

מיון טופולוגי

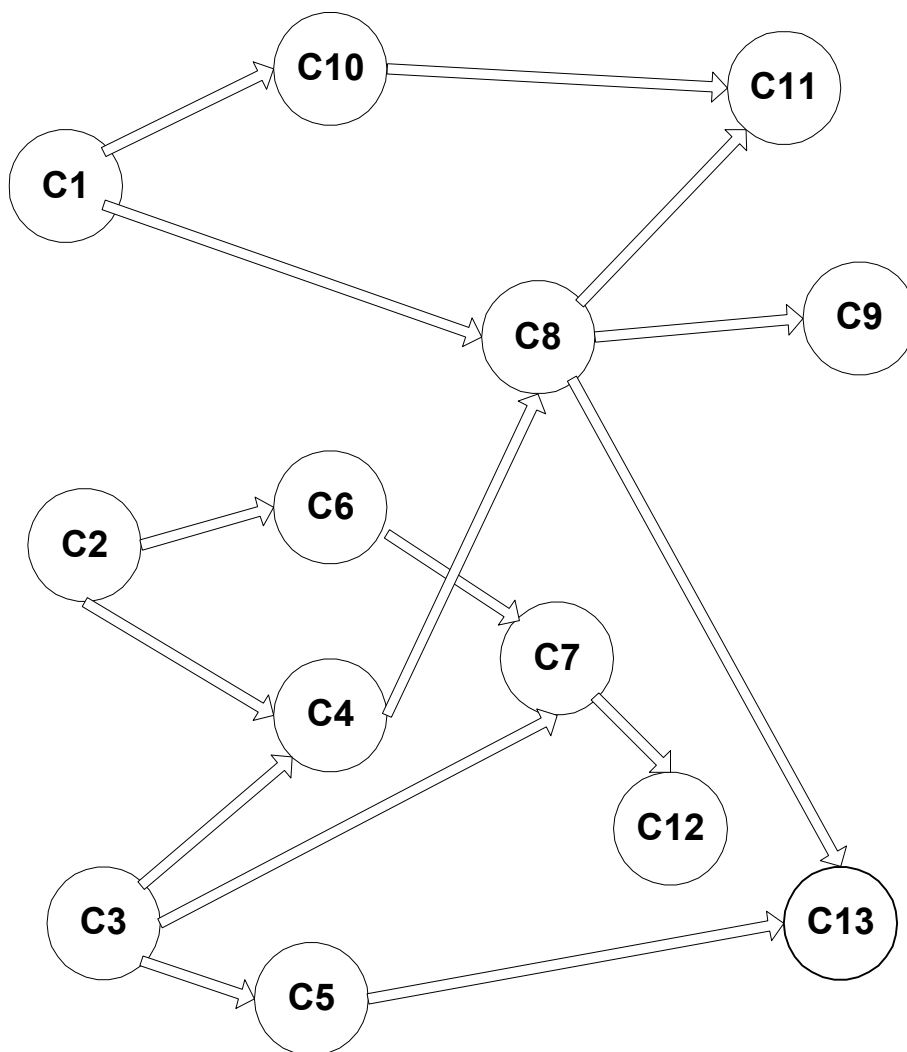
כל הבעיות חוץ מהפשוטות ביותר ניתנות לחלוקה לתתי בעיות שנקראות פעילויות. דוגמא: סטודנט שלומד לקראת תואר במדעי המחשב חייב ללמוד מספר קורסים. במקרה זה הבעיה היא לקבל את התואר, והפעילויות הן הקורסים שהוא צריך לקחת. בטבלה הבאה מפורטים כל הקורסים שיש ללמוד כדי לקבל תואר במדעי המחשב באוניברסיטה מסוימת.

קוד הקורס	שם הקורס	דרישות קדם
C1	מבוא למדעי המחשב	אין
C2	חדו"א 1	אין
C3	אלגברה לינארית 1	אין
C4	חדו"א 2	C2, C3
C5	אלגברה לינארית 2	C3
C6	מתמטיקה דיסקרטית	C2
C7	מבוא להסתברות	C6, C3
C8	מבני נתונים	C1, C4
C9	אלגוריתמיקה מתקדמת	C8
C10	מבנה מחשבים, ותכנות מערכות	C1
C11	מערכות הפעלה	C10, C8
C12	סטטיסטיקה	C7
C13	גרפיקה ממוחשבת	C5, C8

דרישות קדם של קורס מסוים, שהקוד שלו C_i , פירושה שהסטודנט חייב ללמוד את **כל** הקורסים הרשומים בדרישות קדם של הקורס C_i לפני שהוא מתחיל ללמוד את הקורס C_i . סטודנט יכול ללמוד חלק מהקורסים ללא תלות בקורסים אחרים בזמן שלחלק מהקורסים יש מקדימים (דרישות קדם). כך למשל סטודנט יכול ללמוד אלגברה לינארית 1 ($C3$) ללא תלות בקורסים אחרים ואילו הוא יכול ללמוד את הקורס מבני נתונים ($C8$) רק לאחר שיסיים את לימודיו בשני קורסים:

מבוא למדעי המחשב ($C1$) ו חדו"א 2 ($C4$) גם יחד.

את יחס הקדימויות ניתן לייצג על ידי **גרף מכוון** שבו הקודקודים מייצגים קורסים וקשתות מכוונות מייצגות את סדר הקדימות. קשת מכוונת (u, v) פירושה שיש ללמוד את הקורס u לפני שמתחילים ללמוד את הקורס v . להלן גרף המתאר את יחס הקדימויות בין הקורסים השונים שתוארו לעיל.





הגדרה: גרף מכוון G שבו הקודקודים מייצגים משימות או פעילויות והקשתות מכוונות מייצגות יחס קדימויות בין המשימות, נקרא רשת AOV (Activity On Vertex).

הגדרה: קודקוד u ברשת AOV נקרא קודקוד מקדים של הקודקוד v אם ורק אם קיים מסלול מכוון מ- u ל- v .

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

הגדרה: מיון טופולוגי (Topological Sort) 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.



