# סיכום אלגוריתמים

## 1. תכנון לינארי

**מודל :**

א. הגדרת משתני החלטה – משתנים שמייצגים כמויות של נושא מסוים($x1,...,x_n$).
ב. ניסוח פונקציית המטרה Z עם משתני ההחלטה(קשר מסוים ביניהם)
ג. ניסוח האילוצים – המגבלות של כל משתני ההחלטה.
ד. אילוצי אי שליליות – הגדרת התחום של משתני ההחלטה(גדולים, שווים או קטנים מ-0).

**מציאת פתרון אופטימלי בעזרת גרפים :**

א. שרטוט תחום הפתרונות האפשרי – שרטוט האילוצים בגרף+ כיוון.

ב. מציאת הפתרון האופטימלי:
מציאת הנקודה בעלת ערך אופטימלי תלויה ב:
1. מבנה תחום הפתרונות האפשרי :
- תחום חסום : **ייתכנו פתרון יחיד בקודקוד, פתרון מרובה על צלע.**
**דרך א -** נאסוף את כל נקודות החיתוך של הישרים עם עצמם והצירים(נקודות הקצה של התחום) לטבלה ונמצא את הערך שלהם בפונקציית המטרה, לבסוף נבחר את הערך המקסימלי או מינימלי.
***אם נקבל פתרון מקסימלי או מינימלי דומה בשני קודקודים אז נגיד שלבעייה יש פתרון מרובה.**
**דרך ב-** נשרטט את היטלי הגבהים של פונקציית המטרה .היטל גובה או קו גובה של פונקציה הוא אוסף כל הנקודות במישור שעבורן ערך פונקציית המטרה קבוע. דוגמא:

◆עבור הגובה $Z = 2$: נקבל $2 = 5X_1 + 3X_2 - 7$
◆כלומר: $X_2 = -\frac{5}{3}X_1 + 3$

כדי למצוא את הנקודה שבה פונקציית המטרה מקבלת את הערך המקסימלי, יש להתקדם רחוק ככל האפשר בכיוון עליית פונקציית המטרה.(הפוך כלפי מינימום).
***אם נקבל שהיטלי הגובה מקבילים לאחד מהאילוצים אז לבעייה יש פתרונות מרובים**

עבור משוואת הקווים הישרים שלהלן:
◆ $AX_1 + BX_2 = C$
◆ $DX_1 + EX_2 = F$
◆ התנאי המספיק וההכרחי להיותם מקבילים הוא: $\frac{A}{D} = \frac{B}{E}$
◆

- תחום לא חסום : **ייתכנו יחיד, מרובה ולא חסום.**
נשתמש בשיטת היטלי הגבהים -אם נקבל שנקודת הפתרון האופטימלי נמצאת בתחום אזי יש **פתרון יחיד.**
אם הנקודה נמצאת באינסוף אז לבעיה יש **פתרון לא חסום**(אינסוף).
אם היטלי הגובה מקבילים לאילוצים אזי יש **פתרון מרובה.**

- תחום ריק : **לבעיה אין פתרון.**
2. מבנה פונקציית המטרה -מקסימום או מינימום.

# 2. שיטת הסימפלקס

**מודל :**

| | MAX | MIN |
|---|---|---|
| אילוצים | גדולים שווים מ0 | קטנים שווים מ0 |
| משתנים | גדולים שווים ל0 | גדולים שווים ל0 |
| בכל שלב נכניס לבסיס את המשתנה שערכו בפונקציית המטרה : | השלילי ביותר | החיובי ביותר |
| איברים חופשיים(וקטור B) | חיוביים | |
| משתני חוסר | הוספת משתני חוסר לכל אילוץ | |

הבעיה הנתונה היא :

$$\text{Max}\{Z = 20X_1 + 60X_2 + 8X_3\}$$

תחת האילוצים:

$$\begin{aligned} 8x_1 + 2x_2 + 3x_3 &\le 160 \\ 4x_1 + 3x_2 &\le 100 \\ 2x_1 \qquad x_3 &\le 50 \\ x_3 &\le 20 \\ x_1, x_2, x_3 &\ge 0 \end{aligned}$$

**השיטה :**

1. הוספת משתני חוסר לכל אילוץ, משתני אלו יהיו אי שליליים.

א

$$\text{Max}\{Z = 20x_1 + 6x_2 + 8x_3\}$$

תחת האילוצים:

$$\begin{aligned} 8x_1 + 2x_2 + 3x_3 + x_4 \qquad\qquad &= 160 \\ 4x_1 + 3x_2 \qquad x_5 \qquad\qquad &= 100 \\ 2x_1 \qquad x_3 \qquad + x_6 \qquad &= 50 \\ x_3 \qquad\qquad + x_7 &= 20 \\ x_i \ge 0 \qquad i = 1, \ldots\ldots \ldots, 7 & \end{aligned}$$

2. קביעת משתנה נכנס לבסיס – תחילה, המשתנים הבסיסיים יהיו משתני החוסר, נעתיק את כל האילוצים ואת פונקציית המטרה לטבלה(נהפוך את הערכים לשליליים). נכניס כל איטרציה את המשתנה בעל הערך שלילי\החיובי ביותר(בפונקציית המטרה) לבסיס.

ציר ההתמרה

| מחירים מקוריים | 20 | 6 | 8 | 0 | 0 | 0 | 0 | פתרון נוכחי | |
|---|---|---|---|---|---|---|---|---|---|
| המשתנים בבסיס | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | b | $\frac{b_i}{a_{ik}}$ |
| $x_4$ | 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 | $\frac{20}{0} = \infty$ |
| $C'_j = Z_j - C_j$ | -20 | -6 | -8 | 0 | 0 | 0 | 0 | Z=0 | |

משתנה נכנס

16.01.2016 Algorithms © Dr Reuven Hotoveli, 2008 42

3. קביעת המשתנה היוצא- בכל איטרציה נוציא את המשתנה שה-$b_i/a_k$ שלו הכי קטן.
בדוגמא שלנו, נוציא את X4 ונכניס את X1 במקום
4. הכנסת המשתנה הבסיסי החדש וחישוב הטבלה מחדש- (1) מאפסים את הערכים בעמודה של המשתנה הבסיסי החדש חוץ מהערך הפיבוט(הערך המתנגש בעמודה של המשתנה הכנס והשורה של המשתנה היוצא). (2) מחלקים את כל השורה בפיבוט.
(3) כל שאר הערכים מחשבים לפי:**( (פיבוט/כפל של אלכסון משני) - הערך המוחלף ).**

| מחירים מקוריים | 20 | 6 | 8 | 0 | 0 | 0 | 0 | פתרון נוכחי | |
|---|---|---|---|---|---|---|---|---|---|
| המשתנים בבסיס | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $a_6$ | $a_7$ | $b$ | $\frac{b_i}{a_{i2}}$ |
| $x_1$ | 1 | 1/4 | 3/8 | 1/8 | 0 | 0 | 0 | 20 | $\frac{20}{1/4}=80$ |
| $x_5$ | 0 | 2 | -3/2 | -1/2 | 1 | 0 | 0 | 20 | $\frac{20}{2}=10$ |
| $x_6$ | 0 | -1/2 | 1/4 | -1/4 | 0 | 1 | 0 | 10 | - |
| $x_7$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 20 | $\frac{20}{0}=\infty$ |
| $C'_j=Z_j-C_j$ | 0 | -1 | -4/8 | 20/8 | 0 | 0 | 0 | Z=400 | |

משתנה יוצא $x_5$

משתנה נכנס $x_5$

16.01.2016 ...orithms © Dr Reuven Hotoveli, 2008 46

5. **תנאי עצירה -** MAX – כל הערכים בפונקציית המטרה חיוביים .
MIN- כל הערכים בפונקציית המטרה שליליים.
6. **הפתרון האופטימלי –** המשתנים שמופיעים בבסיס והערכים שלהם לפי הטבלה, כאשר קיימנו את תנאי העצירה נבחן :
   א. אם בטבלה האחרונה, שבה מצאנו את הפתרון האופטימלי, אם מקדם של משתנה לא בסיסי מסוים בפונקציית המטרה שווה אפס,אז לבעיה יש **פתרון מרובה.**
   ב. כאשר יש משתנה נכנס אבל אף אחד מהאילוצים לא חסום מלמעלה(שואפים לאינסוף עקב חלוקה במספר שלילי) אז יש **פתרון לא חסום.**
   ג. כאשר א' וב' לא מתקיים אז יש **פתרון יחיד** לפי המשתנים הבסיסיים.

**<u>מקרים מיוחדים :</u>**

1. אם יש משתנה Xi שהוא קטן שווה מ0 נציב : Xi=-X'i.
2. אם יש משתנה שלא מוגבל בסימן נציב במקומו : Xi=X'i-X''i.

## 3. <u>שיטת ה-M הגדול</u>

מטפל במקרים ישנם bים שליליים כתוצאה מכפל ב1- של אילוץ מסוים או כל דבר אחר.

**<u>הכנות :</u>**

1. הפוך כל מרכיב b לאי-שליליים, על ידי הכפלת אי-שוויוניים בהם ה-b ים שליליים ב1-.
2. אם האילוץ גדול שווה – נחסיר משתנה עודף.
אם האילוץ קטן שווה – נוסיף משתנה חוסר.
3. הוספת משתנים מלאכותיים-($y_1,\dots,y_n$) לכל אילוץ רק כאשר האילוץ הוא או קטן או גדול שווה.

מערכת האילוצים שתתקבל היא:

$$\begin{array}{l} 2x_1 + x_2 + 3x_3 - x_4 + y_1 = 2 \\ 5x_1 + x_2 + 2x_3 - x_5 + y_2 = 3 \\ 6x_1 + 3x_2 - x_3 - x_6 + y_3 = 4 \end{array}$$

$$x_j \ge 0 \quad j=1,2,\dots,6 \qquad y_j \ge 0 \quad j=1,2,3$$

4. הוספת המשתנים המלאכותיים לפונקציית המטרה- בעיית מקסימום : נוסיף Yi*(-M)
בעיית מינימום : נוסיף Yi*M

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

## **השיטה :**

1. בניית הטבלה – מעתיקים את הערכים של האילוצים, תחילה המשתנים המלאכותיים בבסיס וערכם בפונקציית המטרה הוא M או M-.
הוספת שורה של Cj שמחושבת כך :
**הערך של Xi\Yi בפונקציית המטרה-((חיבור כל הערכים בעמודה)*M\-M )**

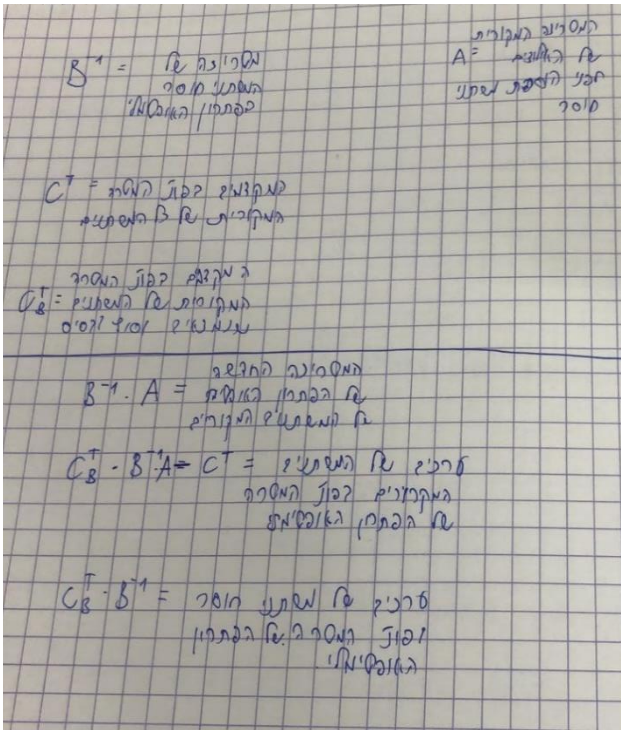
| | | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $y_1$ | $y_2$ | $y_3$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | מחירים | 8 | 4 | 6 | 0 | 0 | 0 | $M$ | $M$ | $M$ | | |
| | בסיס | $\bar{a}_1$ | $\bar{a}_2$ | $\bar{a}_3$ | $\bar{a}_4$ | $\bar{a}_5$ | $\bar{a}_6$ | $\bar{a}_7$ | $\bar{a}_8$ | $\bar{a}_9$ | $\bar{b}$ | $\frac{b_i}{a_{ik}}$ |
| 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 | $9M$ | $=Z$ |

2. ממשיכים לפתור כמו בשיטת הסימפקס.

## **הערות :**

1. אם אחד מהמשתנים המלאכותיים נשאר בבסיס בפתרון האופטימלי אז לבעיה **אין פתרון.**
2. אם יש שוויון באילוץ עדיין מוסיפים משתנה מלאכותי.

