדקוד סמוך ל- u
בצע :
בדוק את הצבע של הקדקוד v :

4.1 אם color[v] הוא צבע לבן
אז בצע : 4.1.1 סמן את הקשת (u,v) כ"קשת עץ" (tree_edge).
4.1.2 p[v] ⟵ u
4.1.3 DFS_Visit(v)

4.2 אם color[v] הוא צבע אפור
אז בצע : סמן את הקשת (u,v) כ"קשת אחורית" (back_edge)

4.3 אם color[v] הוא צבע שחור אז בצע :
אם d[v] > d[u] אז בצע :
סמן את הקשת (u,v) כ"קשת קדימה"
אחרת בצע :סמן את הקשת (u,v) כ"קשת חוצה"

5. color[u] ⟵ שחור
6. time ⟵ time+1
7. f[u] ⟵ time

רכיבי קשירות ורק"חים

הגדרות:

1. גרף לא מכוון נקרא קשיר אם בין כל שני קודקודים קיים מסלול.

2. נתון גרף לא מכוון $G=(V,E)$ רכיב קשיר זוהי קבוצה מקסימלית של קודקודים בגרף כך שבין כל שני קודקודים בקבוצה קיים מסלול.
קודקוד בודד ללא שכנים יקרא גם כן רכיב קשיר.

3. גרף מכוון יקרא "גרף קשיר חזק" אם קיים מסלול מכוון מכל קודקוד לכל קודקוד אחר.

4. נתון גרף מכוון $G=(V,E)$ רכיב קשיר חזק (רק"ח) זוהי קבוצה מקסימלית של קודקודים בגרף G, כך שבין כל שני קודקודים בקבוצה זו קיים מסלול מכוון.

בגרף לא מכוון G, סריקת DFS(G) מוצאת את רכיבי הקשירות.

סריקת BFS(G,a) מוצאת את רכיב הקשירות ש-a נמצא בו.

אפשר לשכתב בקלות את האלגוריתם BFS(G) כך שנוכל למצוא את כל רכיבי הקשירות.

<u>אלגוריתם למציאת רכיבים קשירים בגרף לא מכוון בעזרת BFS:</u>
בתהליך של סריקת קודקודי הגרף בשיטת BFS נשתמש במערך בוליאני Used כך ש-Used[v] מציין האם ביקרנו בקודקוד v או לא. בנוסף נשתמש במשתנה count אשר מונה את מספר הרכיבים הקשירים.

<u>מהלך האלגוריתם:</u>

1. count ← 0
2. לכל קדקוד v בגרף בצע:
   2.1 אם ל- Used[v] ערך FALSE אז בצע:
      2.1.1 קרא לשגרה BFS[G] כאשר קדקוד מקור הינו v.
      2.1.2 count ← count + 1

<u>הערות:</u>

1. סיבוכיות זמן ריצה של האלגוריתם הינה $O(|V|+|E|)$
2. אם ברצוננו להדפיס את כל הקודקודים שנמצאים ברכיב קשיר של v נוכל לעשות זאת בעזרה רקורסיבית בעזרת פרישה של השיטת inorder או preorder בזמן $O(|V|)$.
3. ברצוננו לבדוק האם קיים מסלול בין 2 צמתים בגרף - אם לא קיים נרצה להדפיס הודעה שאין אחרת נדפיס את המסלול עצמו. לכן, ראשית נריץ את האלגוריתם BFS(G) ולאחר מכן נפעיל את השגרה הרקורסיבית אשר מדפיסה את הקודקודים לאורכו של מסלול קצר ביותר מ-s ל-v:

<u>הדפס_מסלול (G,s,v)</u>
- אם v==s אז הדפס את הקדקוד s
- אחרת אם P[v]=nil
- אז הדפס שלא קיים מסלול ועצור!
- אחרת בצע:
- א. **הדפס_מסלול**(G,s,P[v]) (* רקורסיה*)
- **ב.** הדפס את הקדקוד v .

<u>קבלת גרף הפוך G^T מ-G</u> - נתון גרף $G=(V,E)$ נגדיר גרף חדש אשר יקרא את הגרף הפוך ומוגדר כדלקמן:

$G^T=(V,E^T)$ כאשר $E^T=\{(u,v)|(v,u)\in E\}$

כלומר הופכים את כיווני הקשתות

קל לכתוב אלגוריתם בעל זמן הריצה $O(|E|+|V|)$ אשר קולט את הגרף G ויחזיר את הגרף G^T.

<u>אלגוריתם למציאת רכיבי קשירות חזקה בגרפים מכוונים Strong Connected Component (SCC):</u>

- <u>צעד 1.</u> מריצים את DFS(G) ויוצרים רשימת קדקודים (L)
- אשר ממוינת בסדר יורד לפי זמני סיום הטיפול בהם.
- <u>צעד 2.</u> הופכים את הגרף $G=(V,E)$ ומקבלים $G^T=(V,E^T)$

כאשר
$$E^T=\{(u,v)\,|(v,u)\in E\}$$

- כלומר הופכים את קשתות הגרף .
- <u>צעד 3 .</u> מריצים $DFS(G^T)$, כך שהלולאה המרכזית של
- DFS עוברת על קדקודי הגרף לפי הסדר , כפי
- שנקבע בצעד 1 ברשימה L .

סיבוכיות זמן הריצה של האלגוריתם הנ"ל הינו: $O(|E|+|V|)$

<u>גרף על</u> - הוא $G=(V,E)$ כאשר: $V=\{C_1,C_2,C_3,\ldots,C_m \mid m\geq 1$ ו- C_i רק"ח$\}$ $E=\{(C_i,C_j)\mid$ קיימים $x\in C_i$ ו- $y\in C_j$ וקיימת קשת בגרף G $(x,y)\}$

גרף על הוא תמיד גרף מכוון ללא מעגלים.

מיון טופולוגי

כל הבעיות חוץ מהפשוטות ביותר ניתנות לחלוקה לתתי בעיות שנקראות פעולות.

הגדרות מושגים:

1. גרף מכוון G שבו הקודקודים מייצגים משימות או פעולות והקשתות מכוונות מייצגות יחס קדימות בין המשימות, נקרא רשת AOV (Activity on Vertex)

2. קודקוד u ברשת AOV נקרא <u>קודקוד מקדים</u> של קודקוד v אם ורק אם קיים מסלול מכוון מ-u ל-v.

3. קודקוד v ברשת AOV נקרא קודקוד עוקב של הקודקוד u אם הקודקוד u הוא קודקוד מקדים של הקודקוד v.

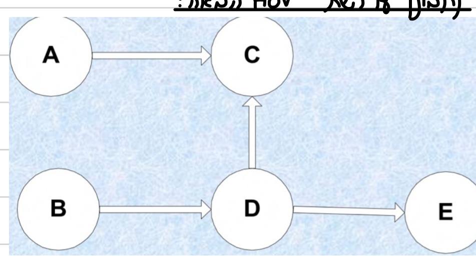
4. מיון טופולוגי (Topological Sort) - של גרף מכוון ללא מעגלים הוא סידור ליניארי של כל קודקודי הגרף כך: שאם בגרף יש קשת מכוונת (i,j) אזי i מופיע לפני j בסידור הנ"ל.

<u>הערה:</u> אם בגרף יש לפחות מעגל אחד אזי לא ניתן לסדר את קודקודי הגרף בשום סדר ליניארי שיכול לייצג את המיון הטופולוגי.

מיון טופולוגי מוגדר בגרפים מכוונים ללא מעגלים!

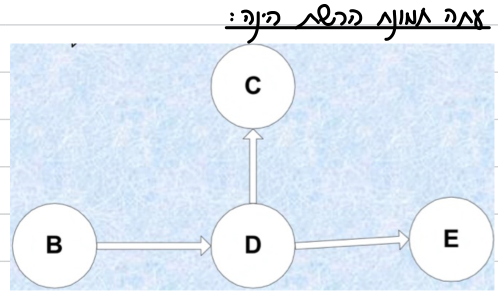
<u>להלן גרסאות שונות של אלגוריתמים הפשוטים - אשר יוצרים סידור ליניארי של קודקודים בגרף מכוון ללא מעגלים כאשר סידור ליניארי זה מייצג את המיון הטופולוגי:</u>

<u>נתבונן על רשת AOV הבאה:</u>



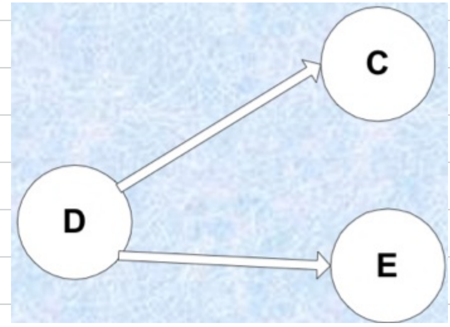
נראה מיון טופולוגי של גרף זה
לקודקודים A ו B אין קודקודים מקדימים לכן נבחר אחד מהם באופן מקרי ונכניס אותו לרשימה המייצגת מיון טופולוגי.
נניח שבחרנו בקודקוד A מבין הקודקודים A ו B שאין להם קודקודים מקדמים.
לאחר שהפעולות A בוצעה נסיר את קודקוד A מהגרף ואת כל הקשתות היוצאות מקודקוד זה בגרף.