בהמשך לדוגמא שראינו בפתיח הפרק הזה להלן שתי אפשרויות למיון טופולוגי, מבין האפשרויות הרבות לסידור טופולוגי, ואלו הן:

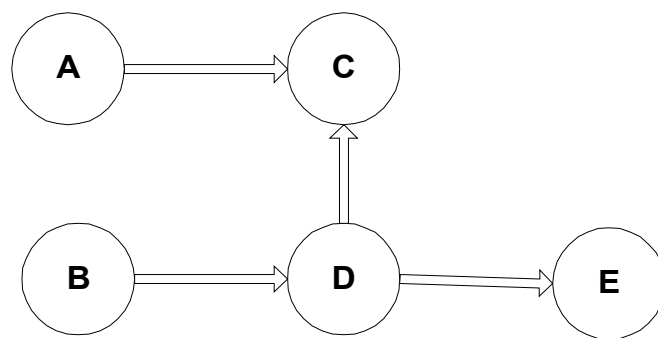
1. $C1, C2, C3, C10, C6, C4, C5, C8, C11, C9, C7, C12, C13$
2. $C1, C10, C3, C2, C4, C8, C9, C11, C6, C5, C13, C7, C12$

הערה: אם בגרף יש לפחות מעגל אחד אזי לא ניתן לסדר את קודקודי הגרף בשום סדר לינארי, שיכול לייצג את המיון הטופולוגי. בדוגמא הנדונה המעגל הוא שקורס כלשהו מהווה דרישת קדם לאותו קורס. כלומר קודקוד u הוא קודקוד מקדים של u . כלומר תלמיד חייב לסיים ללמוד את הקורס u לפני שהוא מתחיל ללמוד את אותו הקורס u . ברור שהדבר בלתי אפשרי ולכן מיון טופולוגי מוגדר בגרפים מכוונים ללא מעגלים.

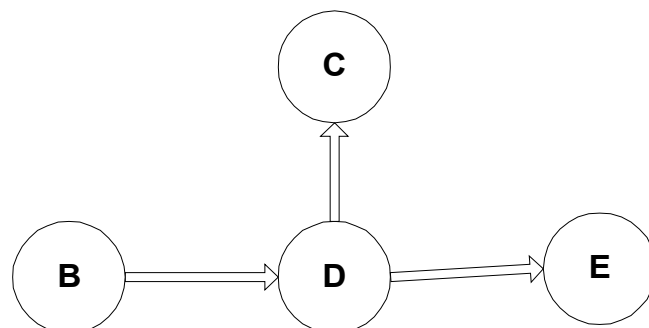


אם מדיניות האוניברסיטה שבדוגמא היא שבכל סמסטר סטודנט יכול ללמוד רק קורס אחד, אז הוא חייב ללמוד אותם על פי הסדר הטופולוגי. להלן גרסאות שונות של אלגוריתמים הפשוטים הבאים, אשר יוצרים סידור לינארי של קודקודים בגרף מכוון ללא מעגלים כאשר סידור לינארי זה מייצג את המיון הטופולוגי.

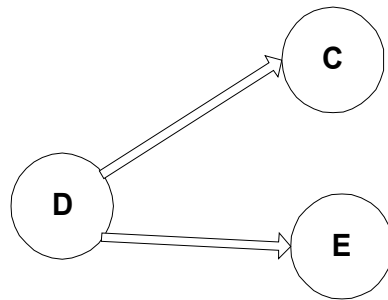
בטרם ננסח את האלגוריתם נתבונן על רשת AOV הבאה:



נראה מיון טופולוגי של גרף זה כסידור של קודקודיו לאורך קו אופקי כך שכל הקשתות פונות משמאל לימין. לקודקודים A ו B אין קודקודים מקדימים, לכן נבחר אחד מהם באופן מקרי ונצרף אותו לרשימה המייצגת מיון טופולוגי. נניח שבחרנו בקודקוד A מבין הקודקודים B, A שאין להם קודקודים מקדימים. לאחר שהפעילות A בוצעה אנו מסירים את קודקוד A מהגרף ואת כל הקשתות היוצאות מקודקוד זה בגרף ולכן עתה תמונת הרשת הינה:



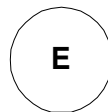
בשלב זה לקודקוד B אין קודקוד מקדים (והוא היחיד). עתה נבצע את הפעילות B. נסיר את הקודקוד B מהגרף ואת כל הקשתות היוצאות מקודקוד זה בגרף ולכן עתה תמונת הרשת הינה:



בשלב זה לקודקוד D אין קודקוד מקדים (והוא היחיד). עתה נבצע את הפעילות D ותמונת הרשת הינה:



בשלב זה לשני הקודקודים C ו E אין קודקודים מקדימים לכן **נבחר באופן מקרי אחד מהם** ונצרף אותו לרשימה המייצגת מיון טופולוגי. נניח שבחרנו קודקוד C מבין הקודקודים C ו E לאחר שהפעילות C בוצעה תמונת הרשת הינה:



בשלב זה לקודקוד E אין קודקוד מקדים (והוא היחיד). עתה נבצע את הפעילות E והרשת הינה גרף ריק. בזה מסתיים התהליך שקובע את המיון הטופולוגי. סופית המיון הטופולוגי שהתקבל הוא:

A B D C E →

יתכן שבשלב מסוים יהיו יותר מקודקוד אחד שאין להם קודקודים מקדימים, אך מאחר שבכל שלב ניתן לבצע רק פעילות אחת בלבד, אז בוחרים באופן מקרי קודקוד אחד מבין הקודקודים שאין להם קודקודים מקדימים 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. באלגוריתם זה נשתמש
בשרותי טיפוס נתון תור (queue), בו נשמור את הקודקודים
שאינן להם מקדימים.

להלן אלגוריתם למיון טופולוגי:

1. - איתור קודקודי הגרף שאינן 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 הקודקוד, שיצא זה עתה מהתור,
ובנוסף למחוק את כל הקשתות היוצאות מקודקוד זה.

2.3 - בגרף החדש, שהתקבל בצעד 2.2, כל קודקוד שאינן

לו קודקודים מקדימים, יש לצרף אותם לתור Q.

3. - בשלב זה התור ריק והאלגוריתם מסתיים.


עתה לפנינו 3 שאלות עיקריות ואלו הן:

1. איך מבטאים את סדר הקדימות בין הפעילויות השונות
בקלט.

