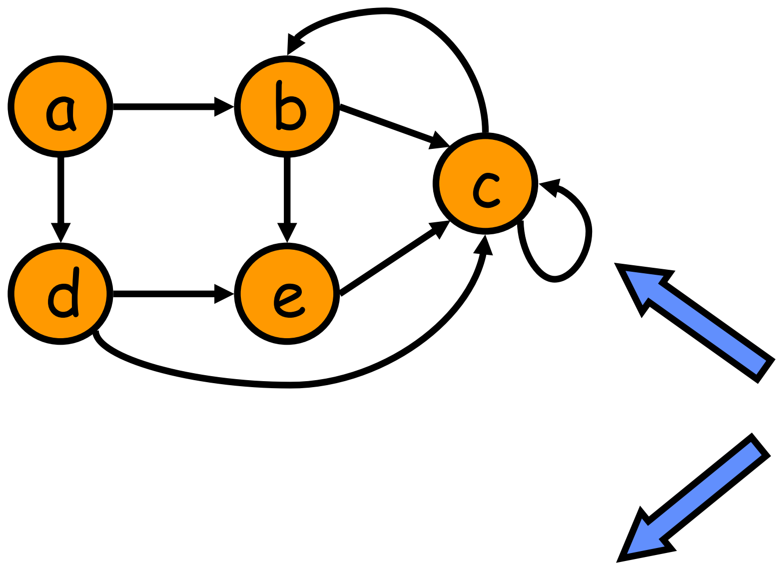
# גרפים מכוונים (Directed Graphs)

גרף מכוון הוא זוג $(V,E)$ המורכב מקבוצת צמתים $V$ וקבוצת קשתות $E \subseteq V \times V$.




חוג עצמי

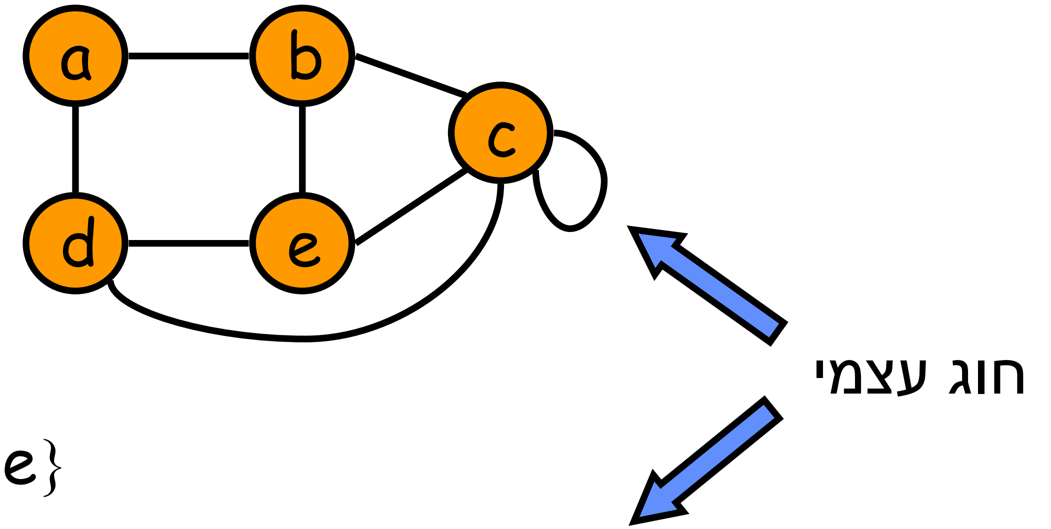
$V = \{a,b,c,d,e\}$

$E = \{(a,b),(a,d),(b,c),(b,e),(c,b),(c,c),(d,c),(d,e),(e,c)\}$

נסמן $|V| = n$ וכן $|E| = m$ . בדוגמא: $n = 5,\ m = 9$.

# גרפים לא-מכוונים (Undirected Graphs)

גרף לא-מכוון הוא זוג (V,E) המורכב מקבוצת צמתים V וקבוצת קשתות E . קשת ב-E היא קבוצה בת שני איברים מתוך V. קשת מסומנת ע"י (i,j) (במקום הסימון המדויק יותר {i,j}).




$V = \{a,b,c,d,e\}$

$E = \{(a,b),(a,d),(b,c),(b,e),(c,c),(d,c),(d,e),(e,c)\}$

נסמן $n = |V|$ וכן $m = |E|$ . בדוגמא: $n = 5$, $m = 8$.

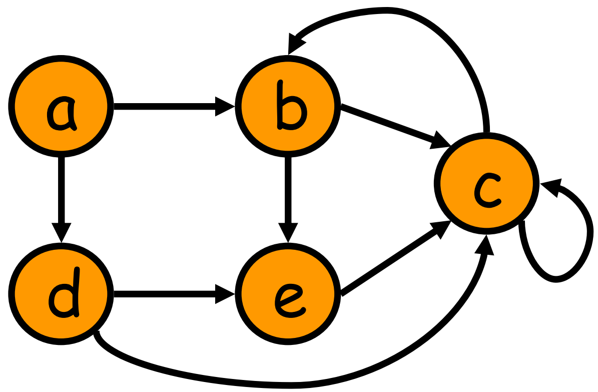
מספר הקשתות m קטן בכל גרף לא מכוון מ- $n^2$ .

# ייצוג גרף מכוון במטריצת סמיכויות (Adjacency Matrix)

נגדיר מטריצה בולאנית $A$ בגודל $n \cdot n$ כך שיתקיים:

$$A[i,j] = 1 \iff (i,j) \in E$$

דוגמא:



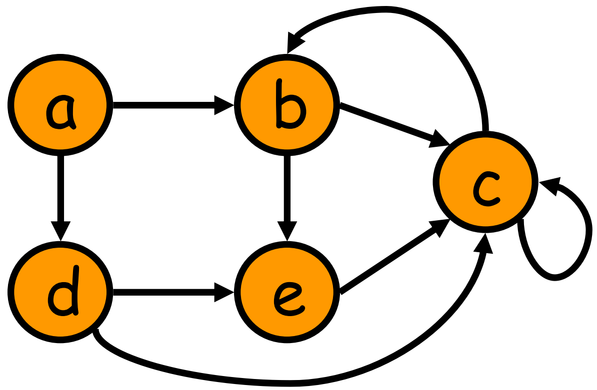

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 1 | 1 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |

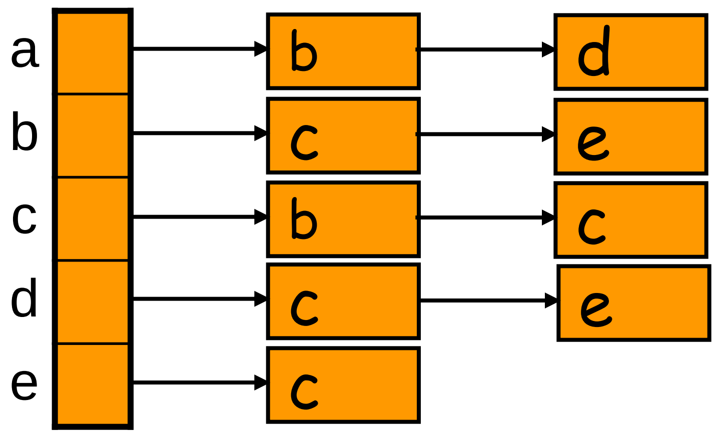
<u>הערה</u>: גרף לא-מכוון אפשר לייצג באמצעות מטריצה סימטרית.

# ייצוג גרף מכוון ברשימות סמיכויות (Adjacency lists)

לכל צומת נשמור את רשימת הצמתים אליהם הוא מצביע.

דוגמא:

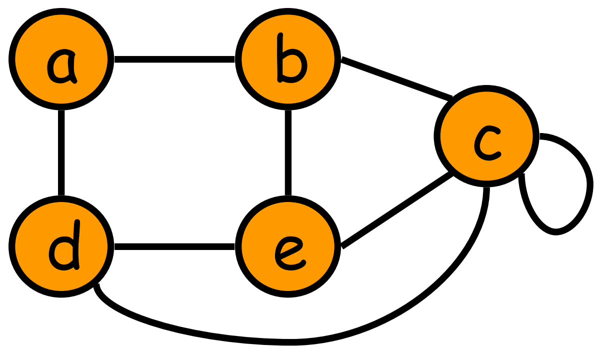


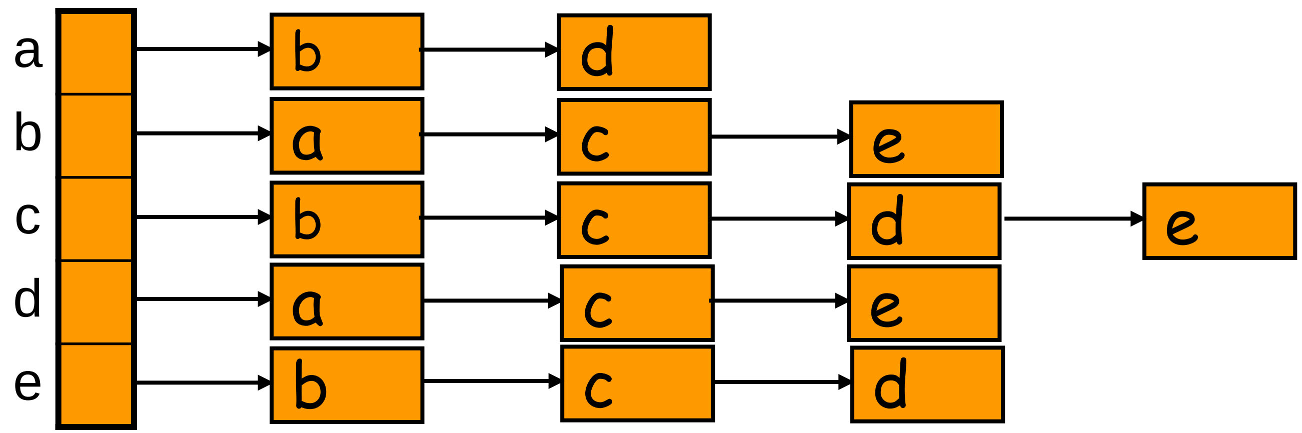



# ייצוג גרף לא-מכוון ברשימות סמיכויות

לכל צומת נשמור את רשימת הצמתים הסמוכים אליו.