<u>עתה תמונת הרשת הינה:</u>



בשלב זה לקודקוד B אין קודקוד מקדים (והוא היחיד) עתה נבצע את הפעולות על B.
נסיר קודקוד B מהגרף ואת כל הקשתות היוצאות מקודקוד זה בגרף.

תמונת הרשת הינה:



בשלב זה ל-D אין קודקוד מקדים (והוא היחיד)
עתה נבצע את הפעולת D.

תמונת הרשת הינה:

בשלב זה לשני הקודקודים C ו-E אין קודקודים מקדמים, לכן נבחר באופן מקרי אחד מהם ונצרף לרשימה המייצגת מיון טופולוגי.
נניח שבחרנו קודקוד C מבין קודקודים E ו-C

לאחר שפעולת C בוצעה תמונת הרשת הינה:

בשלב זה לקודקוד E אין קודקוד מקדים (והוא היחיד)
עתה נבצע את הפעולת E והרשת הינה גרף ריק.
בזה מסיים התהליך שקובע את המיון הטופולוגי.

סופית המיון הטופולוגי שהתקבל הוא: $\xrightarrow{A\ B\ D\ C\ E}$

יתכן שבגרף מסויים יהיה יותר מקודקוד אחד שאין להם קודקודים מקדימים
אך, מאחר שבכל שלב ניתן לבצע רק פעולת אחת בלבד אנו בוחרים באופן מקרי קודקוד אחד מבין הקודקודים
שאין להם קודקודים מקדמים ואותו מצרפים לרשימה המייצגת סדר טופולוגי.
הדבר גורם למספר סידורים טופולוגיים
מיון טופולוגי נוסף עבור הגרף הנתון הוא: BADEC ועוד אפשרויות נוספות.

להלן אלגוריתם למיון טופולוגי:

- נשתמש בשרותי טיפוס נתון **תור (queue)**, בו נשמור את הקדקודים שאין להם מקדימים.
- ***צעד 1***.- איתור קדקודי הגרף שאין להם מקדימים וצירופם לתור Q.
- 2.- כל עוד התור (Q) לא ריק יש לבצע את הצעדים הבאים:
  - 2.1 - הוצא איבר מראש התור וצרף אותו לרשימה המייצגת סדר טופולוגי.
  - 2.2- מחק בגרף את הקדקוד, שיצא זה עתה מהתור, ובנוסף למחוק את כל הקשתות היוצאות מקדקוד זה.
  - 2.3- בגרף החדש, שהתקבל בצעד 2.2 , כל קדקוד שאין לו קדקודים מקדימים, יש לצרף אותם לתור Q.
- 3.- בשלב זה התור ריק והאלגוריתם מסתיים.

כמה שאלות שעלות בעקבות האלגוריתם הנ"ל:

1. איך מבטאים את סדר הקדימות בין הפעולות השונות בקלט?

סדר הקדימויות מתבטא בקלט באמצעות זוגות סדורים כל שורה בקלט מכילה שמות של שני פעולות כאשר הראשונה צריכה להתבצע לפני השנייה.

2. כיצד נייצג את הגרף?

אם מספר הקודקודים בגרף ידוע מראש אז יש אפשרות לייצג את הגרף באמצעות מטריצת סמיכות או רשימות סמיכות
אחרת נשתמש בייצוג המקושר של הגרף.
נניח כי מספר הקודקודים בגרף ידוע מראש והגרף מיוצג באמצעות רשימת סמיכויות.

3. כיצד אפשר לקבוע לאיזה קודקוד אין קודקוד מקדים?

לשם כך נחזיק מערך בשם indegree כך ש indegree[j] מכיל את דרגת הכניסה של קודקוד j

- אם indegree[j]==0 פירוש הדבר שלקודקוד j אין קודקודים מקדימים וניתן לצרף אותו לתור Q מכיוון שבתור Q נמצאים קודקודים נמצאים קודקודים שאין להם מקדימים והם אמורים לצאת מהתור אחד אחרי השני ולהצטרף לרשימה המייצגת מיון טופולוגי.

- כאשר קודקוד j יוצא מהתור יש לעבור על רשימת הסמיכות של הקשתות היוצאות מקודקוד j ולהפחית ב-1 את ערכו של indegree[k] בעבור כל צומת k הסמוך ל-j

להלן האלגוריתם למיון טופולוגי בעזרת התור:

```
TP_Sort(Graph G)
{ Queue Q;
//Find indegree of each vertex
  for each v ∈ V do indegree[v] = 0;
  for each (u,v) ∈ E do
      indegree[v] ← indegree[v] + 1;
/*איתור כל הקדקודים שאין להם מקדימים*/
/*ולצרף אותם לתור Q*/
for each v ∈ V do
      if indegree[v] = = 0 then
          Insert(Q,v);
while Q ≠ Ø
{   v = Delete(Q);   print v;
    for each u ∈ Adj[v] do
        indegree[u] ←— indegree[u] - 1;
        if indegree[u] = 0 then Insert(Q,u);
}
for each v∈V
{   if indegree[v] ≠ 0 then
        print "CYCLE";
}
```

הפעולות הבסיסיות המוגדרות על תור:

- Insert (Q,v) – פעולה זו מוסיפה את האיבר v לעורף התור Q.
- Delete (Q) – פעולה זו מסירה ומחזירה את האיבר שבחזית התור Q.

סיבוכיות זמן הריצה של האלגוריתם למיון טופולוגי הינו: $O(|V|+|E|)=O(max(|V|,|E|))$

גרסא שנייה של האלגוריתם למיון טופולוגי תוך שימוש במחסנית:

הפעם מבנה הנתונים המוצע:

מערך של שני רשימות מקושרות כאשר לכל תא במערך יש 2 שדות והם:

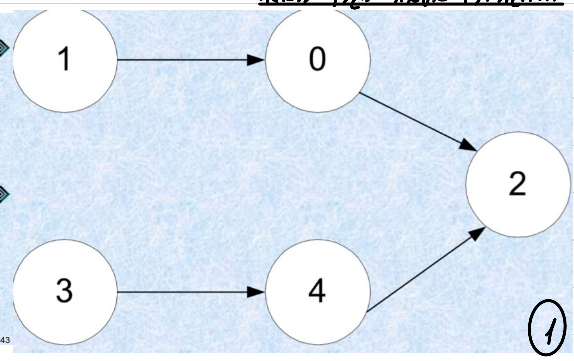
1. count אשר משמש ל- 2 מטרות:
   - כאשר לקודקוד יש קודקוד מקדים אז שדה זה מציין את מספר הקשתות הנכנסות לקודקוד זה.
   - אך כאשר לקודקוד אין קודקודים מקדימים אז שדה זה משמש כשדה לאיחסון המחסנית.
2. link - מצביע לרשימת הקודקודים הסמוכים

כמו כן, לכל צומת ברשימה 2 שדות והם:

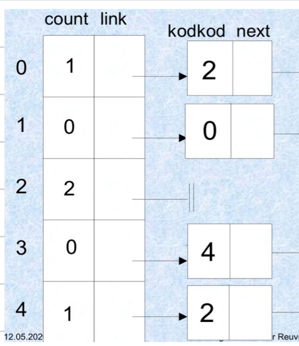
1. kodkod - מציין את מספר הקודקוד.
2. Next - שדה קישור ליצירת הרשימה המקושרת.

האלגוריתם בעבור הגרף הבא:

1



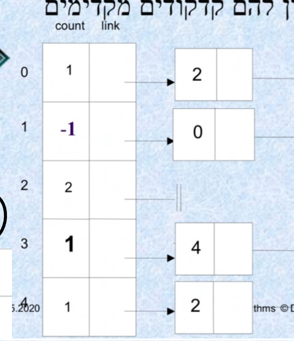
2



- מערך הרשימות נראה כך:
- בהתחלה המחסנית ריקה.

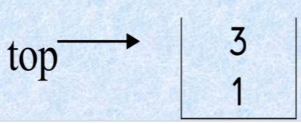
3

- שלב 1
- מאתרים את כל הקדקודים שאין להם קדקודים מקדימים (במקרה שלנו את הקדקודים 1 ו- 3) ומכניסים אותם למחסנית.
- תמונת המצב הינה: ←
- והמשתנה TOP מכיל ערך
- 3, מכיוון שהקדקוד 3
- נמצא בראש המחסנית.