2. כיצד נייצג את הגרף ?

3. כיצד אפשר לקבוע לאיזה קודקוד אין קודקוד מקדים ?

עתה נענה על השאלות האלו.

סדר הקדימות מתבטא בקלט באמצעות זוגות סדורים. כל שורה בקלט מכילה שמות של שתי פעילויות, כאשר הראשונה צריכה להתבצע לפני השנייה. 

הכיצד מיוצג הגרף ?

אם מספר הקודקודים בגרף ידוע מראש אז יש אפשרות לייצג את הגרף באמצעות מטריצת סמיכות או רשימות סמיכות, אחרת נשתמש בייצוג המקושר של גרף. הבא נניח שמספר הקודקודים בגרף ידוע מראש והגרף מיוצג באמצעות רשימות סמיכות.

עתה נענה על השאלה האחרונה:

כיצד אפשר לקבוע לאיזה קודקוד אין קודקוד מקדים ?

לשם כך נחזיק מערך בשם indegree כך ש $\text{indegree}[j]$ מכיל את מספר הקודקודים הקודמים לקודקוד j .
אם $\text{indegree}[j]$ שווה לאפס, פירוש הדבר שלקודקוד j אין קודקודים מקדימים וניתן לצרף אותו לתור Q , מכיוון שבתור Q נמצאים קודקודים שאין להם מקדימים והם אמורים לצאת מהתור אחד אחרי השני ולהצטרף לרשימה המייצגת מיון טופולוגי. בכל פעם שקודקוד j יוצא מהתור יש לעבור על רשימת הסמיכות של הקשתות היוצאות מקודקוד j ולהפחית $\text{indegree}[k]$ של כל צומת K הסמוך ל- j ב-1.

לסיכום, להלן האלגוריתם למיון טופולוגי בעזרת **התור**:

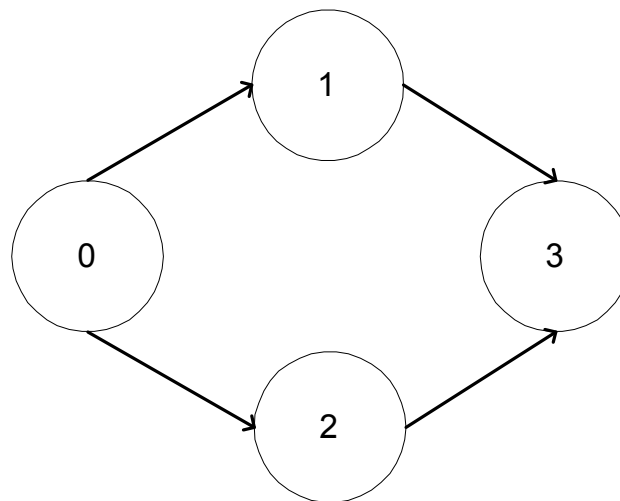
```
TP_Sort(Graph G)
{
    Queue Q;
    //Find indegree of each vertex/
    for each  $v \in V$  do
        indegree[v] = 0;
    for each  $(u,v) \in E$  do
        indegree[v]  $\leftarrow$  indegree[v] + 1;
        /*איתור כל הקודקודים*/
        /*שאין להם מקדימים*/
        /*ולצרף אותם לתור Q*/
    for each  $v \in V$  do
        if indegree[v] = 0 then
            Insert(Q,v);
    while  $Q \neq \emptyset$  do
         $v = \text{Delete}(Q)$ ;
        print v;
        for each  $u \in \text{Adj}[v]$  do
            indegree[u]  $\leftarrow$  indegree[u] - 1;
            if indegree[u] = 0 then
                Insert(Q,u);
    for each  $v \in V$ 
        if indegree[v]  $\neq$  0 then
            print "CYCLE";
}
```

ניזכר כי הפעולות הבסיסיות המוגדרות על התור Q הן:

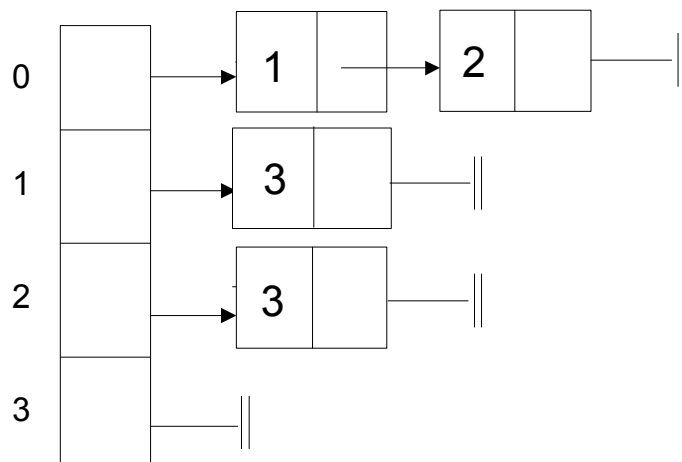
Insert (Q,v) – פעולה זו מוסיפה את האיבר v לעורף התור Q.

Delete (Q) – פעולה זו מסירה ומחזירה את האיבר שבחזית התור Q.

נדגים את הרצת האלגוריתם הנדון על הגרף הבא:



(*)נדגיש כי : הגרף מיוצג באופן הבא:



(*)תמונת המערך indegree , כאשר $\text{indegree}[i]$ מציין את דרגת הכניסה של הקודקוד i , הינה:

indegree	0	1	1	2
אינדקס	0	1	2	3

(*)בתור Q יש רק קודקוד אחד, קודקוד 0, שאין לו קודקוד מקדים.