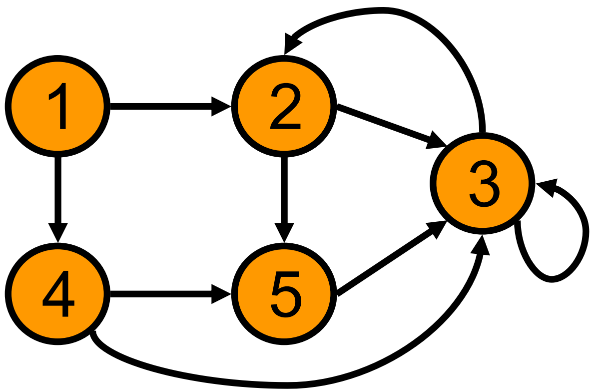
דוגמא:




כל קשת מופיעה פעמיים (מלבד חוגים עצמיים).

# תרגיל



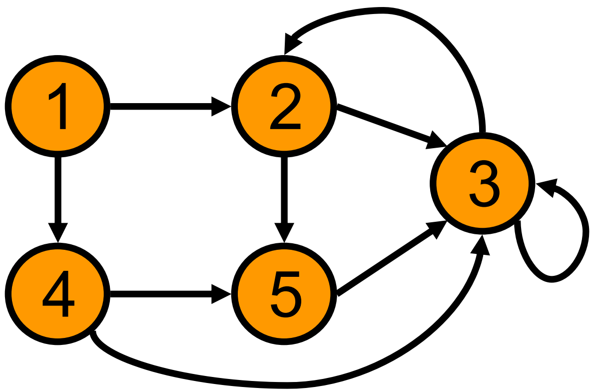

נתונה רשימה לא ממוינת של קשתות.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$.

$$E = \{(5,3),(2,3),(4,5),(4,3),(3,2),(1,2),(3,3),(1,4),(2,5)\}$$

נתונות $m$ קשתות כאשר כל קשת היא זוג מספרים בטווח $n \ldots 1$ .

# תרגיל




נתונה רשימה לא ממוינת של קשתות.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$.

$E = \{(5,3),(2,3),(4,5),(4,3),(3,2),(1,2),(3,3),(1,4),(2,5)\}$

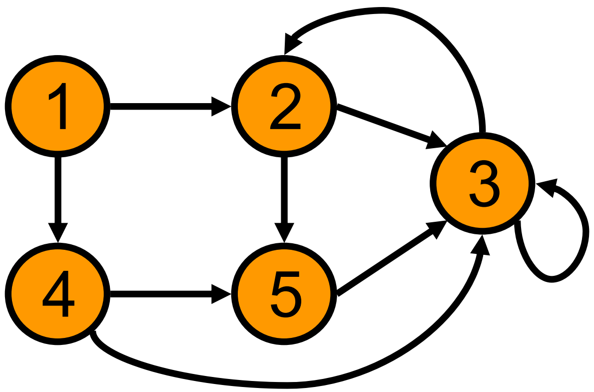
נתונות $m$ קשתות כאשר כל קשת היא זוג מספרים בטווח $n$ ... $1$ .

פתרון: נבצע Radix Sort כאשר הבסיס הוא $n$.

מיון לפי "ספרה" ימנית

| | | | |
|---|---|---|---|
| | (3,3) | | |
| | (4,3) | | |
| (1,2) | (2,3) | | (2,5) |
| (3,2) | (5,3) | (1,4) | (4,5) |

# תרגיל




נתונה רשימה לא ממוינת של קשתות.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$.

$E = \{(5,3),(2,3),(4,5),(4,3),(3,2),(1,2),(3,3),(1,4),(2,5)\}$

נתונות $m$ קשתות כאשר כל קשת היא זוג מספרים בטווח $n$ ... 1 .

פתרון: נבצע Radix Sort כאשר הבסיס הוא $n$.

מיון לפי "ספרה" ימנית

| | | | |
|---|---|---|---|
| | (3,3) | | |
| | (4,3) | | |
| (1,2) | (2,3) | | (2,5) |
| (3,2) | (5,3) | (1,4) | (4,5) |

מיון לפי "ספרה" שמאלית

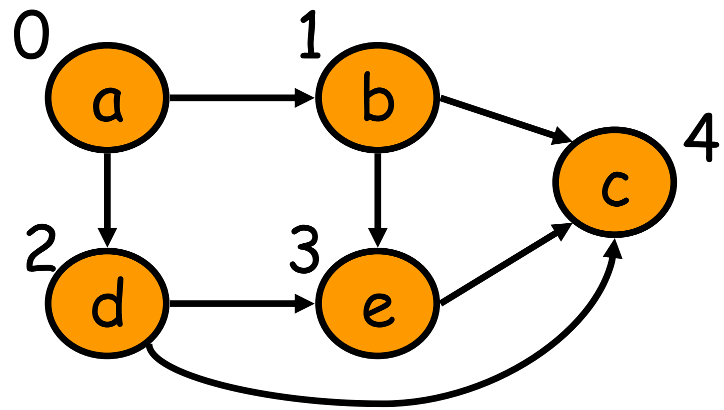
| | | | | |
|---|---|---|---|---|
| (1,4) | (2,5) | (3,3) | (4,5) | |
| (1,2) | (2,3) | (3,2) | (4,3) | (5,3) |

# מיון טופולוגי

קלט: גרף מכוון.

פלט: מספור של צמתי הגרף – לצומת $i$ יינתן המספר $N[i]$ – כך שיתקיים:

$N[i] < N[j] \Rightarrow (i,j) \in E$ או הודעה שלא קיים מספור כזה.




מתי ניתן למצוא מספור כזה ?

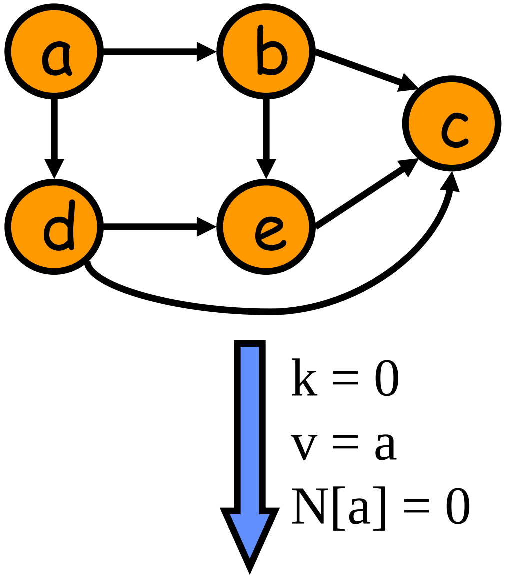
הגדרה: מקור הוא צומת שלא נכנסות אליו קשתות.

אבחנה: לכל גרף מכוון ללא מעגלים מכוונים יש לפחות מקור אחד.