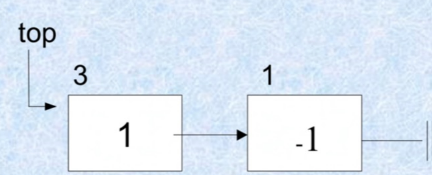
4

- כמו כן הערך של count[3] שווה ל- 1, מכיוון שהקדקוד הבא שנמצא במחסנית הוא קדקוד 1.
- תמונה זו מתארת את המחסנית הבאה:



5

- שים לב ! המחסנית מיוצגת, בעזרת המערך count, כרשימה מקושרת באופן הבא:



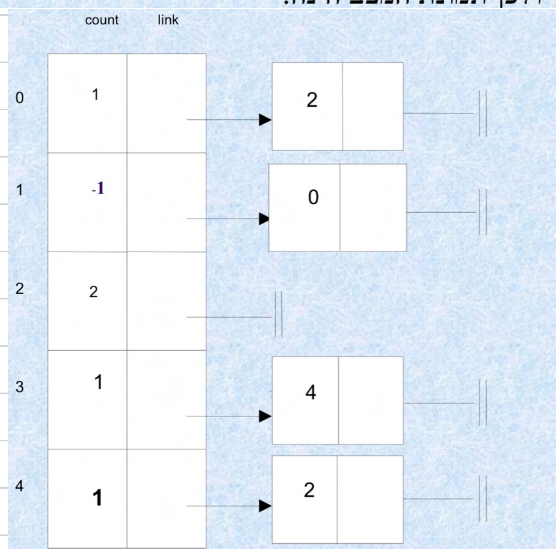
6

- שים לב ! לשינוי שחל ב count[3] .
- ערכו החדש הוא 1 אשר מציין את ה"כתובת" (האינדקס) של הצומת העוקב של count[3] ברשימה המייצגת מחסנית.
- כמו כן יש לשים לב שעבור המחסנית איננו מקצים שטח נוסף לאיחסונה.
- להלן קטע קוד לביצוע משימה זו:

7

```
TOP= -1;
  for (i=0 ; i <= n-1 ; i + +)
    if (count [i] = = 0) {count [i] = TOP;
                          TOP = i;
                         }
```

- <u>שלב 2</u>
- - מוציאים את הקדקוד 3 מהמחסנית.
- - ניגשים למערך של רשימות באינדקס 3 .
- - כך מגלים שהקדקוד הסמוך ל- 3 הוא קדקוד 4.
- עתה יש לבטל את הקשת (3,4) ונקטין את ערכו של _ count[4] ב- 1 .
- אז נקבל כי count[4] שווה ל- 0.
- לכן, הקדקוד 4 יכנס למחסנית מכיוון שאין לו קדקוד מקדים ולכן תמונת המצב הינה:



- והמשתנה TOP מכיל ערך 4, מכיוון שהקדקוד 4 נמצא במחסנית. תמונה זו מתארת את המחסנית הבאה:

- שים לב !
- ערכו של count[4] הוא 1 ,מיכוון שהעוקב של הקדקוד 4 הוא קדקוד 1 ברשימה המייצגת את המחסנית.
- תהליך זה חוזר חלילה.
- לסיכום להלן אלגוריתם נוסף לבעיית מיון טופולוגי אשר משתמש במחסנית כמפורט לעיל.

להלן האלגוריתם לביצוע מיון טופולוגי בעזרת מחסנית:

סיבוכיות זמן הריצה עבור מיון טופולוגי בעזרת מחסנית $O(|V|+|E|)$

```
TP_Sort2 (Graph G)
{ stack s;
 V do count [v] =0;    for each v
 E do count [v] ++;     for each (u,v)
/* Create Empty Stack (s);                */
 TOP = -1;
for (I=0 ; I<=n-1 ; I++)
     if (count[I] = = 0)
       {count[I] = TOP ;
         TOP = I;
       }
/* עד כה הכנסנו למחסנית את כל הקדקודים*/
/* שאין להם קדקודים מקדימים*/
 for (I=0 ; I<=n-1 ; I++)
 {if (TOP = = -1) {printf ("יש מעגל!!!") ;
                        return ;
                       }
 j=TOP;
 TOP = count[TOP]; /*הוצאת איבר מהמחסנית*/
 printf ( "%c",j);
 p=link[j];     //  מצביע לראש רשימת סמיכות      p
 while (p ! =0)  {k=p ⟶ kodkod;   count[k] --;
                       if (count[k] = = 0)
                           {count[k] = TOP;
                             TOP = k;
                           }
                        p=p ⟶ next;
                      }
 }
```

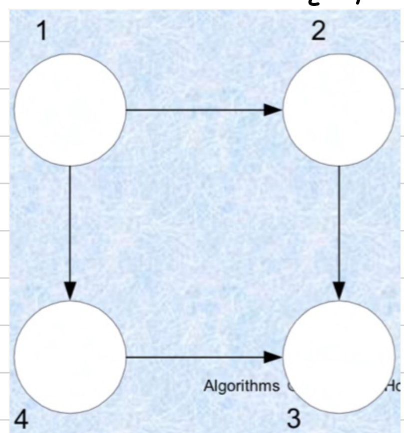
גרסא שישית של האלגוריתם למיון טופולוגי באמצעות סריקת גרף לעומק (DFS):

TP_Sort (Graph G)

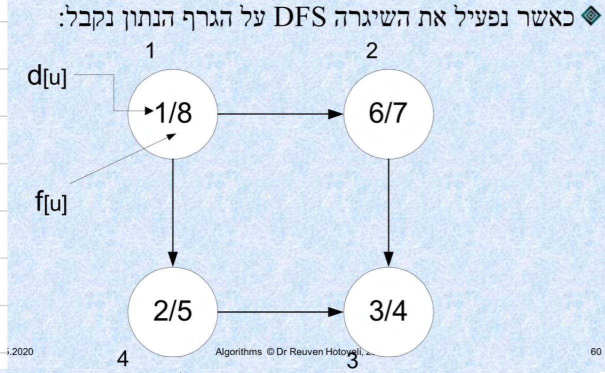
1. קרא לשגרה DFS(G) .
2. השגרה DFS מחשבת את ה- f[v] , המציין את מועד הסיום של הקדקוד v , עבור כל קדקוד v .
3. כאשר הטיפול בקדקוד v מסתיים אז נשים אותו בחזית הרשימה המייצגת את המיון הטופולוגי.
4. הדפס את הרשימה המייצגת את המיון הטופולוגי.

סיבוכיות זמן הריצה של האלגוריתם הנ"ל היא סיבוכיות זמן הריצה של השגרה DFS שהינה: $O(|V|+|E|)$.

להלן דוגמא:



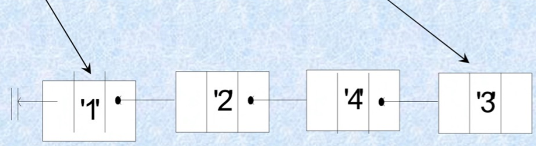
כאשר נפעיל את השיגרה DFS על הגרף הנתון נקבל:



והרשימה שמייצגת את המיון הטופולוגי הינה :

1 2 4 3

בסוף התהליך תמונת הרשימה המייצגת את המיון הטופולוגי הינה : ***מצביע לחזית front*** ***מצביע לעורף rear***



מסלולים קצרים ביותר ממקור יחיד
בגרף מכוון ללא מעגלים (גמ"ל)

Dag Shortest path (G,W,S)

גמ"ל (DAG) - גרף מכוון ללא מעגלים.
האלגוריתם מתחיל במיון טופולוגי של הגמ"ל, כדי לכפות על הקודקודים סדר ליניארי.
אם קיים מסלול מ-u ל-v אזי u יבוא לפני v בסדר הטופולוגי.

<u>מהלך האלגוריתם:</u>

- **<u>צעד 1</u>**: מיון טופולוגי של הקודקודים של G .
- **<u>צעד 2 :</u>** INITIALIZE_SINGLE_SOURCE(G,s)
- **<u>צעד 3</u>** : עבור כל קודקוד u בסדר הטופולוגי בצע:
- 3.1 עבור כל קודקוד v ששכן של u, כלומר $v \in adj[u]$ בצע:
- if $d[v] > d[u] + w(u,v)$ then
- $\{ d[v] \leftarrow d[u] + w(u,v)$ and $P[v] \leftarrow u \}$

<u>הערות:</u>

- מסלולים קצרים ביותר תמיד מוגדרים היטב בגמ"ל שכן גם אם קיימת בו קשתות בעלות משקל שלילי, לא יתכן שיהיו בו מעגלים בעלי משקל שלילי.
- זמן ריצה: כולל O(E+V) מתבצעת איטרציה אחת לכל קודקוד ועבור כל קודקוד כל אחת מן הקשתות היוצאות ממנו נבחנת בדיוק פעם אחת.