## **המשמעות של שיטת הM הגדול :**


$A=$ המטריצה המקורית של האילוצים לפני הוספת משתני מוסף

$B^{-1}=$ מטריצה של המשתני מוסף בפתרון האופטימלי

$C^T=$ המקדמים בפונ' המטרה המקורית של כל המשתנים

$C_B^T=$ המקדמים בפונ' המטרה המקוריות של המשתנים שנמצאים בסוף בבסיס

$B^{-1}\cdot A=$ המטריצה החדשה של הפתרון האופטימלי של המשתנים המקוריים

$C_B^T\cdot B^{-1}\cdot A-C^T=$ ערכים של המשתנים המקוריים בפונ' המטרה של הפתרון האופטימלי

$C_B^T\cdot B^{-1}=$ ערכים של משתני מוסף בפונ' המטרה של הפתרון האופטימלי.

## **<u>בעיה דואלית ופרימלית :</u>**

לכל בעיית תכנות ליניארי קיימת בעיה הקשורה בה ונבנית ממנה, וזוהי הבעיה הדואלית. הבעיה המקורית תיקרא הבעיה הפרימלית

<u>מעבר מפרימלית לדואלית :</u>

**1.** לכל אילוץ מתאימים משתנה דואלי Yi , i= מספר אילוצים.

**2.** אם הבעיה הפרימלית היא מקסימום אז הדואלית מינימום וההיפך.

**3.** לפני המעבר צריכים לדאוג לסימן האילוצים(גדול שווה או קטן שווה).

**4.** בניית האילוצים ופונקציית המטרה של הבעיה הדואלית כך :

*נתונה הבעיה הפרימאלית הבאה:*

$$\mathrm{Max}\{Z = C_1x_1 + C_2x_2 + \ldots + C_nx_n\}$$

$$a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n \le b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n \le b_2$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n \le b_m$$

$$x_j \ge 0 \ldots\ldots (j = 1,2,\ldots,n)$$

דהיינו. הבעיה הדואלית לבעיה הנתונה היא :

$$\mathrm{Min}\{V = b_1y_1 + b_2y_2 + \ldots b_my_m\}$$

$$a_{11}y_1 + a_{22}y_2 + \ldots a_{m1}y_m \ge c_1$$

$$a_{12}y_1 + a_{22}y_2 + \ldots a_{m2}y_m \ge c_2$$

.

.

$$a_{1n}y_1 + a_{2n}y_2 + \ldots a_{mn}y_m \ge c_n$$

$$y_i \ge 0 \qquad (i = 1,2,\ldots..m)$$

## **<u>תכונות הבעיה הדואלית :</u>**

א. הבעיה הדואלית של הבעיה הדואלית היא הבעיה הפרימלית .

ב. בהנחה שקיימים פתרונות אפשריים סופיים לשתי בעיות , אזי קיים פתרון אופטימאלי סופי לשתיהן והוא זהה , דהיינו Min V = V* = Z* = Max Z

ג. פתרון בסיסי אפשרי של הבעיה הפרימלית מהווה חסם תחתון לאופטימום של הדואלית ולהיפך.

ד. אם לאחת הבעיות פיתרון לא חסום, אזי לבעיה המשלימה אין פיתרון אפשרי.

ה.שיטת הסימפלקס פותרת את שתי הבעיות.

ו. הפתרון של הבעיה הדואלית : לכל משתנה דואלי מתאים משתנה חוסר שמתווסף לאילוץ שמתאים לו. ערכו של משתנה החוסר שמופיע בפתרון האופטימלי בפונקציית המטרה הוא ערכו של המשתנה הדואלי.

ז.רק המשתנים הדואלים של האילוצים שמתקיימים כשיוויון הם יהיו בבסיס בפתרון האופטימלי של הבעיה הדואלית ויקבלו ערך חיובי.

ח. אם פותרים בעזרת שיטה גרפית, האילוצים שיוצרים את נקודת החיתוך שהוא הפתרון האופטימלי, אז המשתנים הדואלים שייכים להם יימצאו בבסיס בפתרון האופטימלי וכל שאר המשתנים יתאפסו. יהיו 2 משוואות ו2 נעלמים.

**<u>בעיה קנונית והקשר שלה למעבר בין פרימלי לדואלי :</u>**

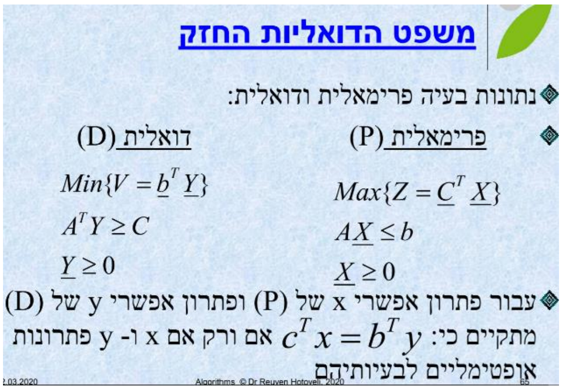
| **<u>אילוץ</u>** | **<u>MAX</u>** | **<u>MIN</u>** |
|---|---|---|
| קטן\שווה | קנוני | אנטי קנוני |
| גדול\שווה | אנטי קנוני | קנוני |
| שווה | ניטרלי | ניטרלי |

| **<u>משתנה</u>** | **<u>פירוש</u>** |
|---|---|
| קנוני | X>=0 |
| אנטי קנוני | X<=0 |
| ניטרלי | X=0 |

מעבר בין הבעיות הוא לפי הטבלה למשל : אם בבעיה פרימלית יש אילוץ ניטרלי אז המשתנה המתאים לו בפרימלית יהיה ניטרלי.(**אילוץ פרימלית->משתנה דואלי**).

אם בבעיה פרימלית יהיה משתנה אנטי קנוני אז האילוץ המתאים לו בבעיה הדואלית יהיה אנטי קנוני.

**<u>משפט הדואליות החלש :</u>** אם קיימים פתרונות אפשריים לבעיה הפרימאלית והבעיה הדואלית שערכי פונקציות המטרה שלהם זהים, אזי שני הפתרונות הם אופטימאליים לבעיותיהם.



# 4. בעיית תובלה

גרסה מקוצרת של שיטת הסימפלקס ובדרך כלל בבעיות תובלה יש מספר גדול של אילוצים ומשתנים ואם נשתמש בשיטת הסימפלקס ייקח לנו הרבה זמן.

**מבנה בעיית תובלה:**

**קבוצת מקורות m-** המייצרת מוצר יחיד.

**קבוצת יעדים n-** הדורשים את אותו מוצר.

לכול מקור יש כמות יצור $s_i$ ולכל יעד יש כמות ביקוש $d_j$.

**הובלה-** הובלת מוצר ממקור מסוים ליעד מסוים כרוכה בעלות מסוימת $C_{ij}$.

**משתני החלטה: הקצאות –** כמות המוצרים שהוחלט להעביר מכל מקור לכל יעד $x_{ij}$.

**פונקציית מטרה: מחיר התובלה-** סך כל עלויות התובלה של ההקצאות מהמקורות אל היעדים (בדרך כלל אנחנו רוצים ערך מינימלי) Z.

**פתרון בסיסי –** זה פתרון בו מספר המשתנים בעלי ערך חיובי הוא m+n-1

**שיטת פתרון:**

1. נשתמש בשיטה "הפינה הצפון-מערבית" ונקבל פתרון בסיסי (לא בטוח אופטימלי)
במקרה שההיצע והביקוש שווים יורדים שורה נותנים לשורה החדשה ערך 0 וממשיכים את השיטה (המספר 0 גם נספר בכמות משתנים)
**סיבוכיות: O(m+n)**

2. **מציאת פתרון אופטימלי:**
הפחתת הערך $u_i$ מכל המחירים בשורה $i$
הפחתת הערך $v_j$ מכל המחירים בעמודה $j$
באופן שיתקבל $C_{ij} - u_i - v_j = 0$ עבור כל משתנה בסיסי.
את ערכי $u_i$ ו- $v_j$ נקבל על ידי פתירת מערכת המשוואות
$C_{ij} - u_i - v_j = 0$ עבור כל $i$ ו-$j$ שעבורם $x_{ij}$ הוא משתנה בסיסי.
עבור כל משתנה לא בסיסי נחשב את המחיר החדש $C_{ij}$ על ידי הפחתת $u_i$ של השורה בה הוא נמצא והפחתת $v_j$ של העמודה בה הוא נמצא.
כלומר נחשב עבור כל משתנה לא בסיסי את ערך הביטוי $C_{ij} - u_i - v_j$
אם ימצא משתנה לא בסיסי שערך זה, כלומר מחירו החדש שלילי יש להוסיפו כמשתנה בסיסי לפתרון ולהוציא משתנה אחר במקומו והפתרון הנוכחי אינו אופטימלי.

**2.1** אם מצאנו משתנה שמחירו שלילי נצטרך להחליט את מי להוציא:

המשתנה שנכניס לבסיסי יסומן ב (+) והמשתנה שנוציא מהבסיס יסומן ב (-)

התא שמסומן ב (+) הופך ל"תא מקבל"

התא שמסומן ב (-) הופך ל"תא תורם"

כל תא תורם מקטין את ההקצאה שלו בדיוק באותו ערך שבו גדל המשתנה הנכנס לבסיס.

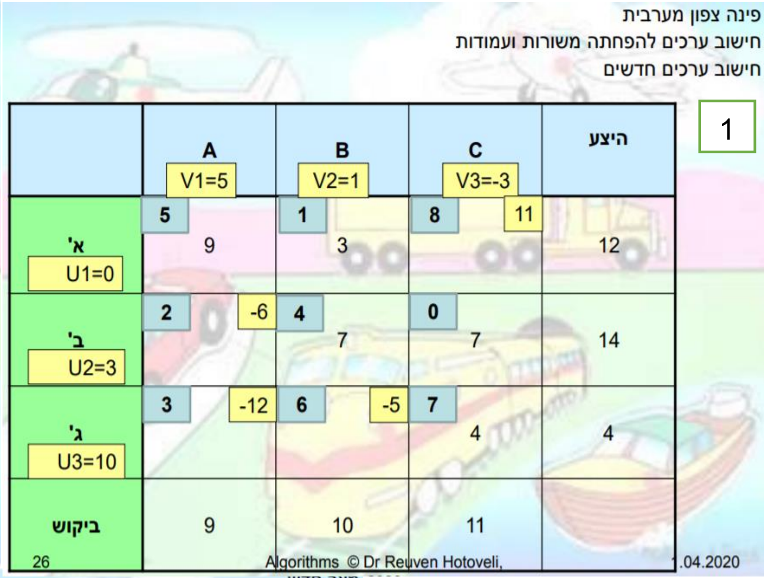
לכן התא התורם שיש בו ההקצאה הקטנה ביותר הוא יגיע ראשון להקצאה אפס ואז הוא יצא מהבסיס.

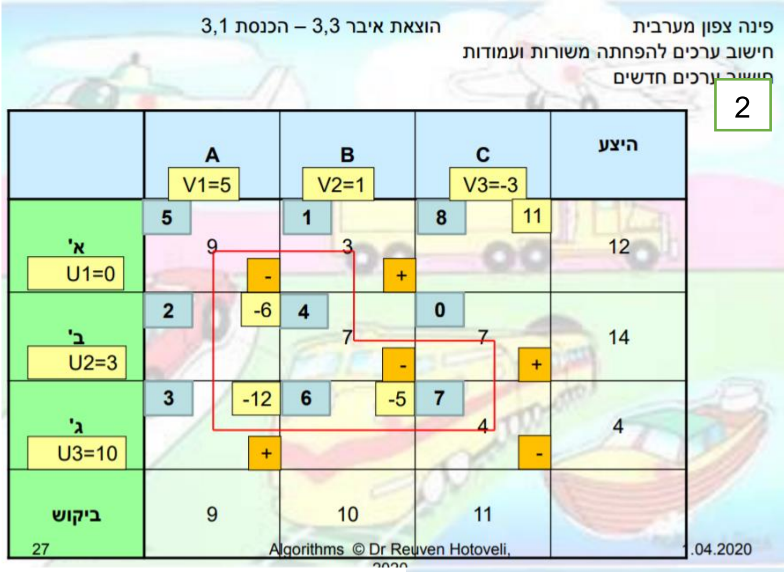
צריך לשים לב שבעקבות הכנסת איבר לבסיס והוצאת אביר מהבסיס יש תגובת שרשרת ויהיו עוד תאים שיהפכו לתאים תורמים או מקבלים כדי לשמור על האילוצים.

אם עדיין יש משתנים שהם לא בבסיסי והם שליליים הם יכולים להקטין לנו את פונקציית המטרה אז נרצה להכניס אותם לבסיס אז נבצע את שלב 2 עוד פעם.