# אלגוריתם למיון טופולוגי

1. אתחול: k = 0.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף
   
   (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

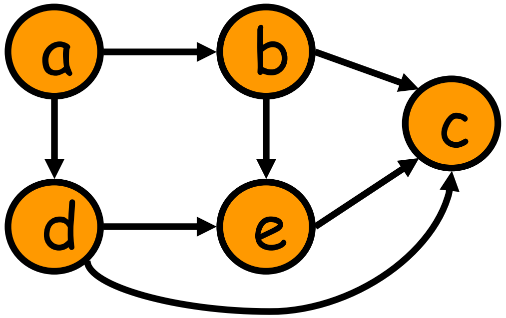
## דוגמא



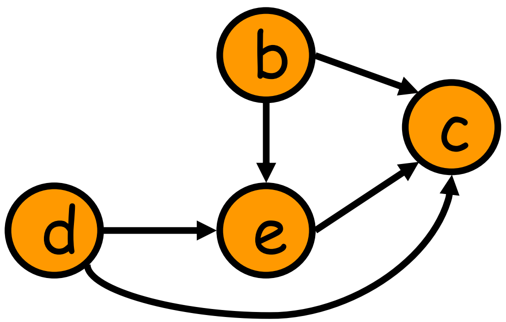
# אלגוריתם למיון טופולוגי

1. <u>אתחול</u>: $k = 0$.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף
   
     (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

## דוגמא



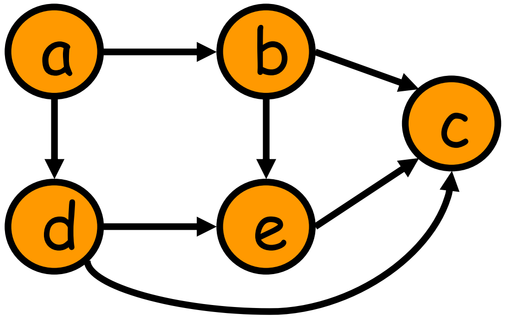

$k = 0$
$v = a$
$N[a] = 0$



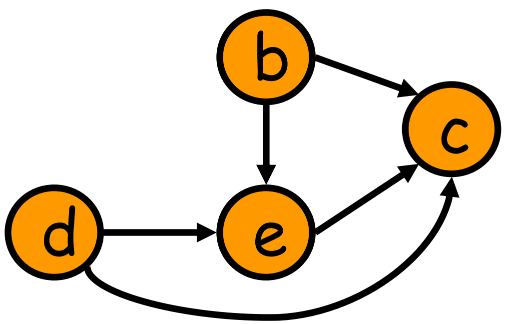

$k = 1$
$v = b$
$N[b] = 1$

# אלגוריתם למיון טופולוגי

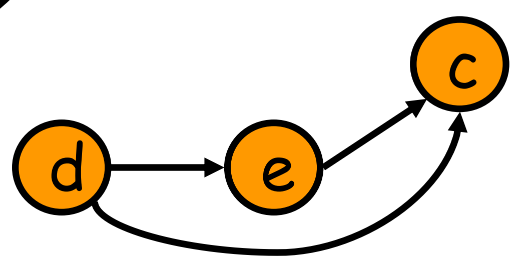
דוגמא




k = 0
v = a
N[a] = 0




k = 1
v = b
N[b] = 1



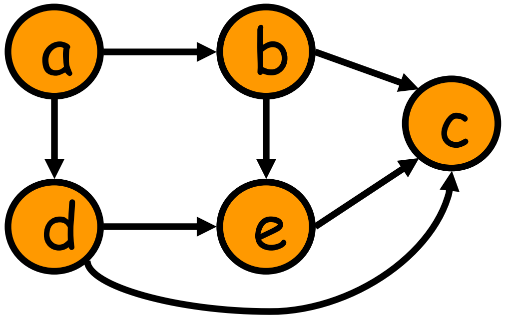

k = 2
v = d
N[d] = 2

1. אתחול: k = 0.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף
   
   (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

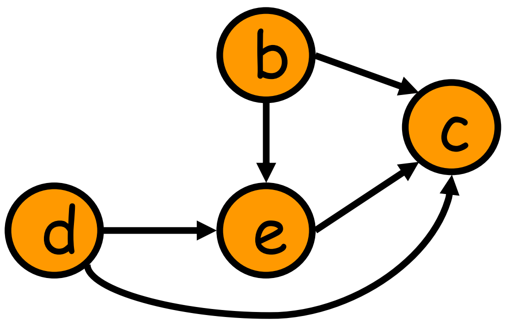
# אלגוריתם למיון טופולוגי

1. <u>אתחול</u>: k = 0.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף
   
     (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

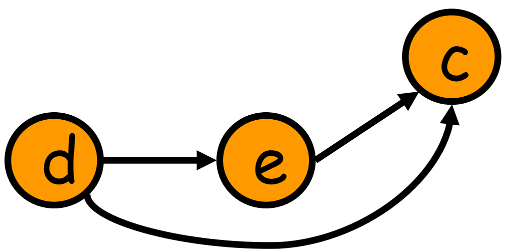
## <u>דוגמא</u>



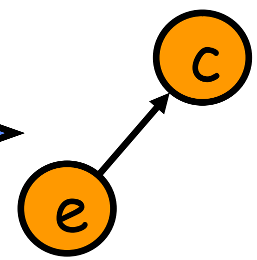

k = 0
v = a
N[a] = 0




k = 1
v = b
N[b] = 1




k = 2
v = d
N[d] = 2

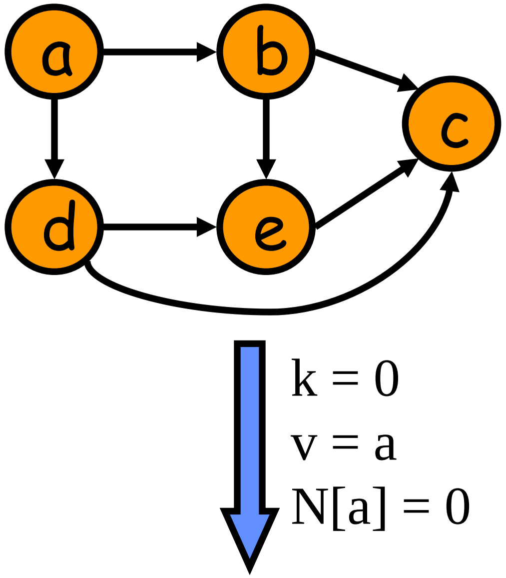



k = 3
v = e
N[e] = 3

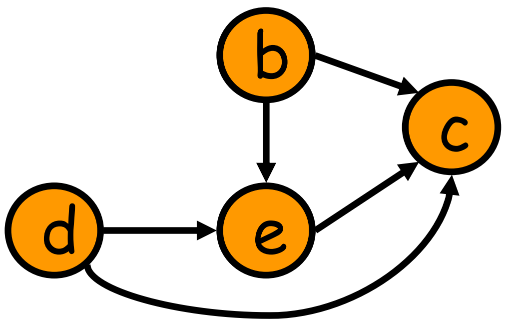
# אלגוריתם למיון טופולוגי

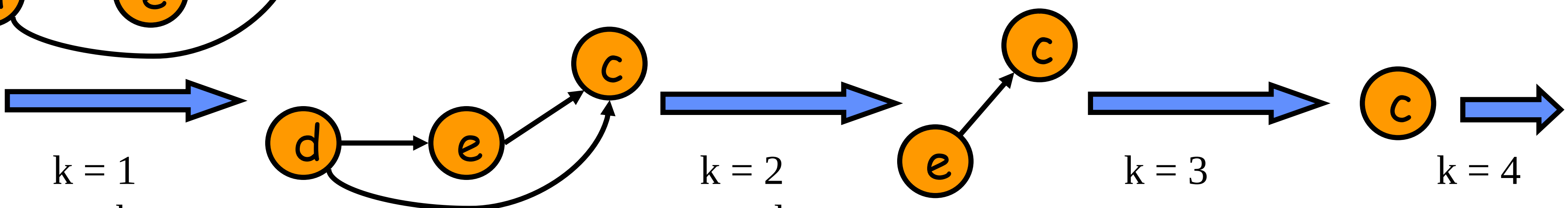
1. <u>אתחול</u>: k = 0.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

## <u>דוגמא</u>



k = 0
v = a
N[a] = 0





k = 1
v = b
N[b] = 1

k = 2
v = d
N[d] = 2

k = 3
v = e
N[e] = 3

k = 4
v = c
N[c] = 4

# מבני נתונים למיון טופולוגי

1. אתחול: $k = 0$.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף (וכן את הקשתות היוצאות ממנו).
3. אם k==n אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

הפעולות הנדרשות

מצא מקור

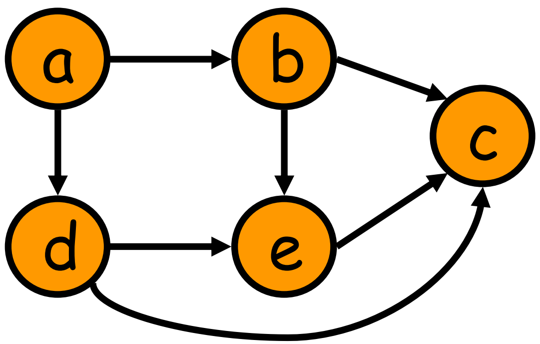
סלק מקור

מימוש 1 בייצוג מטריצת סמיכויות

מימוש 2 בייצוג רשימת סמיכויות

# מימוש בייצוג מטריצת סמיכויות

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |




<u>מימוש הפעולות הנדרשות</u>

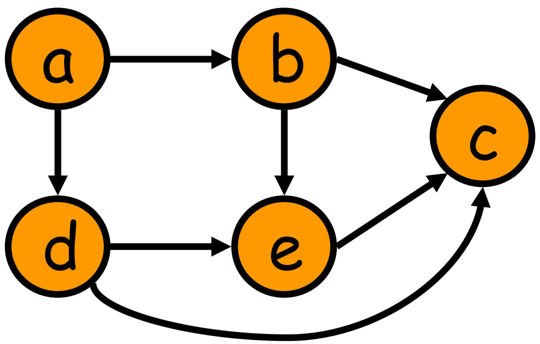
<u>מצא מקור</u>: כדי לבדוק האם צומת $i$ מקור, נבדוק שעמודה $i$ כולה אפסים. זמן $O(n)$.

כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n^2)$.

<u>סלק מקור $i$</u>: אפס שורה $i$. (העמודה כבר מאופסת כיוון שמסלקים מקור). זמן $O(n)$.

# מימוש בייצוג מטריצת סמיכויות

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |




<u>מימוש הפעולות הנדרשות</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת $i$ מקור, נבדוק שעמודה $i$ כולה אפסים. זמן $O(n)$.

כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n^2)$.

<u>סלק מקור $i$</u>: אפס שורה $i$. (העמודה כבר מאופסת כיון שמסלקים מקור). זמן $O(n)$.

<u>זמן כללי</u>: האלגוריתם מבצע $n$ פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן היא $O(n^3)$.

צוואר הבקבוק במימוש זה היא פעולת "מצא מקור" המתבצעת $n$ פעמים.

# מימוש בייצוג מטריצת סמיכויות (שיפור)

בכל השיפורים נבצע עבוד מקדים (Preprocessing). בחישוב זמן הריצה נוסיף את זמן העיבוד המוקדם לזמן ריצת האלגוריתם.

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |

בנה מערך in-degree[i] ובו נשמור את מספר הקשתות הנכנסות לצומת i. זמן הבניה $O(n^2)$.

# מימוש בייצוג מטריצת סמיכויות (שיפור)

בכל השיפורים נבצע עבוד מקדים (Preprocessing). בחישוב זמן הריצה נוסיף את זמן העיבוד המוקדם לזמן ריצת האלגוריתם.

בנה מערך in-degree[i] ובו נשמור את מספר הקשתות הנכנסות לצומת $i$. זמן הבניה $O(n^2)$.

<u>מימוש הפעולות הנדרשות</u>

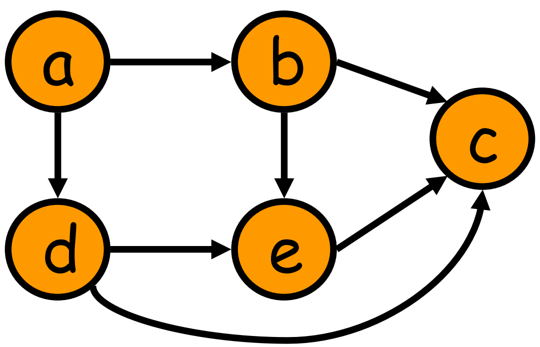
<u>מצא מקור</u>: כדי לבדוק האם צומת $i$ מקור, נבדוק in-degree[i]=0. זמן $O(1)$.

כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n)$.

<u>סלק מקור $i$</u>: אפס שורה $i$.

לכל $A[i,j]=1$ הקטן באחד את in-degree[j]. זמן $O(n)$.

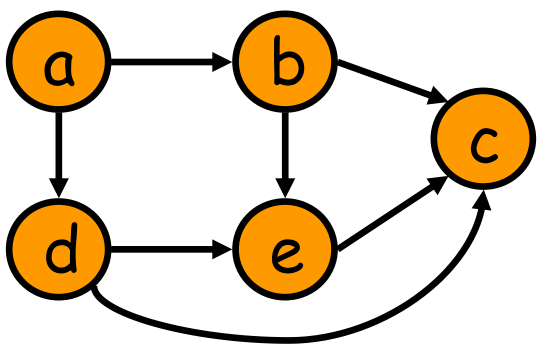
| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |



# מימוש בייצוג מטריצת סמיכויות (שיפור)

בכל השיפורים נבצע עבוד מקדים (Preprocessing). בחישוב זמן הריצה נוסיף את זמן העיבוד המוקדם לזמן ריצת האלגוריתם.

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 0 |
| b | 0 | 0 | 1 | 0 | 1 |
| c | 0 | 0 | 0 | 0 | 0 |
| d | 0 | 0 | 1 | 0 | 1 |
| e | 0 | 0 | 1 | 0 | 0 |




בנה מערך in-degree[i] ובו נשמור את מספר הקשתות הנכנסות לצומת i. זמן הבניה $O(n^2)$.

<u>מימוש הפעולות הנדרשות</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת i מקור, נבדוק in-degree[i]=0. זמן $O(1)$.