מסלול קריטי הינו מסלול ארוך ביותר בגמ"ל.
באמצעות האלגוריתם Dag shortest path נוכל למצוא מסלול קריטי באחת משני הדרכים:
1. היפוך הסימנים של משקלות הקשתות והרצת האלגוריתם Dag shortest path
2. להריץ את האלגוריתם Dag shortest path תוך החלפת ∞ ב- ∞- בצעד 2 והחלפת "<" ב- ">" במשפט if.

מסלולים קצרים לפי דייקסטרה

<u>הגדרה פורמלית של בעיית המסלול הקצר:</u>

נתון גרף משוקלל $G=(V,E)$

לכל קשת מוצמד מספר (אשר יכול לייצג מחיר, מרחק, עלות, ממוצע וכו').

נניח כי המספר שמיוחס לקשת מייצג מרחק בין שני ישובים, כל קודקוד ברשת מייצג ישוב וכל קשת ברשת מייצגת כביש בין שני ישובים.

המספר על הקשת מייצג אורך של הכביש.

הבעיה היא: מהו אורך המסלול הקצר ביותר מקודקוד מקור לקודקוד אחר ברשת, בהתחשב למרחקים שישנם בישובים השונים.

<u>תיאור פורמלי של הבעיה:</u>

נתון גרף $G=(V,E)$ קשיר עם קודקודים ממוספרים באופן אקראי מ-0 עד $n-1$ כלומר $n=|V|$.

עם פונ' משקל $W:E\to R$ אשר מייחסת לכל קשת מספר שנכנה אותו בשם מרחק ועם מרחקים E_{ij} לכל קשת (i,j) המחברת את שני קודקודי הגרף i ו-j.

<u>הגדרה:</u> המשקל של מסלול $p=(v_0,v_1,\ldots\ldots v_k)$ הינו סכום המשקולות המיוחסות לקשתות (v_{i-1},v_i) לכל $1<i<k$

כלומר: $w(p)=\sum_{i=1}^{k} w(v_{i-1},v_i)$

$w(v_{i-1},v_i)$ - מייצג המשקל של הקשת (v_{i-1},v_i)

$w(p)$ - מייצג את המשקל של מסלול p

נגדיר את אורך (משקל) המסלול הקצר ביותר (המינימלי) מקודקוד u לקודקוד v בגרף ע"י:

$$L(u,v)=\begin{cases}\min_{p} w(p) & \text{u לקודקוד v ברשת} \\ \infty & \text{אחרת}\end{cases}$$

<u>הגדרה:</u> מסלול קצר ביותר מקודקוד u אל קודקוד v מוגדר כמסלול p כלשהו שעבורו $w(p)=L(u,v)$

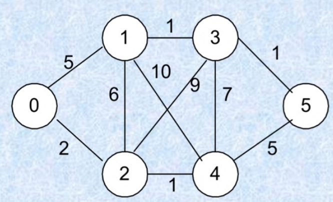
אלגוריתם דיקסטרה - Dijkstra

נתונה רשת $G=(V,E)$ עם פונ משקל $W:E\longrightarrow R^{+}$,לכל קשת מתאימים משקל חיובי:

- להלן מספר דרישות לצורך ביצוע האלגוריתם :
- א . נניח שהגרף מיוצג בעזרת מטריצת סמיכות כדלקמן:

$$a_{ij} = \begin{cases} E_{ij} & \text{(i , j) אם קיימת קשת} \\ 0 & \text{i=j אם} \\ \infty & \text{אחרת} \end{cases}$$

- בהנחה שקיימת קשת (i , j) , <u>אורך המסלול המינימלי הזמני</u> מקדקוד i לקדקוד j הינו המספר שמיוחס לקשת (i , j) .
- אורך המסלול המינימלי של המסלול המעגלי מקדקוד i לעצמו הינו 0 , כיוון שלא מאפשרים מעגלים שאורכם שלילי או אפס.
- קביעה זו די הגיונית כי מחפשים מסלולים בעלי אורך מינימלי שהינם מסלולים פשוטים וללא מעגלים.
- במידה ולא קיימת קשת (i , j) , לא ברור שבעתיד יהיה מסלול מקדקוד i לקדקוד j , לכן המרחק המינימלי הקצר ביותר מקדקוד i לקדקוד j הינו המרחק המינימלי הגרוע ביותר שהינו ∞ .
- לדוגמא עבור הרשת הבאה:



- מטריצת הסמיכות תוגדר כך:

	0	1	2	3	4	5
0	0	5	2	∞	∞	∞
1	5	0	6	1	10	∞
2	2	6	0	9	1	∞
3	∞	1	9	0	7	1
4	∞	10	1	7	0	5
5	∞	∞	∞	1	5	0

- ב.נניח שקדקוד מקור הינו קדקוד 0.
- ג. קבוצת הקדקודים תחולק לשתי קבוצות:
- אחת <u>הקבוצה</u> P(לכבוד "קבוע") אשר תכיל קדקודים, כך שאורך המסלול המינימלי מקדקוד מקור עד אליהם הינו <u>קבוע</u> ולא ישתנה בעתיד עד סוף האלגוריתם.
- והשניה <u>הקבוצה T</u> (לכבוד "זמניים" temporaries) אשר תכיל קדקודים כך שאורך המסלול המינימלי מקדקוד מקור עד אליהם הינו <u>זמני</u> ועשוי להשתנות בעתיד עד סוף האלגוריתם.
- מאחר שאין מסלולים מעגליים בעלי אורך אי חיובי, המסלול בעל אורך המינימלי מקדקוד מקור לעצמו הינו 0 וערך זה לא ישתנה עד סוף האלגוריתם.
- לכן, בהנחה שקדקוד 0 הינו קדקוד מקור, בתחילת האלגוריתם קדקוד 0 ישתייך לקבוצה P ויתר הקדקודים ישתייכו לקבוצה T .
- ד. עבור כל קדקוד u ברשת נרצה לשמור את אורך המסלול הקצר ביותר מקדקוד מקור 0 לקדקוד u ואותו נסמן על ידי d[u] לכבוד המילה (distance).
- בתחילת האלגוריתם לכל קדקוד u , פרט לקדקוד מקור, נבצע את ההשמה הבאה: $d[u] \leftarrow \infty$ ו- $d[0] \leftarrow 0$ מאחר שאורך המסלול הקצר ביותר מקדקוד מקור לעצמו הינו 0.
- ה. כמו כן נרצה לשמור לגבי כל קדקוד מידע על זהות הקדקוד הקודם לו ("הורה" שלו) במסלול הקצר.
- לאור זאת נשתמש במערך Pa , כך שלכל קדקוד u ברשת Pa[u] יציין קדקוד ממנו הגענו ל - u .
- <u>הערה</u>- הסימן Pa, נבע מהסיבה ש - Pa[u] מייצג "הורה" (parent) של קדקוד u בעת סריקה למציאת המסלול הקצר.

- בתחילת האלגוריתם לכל קדקוד u , פרט לקדקוד מקור, נבצע השמה '_'←Pa[u] (משמעותו undefined - עדיין לא מוגדר).לגבי קדקוד מקור 0 נבצע: Pa[0]←nil , המציין שלקדקוד מקור אין אב.
- לאור האמור לעיל בתחילת האלגוריתם ניתן לבצע סדרת ההוראות הבאות:
- - P←{0} ו T={ 1,2,……..,n-1 }
- - d[0] ← 0
- - לכל קדקוד j שאינו קדקוד מקור , כלומר לכל j=1…n-1 בצע:
- $d[j] \leftarrow a_{0j}$ מאחר ש a_{0j} מתאר את אורך המסלול המינימלי **הזמני** העובר דרך הקשת (0,j) .
- - P[0]←nil
- - לכל קדקוד שאינו קדקוד מקור, כלומר לכל j=1…n-1 אם קיימת קשת (0, j) אז בצע : Pa[j]←'0'
- אחרת בצע : Pa[j]←'-'
- בשלב הבא עלינו לשפר את אורכי המסלולים הקצרים d[j] מקדקוד מקור 0 לכל קדקוד אחר j , $1 \le j \le n-1$.
- הדרך לשיפור אורכי המסלולים מתבססת על תהליך איטרטיבי של איתור מסלול מקדקוד מקור לקדקוד j שבעזרתו ניתן לשפר את אורך המסלול המינימלי, עד שלא יהיה מקום לשיפורים נוספים.

<u>הערות:</u>

- באמצעות האלגוריתם ניתן לקבוע מהו המסלול עצמו.
- אלגוריתם של דייקסטרה פועל כהלכה בתנאי שכל המשקולות המיוחסות לקשתות הגרף הינן חיוביות.