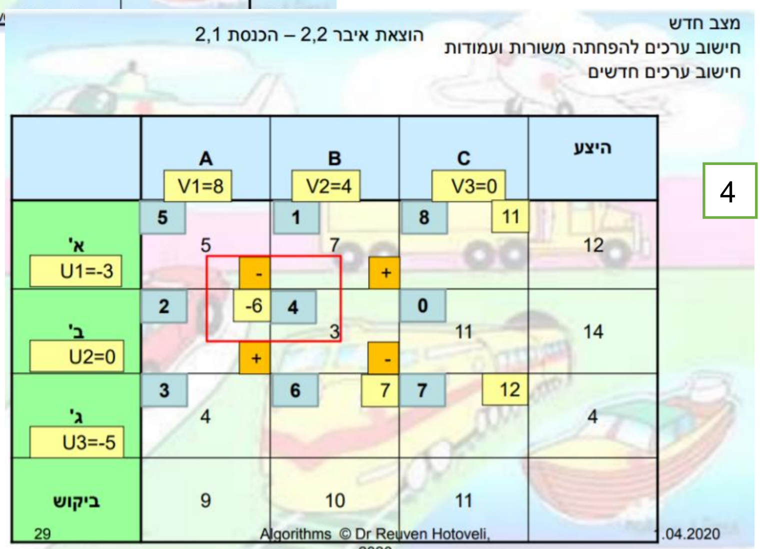
אחרי שכל המשתנים שהם לא בבסיסי הם לא שליליים הגענו לפתרון אופטימלי.

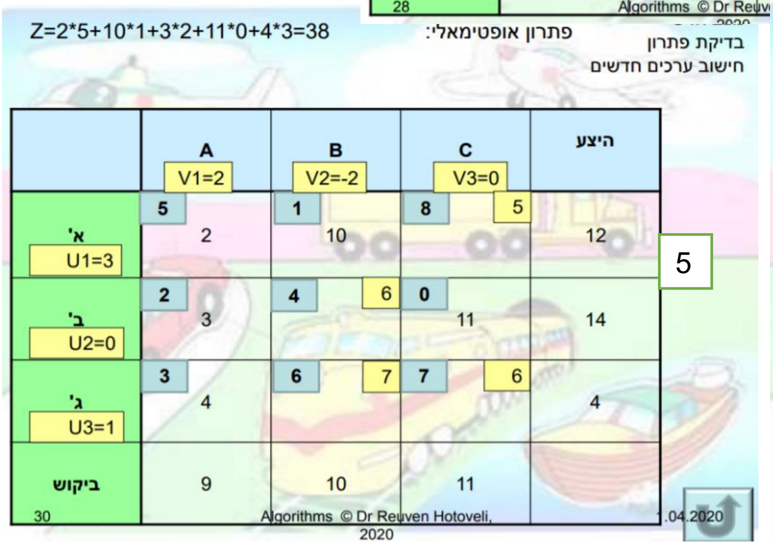
## <u>דוגמה:</u>







# 5. בעיית השמה

בעיית השמה זה שצריך להביא את העלות למינימום בבעיה בה יש לבצע השמה של $n$ עבודות ל - n משאבים כאשר כל משאב מבצע עבודה אחת ולכל עבודה מוקצה משאב אחד.

**סימונים:**

המחירים $C_{ij}$ הם העלויות הקשורות בייעוד אדם $i$ למשימה $j$

$x_{ij}$ הוא משתנה אשר מקבל ערך 1 כאשר אדם $i$ יבצע את המשימה $j$ אחרת הוא יקבל 0.

**אלגוריתם הונגרי:**

1. נחסיר מכול שורה את האיבר המינימלי
2. נחסיר מכול עמודה את האיבר המינימלי
3. נבחר את התאים שיש בהם אפסים בתור פתרון אופטימלי,
   אם יש כמה אפסים באותה שורה אפשר לבחור את אחד מהם
4. אחרי שבחרנו את התאים עם האפסים נחבר את הערך שלהם לפני הפעלת האלגוריתם וזה הפתרון האופטימלי
5. **במקרים מיוחדים:** גם אחרי שנעשה את שלב 4 לא נמצא פתרון.
   במקרה כזה נעביר מספר מינימלי של ישרים אנכיים או אופקיים ונכסה בעזרתם את כל האפסים שבטבלה.
   אם מספר קווים כאלו שווה למספר השורות של המטריצה אז הפתרון שהתקבל הוא אופטימלי ; אחרת לא אופטימלי.
6. במקרה שהפתרון לא אופטימלי ומספר הקווים לא שווה למספר השורות, נחסיר בכל שורה את האיבר המינימלי שלא מכוסה בקו (במקרה שיצא לנו מספר שלילי אחרי החיסור נהפוך אותו לחיובי בזה שנוסיף לשורה/עמודה את אותו מספר)
7. נבצע שוב את שלב 5.

# 6. אלגוריתם חמדני

**הרעיון:**

- בונים את הפלט בהדרגה .
- בכל צעד בוחרים החלטה אופטימלית לאותו הצעד ולא משנים אותה שהיא אופטימלית באופן מקומי.
- בתקווה שבחירה זו תוביל לפתרון אופטימלי כולל.

עתה יש להוכיח שהפתרון המתקבל ע"י האלגוריתם הוא אופטימלי.

**סכמה מומלצת להוכחת נכונות:**

- בד"כ אלגוריתם חמדני בונה את הפתרון בשלבים.
- בכל שלב הפתרון שהאלגוריתם בנה עד כה מוכל בפתרון אופטימאלי כלשהו.
- בסוף ריצת האלגוריתם ערכו של הפתרון שהתקבל זהה לגודל הפתרון האופטימאלי ולכן הם שווים.

**יעילות :**

לעיתים כאשר לא ניתן למצוא את הפתרון האופטימלי בזמן סביר, שימוש באלגוריתם חמדן עשוי לתת קירוב טוב לפתרון המיטבי בזמן קצר. בהתחשב בעובדה שבעיות אופטימיזציה רבות הן קשות מאוד שימוש באלגוריתמים חמדניים יכול להיות יעיל במיוחד.

במקרים מסוימים האלגוריתם החמדן הוא גם האלגוריתם האופטימלי. כלומר, סדרת בחירות חמדניות נותנת לפעמים את הפתרון האופטימלי הכללי לבעיה. למשל האלגוריתם של פרים למציאת עץ פורש מינימלי בגרף

**דוגמא אופטימלית: בעיית התרמיל :**

נתון תרמיל שלו קיבולת משקל W ונתונים n פריטים . n , , , , 3 2 1a a a a לכל פריט ( ) i מיוחס משקל ( ) w ai ורווח a p . והמטרה היא לבחור תת קבוצה של פריטים ,כך שמשיגים רווח מקסימלי מבלי להפר את מגבלת המשקל )קיבולת התרמיל.)

**הקלות:**

תנאי נוסף ,הקלה – מותר לחלק לשבור את הפריטים. בחלוקת פריט ביחס מסוים הרווח מתחלק באותו יחס .הציעו אלגוריתם חמדני לפתרון אופטימלי של הבעיה.

**פתרון:**

נגדיר רווח סגולי רווח ליחידת משקל של כל פריט כיחס בין רווח הפריט למשקלו ai ,כלומר לכל ai

$$u(ai) = \frac{p(ai)}{w(ai)}$$

ברור שככל שהרווח הסגולי גבוה יותר, הפריט משתלם יותר.

**אלגוריתם :**

1. נמיין את כל הפריטים לפי הרווח הסגולי בסדר לא עולה מגבוה לנמוך.
2. נמלא את התרמיל בפריטים לפי הסדר הזה כל עוד הדבר אינו גורם לחריגה מקיבולת התרמיל.
3. אם וכאשר הפריט האחרון at שמנסים להכניס לתרמיל לא נכנס משום שמשקלו p(at)

גדול מהמרווח הפנוי B שנשאר בתרמיל , אז נחתוך מהפריט חלק במשקל B ונכניסו לתרמיל

בזה נמלא את התרמיל עד "אפס מקום "

**נקבל פתרון אופטימלי.**

**<u>דוגמא לא אופטימלית : בעיית הסוכן הנוסע:</u>**

סוכן מכירות רוצה לעבור במספר יישובים כדי למכור את הסחורה שלו. המטרה היא למצוא את המסלול הקצר ביותר שיעבור דרך כל היישובים.

על פי שיטת **האלגוריתם החמדן**, הסוכן הנוסע צריך להסתכל בכל פעם במפה ולנסוע ליישוב הקרוב ביותר בו לא ביקר עדיין.

**<u>הסבר למה לא אופטימלי:</u>**

במקרה זה שיטת **האלגוריתם החמדן** לא תיתן בהכרח את הפתרון הטוב ביותר.

יכול להיות מצב בו הסוכן ידלג על יישוב מסוים משום שישנו יישוב אחר קרוב יותר, כך שיאלץ לחזור ליישוב עליו דילג בסוף המסלול ולעשות דרך ארוכה יותר.

# 7. תכנות דינמי

תכנון דינמי ניתן ליישום כאשר התת-בעיות אינן בלתי תלויות דהיינו , כאשר לתת –בעיות יש תת – בעיות משותפות.

אלגוריתם תכנון –דינמי פותר את כל תת-תת בעיה פעם אחת בלבד , ואז שומר את התשובה בטבלה, וכך חוסך את העבודה הכרוכה בחישובה מחדש של התשובה בכל פעם שמופיעה התת-תת בעיה.

**שלבים לבניית האלגוריתם + דוגמא :**

- נתון מערך A של n מספרים (לא בהכרח חיוביים) $A_1, A_2, \dots A_n$. יש למצוא זוג אינדקסים $i \le j$ שימקסם את הערך של :

$$\sum_{k=i}^{j} A[k] = A[i] + A[i+1] + A[i+2] + \dots + A[j]$$

- הערה : אם כל המספרים שליליים, נסכים שסכום התת סדרה המקסימלי שווה לאפס.

**א. הגדרת תתי בעיות-**

- נגדיר את $c[i]$ כסכום התת סדרה המקסימלית **שמסתיימת בתא ה- i** של המערך A .

**ב. הקשר בין תתי הבעיות שהגדרנו לפתרון השאלה**

- הפתרון של השאלה יהיה :

$$\max_{i=1}^{n} \{c[i]\}$$

**ג. חישוב תתי הבעיות הקטנות ביותר**

$$C[1]=\max\{0, A[1]\}$$

**ד. חישוב תתי בעיות באמצעות תתי בעיות קטנות מהן – נוסחה רקורסיבית**

$$C[i]=\max\{0, c[i-1] + A[i]\}$$

**ה. נכונות** - מספיק לרשום כמה שורות על איך בדיוק הנוסחה הרקורסיבית עובדת.

**ו. ניתוח זמן הריצה**

**ז. האלגוריתם :**

```
int MaxSubSum (const int A[], int n)
{int k, Thissum=0, maxsum=0;
for (k=0; k<n; k++)
 {Thissum +=a[k];
   if (maxsum<Thissum)   maxsum=Thissum;
   else   if (Thissum<0)  Thissum =0;
 }
return maxsum;
}
```

**מסקנה חשובה :**

## החלק הקשה בתכנות דינמי

- החלק הקשה הוא למצוא את ההגדרה הנכונה של תתי הבעיות.
- איך מוצאים את ההגדרה הנכונה?
- פשוט מנחשים!
- מנסים הגדרה אחת לתת בעיה ובודקים האם אפשר למצוא נוסחה רקורסיבית עבורה.
- אם לא מצליחים לגרום לרקורסיה לעבוד אז מנסים לשנות את הגדרת תת הבעיה על מנת שנפתור את הקושי שנתקלים בו.

## כפל מטריצות- תכנות דינמי :

**המטרה :**

- בהינתן סדרה $\langle A_1, A_2, \dots, A_n \rangle$ של n מטריצות , אשר בה ממדיה של המטריצה $A_i$ הם $p_{i-1} \times p_i$ עבור $i = 1,2,\dots\dots\dots,n$ , יש להציב הצבת סוגריים מלאה במכפלה $A_1 \cdot A_2 \cdot \dots\dots \cdot A_n$ באופן שימזער את מספר המכפלות הסקלריות המבוצעות בעת חישוב מכפלת המטריצות.