Q : 0

(*) מיון טופולוגי הינו רשימה ריקה.
שלב 1

- מוציאים את 0 מהתור.
- ניגשים למערך של הרשימות לאינדקס 0 ומגלים את כל הקודקודים הסמוכים ל-0, שהם 1 ו-2.
- עתה יש לבטל את הקשתות (0,1) ו (0,2) ולהפחית $\text{indegree}[1]$ ו- $\text{indegree}[2]$ ב-1.
- תוך כדי הפחתה של ה- $\text{indegree}[1]$ או $\text{indegree}[2]$ ב-1, אם ערכו שווה לאפס אז אין לו קודקוד מקדים ומצרפים אותו לתור Q.
- במקרה שלנו תמונת המצב הינה:

indegree	0	0	0	2
אינדקס	0	1	2	3

Q : $\frac{\quad 2 \quad 1}{\longrightarrow}$

שלב 2

- מוציאים את 1 מהתור.
- ניגשים למערך של רשימות לאינדקס 1 ומגלים שהקודקוד 3 הוא קודקוד סמוך ל-1. לכן יש לבטל את הקשת (1,3) ולהפחית $\text{indegree}[3]$ ב-1.

עתה נקבל:

indegree	0	0	0	1
אינדקס	0	1	2	3

Q : $\frac{\quad 2}{\longrightarrow}$

ברור כי הקודקוד 3 לא הצטרף לתור Q מכיוון שעדיין יש לו קודקוד מקדים (רואים גם ש $\text{indegree}[3]$ שונה מאפס).

שלב 3

- מוציאים את 2 מהתור.
- ניגשים למערך של רשימות לאינדקס 2 ומגלים שהקודקוד 3 הוא קודקוד סמוך ל- 2. לכן יש לבטל את הקשת (2,3) ולהפחית $\text{indegree}[3]$ ב- 1.

עתה נקבל:

indegree	0	0	0	0
אִינְדֵּגְרֵס	0	1	2	3

Q: 3

עתה ברור מדוע הקודקוד 3 הצטרף לתור, מכיוון שבשלב זה אין לו קודקוד מקדים, כי $\text{indegree}[3]$ שווה לאפס.

שלב אחרון

- מוציאים את 3 מהתור.
- ניגשים למערך של רשימות לאינדקס 3 ומגלים שאין קודקודים סמוכים ל- 3 ואין במי לטפל.

עתה תמונת המצב הינה:

indegree	0	0	0	0
אינדקס	0	1	2	3

והתור Q ריק.
והמיון הטופולוגי הוא:

A horizontal number line with arrows at both ends. The line is marked with the integers 0, 1, 2, and 3 from left to right.

טענה: סיבוכיות זמן הריצה של האלגוריתם למיון טופולוגי היא $O(|V| + |E|)$.

הוכחה:

כידוע עבור גרף G שהינו גרף מכוון הסכום של אורכי כל הרשימות הסמיכות הוא $|E|$.

- הלולאה הראשונה – (for) מתבצעת בזמן $O(|V|)$.

- הלולאה השנייה – (for) מתבצעת בזמן $O(|E|)$.

- הלולאה השלישית – (for) מתבצעת בזמן $O(|V|)$.

הגישה לאינדקס מסויים במערך לוקחת זמן $O(1)$ וגם

הפעולה- הוספת איבר לעורף התור לוקחת זמן $O(1)$, ולכן

- מספר הצעדים בלולאה הרביעית (while) הוא כסכום של

אורכי כל הרשימות הסמוכות שהינו $O(|E|)$.

- הלולאה החמישית (for) מתבצעת בזמן $O(|V|)$.

- סופית סיבוכיות זמן הריצה של האלגוריתם הנדון היא:

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

הערה:

במקום התור Q ניתן להיעזר במחסנית אשר מאחסנת בכל שלב של האלגוריתם את קודקודי הגרף שאין להם קודקודים מקדימים.

גירסה שניה של האלגוריתם למיון טופולוגי תוך שימוש במחסנית

הפעם מבנה הנתונים המוצע הינו:

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

1. count אשר משמש לשתי 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 כשטח לאחסון המחסנית. בהמשך נבהיר את זה

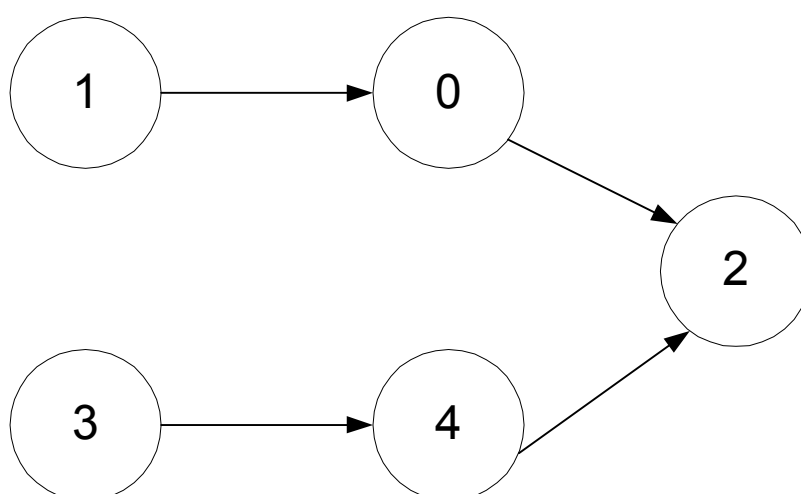


בפרוטרוט.

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 רשימה מקושרת.

כך למשל עבור הגרף הבא:



מערך הרשימות נראה כך:

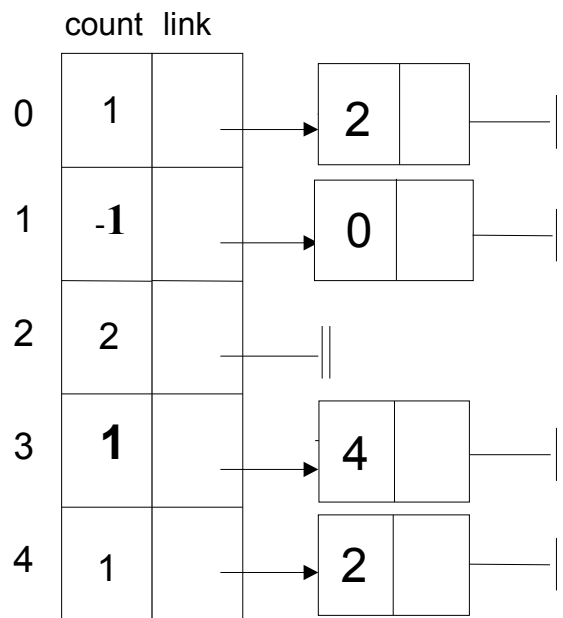
	count	link	kodkod	next
0	1	→	2	
1	0	→	0	
2	2			
3	0	→	4	
4	1	→	2	

בהתחלה המחסנית ריקה.