סיכום דיקסטרה:

- צעד 0
- 0.1 $T=\{1,2,\ldots,n-1\}$ $P=\{0\}$
- 0.2 $d[0]\leftarrow 0$
- 0.3 לכל קדקוד , שאינו קדקוד מקור , כלומר לכל $j=1,\ldots,n-1$ בצע: $d[j]\leftarrow a[0,j]$
- 0.4 סוף הלולאה
- 0.5 $Pa[0]\leftarrow nil$
- 0.6 לכל קדקוד $j=1,\ldots,n-1$ בצע:
- אם קיימת קשת $(0,j)$
- אז בצע: $Pa[j]\leftarrow$'0'
- אחרת בצע: $Pa[j]\leftarrow$'-'
- 0.7 סוף לולאה.

- צעד 1
- 1.1 מצא קדקוד k מתוך קבוצת הקדקודים "הזמניים"- T בעל ערך d[k] מינימלי, כלומר: לכל $j\in T$ $d[k]=\min\{d[j]\}$
- 1.2 צרף את הקדקוד k לקבוצה p כלומר $p\leftarrow p+\{k\}$
- 1.3 להוריד את הקדקוד k מקבוצה T כלומר $T\leftarrow T-\{k\}$
- 1.4 אם $T=\emptyset$ (T הינה קבוצה ריקה) אזי סיים !
- אחרת עבור לצעד 2.

- צעד 2
- 2.1 לכל קדקוד $j\in T$ בצע :
- 2.1.1 אם $d[k]+a[k][j] < d[j]$
- אז בצע: $Pa[j]\leftarrow k$
- 2.1.2 $d[j]=\min\{d[j], d[k]+a[k][j]\}$
- 2.2 סוף הלולאה
- 2.3 חזור לצעד 1 .

סיבוכיות זמן הריצה של האלגוריתם הנ"ל הינו: $O(V^2+E)=O(V^2)$

למעשה ניתן להשיג זמן ריצה של $O(E\cdot\log|V|)$ כאשר הקבוצה T ממומשת בעזרת מבנה נתונים ערימה בינארית.

- לבנות ערמה דורשת $0(|V|)$.
- איתור וסילוק האיבר הקטן ביותר שבערמה דורשת
- זמן $0(\log|V|)$.
- צעד 2.1 מתבצע בזמן $0(\log|V|)$.
- <u>צעד 1</u>
- מתבצע $|V|$ פעמים ובכל צעד נדרש זמן $0(\log|V|)$. לכן הזמן הכולל של צעד 1 הוא $0(|V|\log|V|)$.
- <u>צעד 2</u>
- עדיין כל קדקוד מוכנס לקבוצה P בדיוק פעם אחת .
- לכן כל קשת ברשימת הסמיכות נבחנת בדיוק פעם אחת במהלך האלגוריתם .
- כאמור מס' הקשתות הכולל ברשימת הסמיכות הוא $0(|E|)$.
- לכן צעד 2 מתבצע $0(|E|)$ פעמים ובכל צעד נדרש זמן $0(\log|V|)$ עדכון הערך והכנסתו לערמה מחדש עם הערך החדש. לכן זמן הריצה של צעד 2 הוא $0(|E|\log|V|)$.
- סופית , זמן הריצה של האלגוריתם כולו הוא : $0(|E|\log|V|+|V|\log|V|)=0((|E|+|V|)\log|V|)$
- בגרף קשיר ,מאחר ש $|E|\geq|V|$, סיבוכיות זמן הריצה של האלגוריתם דיקסטרה היא: $0(|E|\log|V|)$.
- רואים כי כדאי לממש את T כערמה בעבור גרפים דלילים בלבד.
- הערה חשובה: קיים מבנה נתונים מתקדם הנקרה ערמת פיבונצ'י ובאמצעותו ניתן להשיג זמן ריצה O(VLogV+E)
- וזהו מימוש מהיר יותר אסימפטוטית.

<u>ניסוח אחר של האלגוריתם:</u>

- INITIALIZE-SINGLE-SOURCE(G,s)
- 1. for each vertex v∈V do
- 1.1 d[v]← ∞
- 1.2 $\pi[v]\leftarrow NIL$
- 2. d[s]← 0
- כאשר $\pi[v]$ הוא קדקוד "קודם" של v .
- טכניקת ההקלה (relaxation) :

RELAX(u,v,w)

- 1. if d[v] > d[u] + w(u,v) then
- 1.1 d[v] ← d[u] + w(u,v)
- 1.2 $\pi[v]\leftarrow u$

DIJKSTRA(G,w,s)

- 1. INITIALIZE-SINGLE-SOURCE(G,s)
- 1. S← Ø
- 2. Q← V {Q - הוא תור של קדקודים}
- 3. While Q ≠ Ø do
- 3.1 u← Extract_Min(Q)
- 3.2 $S\leftarrow S\cup\{u\}$
- 3.3 for each vertex $v\in Adj[u]$
- do RELAX(u,v,w)

משפטים:

1. נתון גרף $G=(V,E)$ עם פונ' משקל $W:E\rightarrow R^+$ יהי $P=\{v_0,v_1,v_2,...,v_k\}$ מסלול הקצר ביותר מקודקוד V_0 לקודקוד V_k
אם $P_{ij}=(v_i,v_{i+1},......v_j)$ הינו תת מסלול מקודקוד V_i לקודקוד V_j עבור $0\leq i\leq j\leq k$ אז P_{ij} הינו המסלול בעל אורך מינימלי מקודקוד V_i לקודקוד V_j.

מסקנות ממשפט 1:

1. נתון גרף $G=(V,E)$ עם פונ' משקל $W:E\rightarrow R^+$ נניח כי s קודקוד מקור
יהי P מסלול בעל אורך מינימלי מקודקוד s לקודקוד a בגרף.
מסלול P ניתן לפירוק באופן הבא: $s\rightsquigarrow v\longrightarrow a$
ומתקיים: $L(s,a)=L(s,v)+W(v,a)$

כלומר אורך המסלול המינימלי מקודקוד s לקודקוד a הינו אורך המסלול המינימלי מקודקוד s לקודקוד v בתוספת המשקל של הקשת (v,a) בתנאי שקיימת קשת מקודקוד v לקודקוד a.

2. לכל קשת $(u,a)\in E$ בגרף מתקיים: $L(s,a)\leq L(s,u)+W(u,a)$

3. תהי $(u,v)\in E$ קשת בגרף בתום צעד מספר 2 של האלגוריתם מתקיים: $d[v]\leq d[u]+W(u,v)$

2. נתון גרף $G=(V,E)$ עם פונ' משקל $W:E\rightarrow R^+$
יהי s קודקוד מקור באלגוריתם למציאת מסלולים בעלי משקל מינימלי, בתחילת האלגוריתם מתקיים: $d[v]\geq L(s,v)$ לכל קודקוד v בגרף
אם $d[v]=L(s,v)$ אז הערך של $d[v]$ לא ישתנה עד סוף האלגוריתם.

3. יהי מסלול בעל אורך מינימלי $s\rightsquigarrow u\longrightarrow v$ בגרף $G=(V,E)$ אם $d[u]=L(s,u)$ אז- $d[v]=L(s,v)$

4. (משפט מרכזי להוכחת נכונות האלגוריתם של דיקסטרה) לאחר הרצת האלגוריתם של דיקסטרה על הגרף $G=(V,E)$ עם פונ' משקל W (משקולות חיוביות) וקודקוד s שהינו קודקוד מקור, לכל קודקוד $u\in V$ מתקיים $d[u]=L(s,u)$

מסלולים קצרים לפי בלמן פורד
Bellman-ford

מציאת משקל המסלולים הקצרים ביותר מקודקוד מקור יחיד

נתון גרף $G=(V,E)$ מכוון עם פונ' משקל $w:E\rightarrow R$ כלומר לכל קשת מתאימים מספר ממשי
להלן מספר דרישות לצורך ביצוע אלגוריתם
הגרף G מיוצג בעזרת מטריצת סמיכות a_{ij} המוגדרת:
א.

$$a_{ij}=\begin{cases} E_{ij}\ \text{((i,j) משקל של הקשת)} & \text{אם קיימת קשת מכוונת } (i,j) \\ 0 & \text{אם } i=j \\ \infty & \text{אחרת} \end{cases}$$

ב. $d[v]$ הוא המשקל המסלול הקצר ביותר מקודקוד מקור 1 לקודקוד v

הערה: האלגוריתם של בלמן-פורד פועל כהלכה בתנאי שהגרף מכוון.

סיבוכיות זמן ריצה של האלגוריתם: $O(V\cdot E)$

- Bellman-Ford(G,w,s)
- 1. INITIALIZE-SINGLE-SOURCE(G,s)
- 2. for i ← 1 to |V|-1 do
-     for each edge $(u,v)\in E$
      do Relax(u,v,w)
- 3. for each edge $(u,v)\in E$ do
-     if d[v] > d[u] + w(u,v)
-         then return FALSE
- 4. return TRUE

משפט 3 בודק אם קיים מעגל בעל משקל שלילי ומחזיר את הערך הבוליאני המתאים.