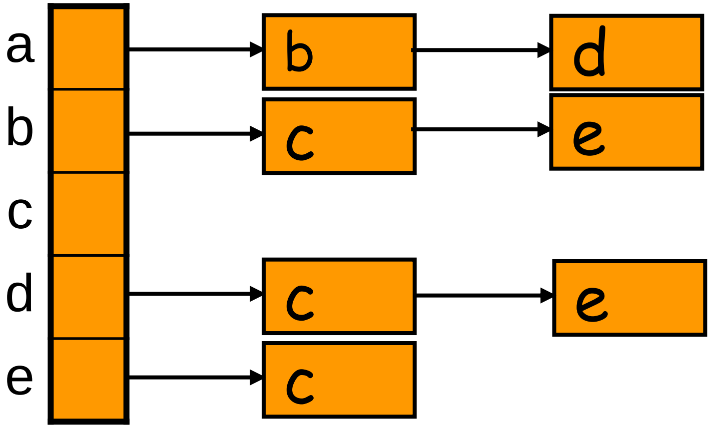
כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n)$.

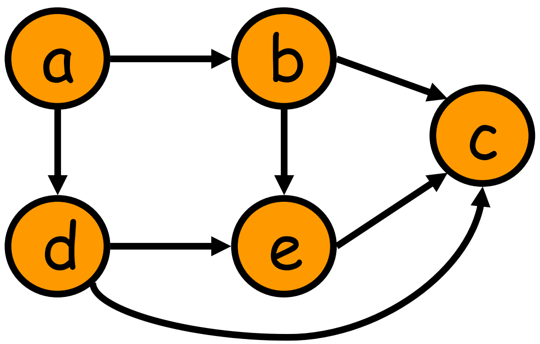
<u>סלק מקור i</u>: אפס שורה i.

לכל $A[i,j]=1$ הקטן באחד את in-degree[j]. זמן $O(n)$.

<u>זמן כללי</u>: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן היא $O(n^2)$.

# מימוש בייצוג רשימת סמיכויות




מימוש הפעולות הנדרשות

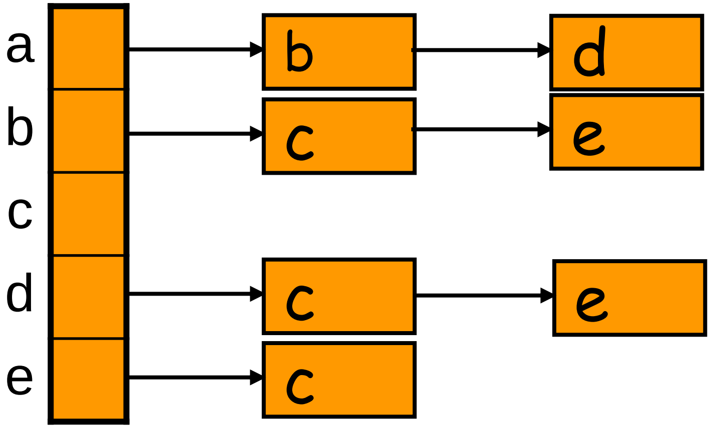
מצא מקור: כדי לבדוק האם צומת i מקור, נעבור על כל הרשימות. אם הצומת לא נמצא אז הוא מקור. זמן $O(m)$.

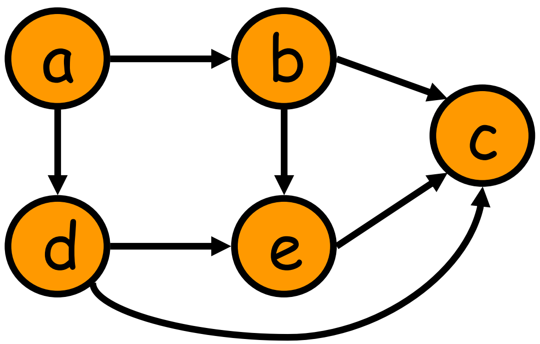
כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n \cdot m)$.

סלק מקור i: סמן שצומת i סולק. זמן $O(1)$.

זמן כללי: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן היא $O(n^2 \cdot m)$.

# מימוש בייצוג רשימת סמיכויות




<u>מימוש הפעולות הנדרשות</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת i מקור, נעבור על כל הרשימות. אם הצומת לא נמצא אז הוא מקור. זמן $O(m)$.

כדי למצוא מקור נבדוק את כל הצמתים. סה"כ $O(n \cdot m)$.

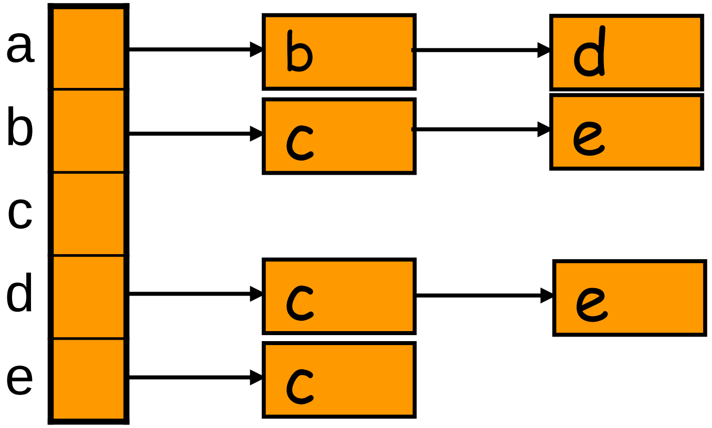
<u>סלק מקור i</u>: סמן שצומת i סולק. זמן $O(1)$.

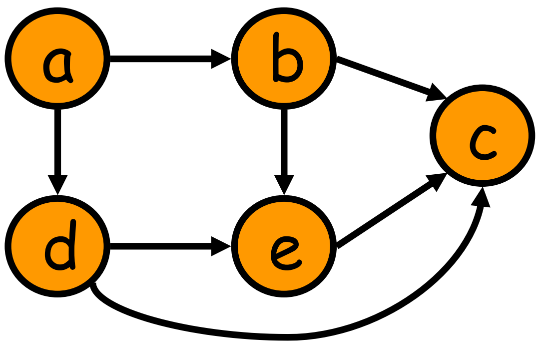
<u>זמן כללי</u>: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן היא $O(n^2 \cdot m)$.

<u>שיפור ראשון</u>:

מצא מקור: כדי למצוא מקור נעבור על כל הרשימות ונשמור מערך בולאני ובו 1 לכל צומת בו נתקלנו ו- 0 אחרת. זמן $O(n+m)$. לפיכך זמן כללי $O(n(n+m))$.

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שני: נבנה ונשמור in-degree[i].

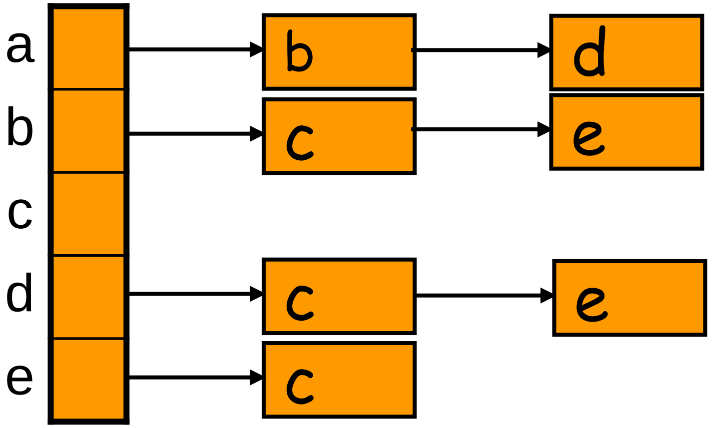
בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. זמן $O(n+m)$.

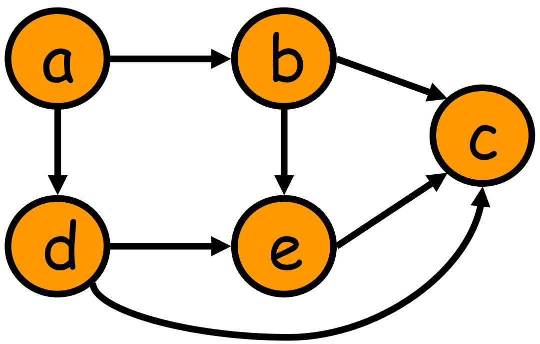
<u>מצא מקור</u>: עבור על המערך in-degree בזמן $O(n)$.

<u>סלק מקור i</u>: עבור על הרשימה ה- i ועדכן את המערך in-degree. זמן $O(\text{out-degree}(i))$.

<u>זמן כללי</u>: האלגוריתם מבצע $n$ פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות זמן היא:

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שני: נבנה ונשמור in-degree[i].

בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. זמן O(n+m).

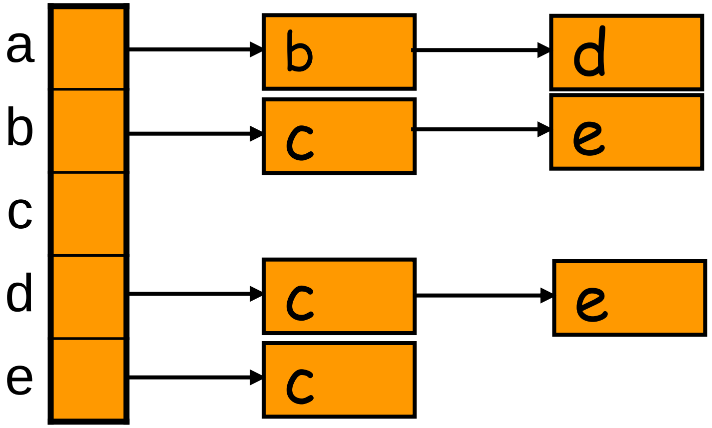
מצא מקור: עבור על המערך in-degree בזמן O(n).

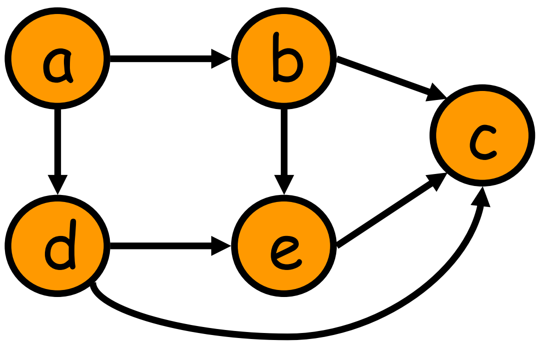
סלק מקור i: עבור על הרשימה ה- i ועדכן את המערך in-degree. זמן O(out-degree(i)).

זמן כללי: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות זמן היא:

$$\sum_{i=1}^{n} O(n+\text{outdegree}(i)) = O(n^2+\sum_{i=1}^{n}\text{outdegree}(i))$$

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שני: נבנה ונשמור in-degree[i].

בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. זמן O(n+m).