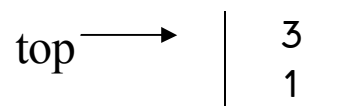
שלב 1

מאתרים את כל הקודקודים שאין להם קודקודים מקדימים
(במקרה שלנו את הקודקודים 1 ו-3) ומכניסים אותם
למחסנית.

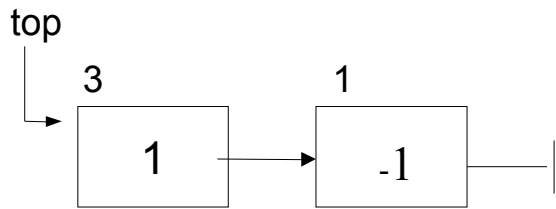
תמונת המצב הינה:



והמשתנה TOP מכיל ערך 3, מכיוון שהקודקוד 3 נמצא
בראש המחסנית. כמו כן הערך של `count[3]` שווה ל-1,
מכיוון שהקודקוד הבא שנמצא במחסנית הוא קודקוד 1.
תמונה זו מתארת את המחסנית הבאה:



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



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

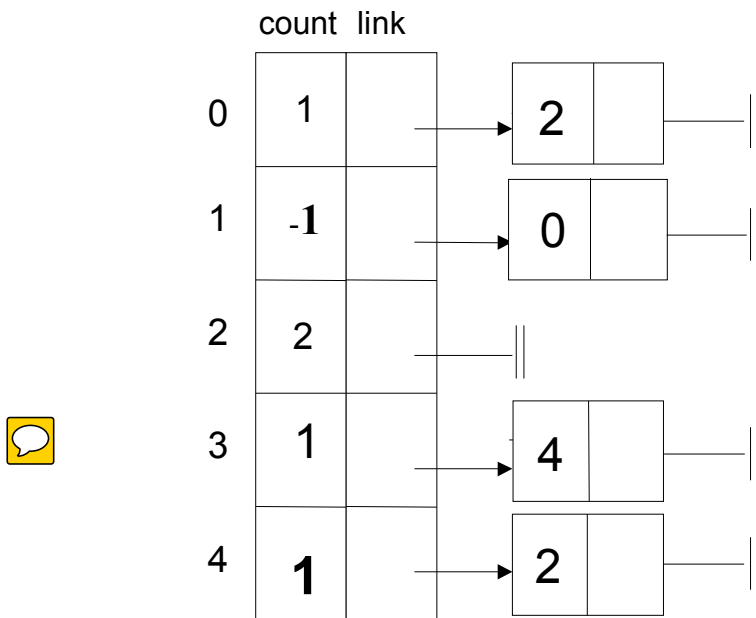
```

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

```

שלב 2

- מוציאים את הקודקוד 3 מהמחסנית.
 - ניגשים למערך של רשימות לאינדקס 3 ומגלים שהקודקוד הסמוך ל- 3 הוא קודקוד 4.
 עתה יש לבטל את הקשת (3,4) ולהפחית $\text{count}[4]$ ב- 1 ואז נקבל כי $\text{count}[4]$ שווה ל- 0.
 אי לכך הקודקוד 4 יכנס למחסנית מכיוון שאין לו קודקוד מקדים ולכן תמונת המצב הינה:



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



שים לב ! ערכו של count[4] הוא 1, מיכיוון שהעוקב של הקודקוד 4 הוא קודקוד 1 ברשימה המייצגת את המחסנית. התהליך חוזר חלילה.

לסיכום להלן אלגוריתם נוסף לבעיית מיון טופולוגי אשר משתמש במחסנית כמפורט לעיל.

TP_Sort2 (Graph G)

```
{ stack s;
  for each  $v \in V$  do count [v] =0;
  for each  $(u,v) \in E$  do count [v] ++;
  TOP = -1;          /* Create Empty Stack (s); */
  for (I=0 ; I<=n-1 ; I++)
    if (count[I] == 0)
      {count[I] = TOP ;
       TOP = I;
      }
  /* עד כה הכנסנו למחסנית את כל הקודקודים */
  /* שאין להם קודקודים מקדימים */
  for (I=0 ; I<=n-1 ; I++)
    {if (TOP == -1) {printf ("!!!מעגל");
                      return ;
                    }

    j=TOP;
    TOP = count[TOP];    /*הוצאת איבר מהמחסנית*/
    printf ( "%c",j);
    p=link[j];          /* מצביע לראש רשימת קודקודים
                          j סמיכים לקודקוד */
    while (p !=0) {k=p → kodkod;
                  count[k] --;
                  if (count[k] == 0)
                    {count[k] = TOP;
                     TOP = k;
                    }
                  p=p → next;
                }
    }
```

באותו אופן ניתן להראות שסיבוכיות זמן הריצה של האלגוריתם למיון טופולוגי בעזרת מחסנית הינה $O(|V| + |E|)$.

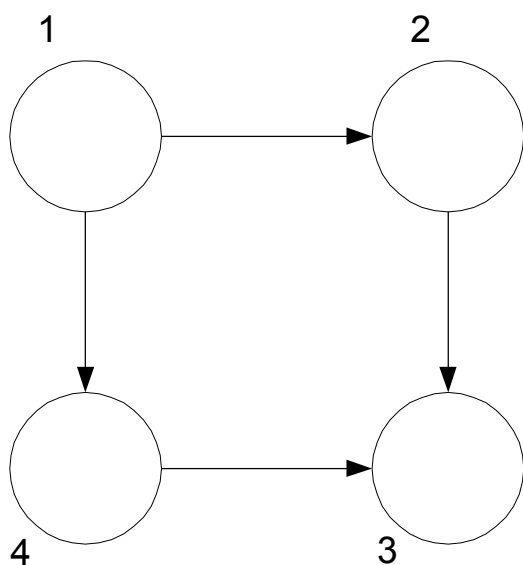
גירסה שלישית של האלגוריתם למיון טופולוגי באמצעות סריקת גרף לעומק (DFS).