ניסוח אחר של אלגוריתם בלמן-פורד

האלגוריתם מחזיר ערך בוליאני המציין אם קיים או לא קיים בגרף מעגל בעל משקל שלילי שניתן להגיע אליו מן המקור.
אם קיים מעגל כזה האלגוריתם מודיע שלא קיים פתרון לבעיה
אם לא קיים מעגל כזה האלגוריתם יוצר את המסלולים הקצרים ביותר ואת משקליהם.

- INITIALIZE-SINGLE-SOURCE(G,s)
- 1. for each vertex v∈V do
-     1.1 d[v] ← ∞
-     1.2 $\pi[v]\leftarrow NIL$
- 2. d[s] ← 0
- כאשר $\pi[v]$ הוא קדקוד "קודם" של v .
- טכניקת ההקלה (relaxation) :
- RELAX(u,v,w)
- 1. if d[v] > d[u] + w(u,v) then
-     1.1 d[v] ← d[u] + w(u,v)
-     1.2 $\pi[v]\leftarrow u$

סיבוכיות זמן ריצה של האלגוריתם: $O(V\cdot E)$

- כאשר נתון גרף מכוון ללא מעגלים ומותר משקל שלילי נשתמש ב-DAG $O(V+E)$
- אם יש משקלים שליליים, אבל יש מעגלים לא שליליים נשתמש ב-בלמן-פורד $O(E \cdot V)$
- אם יש משקלים חיוביים ולא שליליים - דייקסטרה $O(E+V^2)$
- אם יש טווח משקל שמגבה את הערכים לקבוצה סופית של ערכים שלמים וחיוביים נשתמש- דייקסטרה משופר $O(E+V)$

מספר הקשתות במעגל כלשהו הוא לכל היותר $|V|-1$.

עץ פורש מינימלי

<u>הצגת הבעיה:</u> נתון גרף בלתי מכוון קשיר $G=(V,E)$ כאשר V מבטא קבוצת קודקודים ו E קבוצת קשתות, פונ' משקל $W:E\longrightarrow R$ קובעת משקל לכל קשת בגרף G וכל קשת $(u,v)\in E$ היא בעלת משקל $w(u,v)$
הבעיה היא: נתון גרף $G=(V,E)$ עלינו למצוא תת גרף $T=(V,E_T)$ ללא מעגלים (עץ) של קשתות המחברות את כל קודקודי הגרף הנתון G ואשר משקלה הכולל $w(E_T)=\sum_{(u,v)\in E_T} w(u,v)$ הוא מינימלי.

מכיוון ש- E_T מחברת את כל קודקודי הגרף ולא מכילה מעגלים T יוצר עץ, עץ כזה נקרא עץ פורש.
מכיוון שאנו רוצים למצוא עץ פורש T כזה שבו סכום המשקלות המיוחסות לקשתות העץ T $w(E_T)$ יהיה מינימלי, מציאת עץ T כזה מכונה בשם בעיית העץ הפורש המינימלי

<u>עץ פורש (Spanning Tree)</u>- של גרף G קשיר הוא עץ אשר מכיל את כל קודקודי הגרף G ועץ זה הוא גם כן גרף קשיר.

<u>עץ פורש מינימלי (Minimum Spanning Tree)</u>- הינו עץ פורש וסכום המשקולות הרשומות בצד הקשתות של העץ הינו מינימלי.

בעיית העץ הפורש המינימלי שייכת לבעיות אופטימיזציה שיש להן פתרונות אפשריים רבים לכל פתרון יש ערך ואנו נרצה למצוא את הפתרון בעל ערך אופטימלי - המקסימלי או המינימלי.
פתרון אופטימלי אינו הפתרון האופטימלי מאחר ויתכן שיהיו מספר פתרונות שערכם אופטימלי

<u>הגדרת תכונות יסוד של עצים:</u>

<u>הגדרות:</u>
1. גרף G יקרא עץ אם הוא גרף קשיר וחסר מעגלים
2. גרף G יקרא יער אם הוא גרף לא קשיר וכל רכיב בו הוא עץ
3. נתון גרף בלתי מכוון וקשיר $G=(V,E)$ הקשת e בגרף G נקראת קשת מפרידה אם הסרתה בגרף G תיצור גרף שבו שני רכיבים

משפטים:

1. בכל עץ T בין שני קודקודים כלשהם u,v קיים מסלול אחד ויחיד.
2. נתון עץ פורש $T=(V,E_T)$ הגרף $G'=(V,E')$ שבו $E'=E_T-e$ כאשר $e=(a,b)$ קשת כלשהי כלומר $e=(a,b)\in E_T$ הוא יער המכיל בדיוק שני עצים.
3. נתון גרף $G=(V,E)$ עבור עץ פורש $T=(V,E_T)$ של הגרף G מתקיים $|E_T|=|V|-1$ כלומר מס' הקשתות בעץ T קטן ב-1 ממספר קודקודיו.
4. גרף G הוא עץ $T=(V,E_T)$ אם ורק אם G חסר מעגלים והוספת קשת $e=(a,b)$ בין שני קודקודים שלא מחוברים a,b ב-G יוצרת מעגל אחד ויחיד, כלומר בגרף $G'=(V,E_T\cup(a,b))$ כאשר $(a,b)\notin E_T$ קיים מעגל אחד ויחיד המכיל את הקשת (a,b).
5. נתון גרף בלתי מכוון קשיר $G=(V,E)$ הקשת כלשהי $e\in E$ היא קשת מפרידה אם ורק אם e אינה שייכת לקבוצת הקשתות כלשהי המהוות מעגל בגרף G

אלגוריתם של קרוסקל (kruskal) למציאת עץ פורש מינימלי

באלגוריתם זה בוחרים תחילה את הקשת בעלת משקל מינימלי, לאחר מכן את הקשת בעלת משקל מינימלי מבין הקשתות הנותרות וכו', בלבד שלא ניצור מעגל.

נשים לב כי עץ פורש מינימלי אינו בהכרח אחד ויחיד

צעד 1

1.1 $E_T=\emptyset$

(*בתחילת האלגוריתם קבוצת הקשתות בעץ פורש מינימלי הינה קבוצה ריקה*)

1.2 $k\leftarrow 0$

(*עד כה ביצענו אפס צעדים באלגוריתם *)

צעד 2

2.1 מצא קשת a השייכת ל-$(E-E_T)$ כך שלכל קשת b אשר שייכת ל-$(E-E_T)$ מתקיים:

$$w(a)=\min_b w(b)$$

(*כלומר מאתרים קשת, מקבוצת E בגרף G, שהינה בעלת משקל מינימלי ולא נבחרה עד כה*)

2.2 אם לא קיימת קשת כזו אזי אלגוריתם מסתיים!. אחרת

2.2.1 אם קבוצה $E_T\cup\{a\}$ יוצרת מעגל

2.2.1.1 אז בצע:

2.2.1.1.1 $E\leftarrow E-\{a\}$

2.2.1.1.2 חזור לצעד 2.1

2.2.1.2 אחרת בצע:

2.2.1.2.1 $E\leftarrow E-\{a\}$

2.2.1.2.2 $E_T\leftarrow E_T+\{a\}$

2.3 אם $k=|V|-1$ אזי הגרף $G=(V,E_T)$ הינו עץ פורש מינימלי ועוצר את האלגוריתם.

2.4 אם $k<|V|-1$ והאלגוריתם הסתיים אזי עצור את האלגוריתם ותודיע: "לא קיים עץ פורש ב-G".

צעד 3

3.1 $k\leftarrow k+1$

3.2 חזור לצעד 2

בכל שלב של אלגוריתם קרוסקל פרט לשלב האחרון, אנו מקבלים יער שהוא גרף לא קשיר וכל רכיב בו הוא עץ
כל רכיב הינו קבוצה של קודקודים והרכיבים הם זרים זה לזה.
לכן, יש צורך: לייצג את היער באמצעות מבנה נתונים המתחזק קבוצות זרות של איברים כל קבוצה תכיל את הקודקודים של עץ אחד ביער.
על מבנה זה נגדיר את מספר הפעולות הבסיסיות הבאות:

MAKE-SET(v) – יצירת רכיב קשירות חזקה כקבוצה בת איבר אחד – $\{v\}$.

FIND(v) – פעולה זו מחזירה מספר הקבוצה , אליה שייך הקודקוד v .

UNION(u,v) – פעולה זו מקבלת 2 רכיבים קשירים u ו v וגורמת לקריסת 2 רכיבים אלו לרכיב קשירות אחת .