**הגדרות :**

- יהי $m[i,j]$ המספר המינימלי של המכפלות הסקלריות הנדרשות לחישוב המטריצה $A_{i..j}$
- נגדיר את $s[i,j]$ כערך של $k$ שבו נוכל לפצל את המכפלה $A_iA_{i+1}\cdots A_j$ כדי לקבל הצבת סוגריים אופטימלית.

$$m[i,j] = \begin{cases} 0 & \text{אם } i = j \\ \min\limits_{i \le k < j} \{ m[i,k] + m[k+1,j] + p_{i-1}p_kp_j \} & \text{אם } i < j \end{cases}$$

**האלגוריתם:**

```
MATRIX-CHAIN-ORDER(P)
n ← length[p] - 1
for i ← 1 to n
  do m[i, i] ← 0
for L ← 2 to n
  do for i ← 1 to n - L + 1
       do j ← i + L-1
          m[i, j] ← ∞
          for k ← i to j - 1
             do q= m[i, k] + m[k + 1, j] + p_{i-1} p_k p_j
                if q < m[i, j]
                   then m[i, j] ← q
                        s[i, j] ← k
return m and s
```



**מציאת הפתרון האופטימלי :**

```
MATRIX-CHAIN-MULTIPLY(A, s, i ,j)
{
if j >i
{ X ← MATRIX-CHAIN-MULTIPLY(A, s, i, s[i, j])
Y ← MATRIX-CHAIN-MULTIPLY(A, s, s[i, j] + 1, j)
return MATRIX-MULTIPLY(X, Y)
}
else return A_i
}
```

**קריאת הפונקציה הזאת עם ערכים של i=1 ו-j=n מחשבת מכפלת המטריצות על פי הסדר האופטימלי.**

**זמן הריצה :** O(N^3).

# מציאת תת סדרה משותפת גדולה ביותר(תמ"א) – תכנות דינמי :

**הגדרות:**

- נסמן ב-$c(i, j)$ את האורך של תמ"א של $X_i$ ו- $Y_j$
- אז אורך פתרון הבעיה המקורית הוא $c(m, n)$

- נפתח נוסחת נסיגה לחישוב של $c(i, j)$:
  - אם $i = 0$ או $j = 0$ (או שניהם), אזי $c(i, j) = 0$ (תנאי שפה)
  - אחרת $(i, j > 0)$, לפי המשפט:
    - אם $x_i = y_j$, אז $c(i, j) = c(i-1, j-1) + 1$
    - אחרת $(x_i \neq y_j)$, אז $c(i, j) = \max\{c(i-1, j), c(i, j-1)\}$

**האלגוריתם** :

```
LCS-LENGTH (X, Y)
    m ← length [X]
    n ← length [Y]
    for i ← 1 to m  do
        c[i, 0] ← 0
    for j ← 0 to n  do
        c[0, j] ← 0

    for i ← 1 to m  do
        for j ← 1 to n  do
            if (x_i = y_j)  then
                c[i, j] ← c[i-1, j-1] + 1
                b[i, j] ← " ↖ "
            else if (c[i-1, j] ≥ c[i, j-1])  then
                c[i, j] ← c[i-1, j]
                b[i, j] ← "↑"
            else
                c[i, j] ← c[i, j-1]
                b[i, j] ← "←"
    return  c and b
```

כדי לשחזר את האיברים של תמ"א יש לעקוב אחר החיצים b[I, j] החל מהמשבצת הימנית התחתונה בטבלה .

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | A | 0 | ↑ | ↑ | ↑ | ↖ | ← | ↖ |
| 2 | **B** | 0 | ↖ | ← | ← | ↑ | ↖ | ← |
| 3 | **C** | 0 | ↑ | ↑ | ↖ | ← | ↑ | ↑ |
| 4 | **B** | 0 | ↖ | ↑ | ↑ | ↑ | ↖ | ← |
| 5 | D | 0 | ↑ | ↖ | ↑ | ↑ | ↑ | ↑ |
| 6 | **A** | 0 | ↑ | ↑ | ↑ | ↖ | ↑ | ↖ |
| 7 | B | 0 | ↖ | ↑ | ↑ | ↑ | ↖ | ↑ |

**זמן הריצה**: O(nm).

# 8. אלגוריתם הופמן :

מטרה : הצפנה של הודעה , כמחרוזת ארוכה של סיביות .

חשוב לזכור : קוד של סימן אחד אסור שיהיה תחילתו של קוד של סימן אחר .

לכן המטרה היא לקבוע קוד כזה שההודעה תפורש באופן חד משמעי.

האלגוריתם של הופמן – הוא דוגמא לאלגוריתם חמדן , אשר משתמש בטבלת שכיחויות של מופעי התווים לבניית דרך אופטימלית לייצוג כל תו כמחרוזת בינארית .

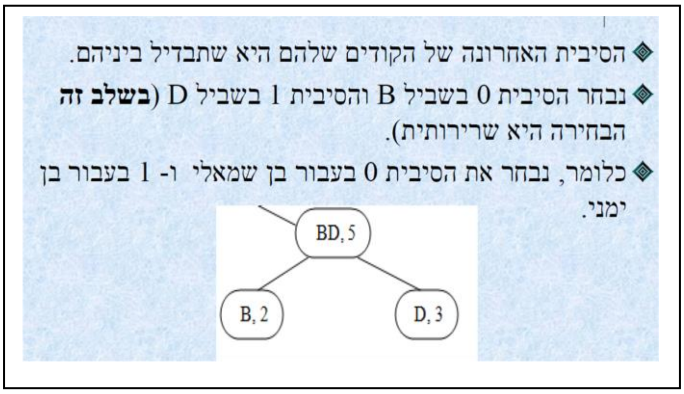
דרך פעולה :

1. נקבל טבלת של תווים ושכיחויות של כל תו .
2. תמיד ניקח את שני התווים עם השכיחות הנמוכה ביותר ונרכיב מהם עץ שבראשו שני התווים מתחברים למחרוזת והתו בעל השכיחות הנמוכה ביותר יהיה בן שמאלי והשני בן ימני
3. לבן השמאלי ניתן סיבית 0 , ולבן השמאלי ניתן סיבית אחד .
4. ככה נעשה עד שנסיים לעבור על כל הטבלה .

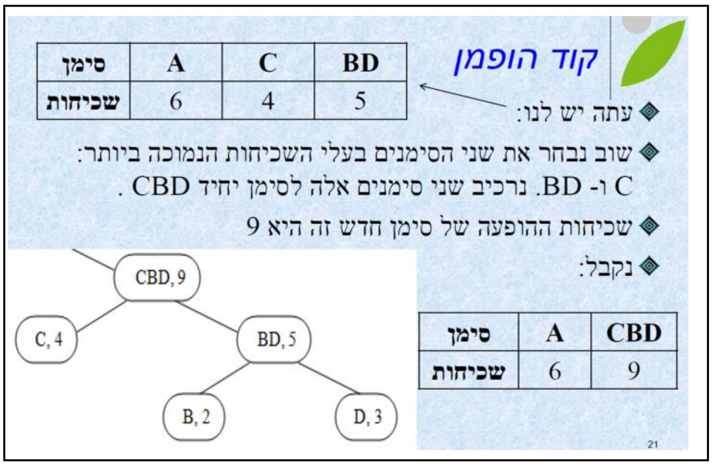
דוגמא :

שלב 1 :

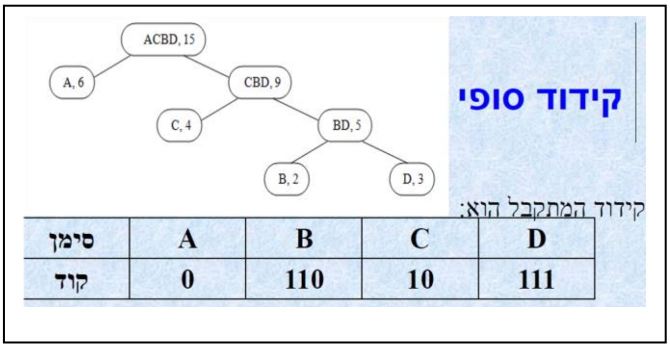

שלב 2:




שלב 3:




מטרה סופית :




לכל עלה בו משוכן תו מסוים , אורכו של קוד התו שווה לעומק העלה .

לפי הדוג' שלנו $d_T(A) = 1$

בתהליך שלנו , סימנים המופיעים לעיתים דחופות בהודעה מקבלים קודים קצרים יותר מאלו המופיעים לעיתים פחות תכופות .

**תכונות העץ :**

- עצי הופמן הם לא עצי חיפוש בינאריים
- העלים לא חייבים להיות ממוינים
- הצמתים הפנימיים לא מכילים תווים
- קוד אופטימלי עבור קובץ מיוצג תמיד ע"י עץ בינארי מלא , אשר בו לכל צומת שאינו עלה יש שני בנים .

חשוב לזכור : בכל עץ בינארי ,מספר העלים גדול ב1 ממספר הצמתים בעלי דרגה 2 .

מספר הסיביות הדרושות לקידוד קובץ אשר מורכב מכל האלפבית שמורים בקבוצה C הינו : $B(T) = \sum_{c \in C} f[c] d_T(c)$ $d_T$: עומק של העלה

מספר זה הינו העלות של העץ $T$

**אלגוריתם של הופמן עם ערימה :**

```
BUILD-HEAP(A)
n ← |C|
Q ← C
for i ← 1 to n - 1
    do z ← ALLOCATE-NODE()
        x ← left[z] ← EXTRACT-MIN(Q)
        y ← right[z] ← EXTRACT-MIN(Q)
        f[z] ← f[x] + f[y]
        INSERT(Q, z)
return EXTRACT-MIN(Q)
```

# 9. גרפים :

## מושגים :

**גרף –** G(V,E) , V- קבוצת הקודקודים, E – קבוצת הקשתות.

**דרגת כניסה –** כמה קשתות נכנסות לצומת.

**דרגת יציאה –** כמה קשתות יוצאות מהצומת.

**דרגה –** דרגת כניסה + דרגת יציאה.

**לולאה** – קשת מצומת לעצמה.

**מעגל** – מסלול בגרף מקודקוד כלשהו לעצמו, כאשר בגרף יש מעגל הוא נקרא גרף מעגלי.

**מסלול פשוט –** מסלול ובו אף קשת לא מופיעה יותר מפעם אחת.

**גרף שלם –** הינו גרף שבו כל קודקוד מחובר לשאר הקודקודים, גרף מלא שבו n קודקודים יסומן בKn. ( n מעל 2 צלעות).

**רכיב קשיר –** קבוצה מקסימלית של קודקודים בגרף לא מכוון שבה יש מסלול פשוט בין כל 2 קודקודים בגרף, כנ"ל גם קודקוד בודד.

**גרף קשיר –** קיים מסלול פשוט בין כל שני קודקודים בגרף.

**רק"ח** – רכיב קשיר מכוון בחוזקה – רכיב קשיר בגרף מכוון.

**גרף דו צדדי** – גרף בעל קבוצת קודקודים המתחלקת לשתי קבוצות זרות כך שלא קיימת קשר בתוך קודקודי הקבוצות אלא רק קשתות בין הקבוצות.

**גרף משלים של G** – גרף שבו כל הצלעות שחסרות בG.

**עלה –** צומת בעל דרגה 1 בדיוק.

## משפטים :

א. בגרף קשיר לא מכוון שבו מספר הקודקודים גדול ממספר הצלעות יש מעגל.

ב. התנאים הבאים שקולים :

- גרף G הינו עץ.
- G קשיר ובעל n-1 קשתות.
- G חסר מעגלים ובעל n-1 קשתות.

ג. בכל גרף פשוט מתקיים שסכום דרגות הקודקודים שווה לפעמיים מספר הצלעות.

## **<u>מפרט :</u>**

**<u>הוסף קשת (x, y ,G)</u>** – מוסיפה קשת מצומת X לY.

**<u>הוסף קשת משוקללת(x,y,w,G)-</u>** מX לY עם משקל W.

**<u>הסר קשת (x, y ,G) –</u>** מסירה קשת מצומת X לY.

**<u>הסר קשת משוקללת(x,y,w,G)-</u>** מX לY עם משקל W.

**<u>שכנים?(x,y, G)-</u>** האם X שכן של Y.

**<u>גרף ריק? (G)</u>**

**<u>הוסף צומת (G,X) –</u>** הוסף צומת עם מידע X.

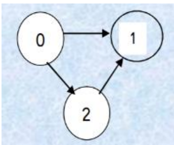
**<u>הסר צומת (G,X) –</u>** מסיר צומת עם מידע X.

## **<u>ייצוגים של גרפים :</u>**

### **<u>מטריצת סמיכות :</u>**

הדרך הטבעית להצגה של גרף הוא מערך דו-מימדי מסדר. n x n נסמן אותו בg.

השורות הן הקודקודים מהם יוצאות הקשתות והעמודות הן הקודקודים בהם נכנסות הקשתות.

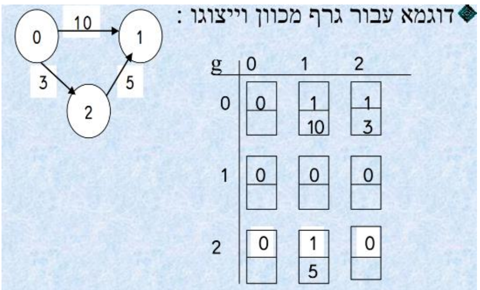



| g | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 |

### **<u>מטריצת המשקלות :</u>**

לגבי כל אלמנט במטריצה זו נחזיק מספר מאפיינים:

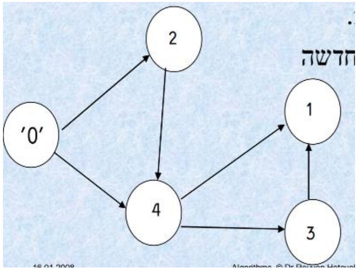
1. האם קיימת קשת בין זוג סדור של קודקודים .
2. במידה וקיימת קשת בין זוג סדור של קודקודים נרצה לדעת מהו המידע המיוחס לה.