TP_Sort (Graph G)

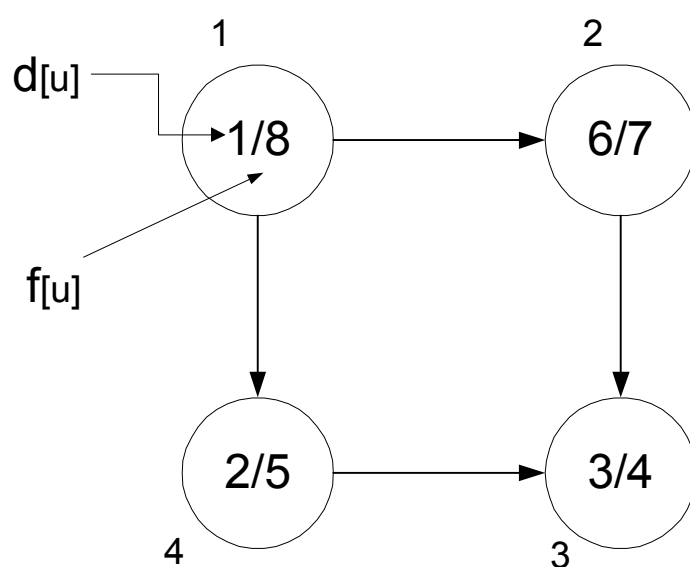
1. קרא לשגרה $DFS(G)$. השיגרה DFS מחשבת את ה- $f[v]$, המציין את מועד הסיום של הקודקוד v , עבור כל קודקוד v . כאשר הטיפול בקודקוד v מסתיים אז נשים אותו בחזית הרשימה המייצגת את המיון הטופולוגי.
2. הדפס את הרשימה המייצגת את המיון הטופולוגי.

הערות:

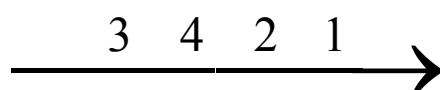
1. קל לראות שסיבוכיות זמן הריצה של האלגוריתם הזה היא כסיבוכיות הריצה של השיגרה DFS שהינה: $O(|V| + |E|)$.
2. נתון הגרף הבא:



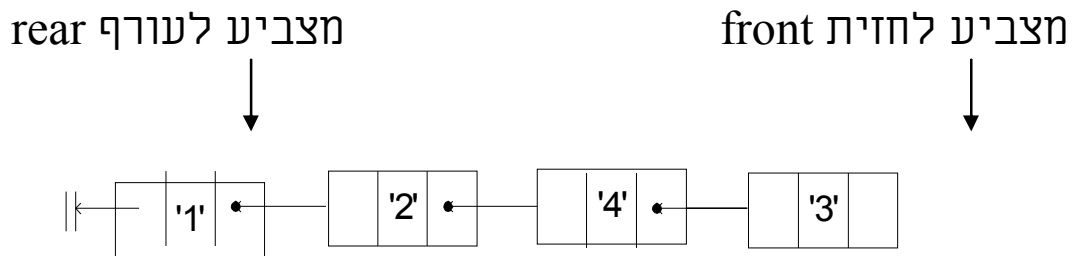
כאשר נפעיל את השיגרה DFS על הגרף הנתון נקבל:



והרשימה שמייצגת את המיון הטופולוגי הינה :



בסוף התהליך תמונת הרשימה המייצגת את המיון הטופולוגי הינה :




זאת מכיוון שבשלב הראשון הטיפול הסתיים בקודקוד 3 ,
 אחריו הטיפול הסתיים בקודקוד 4 , אחריו הטיפול הסתיים
 בקודקוד 2 , ולבסוף הטיפול הסתיים בקודקוד 1 .
 צירוף הקודקוד החדש לעורף הרשימה – פעולה זו דורשת
 זמן שהינו בעל סדר גודל $O(1)$.

תרגילים

1. נתון המידע הבא עבור המשתנים A,B,C,D,E,F :

$$A < B ; C < B ; C < F ; A < F ; D < E ; E < F ; C < D$$

- א. אפשר לייצג מידע זה על ידי גרף מכוון,בדרך הבאה:
 כל משתנה מיוצג על ידי קודקוד בגרף.
 כל יחס $X < Y$ מיוצג על-ידי קשת מכוונת מ-X ל-Y .
 צייר את הגרף המייצג את המידע הנתון.

- ב. (1) כיצד תוכל לדעת, מתוך הסתכלות על הגרף  בלבד,בכמה יחסים משתתף כל משתנה ?

- (2) בזיכרון המחשב מאוחסנת מטריצה $M(10, 10)$,
 המייצגת **גרף מכוון**, המכיל מידע (אוסף יחסים
 $X < Y$) על עשרה משתנים A,B,C...,J.
 המטרה לכתוב שגרה המוצאת את המשתנה,
 שמשתתף במספר מקסימלי
 של יחסים (אם יש יותר ממשתנה אחד כזה,
 השגרה תמצא את כולם).

להלן אלגוריתם שהוצע עבור הבעיה :

1. איפוס מערך שורות – Shurot { גודלו (10) Shurot }.
2. איפוס מערך עמודות – Amudot { גודלו (10) Amudot }.
- 3.

```
for ( i = 0 ; i<n , i++)  
for ( j = 0 ; j<n , j++)  
Shurot[i] +=M[i][j] ;  
Amudot[j] +=M[i][j] ;
```

4.	(1)	.
5.	(2)	.
6.	(3)	.

באלגוריתם זה חסרים 3 ביטויים, המסומנים בספרות בין סוגריים עגולים. רשום במחברת את מספרי הביטויים החסרים (1)-(3) בלבד, בסדר עולה, וכתוב לצד כל מספר את הביטוי החסר שהוא מייצג.

ג. האם ייתכן שישנו מעגל (מכוון) בגרף כזה ? נמק.