SORT(E) – פעולה זו ממיינת את הקשתות לפי סדר לא יורד על סמך המשקולות שעליהן.

אלגוריתם מילולי למימוש אלגוריתם של קרוסקל:

צעד 1 בעץ פורש קבוצת הקשתות הינה קבוצה ריקה ($E_T \leftarrow \emptyset$)

צעד 2 צור |v| רכיבי קשירות (עצים), מאחר שכל קודקוד בגרף הנתון מגדיר עץ שהינו קבוצה בת איבר אחד.

צעד 3 מיין את קשתות העץ לפי סדר לא יורד על סמך המשקולות המיוחסות להן.

צעד 4 לכל קשת (u,v) , לפי הסדר שנקבע בצעד 3 , בצע :
4.1 אם קבוצה שאליה שייך קודקוד u שונה מהקבוצה אליה שייך קודקוד v אז בצע :
3.1.1 $E_T \leftarrow E_T \cup \{(u,v)\}$
3.1.2 שני רכיבי קשירות של u,v קורסים לרכיב קשירות אחת. (UNION(u,v)).

צעד 5 להחזיר עץ פורש שהינו E_T.

אלגוריתם של קרוסקל:

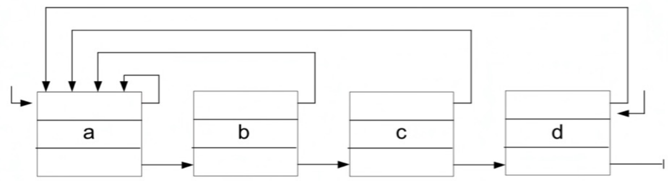
נתון גרף $G = (V,E)$.
המטרה לבנות עץ פורש מינימלי $T = (V, E_T)$.

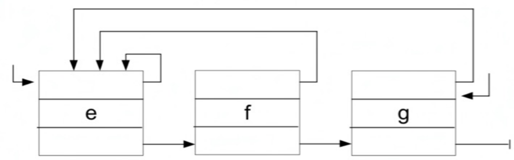
```
KRUSKAL(G)
//PRE-ALGORITHM

L ← SORT(E)
for each v ∈ V do MAKE-SET(v)
E_T ← Φ

//GROW TREE

for each (u,v) ∈ L do
    u' ← FIND(u)
    v' ← FIND(v)
    if u' ≠ v' then
        E_T ← E_T ∪ {(u,v)}
        UNION(u',v')
return G(V,E_T)
```

יעילות האלגוריתם של קרוסקל:

קל לראות כי זמן הריצה של האלגוריתם-קרוסקל אשר פועל על גרף בלתי מכוון קשיר $G=(V,E)$ תלוי במימוש מבנה נתונים יער שהינו קבוצות זרות.

א. __ייצוג קבוצות זרות על ידי רשימות מקושרות.__
כל קבוצה מיוצגת על ידי רשימה מקושרת.והאיבר הראשון בכל רשימה משמש כנציג הקבוצה ,המיוצגת באמצעות הרשימה.
התרשים הבא מתאר את הייצוג של שתי קבוצות זרות על ידי שתי רשימות מקושרות:

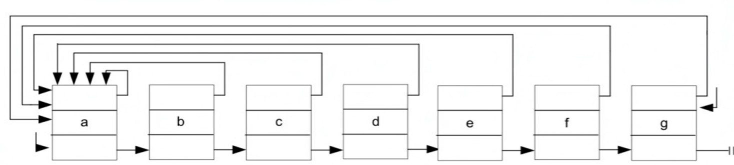




ברשימה מקושרת כזו כל צומת של הרשימה שייך לאיבר אחד של קבוצה. בכל צומת ברשימה 3 שדות עיקריים:
האחד – מצביע לצומת הבא המכיל איבר כלשהו של אותה קבוצה.
השני – מצביע לצומת הראשון ברשימה המכיל איבר שהוא הנציג של הקבוצה.
השלישי – מכיל את האיבר.

שים לב לכך:

1. הרשימה הראשונה מכילה את איברי הקבוצה $\{a,b,c,d\}$, והרשימה השניה מכילה את איברי הקבוצה $\{e,f,g\}$.
2. לכל רשימה יש מצביע לראשה וגם לסופה.
3. הרשימה המתקבלת מפעולת Union (b,g) היא :



בייצוג כזה של קבוצות זרות על ידי רשימות מקושרות נקבע את סיבוכיות זמן הריצה של הפעולות הבסיסיות המוגדרות על טיפוס נתון יער.

__עבור הפעולה__	סיבוכיות זמן הריצה __במקרה הגרוע ביותר__
MAKE_SET(v)	$O(1)$
FIND(v)	$O(1)$
UNION(x,y)	$O(y)\cong O(n^2)$בדוק!

לאור האמור לעיל עתה נוכל לחקור את סיבוכיות זמן הריצה של האלגוריתם קרוסקל.

__צעד__	__דורש זמן__
1.	$O(1)$
2.	$O(\lvert V\rvert)$
3.	$O(\lvert E\rvert\log\lvert E\rvert)$
4.	$O(\lvert E\rvert\cdot(1+\lvert V\rvert^2))=O(\lvert V\rvert^2\cdot\lvert E\rvert)$

לכן סיבוכיות זמן הריצה של האלגוריתם קרוסקל הוא $O(\lvert V\rvert^2\cdot\lvert E\rvert)$.

במדעי המחשב ברמה אקדמית ניתן לייצג קבוצות זרות באמצעות מבנה נתונים מיוחד הנקרא עצים מושרשים. כאן לא נקיים דיון אודות עצים מושרשים מכיוון שנושא זה חורג מדרישות הקורס . נספר רק על העובדה כי באמצעות מבנה נתונים " חכם " ניתן לבצע את צעד 4 של האלגוריתם קרוסקל בזמן $O(\lvert E\rvert\log\lvert E\rvert)$ ולכן סיבוכיות זמן הריצה של האלגוריתם הוא: $O(\lvert E\rvert\log\lvert E\rvert)$.

$\boxed{\text{2500 - 3 פעמים בשבוע לחצי שנה}}$

$\boxed{\text{4500 - לשנה}}$

האלגוריתם של קרוסקל הוא אלגוריתם חמדן מאחר שבכל צעד הוא מוסיף ליער קשת בעלת משקל קטן ככל האפשר בחירת הקשת בעלת המשקל הכי קטן ככל האפשר זוהי האפשרות הטובה ביותר באותו הרגע (בכל צעד וצעד) מכיוון שברצוננו למצוא עץ המחבר בין כל הקדקודים בגרף וסכום המשקולות הרשומות על הקשתות-מינימלי.
הגישה החמדנית הזו נותנת פתרון אופטימלי.

משפט: לאחר הרצת אלגוריתם של קרוסקל על הגרף $G=(V,E)$ מתקיים:

א. מספר הצעדים באלגוריתם של קרוסקל הוא $|V|-1$ לכל היותר

ב. אם האלגוריתם מסיים לאחר m איטרציות אז:

אם $m=|V|-1$ אז $T=(V,E_T)$ הינו העץ הפורש המינימלי של G

אחרת $m<|V|-1$ הגרף G לא מכיל עץ פורש.

ניתן לנסח אלגוריתם כללי למציאת עץ פורש מינימלי:

1. $(E_T \leftarrow \varnothing)$
2. בצע לולאה $|V|-1$ פעמים על:

2.1 מצא קשת (a,b) לפי האסטרטגיה המתאימה כך ש $E_T + \{(a,b)\}$ הינה תת קבוצה של קשתות של עץ פורש מינימלי.

2.2 $E_T \leftarrow E_T + \{(a,b)\}$.

3. החזר את E_T.

הגדרות:

1. חתך בגרף לא מכוון קשיר $G=(V,E)$ מחלק את קבוצת הקודקודים V לשני תתי קבוצות זרות X ו $\bar{X}$ כך ש: $X\cap\bar{X}=\varnothing$ ו $X\cup\bar{X}=V$

2. נתון גרף לא מכוון קשיר $G=(V,E)$ קשת כלשהי $e\in E$ בגרף נקראת קשת חוצה את החתך $(X,\bar{X})$ אם אחת מהקצוות שלה ב-X והאחרת ב-$\bar{X}$.

משפט: נתון גרף לא מכוון קשיר $G=(V,E)$ עם פונ' משקל $W:E\rightarrow R$ יהי $E_T\subseteq E$ כך ש E_T מוכלת בעץ פורש מינימלי כלשהו של G
יהי החתך $(X,\bar{X})$ כך שלא קיימת קשת ב-E_T החוצה את החתך.
אם קשת (a,b) כלשהי בגרף חוצה חתך ומשקלה מינימלי מבין משקלותיהן של כל הקשתות החוצות את החתך $(X,\bar{X})$ אזי $E_T+\{(a,b)\}$ גם כן מוכלת בעץ פורש מינימלי של G.