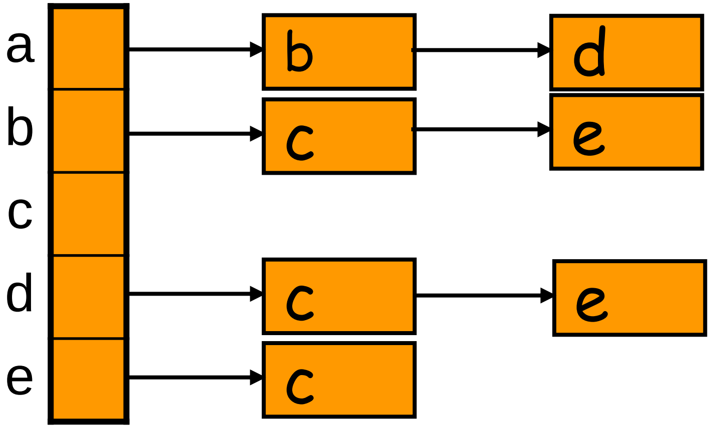
מצא מקור: עבור על המערך in-degree בזמן O(n).

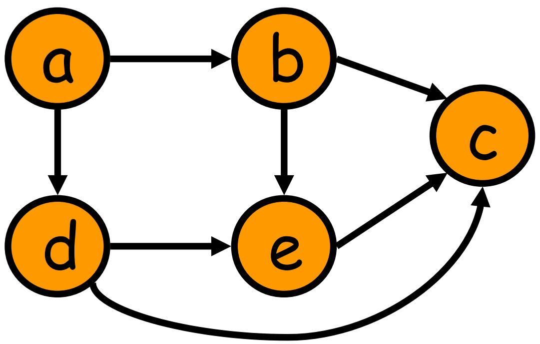
סלק מקור i: עבור על הרשימה ה- i ועדכן את המערך in-degree. זמן O(out-degree(i)).

זמן כללי: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות זמן היא:

$$\sum_{i=1}^{n} O(n+\mathrm{outdegree}(i)) = O(n^2+\sum_{i=1}^{n}\mathrm{outdegree}(i)) \qquad = O(n^2+m)$$

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שלישי: נשמור צמתים עבורם מתקיים in-degree[i]=0 במחסנית, תור, או רשימה.

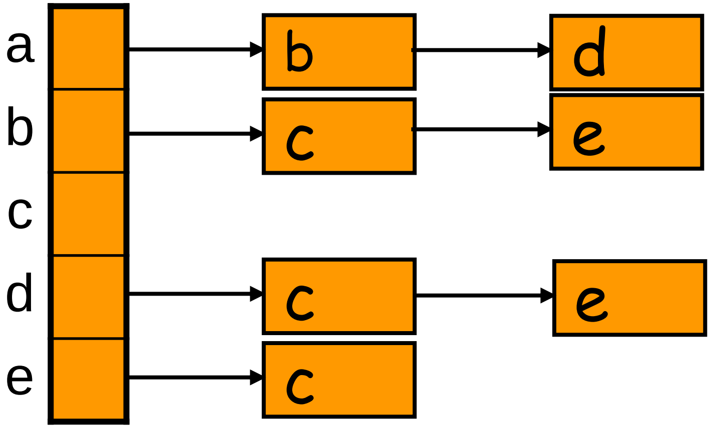
בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. כמו קודם. זמן $O(n+m)$.

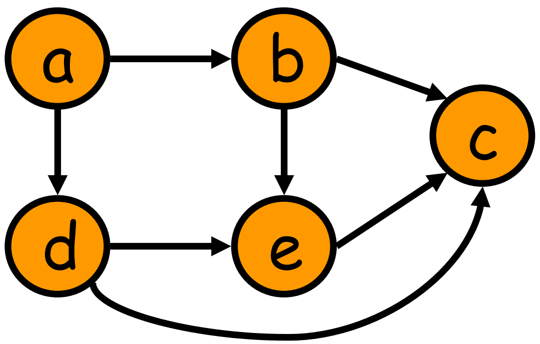
מצא מקור: איבר ראשון במחסנית. זמן $O(1)$.

סלק מקור i: עבור על הרשימה ה- i ועדכן את המערך in-degree. הכנס למחסנית איברים עבורם דרגת הכניסה מתאפסת. זמן

$O(\text{out-degree}(i))$.

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שלישי: נשמור צמתים עבורם מתקיים in-degree[i]=0 במחסנית, תור, או רשימה.

בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. כמו קודם. זמן $O(n+m)$.

<u>מצא מקור</u>: איבר ראשון במחסנית. זמן $O(1)$.

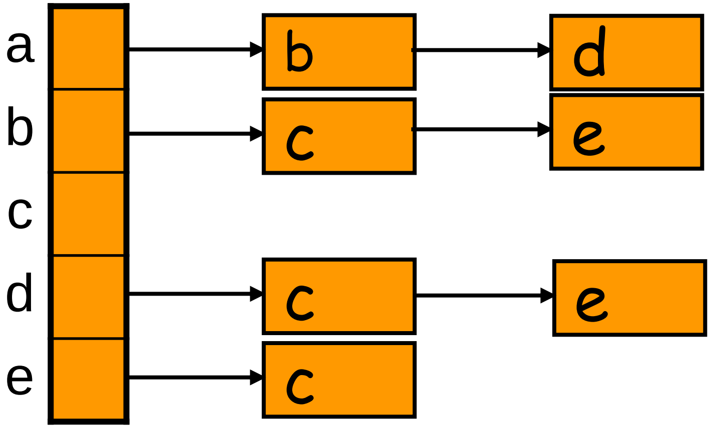
<u>סלק מקור i</u>: עבור על הרשימה ה- i ועדכן את המערך in-degree. הכנס למחסנית איברים עבורם דרגת הכניסה מתאפסת. זמן

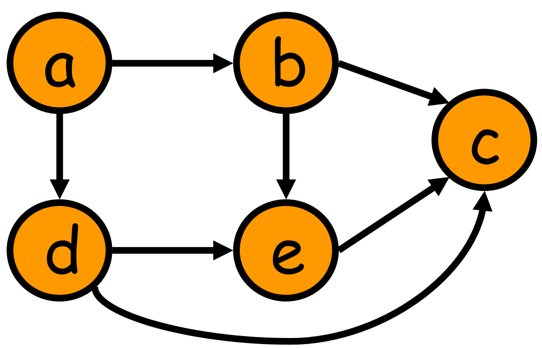
$O(\text{out-degree}(i))$.

<u>זמן כללי</u>:

$$\sum_{i=1}^{n} O(1+\text{outdegree}(i)) = O(n+\sum_{i=1}^{n}\text{outdegree}(i))$$

# מימוש בייצוג רשימת סמיכויות (שיפור)




שיפור שלישי: נשמור צמתים עבורם מתקיים in-degree[i]=0 במחסנית, תור, או רשימה.

בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. כמו קודם. זמן $O(n+m)$.

מצא מקור: איבר ראשון במחסנית. זמן $O(1)$.

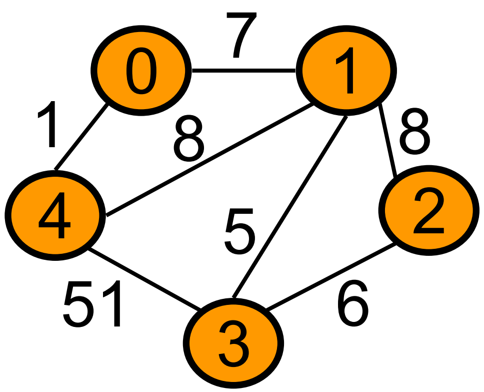
סלק מקור i: עבור על הרשימה ה- i ועדכן את המערך in-degree. הכנס למחסנית איברים עבורם דרגת הכניסה מתאפסת. זמן $O(\text{out-degree}(i))$.

זמן כללי: $$\sum_{i=1}^{n} O(1+\text{outdegree}(i)) = O(n+\sum_{i=1}^{n}\text{outdegree}(i)) = O(n+m)$$

# גרפים ממושקלים

גרף יקרא <u>ממושקל</u> אם לכל קשת $e$ בקבוצה $E$ קיים משקל חיובי $w(e)$.

נתמקד בגרפים ממושקלים לא-מכוונים. גרפים אלה ניתן לייצג בעזרת מטריצת סמיכויות סימטרית או רשימות סמיכויות.




דוגמא:

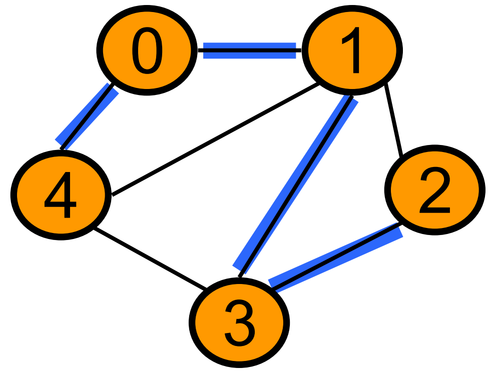
$$A[i,j] = \begin{cases} 0 & e \notin E \\ w(i,j) & \text{אחרת} \end{cases}$$

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 7 | 0 | 0 | 1 |
| 1 | | 0 | 8 | 5 | 8 |
| 2 | | | 0 | 6 | 0 |
| 3 | | | | 0 | 51 |
| 4 | | | | | 0 |

מטריצה $A$ ממומשת ע"י ייצוג של מטריצה סימטרית. האלכסון מציין חוגים עצמיים. מספר האיברים הנדרש הוא $n(n+1)/2$.

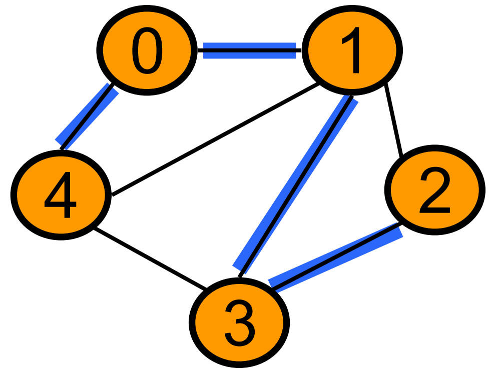
# עץ פורש מינימום

עץ פורש של גרף לא-מכוון $G$ הוא עץ שצמתיו הם הצמתים של $G$ וקשתותיו הן קשתות של $G$.

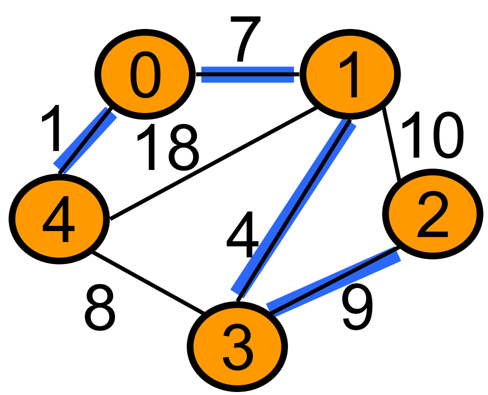


# עץ פורש מינימום

עץ פורש של גרף לא-מכוון $G$ הוא עץ שצמתיו הם הצמתים של $G$ וקשתותיו הן קשתות של $G$.