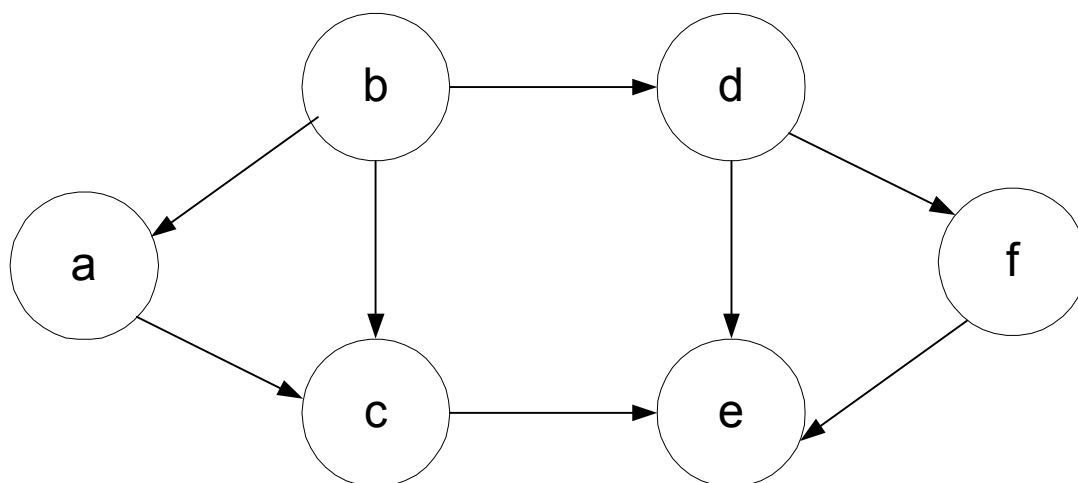
2. הראו ריצה יבשה של אלגוריתם המיון הטופולוגי על הקלט הנתון להלן, בצורת רשימות סמיכות. התשובה צריכה להינתן ע"י הטבלה. הטבלה צריכה לתאר את תוכן התור וערכי indgree של הקודקודים בכניסה לכל איטרציה של לולאת ה-while, וכן את הפלט. מומלץ לצייר את הגרף, אך שימו לב, על הריצה להראות כיצד יפעל האלגוריתם על מבנה הנתונים הספציפי הנתון.

קלט : (שמות הקודקודים ניתנים כאן כאותיות)

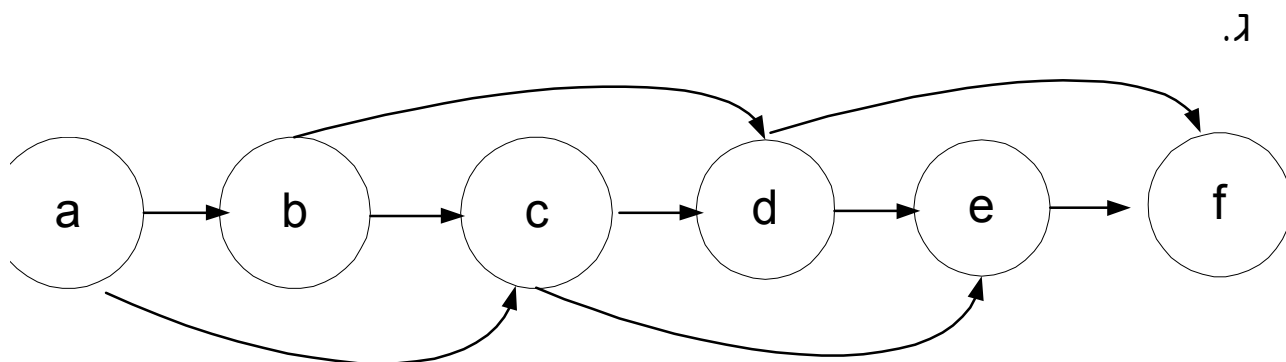
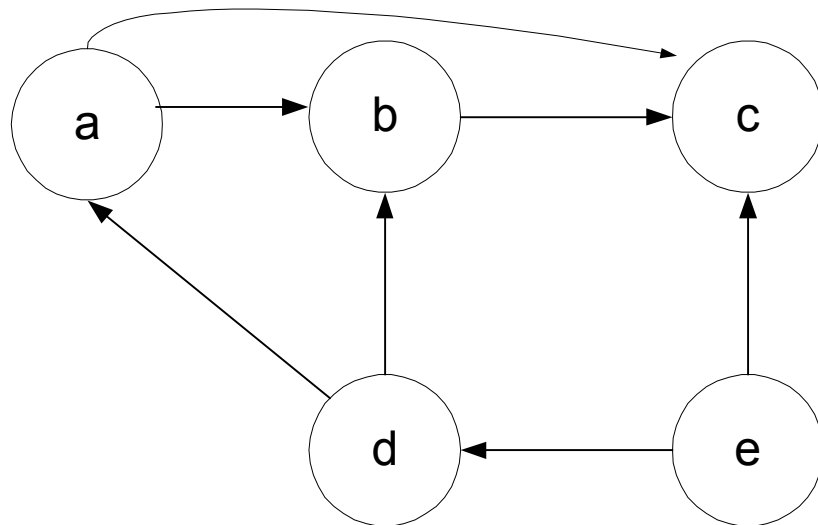
A	D E
B	D
C	D
D	
E	C B
F	D E

3. הראה את סידור הקודקודים הנוצר ע"י האלגוריתם מיון טופולוגי כאשר מריצים אותו על הגרפים הבאים :
אם לא ניתן למצוא את הסידור המבוקש , ציין מה הסיבה לכך.

א.

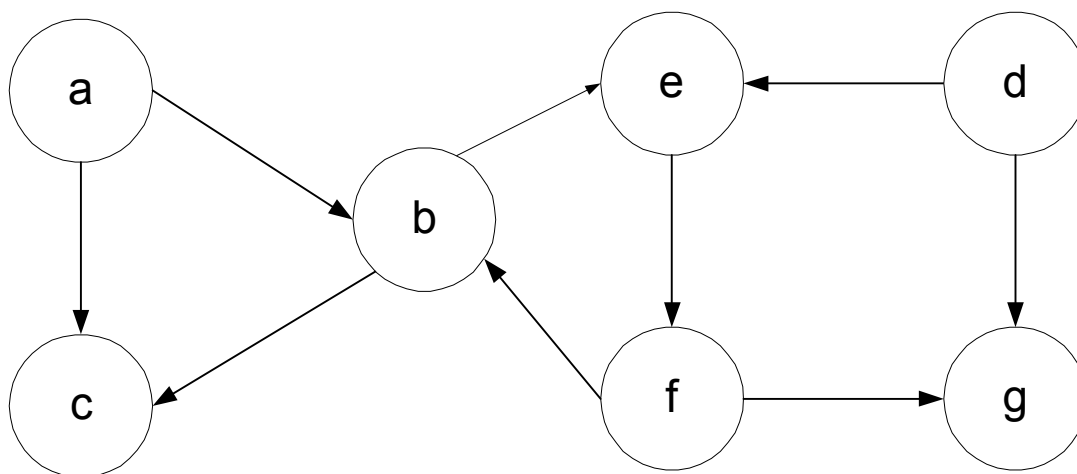


ב.