מסקנה:

נתון גרף $G=(V,E)$ בלתי מכוון וקשיר עם פונקצית משקל $\mathbf{W: E \rightarrow R}$. $E_T \subset E$ והיא מוכלת בעץ פורש מינימלי כלשהו של G. כמו כן T_1 הוא עץ (רכיב קשיר) ביער. אם הקשת (a,b) בעלת משקל מינימלי המחברת את עץ T_1 לרכיב (עץ) אחר ביער אזי $E_T + \{(a,b)\}$ גם כן מוכלת בעץ פורש מינימלי כלשהו של G.

הוכחה

לא קיימת ב E_T קשת החוצה את החתך $(\mathbf{T1, V - T1})$ ולכן הקשת (a,b) היא קשת הקלה עבור חתך זה, לכן לפי המשפט הקודם $\mathbf{E_T + \{(a, b)\}}$ גם כן מוכלת בעץ פורש מינימלי של G.

אלגוריתם של פרים - prim למציאת עץ פורש מינימלי

האלגוריתם של פרים הוא גם אלגוריתם חמדני מאחר ובכל צעד מוסיף קשת בעלת משקל קטן ככל האפשר ולכן אסטרטגיה כזו מבטיחה שבתום האלגוריתם E_T תכיל קשתות המהוות עץ פורש מינימלי.

כאמור מטרתנו למצוא עץ המכיל את כל קודקודי הגרף. לשם פיקוח על כך נשתמש בתור Q, כך ש: V - Q מייצגת קבוצת הקודקודים אשר טופלו ונמצאים בעץ פורש ואילו Q מייצגת קבוצת הקודקודים שעדיין לא טופלו ולא נימצאים בעץ פורש.

מאחר שבתחילת האלגוריתם אף קודקוד של גרף נתון לא טופל כל הקודקודים יהיו בתור ובתום האלגוריתם התור Q חייב להישאר ריק, כיוון שכל הקודקודים חייבים להיות בעץ פורש.

עבור כל קודקוד v נשמור בתור Q את K[v] אשר יכיל את המשקל המינימלי מבין משקלי הקשתות המחברות את הקודקוד v לקודקודים השייכים לעץ.

ברור כי בתחילת האלגוריתם $K[v] \leftarrow \infty$.

בנוסף, עבור כל קודקוד v נשמור מידע נוסף P[v], שהינו ה"הורה" של v בעת בניית עץ פורש. כלומר P[v] מציין את האבא של P[v].

הערה: הסימון P, נבע מהסיבה ש - P[v] מייצג "הורה" (Parent) של קודקוד v.

<u>אלגוריתם Prim (G , r)</u>

האלגוריתם מקבל כקלט את הגרף הבלתי מכוון וקשיר G ואת השורש r של העץ הפורש המינימלי שהאלגוריתם צריך לבנות ולהחזיר.

<u>צעד 1</u>.

לכל קודקוד v בצע :

1.1 $K[v] \leftarrow \infty$

1.2 הכנס את K[v] לתור Q.

סוף לולאה.

<u>צעד 2</u>.

2.1 $K[r] \leftarrow 0$

2.2 $P[r] \leftarrow nil$

<u>צעד 3</u>.

כל עוד התור Q לא ריק (כלומר לא כל הקודקודים נמצאים עדיין בעץ פורש T) בצע :

3.1 הוצא קודקוד u מהתור Q, כך שלכל קודקוד $v \in Q$ $K[u] = \min_{v}\{K[v]\}$ (כלומר מוציאים מהתור Q קודקוד u כאשר $K[v] > K[u]$ לכל קודקוד v בתור).

3.2 עבור כל קודקוד v שהינו שכן של קודקוד u ועדיין נמצא בתור Q בדוק:

אם $w(u,v) < K[v]$ (כלומר הקשת הנוכחית (u, v) בעלת משקל יותר קטן מבין המשקולות שעל הקשתות שבחרנו עד כה ונגעו ב - v)

אז בצע :

3.2.1 $P[v] \leftarrow u$ (* נקבע הורה חדש *)

3.2.2 $K[v] \leftarrow w(u,v)$ (* נקבע משקל חדש *)

סוף לולאה.

סוף לולאה.

יש לשים לב שמטפלים בקודקוד u_i וברשימת הסמיכות שלו רק פעם אחת לכן בצעד 3 בהיטרציה כלשהי i הזמן הנדרש הוא

$$O(\log|V|) + adj[u]\cdot[O(1) + O(\log|V|)]$$
$$\cong O(\log|V|) + adj[u]\cdot O(\log|V|)$$

מאחר שהצעד 3.2 מתבצע $|v|$ פעמים אז:
סך הכל מספר הצעדים בכל האיטרציות הוא :

$$\sum_{u\in V} adj[u]\cdot O(\log|V|) = O(\log|V|)\cdot \sum_{u\in V} adj[u]$$

מחד $\sum_{u\in V} adj[u]$ הוא הסכום של אורכי כל הרשימות הסמיכות בעבור כל קודקודי הגרף הנתון, מאידך ידוע כי בגרף לא מכוון מתקיים: $\sum_{u\in V} adj[u] = 2|E|$.

לכן צעד 3.2 דורש זמן $O(|E|\log|V|)$.

מאחר שצעד 3.1 מתבצע $|V|$ פעמים ובכל איטרציה הזמן הנדרש הוא $(\log|V|)$ אזי סך הכל זמן הריצה של הצעד 3.1 הוא $O(|V|\log|V|)$.

לכן סך הכל הזמן הנדרש לצעד 3 הוא:

$$O(|V|\log|V|) + O(|E|\log|V|)$$
$$\approx O(|E|\log|V|)$$

סופית סיבוכיות זמן הריצה של האלגוריתם הנתון הינה :

$$O(\max(|V|, |V|, |V|\log|V|, |E|\log|V|)) = O(|E|\log|V|)$$

זמן ריצה של אלגוריתם קרוסקל הינו: $\mathbf{O(|E|\log|E|)}$

ושל Prim: $O(|E|\log|V|)$

אך $|\mathbf{E}| \le \mathbf{n^2}$

ו $|V| \le n$

לכן $O(\log|E|) \approx O(\log|V|)$

לכן זמני הריצה של קרוסקל ושל פרים אסימפטוטית זהים.

הערה: אם נממש את התור Q בעזרת מבני נתונים מתקדם הנקרא "ערמות פיבונצ'י" שבאמצעותו ניתן לממש את הצעד 3.2.1.3 בזמן $O(1)$. לכן סיבוכיות זמן הריצה של אלגוריתם Prim הינו: $O(|V|\log|V| + |E|)$.

הערות חשובות:

1. משקלי הקשתות של הגרף אינם חייבים להיות חיוביים

2. ניתן למצוא עץ פורש מינימלי בעזרת אלגוריתם הבא:

א. חזור על התהליך הבא- מצא קשת בעלת משקל הגדול ביותר בגרף הנתון והסר אותה מהגרף בתנאי שהגרף נשאר קשיר.

ב. בתום שלב זה העץ הפורש המתקבל הינו עץ פורש מינימלי.

3. כל האלגוריתמים למציאת עץ פורש מינימלי ניתנים להמרה ע"י מציאת עץ פורש מקסימלי.

דוגמאות ל-2 גרסאות אשר מוצאים עץ פורש מקסימלי

גירסה I :

- חזור על התהליך הבא :
  מצא קשת בעלת המשקל הקטן ביותר בגרף הנתון והסר אותו מהגרף בתנאי שהגרף נשאר קשיר.
- עתה העץ הפורש שמתקבל הוא עץ פורש מקסימלי.

גירסה II :

עץ פורש מקסימלי ניתן לבנות גם כך: בהתחלה בוחרים את הקשת בעלת העלות המקסימלית, לאחר מכן את זו בעלת העלות הגדולה ביותר מבין הקשתות הנותרות וכך הלאה ובלבד שבכל שלב הקשת שנבחרה לא סוגרת (לא יוצרת) מעגל.

4. נתון גרף קשיר ולא מכוון $G=(V,E)$ עם פונ' המשקל $W:E\to R$ נתונה פונ' מונוטונית עולה ממש $f(x)$ נגדיר פונ' משקל חדשה כך: $w'(x)=f(w(x))$
כל עץ פורש מינימלי לפי w נשאר עץ פורש מינימלי לפי w'.

5. נתונים שני עצים פורשים מינימלים T_1, T_2 של גרף נתון $G=(V,E)$ עם פונ' משקל $W:E\to R$ אם נסדר לכל עץ את סדרת משקלותיו בסדר עולה אזי הסדרות זהות.

6. אפשר למצוא עץ פורש מינימלי עם עדיפות לקשתות (הדוגמא עם הקשתות הצהובות בשעור)