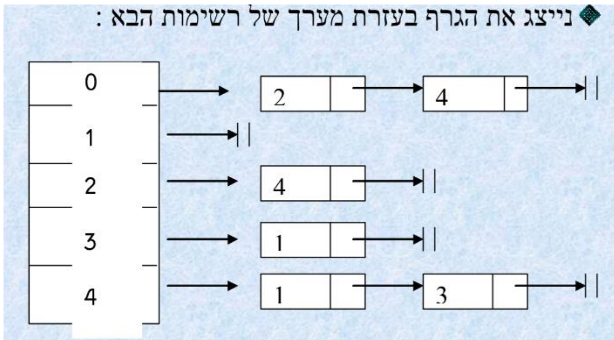



| g | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 / 10 | 1 / 3 |
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 / 5 | 0 |

**רשימת סמיכות :**

- הקודקודים של הגרף מיוצגים על ידי מערך.
- כל אינדקס של המערך מייצג צומת (קודקוד) בגרף .
- הקשתות של הגרף מיוצגות על ידי רשימה הנקראת רשימת סמיכות (שכנות/קשירות) .
- כל צומת ברשימת סמיכות מייצג קשת בגרף.




**משפטים על גרף דו צדדי :**

1. G גרף דו-צדדי וקשיר ⟵ G אינו מכיל מעגל באורך אי-זוגי.
2. G גרף קשיר שאינו מכיל מעגל באורך אי-זוגי ⟵ G גרף דו-צדדי.

**מטריצת מסלולים :**

משפט:
אם מטריצת סמיכות המייצג גרף פשוט אזי $A^n[i,j]$ מציין מספר המסלולים באורך n מקדקוד i לקדקוד j.

כאשר ברצוננו לדעת האם קיים מסלול כלשהו בין קדקודים כלשהם בגרף, עלינו למצוא את $A^{\le n}$, כאשר $A^{\le n} = A + A^2 + \ldots + A^n$

<u>טענה:</u>
נתון גרף $G=(V,E)$ כאשר $|V|=n$ ($n$ קדקודים בגרף). כל מסלול אפשרי מכל קדקוד לכל קדקוד בגרף הוא בעל אורך $n$ לכל היותר.

**דוגמא :**

נכפיל את שני הווקטורים הבוליאניים (כפל סקלרי).

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

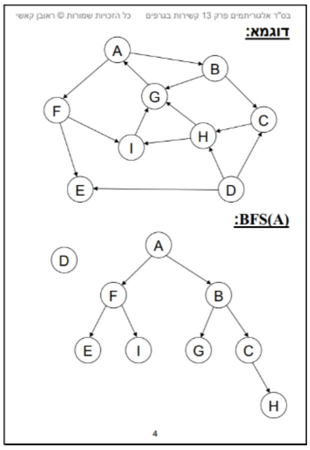
$$A^2 = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

1.5.2020 Algorithms © Dr Reuven Hotoveli, 2020 19

מסקנה:
לכל שורה i (וקטור i) במטריצה $A^2$ ולכל עמודה j (וקטור j) במטריצה A, לכפל הסקלרי i*j יש ערך 1 (TRUE) **אם ורק אם** קיים מסלול באורך 3 מקדקוד i לקדקוד j בגרף.

# שיטות סריקה:

## BFS (רוחב):






- **אלגוריתם BFS:**

BFS($v$)

1. Visit & mark $v$
2. $Q \leftarrow insert(v)$
3. While $Q \neq \emptyset$ do
   - 3.1 $u \leftarrow head(Q)$
   - 3.2 For all unmarked $w \in Adj[u]$ do
     - 3.2.1 Visit & mark $w$
     - 3.2.2 $Q \leftarrow insert(w)$
   - 3.3 $Pop(Q)$

שתי מטרות עיקריות:

1. סריקה שיטתית של הגרף.
2. מציאת מסלולים קצרים.

3

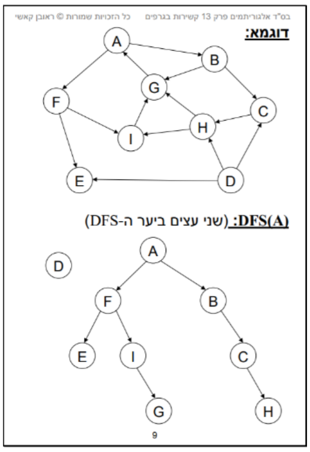
## תיאור האלגו':

מחלקים את קודקודי הגרף לשכבות באופן הבא:
בשכבה ראשונה יהיו קודקודים שניתן להגיע אליהם, מקודקוד מקור S, בעזרת מסלול שאורכו אחד.
בשכבה שנייה יהיו קודקודים שניתן להגיע אליהם, מקודקוד מקור S, בעזרת מסלול שאורכו 2.
ובאופן כללי : בשכבה i , לכל $i \geq 1$, יהיו קודקודים כך שניתן להגיע אליהם, מקודקוד 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 השנייה
וכך נמשך תהליך הביקור בקודקודי הגרף עד שנבקר בכל קודקודי הגרף פעם אחת בלבד.

## DFS(עומק):

להלן השגרה המשוכתבת DFS_Visit(u)

1. אפור ← color[u]
2. time+1 ← time
3. time ← d[u]
4. עבור כל קודקוד v שהינו קודקוד סמוך ל- u
   בצע :
   בדוק את הצבע של הקודקוד 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 u ← p[v]
     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. שחור ← color[u]
6. time+1 ← time
7. time ← f[u]




- באלגוריתם זה הקדקודים נצבעים במטרה לציין את מצבם. כל קדקוד נצבע באחד מבין הצבעים הבאים : לבן, שחור ואפור.
- בתחילת האלגוריתם כל קדקוד יהיה צבוע בצבע לבן. כאשר קדקוד מתגלה, כלומר מגיעים אליו תוך כדי הסריקה הוא נצבע בצבע אפור.
- כאשר אנו מסיימים את הטיפול בקדקוד (עוזבים אותו ולא חוזרים אליו לטיפול) אז הקדקוד יצבע בצבע שחור.
- נגדיר :
- d[u] מציין את מועד הגילוי של הקדקוד u בעת הסריקה.
- f[u] מציין את מועד סיום הטיפול בקדקוד u בעת הסריקה.



- קשת קדימה F - קשת $(u,v)$ היא קשת קדימה אם היא קשת מאב קדמון $u$ בעץ ה-DFS לצאצא שלו $v$ בעץ ה-DFS.
- קשת לרוחב (הצידה) C - קשת בין שני תתי עצים או בין שני עצים שונים ביער ה-DFS.
- המאפיין קשתות $(u,v)$ אלו הוא שהן קשתות מקודקוד מסומן $u$ לקודקוד $v$ שכבר הסתיים החיפוש ממנו (ולכן הוא כבר מסומן ומחוץ למחסנית).
- ההבדל הוא שקשת $(u,v)$ היא קשת קדימה אם חיפוש ה-DFS גילה קודם את קודקוד $u$ לפני הגילוי של קודוקד $v$.
- לעומת זאת שקשת $(u,v)$ היא קשת לרוחב אם בחיפוש ה-DFS קודם גילינו את $v$ ובהמשך ה-DFS גילינו את $u$ .

- קשת בעץ T - קשת $(u,v)$ היא קשת בעץ אם היא קשת מקודקוד $u$ שמסומן ועדיין לא סיימנו את החיפוש ממנו (כלומר $u$ נמצא במחסנית), לקודקוד $v$ שמגלים אותו בפעם הראשונה (עדיין לא היה במחסנית).
- קשת אחורה B - קשת $(u,v)$ היא קשת אחורה אם היא קשת מקודקוד $u$ לאב קדמון שלו $v$ בעץ ה-DFS (כלומר מקודקוד מסומן $u$ שעדיין לא סיימנו איתו לקודקוד מסומן $v$ שעדיין לא סיימנו איתו).

**למה:** בביצוע DFS על גרף לא מכוון G, כל קשת של G היא או קשת בעץ או קשת אחורה.

**לסיכום - סוגי קשתות ביער ה-DFS:**

גרף לא מכוון:

כל קשת שמופיעה בגרף או:

1. תופיע בתור קשת בעץ.
2. היא פונה אחורה בעץ.

גרף מכוון:

כל קשת שמופיעה בגרף או:

1. תופיע בתור קשת בעץ.
2. קשת קדימה.
3. קשת אחורה.
4. קשת לרוחב (הצידה).

# 10. מיון טופולוגי:

גרף מכוון G שבו הקודקודים מייצגים משימות או פעילויות והקשתות מכוונות מייצגות יחס קדימויות בין המשימות נקרא רשת AOV.

מיון טופולוגי של גרף מכוון ללא מעגלים הוא סידור לינארי של כל קודקודי הגרף כך שאם בגרף יש קשת מכוונת (i,j) אזי i מופיע לפני j בסידור זה..

יש 3 סוגי אלגוריתמים של מיון טופולוגי : שימוש בתור, שימוש במחסנית , באמצעות סריקת גרף לעומק (DFS)

ל-3 האלגוריתמים אותם סיבוכיות.

נביא אחד מהאלגוריתמים: נראה את האלגוריתם שמשתמש בתור.

**<u>אלגוריתם:</u>**

נשתמש בשרותי טיפוס נתון תור (queue), בו נשמור את הקודקודים שאין להם מקדימים.

1. איתור קודקודי הגרף שאין להם מקדימים וצירופם לתור Q.
2. כל עוד התור (Q) לא ריק יש לבצע:

2.1 הוצא איבר מראש התור וצרף אותו לרשימה המייצגת סדר טופולוגי.

2.2 מחק בגרף את הקודקוד, שיצא זה עתה מהתור, ובנוסף למחוק את כל הקשתות היוצאות מקודקוד זה.

2.3 בגרף החדש, שהתקבל בצעד 2.2 , כל קדקוד שאין לו קודקודים מקדימים, יש לצרף אותם לתור Q.

3. בשלב זה התור ריק והאלגוריתם הסתיים.

**<u>סיבוכיות:</u> O(V+E)**

# 11. רק"ח (רכיב קשירות חזק)

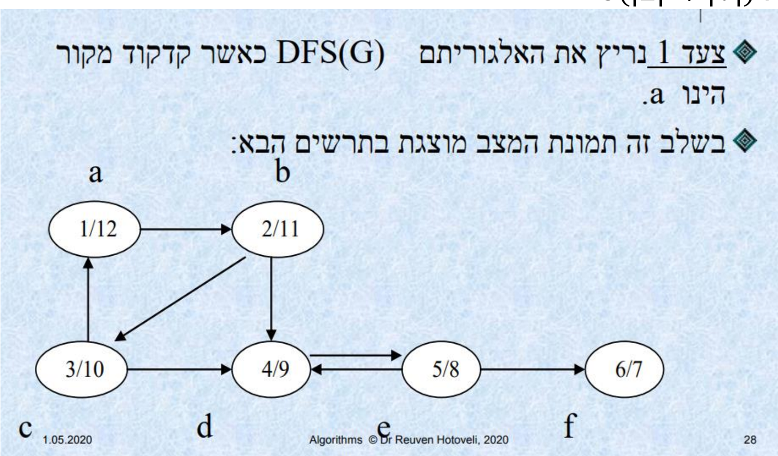
- רכיב קשיר חזק – קבוצה מקס' של קודקודים בגרף ,כך שבין כל שני קודקודים בקבוצה זו קיים מסלול מכוון .
- קודקוד בודד ללא שכנים ייקרא רק"ח
- בגרף לא מכוון G , DFS(G) מוצא את רכיבי הקשירות .
- בגרף לא מכוון, BFS(G,a) מוצא את רכיב הקשירות ש- a נמצא בו .