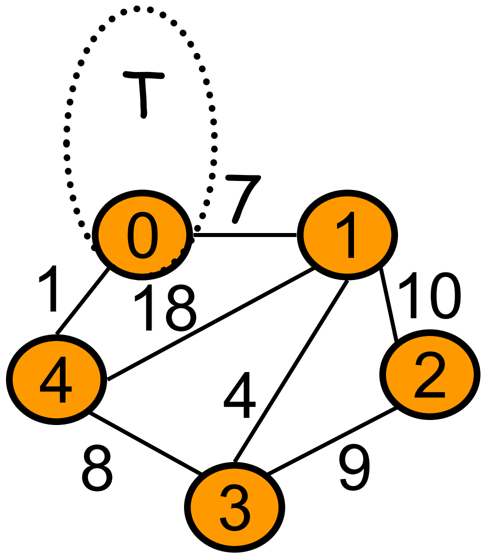



<u>עץ פורש מינימום</u> של גרף ממושקל לא-מכוון $G$ הוא עץ פורש של $G$ שסכום משקלי קשתותיו מינימלי.



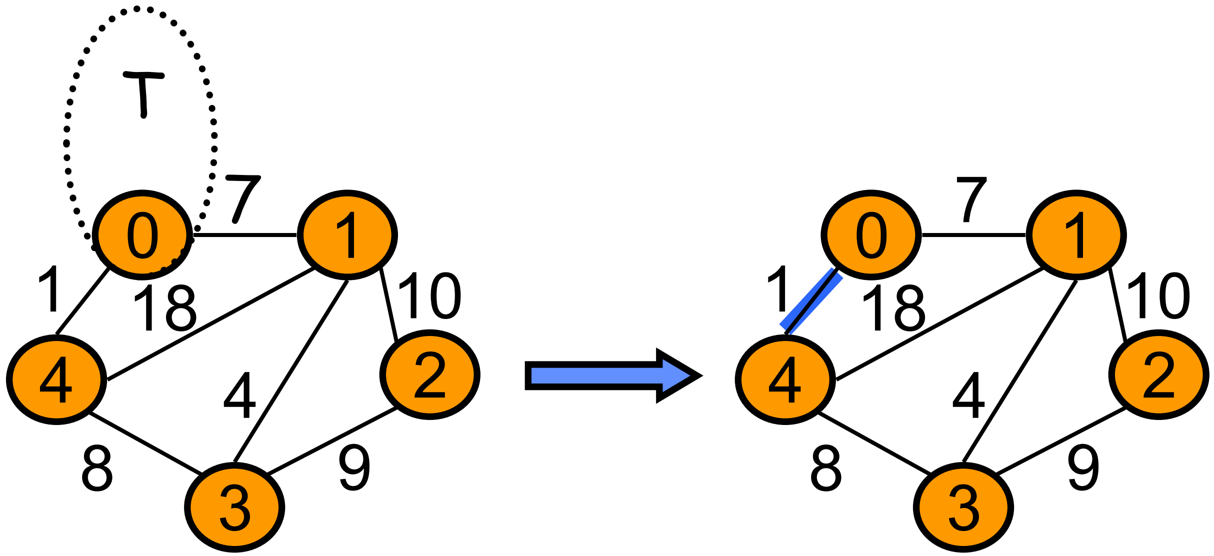
# מציאת עץ פורש מינימום

<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



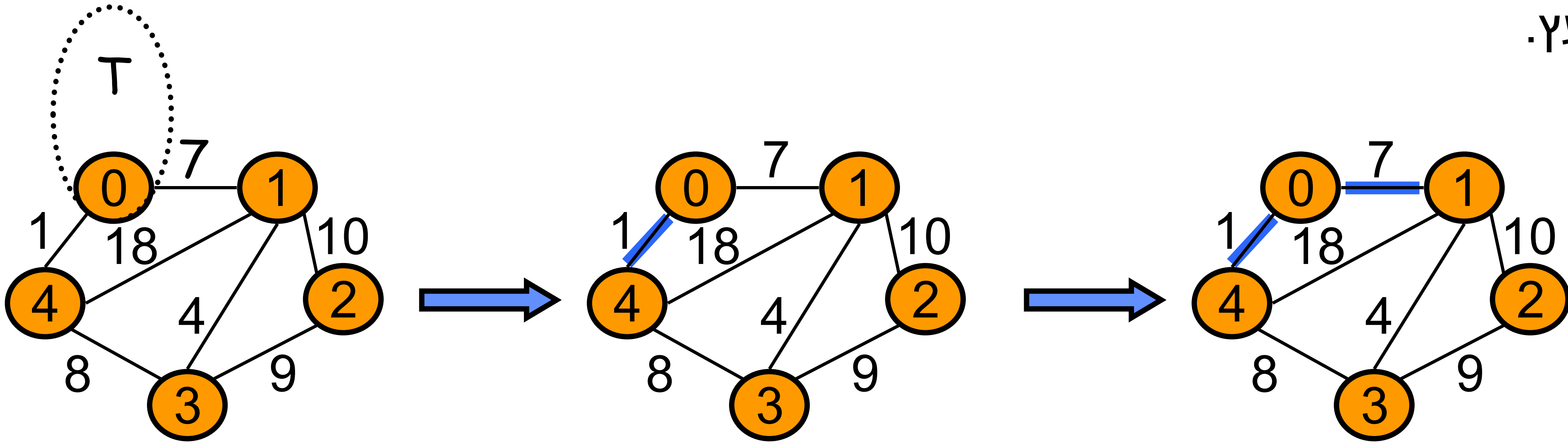
# מציאת עץ פורש מינימום

<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



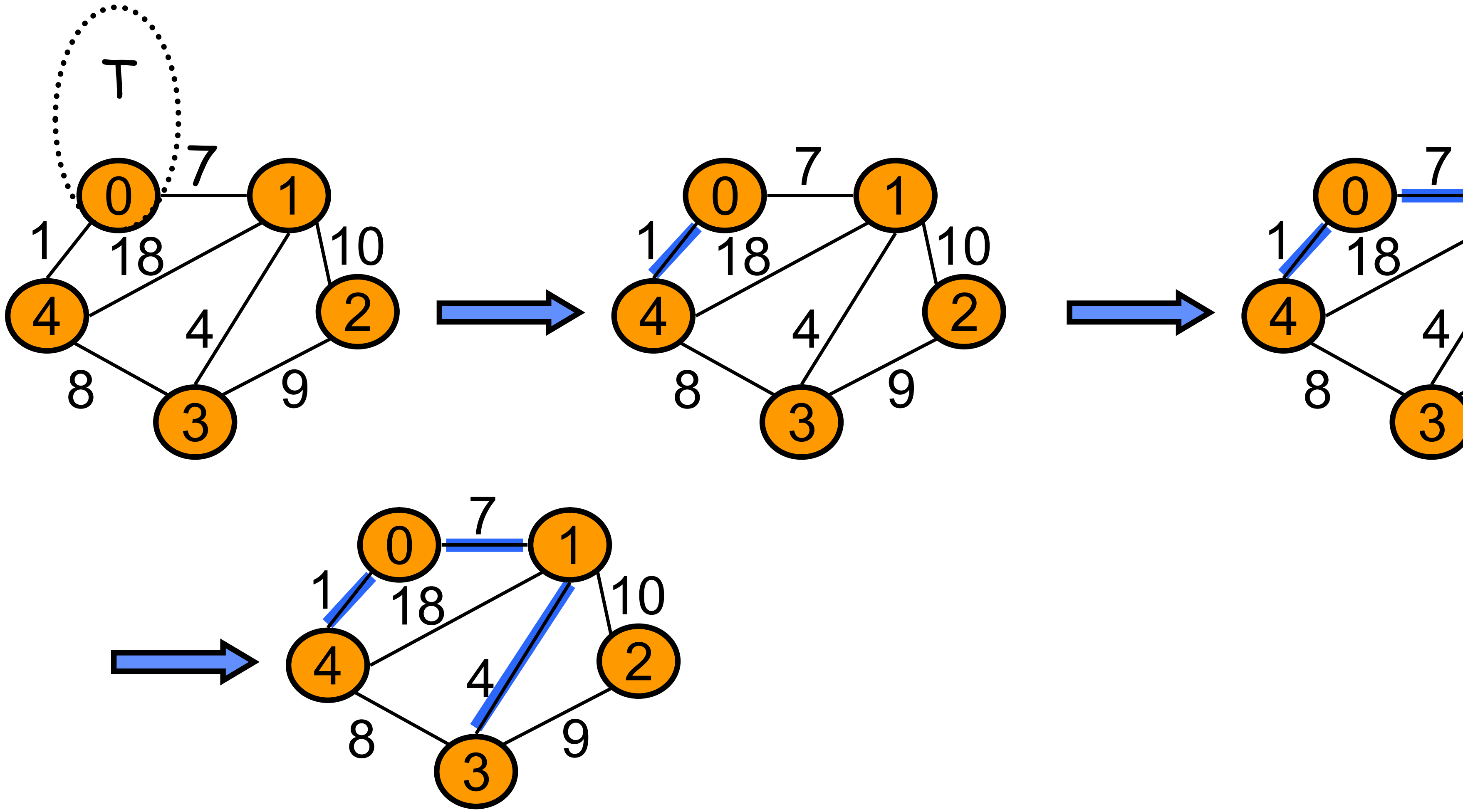
# מציאת עץ פורש מינימום

<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