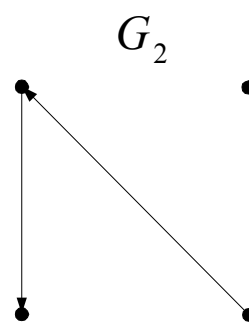
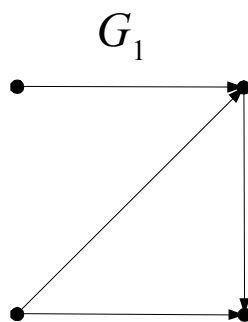
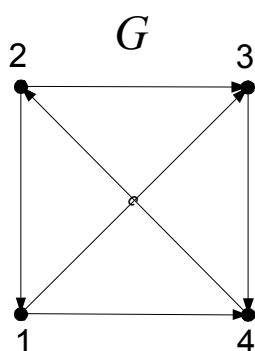
.λ

.T



4. יהי $G = (V, E)$ גרף מכוון פשוט. "פירוק של G לשני גרפים אציקליים" הוא חלוקה של קבוצת הקשתות E לשתי קבוצות E_1 ו- E_2 זרות, כך ששני תת-הגרפים: $G_1 = (V, E_1)$ ו- $G_2 = (V, E_2)$ הינם אציקליים.

דוגמא:



טענה: **אפשר לפרק כך כל גרף מכוון**. הוכיחו את הטענה
 ע"י הצגת אלגוריתם המבצע זאת. הדגימו אותו על קלט
 המתאר את הגרף שלמעלה (כמובן שהפלט לא חייב
 להיות הפירוק שלמעלה, כי יש עוד אפשרויות). הוכיחו
 את נכונותו ונתחו את יעילותו - רצוי להגיע לזמן ליניארי.

5. הוכיחו כי גרף מכוון הוא לא מעגלי אם ורק אם ניתן
 למספר את קודקודי הגרף כך שלכל קשת (i, j)
 מתקיים $i < j$.

6. נתון גרף מכוון. להלן אלגוריתם הממספר את הקודקודים
 של הגרף כך שעבור כל קשת (i, j) מתקיים $i < j$.
 הנח כי הגרף מיוצג באמצעות **מטריצת סמיכות**.

להלן הסימונים שנשתמש בהם:

$v(k)$ - מציין המספור החדש של הקודקוד k .
 $d[j]$ - מציין את דרגת הכניסה של הקודקודים j , כלומר
 מספר הקשתות הנכנסות לקודקוד j .

להלן האלגוריתם:

צעד 1

1.1 לכל קודקוד j , $j = 1 \dots n$,
 בצע:

$$d[j] = \sum_{i=1}^n a_{ij}$$

$$N = \{1, 2, 3, \dots, n\} \quad 1.2$$

$$m = 1 \quad 1.3$$

צעד 2

2.1 מצא קודקוד k , מבין הקודקודים שבקבוצה N ,
כך ש - $d[k] = 0$.

2.2 אם אין k כזה אזי עצור ותודיע שהגרף המכוון מעגלי.

$$v(k) = m \quad 2.3$$

$$\frac{(1)}{\quad} \quad 2.4$$

$$N = \frac{(2)}{\quad} \quad 2.5$$

2.6 אם N היא קבוצה ריקה אז סיים את האלגוריתם.

צעד 3

3.1 עבור כל קודקוד j , כאשר $j \in N$,

בצע: $d[j] = \frac{(3)}{\quad}$

3.2 חזור על צעד 2.

בקוד הנתון חסרים 3 ביטויים המסומנים בספרות בין סוגריים עגולים. השלם את החסר.

- 7*. מסלול המילטון הוא מסלול העובר **בכל הקודקודים**, בכל קודקוד פעם אחת בדיוק. המסלול יכול להתחיל בכל קודקוד שהוא ולהסתיים בכל קודקוד אחר. בגרף מכוון, גם על המסלול להיות מכוון.
- א. מיצאו מסלול המילטון בגרף G המצויר בשאלה 4.
- ב. הוכיחו: אם בגרף מכוון אציקלי יש מסלול המילטון, נניח $(v_1 \ v_2 \ \dots \ v_n)$, אז יש לו מיון טופולוגי יחיד (כלומר: סידור אחד ויחיד של הקודקודים מהווה מיון טופולוגי חוקי).
- ג. אם ידוע שלגרף יש לו מיון טופולוגי יחיד, האם נובע מכך שיש מסלול המילטון? הוכיחו או הראו דוגמא נגדית.
- ד. תנו אלגוריתם יעיל למציאת מסלול המילטון בגרף מכוון אציקלי. האלגוריתם נדרש למצוא מסלול כזה אם ישנו ולהודיע שאין אם אין מסלול כזה.

8*. נתון גרף מכוון. כמה זמן דרוש למציאת כל הצמתים

שמכל אחד מהם ניתן להגיע אל כל צמתי הגרף?
הוכיחו את תשובתכם. 

- *9. א- הוכח כי גרף מכוון G אינו מכיל מעגלים אם ורק אם $DFS(G)$ אינו מניב קשתות אחורה.
ב- האם האלגוריתם מיון טופולוגי יוצר יחס כזה:
 $f[v] < f[u]$ כאשר (u, v) הינה קשת בגרף מכוון ללא מעגלים? נמק את תשובתך.
ג- הוכח כי "האלגוריתם מיון טופולוגי" יוצר מיון טופולוגי של גרף מכוון ללא מעגלים.