אלגוריתם למציאת רכיבי קשירות חזקה בגרפים מכוונים (SCC):

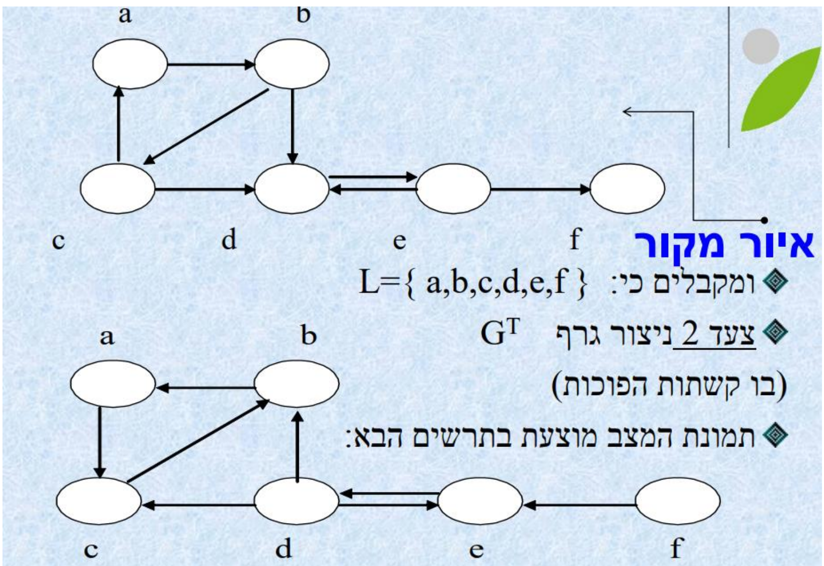
1. מריצים את DFS על גרף G ויוצרים רשימת קדקודים L אשר ממוינת בסדר יורד לפי זמני סיום הטיפול בהם.
2. הופכים את הגרף G ומקבלים Gt $E = \{ (u,v) \,|(v,u)\; E\}$ במילים אחרות הופכים את הקשתות כל קשת שהייתה מ u ל v , נשנה ל v ל u
3. מריצים DFS על הגרף Gt החדש , כך שהלולאה עוברת על קודקודי הגרף לפי הסדר , כפי שנקבע בצעד 1 ברשימה L

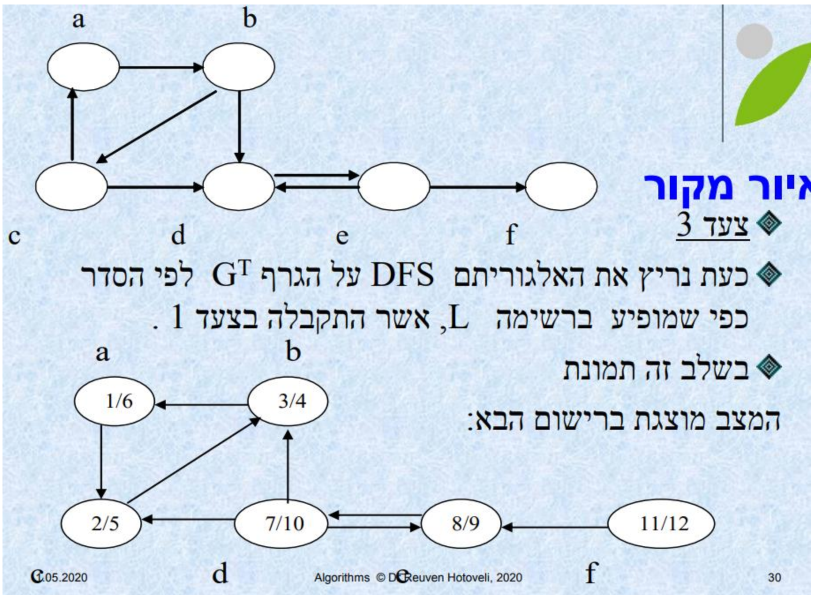
סיבוכיות : $O(|E| + |V|)$ .

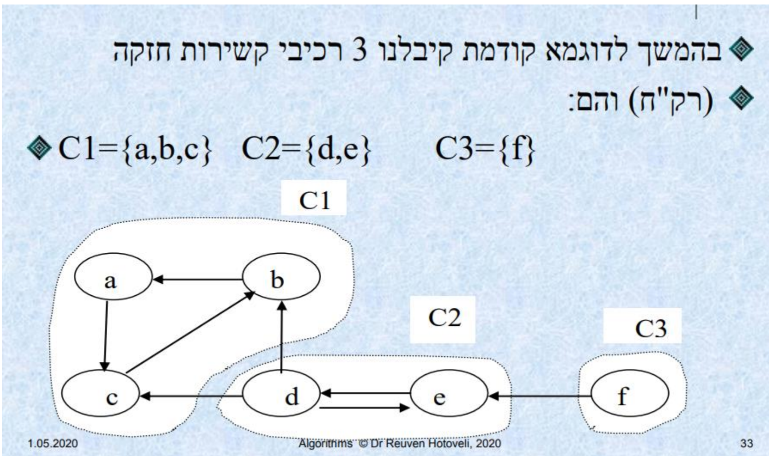
דוגמא :




אפשר לשים לב כי הכיוונים של החיצים השתנו , יצרנו גרף מכוון חדש הפוך לגרף הקודם





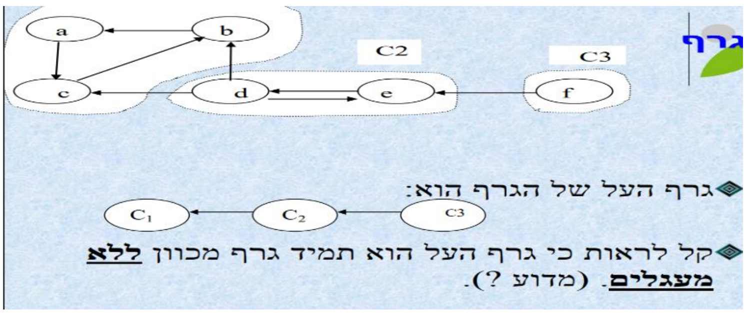




**גרף על :** G הינו גרף על אם מתקיים :

$$E = \{(C_i, C_j) / (x,y) \text{ וקיימת קשת } y \in C_j \text{ and } x \in C_i \text{ קיימים}\}$$

$$V = \{C_1, C_2, \ldots\ldots\ldots C_m \ / \ m \geq 1 \text{ and } C_i \text{ רקח}\}$$

במילים שלנו , רק אם הוא מורכב מרקח"ים ולכל זוג קודקודים אשר נמצאים ברקח"ים שונים יהיה קשת שתחבר בין רכיבי הרקח .



לדוגמא : נרצה אלגוריתם אשר מוצא את כל הקודקודים בגרף מסוים אליהם אפשר להגיע במסלול מכוון מכל קודקודי הגרף

פתרון : נשתמש בגרף על .

הסבר:

1. נמצא את הרקח"ים עם אלגוריתם SCC
2. נבנה גרף על
3. נעבור על גרף העל ונספור את מספר הצמתים להם יש דרגת יציאה 0

מסקנה : אם קיים יותר מקודקוד אחד שיש לו דרגת יציאה 0 , הקבוצה המבוקשת לא קיימת.

# 12. <u>בעיית המסלולים הקצרים (דייקסטרה)</u>

<u>הבעיה :</u> למצוא מסלול בעל עלות נמוכה ביותר בגרף ממושקל.

<u>**אלגוריתם דייקסטרה דרישות לביצוע :**</u>

- א . נניח שהגרף מיוצג בעזרת מטריצת סמיכות כדלקמן:
- אם קיימת קשת ( i , j )
- אם i=j
- אחרת

$$a_{ij} = \begin{cases} E_{ij} \\ 0 \\ \infty \end{cases}$$

- ב.נניח שקודקוד מקור הינו קודקוד 0.
- ג. קבוצת הקודקודים תחולק לשתי קבוצות:
- אחת <u>הקבוצה</u> P(לכבוד "קבוע") אשר תכיל קודקודים, כך שאורך המסלול המינימלי מקודקוד מקור עד אליהם הינו <u>קבוע</u> ולא ישתנה בעתיד עד סוף האלגוריתם.
- והשניה <u>הקבוצה T</u> (לכבוד "זמניים" temporaries ) אשר תכיל קודקודים כך שאורך המסלול המינימלי מקודקוד מקור עד אליהם הינו <u>זמני</u> ועשוי להשתנות בעתיד עד סוף האלגוריתם.
- ד. עבור כל קודקוד u ברשת נרצה לשמור את אורך המסלול הקצר ביותר מקודקוד מקור 0 לקודקוד u ואותו נסמן על ידי d[u] לכבוד המילה ( distance).
- בתחילת האלגוריתם לכל קודקוד u , פרט לקודקוד מקור, נבצע את ההשמה הבאה: $d[u] \leftarrow \infty$ ו- $d[0] \leftarrow 0$ מאחר שאורך המסלול הקצר ביותר מקודקוד מקור לעצמו הינו 0.
- ה. כמו כן נרצה לשמור לגבי כל קודקוד מידע על זהות הקודקוד הקודם לו ( "הורה" שלו) במסלול הקצר.
- לאור זאת נשתמש במערך Pa , כך שלכל קודקוד u ברשת Pa[u] יציין קודקוד ממנו הגענו ל - u .

**סיבוכיות :** בעזרת מערך- O(v^2)

בעזרת ערמת מינימום – O(e*log(v))

**האלגוריתם :**

- INITIALIZE-SINGLE-SOURCE(G,s)
- 1. for each vertex $v \in V$ do
- 1.1 d[v]← ∞
- 1.2 $\pi[v] \leftarrow NIL$
- 2. d[s]← 0
- כאשר $\pi[v]$ הוא קודקוד "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← $\emptyset$
- 2. Q← V {Q - הוא תור של קודקודים}
- 3. While Q $\neq \emptyset$ 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)

# 13. מסלולים קצרים לפי בלמן פורד

**דרישות :**

- נתון גרף G= (V,E) *מכוון* עם פונקצית משקל W:E→R.
- aij=
- אם קיימת קשת מכוונת (i,j) — $E_{ij}$ ((i,j) משקל על הקשת)
- אם i=j — 0
- אחרת — ∞
- ב. d[v] הוא משקל המסלול הקצר ביותר מקדקוד מקור 1 לקדקוד v .
- ג. המשקל על הקשת יכול להיות חיובי או שלילי
- בגרף אין מעגלים בעלי משקל שלילי.
- ד. עבור כל קדקוד v , נגדיר: $d^{(m)}[v]$ כמסלול הקצר ביותר מקדקוד מקור 1 לקדקוד v , <u>בהנחה שהמסלול מכיל לא יותר מ- m קשתות.</u>
- לכן $d^{(1)}[1] = 0$ ולכל צומת $j \neq 1$ $d^{(1)}[j] = a_{1j}$

**האלגוריתם:** **סיבוכיות :** O(VE)

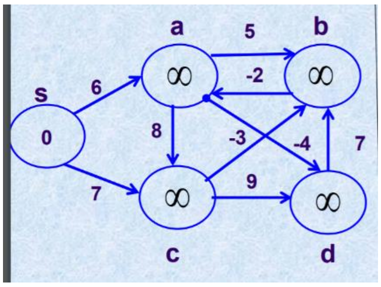
- Bellman-Ford(G,w,s)
- 1. INITIALIZE-SINGLE-SOURCE(G,s)
- 2. for i ← 1 to |V|-1 do
-     for each edge $(u,v) \in E$
      do Relax(u,v,w)
- 3. for each edge $(u,v) \in E$ do
-     if d[v] > d[u] + w(u,v)
-         then return FALSE
- 4. return TRUE

- 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 .

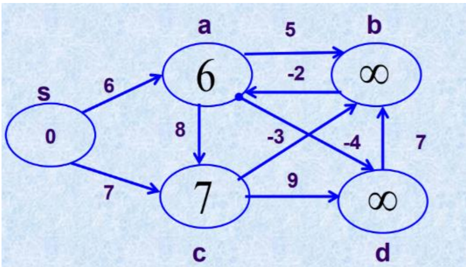
- טכניקת ההקלה (relaxation) :

- 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 $\pi[v] \leftarrow u$

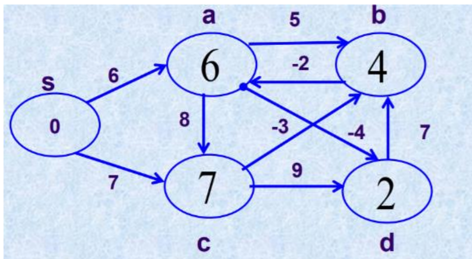
המצב ההתחלתי



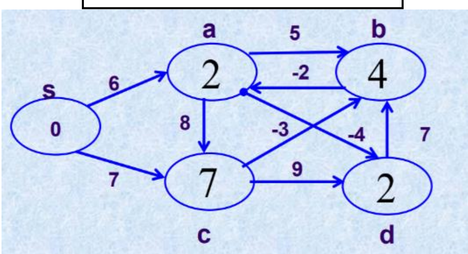

איטרציה ראשונה m=1



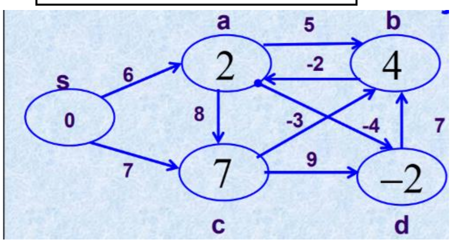

איטרציה שנייה m=2




איטרציה שלישית m=3




איטרציה רביעית m=4




- קיבלנו $d^{(4)}[v]$ כמסלול הקצר ביותר מקדקוד מקור s לקדקוד v , <u>כאשר שהמסלול מכיל לא יותר מ-4 קשתות.</u>
- <u>מסלולים אלו מק"ב-ים בעבור הגרף הנתון.</u>

# 14. מסלולים קצרים בגמ"ל

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

◈ **Dag _Shortest _Path (G,w,s)**

◈ גמ"ל (DAG) – הוא גרף מכוון ללא מעגלים.