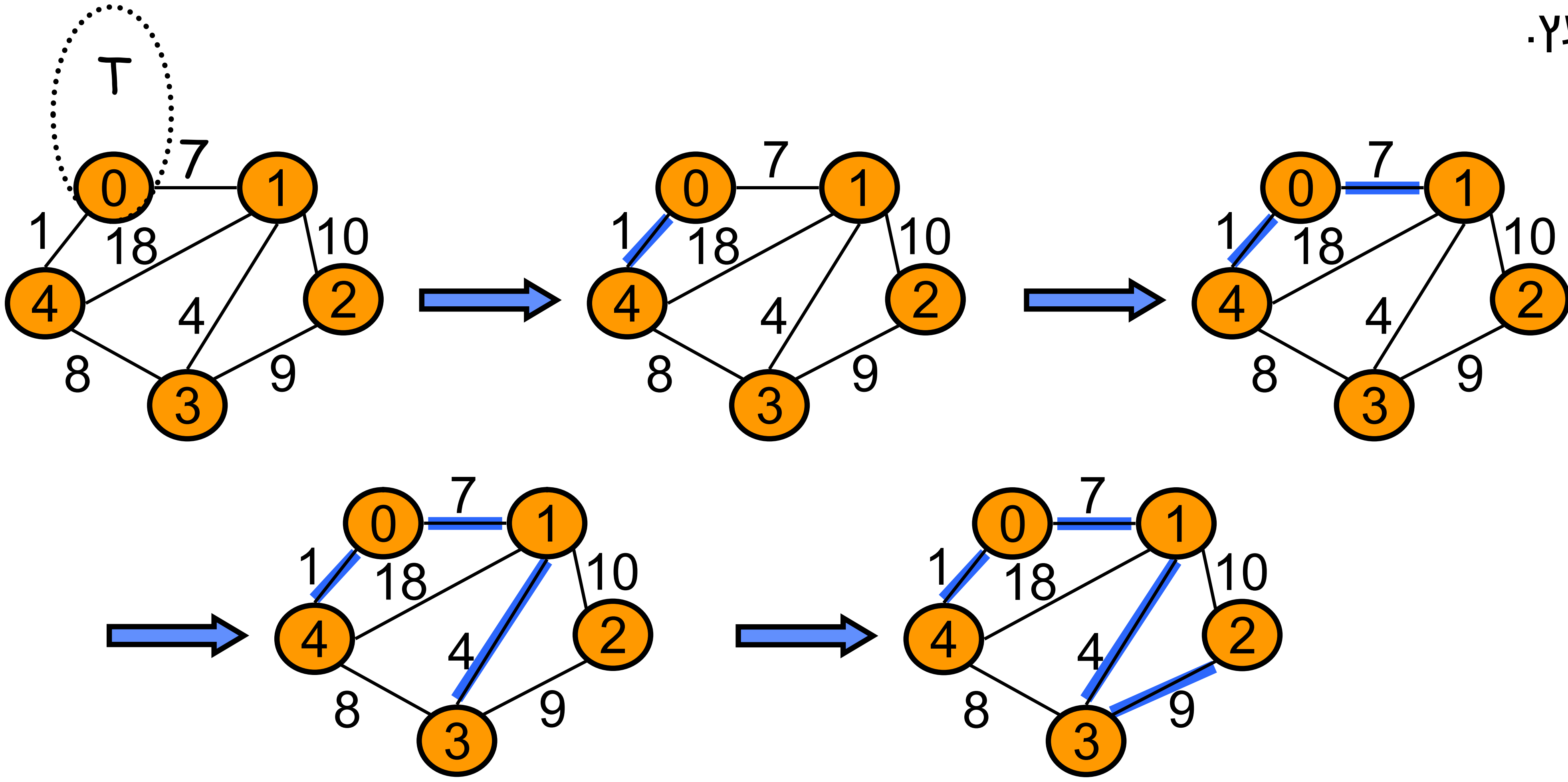
# מציאת עץ פורש מינימום

<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



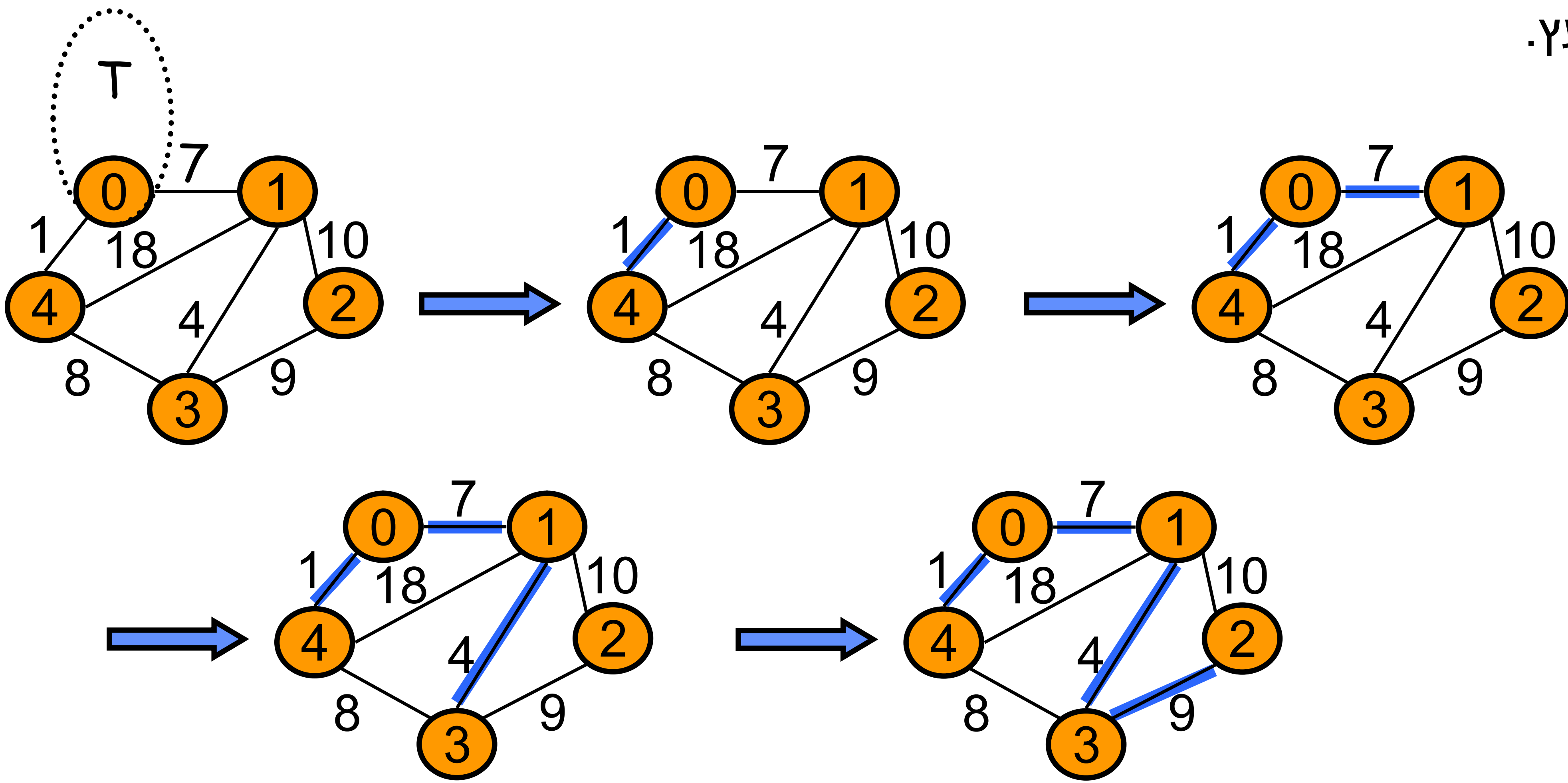
# מציאת עץ פורש מינימום

<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



# מציאת עץ פורש מינימום

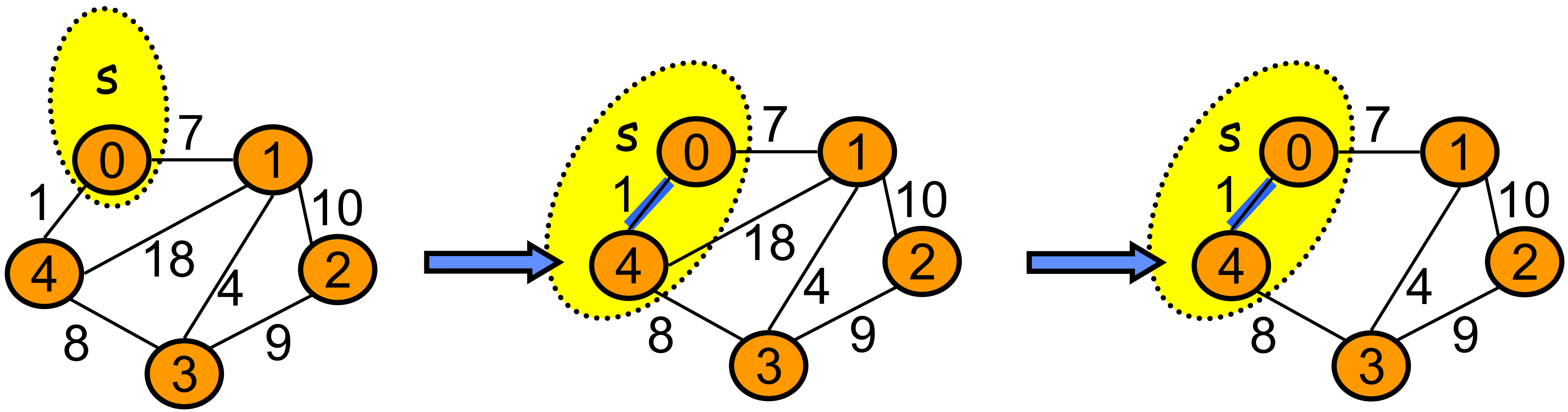
<u>רעיון (Prim)</u>: הכנס צומת כלשהו לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.




<u>הערה</u>: כמובן שיש להוכיח את נכונות האלגוריתם (ראו פרק 24), אך אנו נתרכז בפרטי המימוש בלבד.