◈ האלגוריתם מתחיל במיון טופולוגי של הגמ"ל , כדי לכפות על הקודקודים סדר ליניארי.

◈ אם קיים מסלול מ-u אל v , אזי u יבוא לפני v בסדר הטופולוגי.

**האלגוריתם** :

◈ **צעד 1**: מיון טופולוגי של הקודקודים של G .

◈ **צעד 2 :** INITIALIZE_SINGLE_SOURCE(G,s)

◈ **צעד 3** : 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

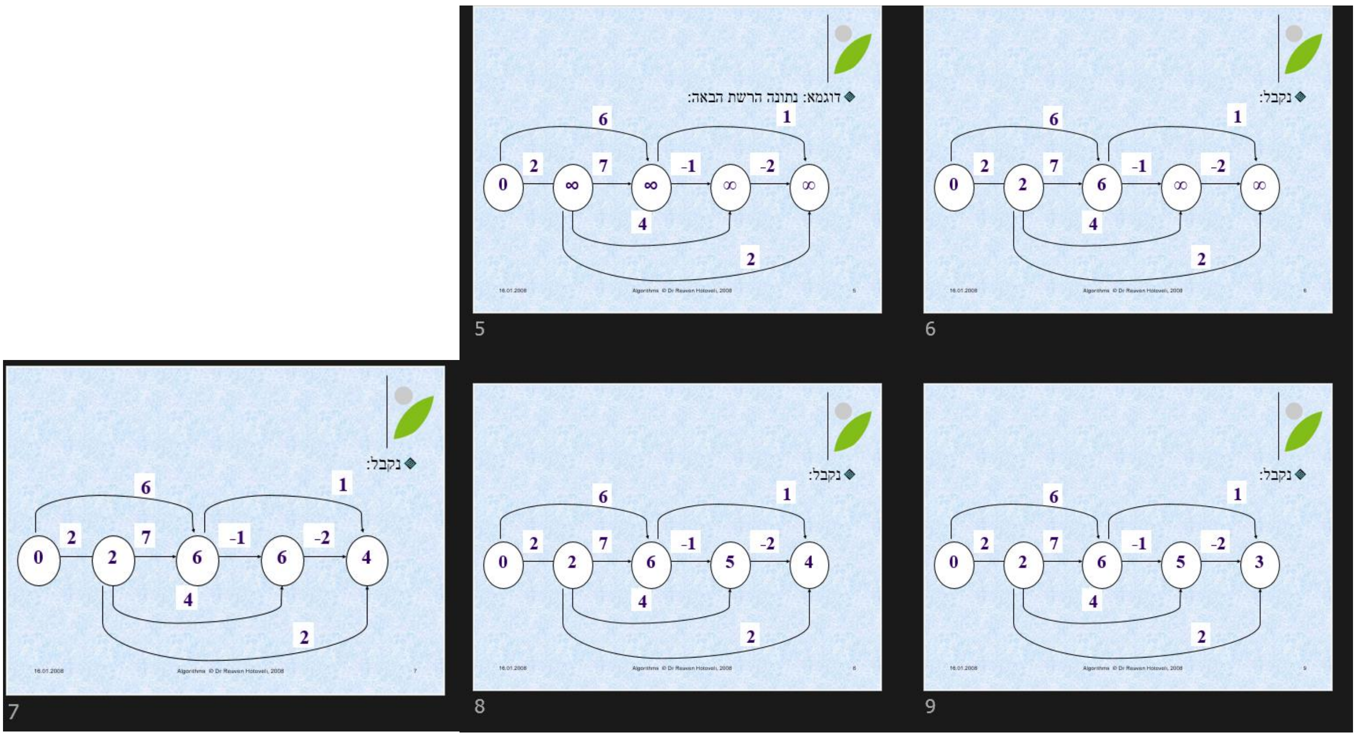
◈ { $d[v] \leftarrow d[u] + w(u,v)$ and $P[v] \leftarrow u$ }

◈ הערות:

◈ 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 הכולל: $\Theta(E+V)$ (בדוק!)

**דוגמא :**



# 15. <u>מסלול אוילר</u>

**<u>הגדרות</u>:**

**מסלול אוילר -** הוא מסלול בגרף העובר דרך כל קשת בגרף בדיוק פעם אחת.

**מעגל אוילר -** הוא מעגל בגרף  העובר דרך כל קשת בגרף בדיוק פעם אחת.

**משפט 1 -** יהי גרף G=(V,E) קשיר אם כל דרגות קודקודיו זוגיות , אז קיים בגרף מעגל פשוט.

**משפט 2 -** בגרף לא מכוון קיים מעגל אוילר אם ורק אם לכל קודקוד בגרף דרגה זוגית.

**משפט 3 -** בגרף קשיר לא מכוון קיים מסלול אוילר אם ורק אם יש בו 0 או 2 קודקודים בעלי  דרגה אי זוגית.

**משפט -** נתון G גרף ובו בדיוק  2k קודקודים בעלי דרגה אי זוגית k=<1, ומספר טבעי

אפשר לצייר את G ב k -משיכות קולמוס ואי אפשר לצייר את G בפחות מ k -משיכות קולמוס

## <u>מסקנות:</u>

- **גרף קשיר** לא מכוון הוא **אוילריאני** כלומר יש בו מעגל אוילרי אם ורק אם דרגות כל הצמתים זוגיות
- **גרף קשיר** לא מכוון הוא **חצי אוילריאני** כלומר יש בו מסלול ולא מעגל אוילרי אם ורק אם דרגות כל הצמתים זוגיות פרט לשניים צומת התחלה וסיום.
- **גרף מכוון** הינו גרף **אוילריאני** אם עבור כל קודקוד בגרף דרגת הכניסה שווה לדרגת היציאה.
- **גרף מכוון** הינו גרף **חצי אוילריאני** אם בעבור קודקוד מקור (התחלה) דרגת היציאה גדולה באחת מדרגת הכניסה ובעבור קודקוד היעד (סיום) דרגת הכניסה גדולה באחת מדרגת היציאה.
עבור כל קודקוד אחר, דרגת הכניסה שווה לדרגת היציאה.

| סוג הגרף / סוג המסלול | מעגלי | מסלול שאינו מעגלי |
|---|---|---|
| **לא מכוון** | לכל הקודקודים בגרף דרגה זוגית | לכל הקודקודים פרט לשניים בגרף דרגה זוגית |
| **מכוון** | לכל קודקוד בגרף $v \in V$ $d_{in}(v) = d_{out}(v)$ | עבור קודקוד מקור (s) קודקוד יעד (t) ועבור כל קודקוד v בגרף השונה מ-s ו-t מתקיים: $d_{in}(v) = d_{out}(v)$ $d_{out}(s) = d_{in}(s)+1,$ $d_{out}(t) = d_{in}(t)-1$ |

- $d_{in}(v)$ מציין דרגת הכניסה של הקודקוד V בגרף מכוון.
- $d_{out}(v)$ מציין דרגת היציאה של הקודקוד V בגרף מכוון
- $d(v)$ מציין דרגתו של קודקוד של V בגרף מכוון/לא מכוון.

**<u>דרך למציאת לולאת/מסלול אוילר:</u>**

להתחיל מקודקוד מסוים K (בעל דרגה אי זוגית, אם קיים קודקוד כזה בגרף) ולמחוק כל קשת לאחר שעוברים דרכה, כך ממשיכים ובכל פעם לאחר שעוברים דרך הקשת מוחקים אותה.

צריך ליזכור שאסור לעבור על קשת שמחיקתה תיצור 2 גרפים מנותקים.

הסריקה הזאת חייבת להסתיים בקודקוד אחר שדרגתו אי זוגית (אם קיים קודקוד כזה בכלל)

יתכן שהתחלנו את הסריקה מקודקוד V (שדרגתו אי זוגית, אם קיים קודקוד כזה) וסיימנו את הסריקה בקודקוד W (גם כן בעל דרגה אי זוגית אם קיים קודקוד כזה),אך ייתכן שתוך כדי הסריקה לא עברנו דרך כל הקשתות שבגרף.

מאחר שהגרף קשיר, חייב להיות קודקוד בגרף שממנו ניתן לסרוק ,לגלות ולעבור דרך הקשתות שעדיין לא נמחקו מהגרף.

לאור זאת נצטרך לבדוק לאיזה קודקוד יש עדיין קשתות )שעדיין לא נמחקו( ונתחיל את ביצוע המשימה מקודקוד זה.

את התת המסלול הזה (שחייב להיות מעגל) נוסיף למסלול הראשון , האלגוריתם יסתיים כאשר כל הקשתות בגרף נמחקו E=0 .

**<u>סיבוכיות:</u>** O(|E|)

# 16. גרף המילטוני

מעגל (מסלול) המילטון מוגדר כמעגל (מסלול) פשוט העובר דרך כל קודקודי גרף נתון G=(E,V) פעם אחת בדיוק.

למרות שבעיית אפיון הגרפים המילטוניים מזכירה את זו של גרפים אוליריאנים הבעיות הקשורות לגרפים המילטוניים הן בעלי סיבוכיות אקספוננציאליות ואינן פשוטות לפתרון.

למעגל המילטון חשיבות רבה בתורת הרשתות , אחת הבעיות שבה משתמשים במעגל המילטוני הנה הסוכן הנוסע (traveling sales problem (tsp

## הגדרות:

**מעגל המילטון -** הוא מעגל המכיל את כל הקודקודים ,כל קודקוד פעם אחת בלבד.

**מסלול המילטון -** הוא מסלול המכיל את כל הקודקודים, כל קודקוד פעם אחת בלבד.

גרף המכיל מעגל המילטון נקרא גרף המילטוני.

גרף המכיל מסלול המילטון נקרא גרף חצי המילטוני.

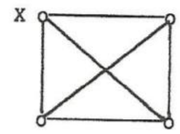
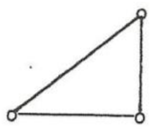
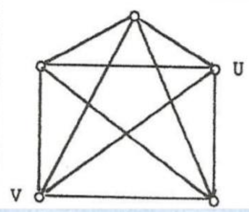
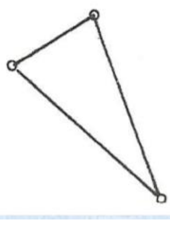
## טענות:

1. בכל גרף שלם $k_n$ n≥3 קיים מעגל המילטון.
2. בגרף דו צדדי שלם $k_{m,n}$ **יש מעגל המילטון אם ורק אם** m=n≥2
3. עבור m שונה מ- n **גרף דו צדדי שלם חצי המילטוני אם ורק אם ההפרש בין** m ל- n הוא 1.

תנאי הכרחי לגרף המילטוני:

אם גרף פשוט וקשיר המילטוני אזי מספר הרכיבים ב-*G המתקבל מ-G על ידי השמטת k>0 קודקודים (וכל הקשתות שלהם) **אינו** עולה על k.

טבלה 1 הסרת קדקודים מגרף קשיר נתון ומספר הרכיבים המתקבלים מכך

| הגרף G | קדק' להסרה | הגרף הנותר *G | מס' רכיבים |
|---|---|---|---|
|   | X |  | 1 |
|   | U , V |  | 1 |

**משפט דירק (Dirac) –** נתון G גרף פשוט בעל n>=3 קודקודים הדרגה של כל קודקוד היא לפחות n/2 אז הגרף G המילטוני.

לשים לב שהתנאי הוא תנאי מספיק ע"מ ש G-יהיה המילטוני אך אין הוא הכרחי.

# 17. עץ פורש

**הגדרה:** $T=(V,E_T)$ ללא מעגלים (בקיצור 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.
מכוון ש $E_T$ מחברת את כל קודקודי הגרף ולא מכילה מעגלים נראה בהמשך כי T יוצר עץ. עץ כזה נקרא <u>עץ פורש</u> (Spanning Tree).

| |
|---|
| הגדרה:<br>עץ פורש ( Spanning Tree) של גרף G קשיר הוא עץ אשר מכיל את כל קודקודי הגרף G , ועץ זה הוא גם כן גרף קשיר. |

| |
|---|
| הגדרה:<br>עץ פורש מינימלי ( Minimum Spanning Tree )-הינו עץ פורש וסכום המשקולות הרשומות בצד הקשתות של העץ הינו מינימלי. |

הגדרה : נתון גרף בלתי מכוון קשיר $G=(V,E)$ .
קשת e בגרף G נקראת <u>קשת מפרידה</u> אם הסרתה בגרף G תיצור גרף שבו שני רכיבים.

## 1. <u>האלגוריתם של קרוסקל :</u>

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

6.3.1<u>אלגוריתם של קרוסקל</u>
<u>צעד 1</u>
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 *)
<u>צעד 2</u>
2.1 מצא קשת a השייכת ל-$(E-E_T)$ כך שלכל קשת b אשר שייכת ל -$(E-E_T)$ מתקיים :

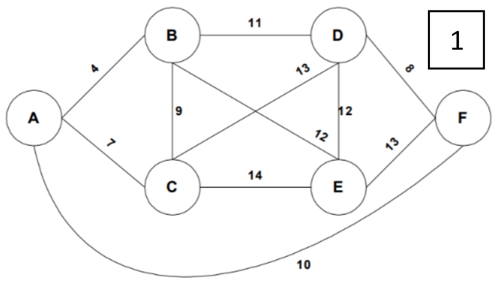
$$w(a)=\min_{b} w(b)$$

(*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 ".
<u>צעד 3</u>
3.1 $k\leftarrow k+1$
3.2 חזור לצעד 2

לכן סיבוכיות זמן הריצה של האלגוריתם קרוסקל הוא $O\left(|V|^2\cdot|E|\right)$.

## **דוגמא:**

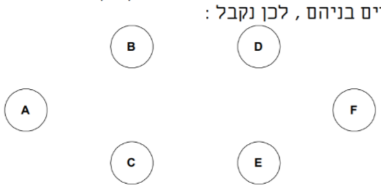
1



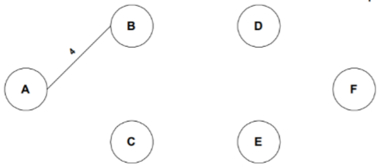

בגרף זה 6 קודקודים , לכן בעץ הפורש יהיו 5 קשתות (לפי משפט 6.2.3) .

2

בהתחלה מתחילים עם יער שבו כל הקודקודים אינם מחוברים בניהם , לכן נקבל :

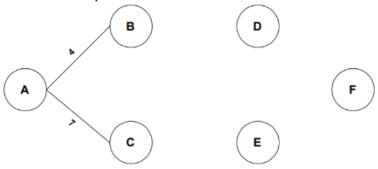



באיטרציה ראשונה מוצאים את הקשת (A,B) שהינה בעלת המשקל הקטן ביותר.לכן בתום האיטרציה הראשונה היער יראה כך :

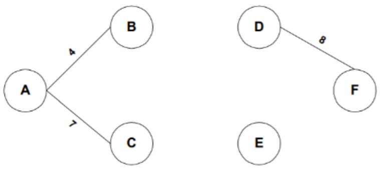



3

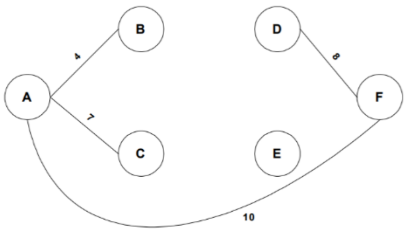
באיטרציה שניה מוצאים את הקשת (A,C) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
לכן בתום האיטרציה השניה היער יראה כך :




באיטרציה השלישית מוצאים את הקשת (D,F) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
לכן בתום האיטרציה השלישית היער יראה כך :

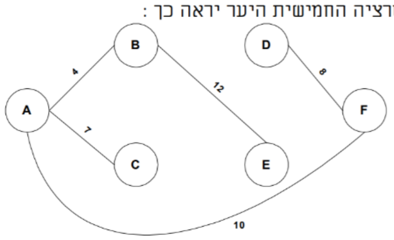



באיטרציה רביעית מוצאים את הקשת (B,C) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
אך לא נוסיף ליער את הקשת (B,C) , מכיוון שהוספתה יצור מעגל.לכן עתה נמצא את הקשת (A,F) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.לכן בתום האיטרציה הרביעית היער יראה כך :




4

באיטרציה חמישית מוצאים את הקשת (B,D) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
אך לא נוסיף ליער את הקשת (B,D) , מיכוון שהוספתה יצור מעגל.לכן עתה נמצא את הקשת (B,E) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.לכן בתום האיטרציה החמישית היער יראה כך :




כמובטח לאחר חמש איטרציות קיבלנו עץ פורש מינימלי.

## 2. אלגוריתם פרים

אלגוריתם פרים הוא אלגוריתם חמדני למציאת עץ פורש מינימום.

1. נתחיל עם צומת רנדומלי $u$ ונבדוק את כל הקשתות המחוברות לצומת זה.
2. עבור כל צומת v שמחובר בקשת לצומת u נגדיר ההורה של v הוא u (v.parent=u)
3. כעת נבדוק מי הצומת השכן x של u עם הקשת הכי זולה ונוסיף את הצומת לעץ
4. עבור כל השכנים נבדוק האם יש קשת זולה יותר מאשר זו המחוברת .
   אם כן , x ההורה החדש שלו ואם לא , לא נעשה כלום .
   כלומר -לכל צומת שלא בעץ , שמרנו מה הקשת הזולה ביותר ממנו אל העץ .
5. נחזור על שלבים 3-4 עד שעברנו על כל הצמתים בעץ .
6. נחזיר את העץ

<u>אלגוריתם מילולי</u>

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

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

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

<u>צעד 4</u> 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)).

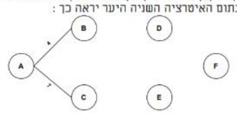
המטרה היא לקבל עץ פורש מינימלי

סיבוכיות: $O(E \log E)$

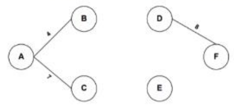
דוגמא

שלב 1

באיטרציה שניה מוצאים את הקשת (A,C) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
לכן בתום האיטרציה השניה היער יראה כך :



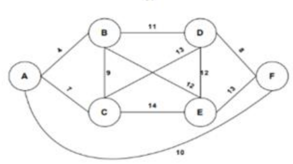
באיטרציה השלישית מוצאים את הקשת (D,F) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
לכן בתום האיטרציה השלישית היער יראה כך :



באיטרציה רביעית מוצאים את הקשת (B,C) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
אך <u>לא נוסיף</u> ליער את הקשת (B,C) , <u>מכיוון שהוספתה יצור מעגל</u>.לכן עתה נמצא את הקשת (A,F) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.לכן בתום האיטרציה הרביעית היער יראה כך :

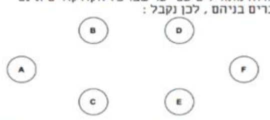
שלב 2

14

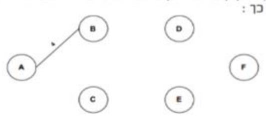


בגרף זה 6 קודקודים , לכן בעץ הפורש יהיו 5 קשתות (לפי משפט 6.2.3) .

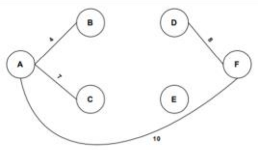
בהתחלה מתחילים עם יער שבו כל הקודקודים אינם מחוברים בניהם , לכן נקבל :



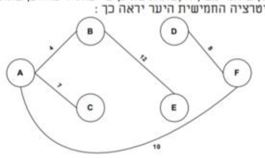
באיטרציה ראשונה מוצאים את הקשת (A,B) שהינה בעלת המשקל הקטן ביותר.לכן בתום האיטרציה הראשונה היער יראה כך :



שלב 3



באיטרציה חמישית מוצאים את הקשת (B,D) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.
אך <u>לא נוסיף</u> ליער את הקשת (B,D) , <u>מיכוון שהוספתה יצור מעגל</u>.לכן עתה נמצא את הקשת (B,E) שהינה בעלת המשקל הקטן ביותר מבין הקשתות שאינן שייכות ליער.לכן בתום האיטרציה החמישית היער יראה כך :



כמובטח לאחר חמש איטרציות קיבלנו עץ פורש מינימלי.

שים לב לכך , שעץ פורש מינימלי אינו בהכרח אחד ויחיד.
לדוגמא , באיטרציה החמישית אפשרי לבחור את הקשת

| אלגוריתם | דברים שצריך לדעת \תוספות | שימוש\איך עובד\תוצאה | סיבוכיות |
|---|---|---|---|
| BFS<br>חיפוש לרוחב | 1. כל תת מסלול של מק"ב הוא מק"ב<br>2. אם u נגיש מ קוד' מקור אז האלגוריתם מבקר בצומת u<br>3. מסדר לפי השכבה לדוגמא קוד' מקור יהיה בשכבה 0 | • שאלת נגישות מקוד' מקור<br>• מציאת מסלולים קצרים ביותר מקוד' מקור<br>• מציאת רכיבים קשירים בגרף לא מכוון | $O(V+E)$<br>לינארית |
| DFS<br>חיפוש לעומק | בשונה מ bfs בעת מציאת שכן של צומת, לא נמשיך לשאר שכניו אלא נמשיך לפתח את אותו שכן. רק כאשר יתקע יחזור אחורה וימשיך לשאר. | 1. מציאת רכיבי קשירות בגרף לא מכוון<br>2. מציאת רכיבי קשירות חזקה בגרף מכוון<br>3. בדיקת שייכות שני צמתים לאותו רכיב קשירות<br>4. האם הגרף (מכוון או לא ) מכיל מעגל<br>5. סיווג קשתות(אחורית , קדמית, חוצה ) | $O(V+E)$<br>לינארית |
| **SCC**<br>גרפים מכוונים | מציאת רכיבי קשירות חזקים ע"י DFS בגרפים מכוונים. | מקבלים יער פרישה , שכל עץ בו מייצג רק"ח .<br>יעזור לנו להגיע בסוף לגרף על. | $O(V+E)$<br>לינארית |
| **מיון טופולוגי**<br>בגרפים מכוונים ללא מעגלים | 1. מיון טופולוגי הוא סידור של קבוצת איברים שקיימת ביניהם תלות כך שאף איבר לא יופיע לפני איבר בו הוא תלוי<br>2. אם יש לנו מעגל לא יכול להיות לנו מיון טופולוגי. | 1. יצירת סידור לינארי של קודקודים בגרף .<br>2. יכולים להתקבל מספר סידורים טופולוגים<br>3. סדר כרונולוגי של משימות | $O(V+E)$<br>לינארית |
| **דייקסטרה** –<br>גרפים מכוונים ללא משקלים שליליים | **INITIALIZE-SINGLE-SOURCE**-איתחול מטריצת סמיכויות (הורים , משקלים וכו)<br>**RELAX**(u,v,w)- בודקת האם המשקל החדש יותר טוב ממה שהיה ומשנה אם כן . | 1. מציאת המסלול הקל ביותר מנקודה בגרף ליעד בגרף ממושקל.<br>2. מציאת עץ המסלולים הקצרים . | עם מערך :<br>$O(V^2)$<br>עם ערימה :<br>$O(E\log V)$ |
| **בלמן -פורד**<br>גרפים מכוונים ללא מעגלים שליליים | הערות : המסלול הקצר ביותר חייב להיות פשוט ובעל אורך של n-1 כי אם לא יש מעגל .<br>משתמש באותן פונקן כמו דייקסטרה . | מציאת המסלול הממושקל הקל והקצר ביותר .<br>יכול להתמודד עם משקלים שליליים .<br>בדיקה האם קיים מעגל שלילי בגרף . | $O(VE)$ |
| **DAG**– גרף מכוון ללא מעגלים | ממיין טופולוגית, לפי סדר המיון, לכל צומת *u* נבצע *relax(u,v)*<br>לכל קשת שיוצאת ממנו.<br>יכול לעבוד על קשתות שליליות . | מציאת מסלולים קצרים .<br>מציאת מסלול קריטי(ארוך ביותר)ע"י כמה שינויים קטנים . | $O(V+E)$ |

| | | | |
|---|---|---|---|
| קרוסקל-<br>גרף ממושקל, קשיר, לא מכוון. | מתמקד בקשתות<br>באלגוריתם זה בוחרים תחילה את הקשת בעלת משקל מינימלי ,<br>לאחר מכן את הקשת בעלת משקל מינימלי מהקשתות הנותרות וככה הלאה , רק שלא ניצור מעגל . | מציאת עץ פורש מינימלי .<br>יצירת גרף חדש עפ"י קרוסקל .<br>האלגוריתם עובד בצורה חמדנית מכיוון שבכל פעם הוא בוחר את הצלע המינימלית . | $O(E\log E)$ |
| פרים | מתמקד בקודקודים.<br>אלגו' שמשלב את השיטה החמדנית +שמירת ההורה והבנת לפי התור\מערך של קודקודים כמה נשאר לסוף התהליך . | מציאת עץ פורש מינימלי. | $O(E\log V)$ |