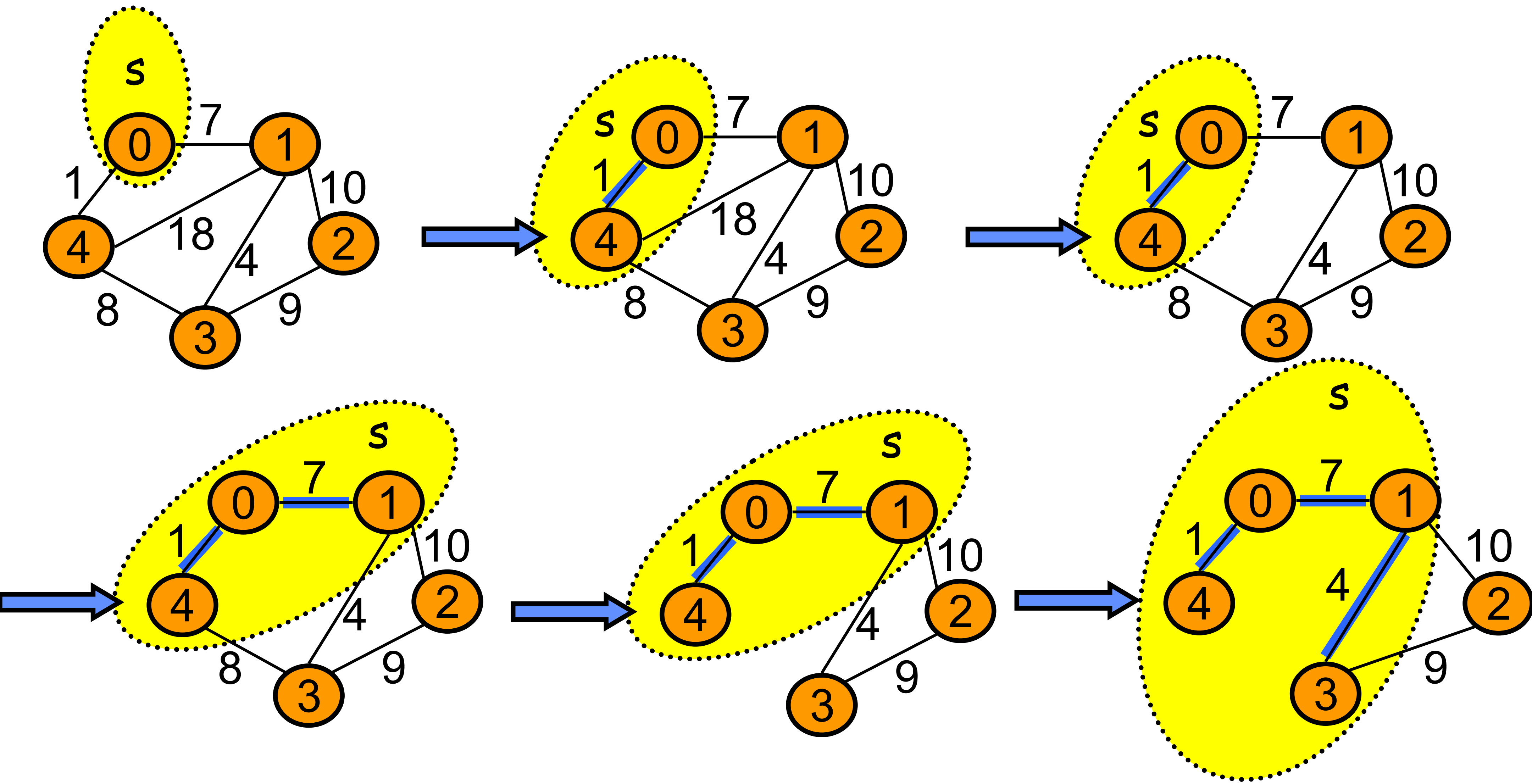
# צעד ראשון לקראת מימוש

רעיון: בכל שלב נבחר קשת (s,i) להכנסה לעץ T ונשנה את הגרף המקורי על ידי אחוד הצומת s והצומת i לצומת בודד ששמו s ועדכון הקשתות היוצאות מצומת זה.



# צעד ראשון לקראת מימוש

<u>רעיון</u>: בכל שלב נבחר קשת $(s,i)$ להכנסה לעץ $T$ ונשנה את הגרף המקורי על ידי אחוד הצומת $s$ והצומת $i$ לצומת בודד ששמו $s$ ועדכון הקשתות היוצאות מצומת זה.



# אלגוריתם למציאת עץ פורש מינימום (Prim)

איתחול: בחר צומת התחלה כלשהו s. הכנס את s לקבוצת צמתי העץ $V_T$. צור קבוצה ריקה $E_T$ עבור קשתות העץ.

כל עוד קיימות קשתות היוצאות מהצומת s בצע:

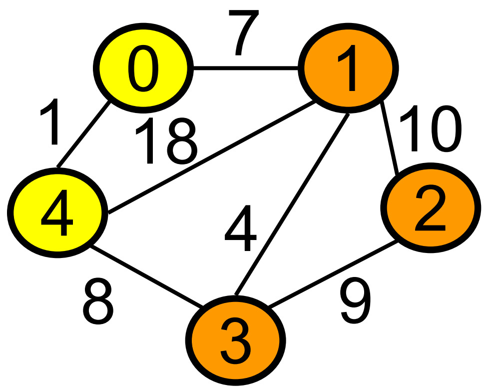
1. מצא קשת (s,i) בעלת משקל מינימלי.
2. הוסף קשת (s,i) לקבוצה $E_T$ (עם הקצוות המקוריים) ואת הצומת i לקבוצה $V_T$.
3. (עדכן את G) סלק צומת i ואת הקשתות הסמוכים ל- i מהגרף G.

   לכל קשת (i,j) מלבד הקשת (i,s) בצע:

   1. אם הקשת (s,j) לא נמצאת ב- G אז הוסף אותה עם המשקל w(i,j)
   2. אם הקשת (s,j) נמצאת ב- G ומתקיים $w(i,j) < w(s,j)$, אז הקטן את משקלה של (s,j) להיות w(i,j).

# מימוש עבור גרף נתון במטריצת סמיכויות

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s = 0 | - | 7 | - | - | 1 |
| 1 | 7 | - | 10 | 4 | 18 |
| 2 | - | 10 | - | 9 | - |
| 3 | - | 4 | 9 | - | 8 |
| 4 | 1 | 18 | - | 8 | - |
| Origin: | | 0 | - | - | 0 |



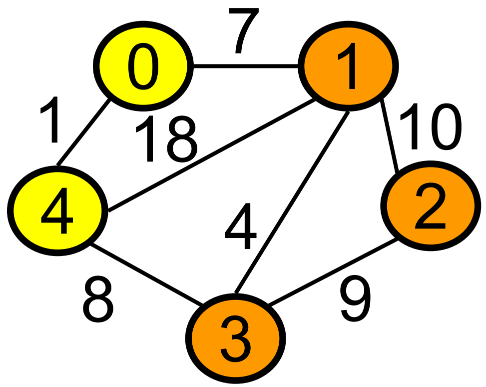

הפעולות

מציאת קשת קלה ביותר היוצאת מצומת s: ע"י מעבר על השורה המתאימה לצומת s.

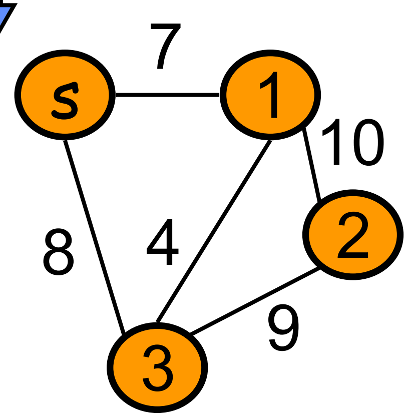
סילוק צומת i: ע"י מיזוג שורה i עם השורה של צומת s.

# מימוש עבור גרף נתון במטריצת סמיכויות

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| s = 0 | - | 7 | - | - | 1 |
| 1 | 7 | - | 10 | 4 | 18 |
| 2 | - | 10 | - | 9 | - |
| 3 | - | 4 | 9 | - | 8 |
| 4 | 1 | 18 | - | 8 | - |
| Origin: | | 0 | - | - | 0 |




| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | - | 7 | - | 8 | |
| 1 | 7 | - | 10 | 4 | |
| 2 | - | 10 | - | 9 | |
| 3 | - | 4 | 9 | - | |
| | | | | | |
| Origin: | | 0 | - | 4 | |




<u>הפעולות</u>

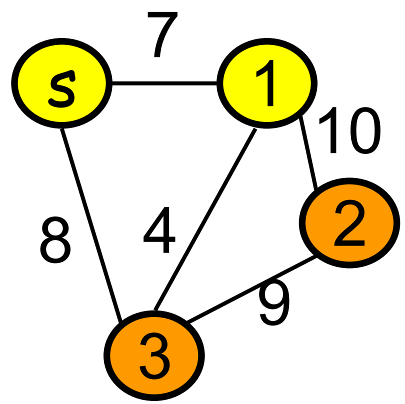
<u>מציאת קשת קלה ביותר</u>
<u>היוצאת מצומת s</u>: ע"י מעבר
על השורה המתאימה
לצומת s.

<u>סילוק צומת i</u>: ע"י מיזוג
שורה i עם השורה של
צומת s.

המערך origin שומר את הקצה של הקשת שקבעה את המרחק מהצומת s.

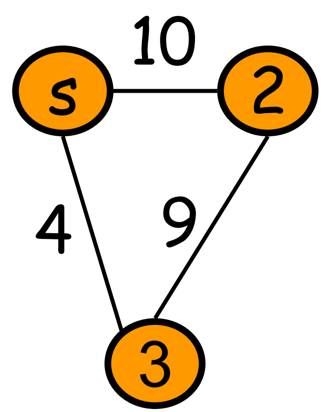
# מימוש עבור גרף נתון במטריצת סמיכויות

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | - | 7 | - | 8 | |
| 1 | 7 | - | 10 | 4 | |
| 2 | - | 10 | - | 9 | |
| 3 | - | 4 | 9 | - | |
| | | | | | |




Origin: 0 - 4

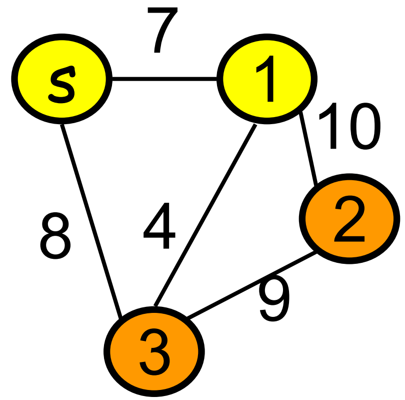
| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | - | | 10 | 4 | |
| 1 | | | | | |
| 2 | 10 | | - | 9 | |
| 3 | 4 | | 9 | - | |
| | | | | | |



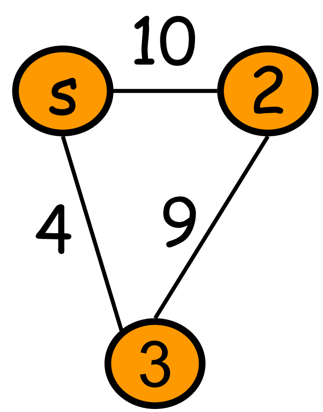

Origin: 1 1

# מימוש עבור גרף נתון במטריצת סמיכויות

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | - | 7 | - | 8 | |
| 1 | 7 | - | 10 | 4 | |
| 2 | - | 10 | - | 9 | |
| 3 | - | 4 | 9 | - | |
| | | | | | |
| Origin: | | 0 | - | 4 | |




| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | - | | 10 | 4 | |
| 1 | | | | | |
| 2 | 10 | | - | 9 | |
| 3 | 4 | | 9 | - | |
| | | | | | |
| Origin: | | | 1 | 1 | |




## זמנים

מציאת קשת קלה ביותר היוצאת מצומת s: ע"י מעבר על השורה המתאימה לצומת s.

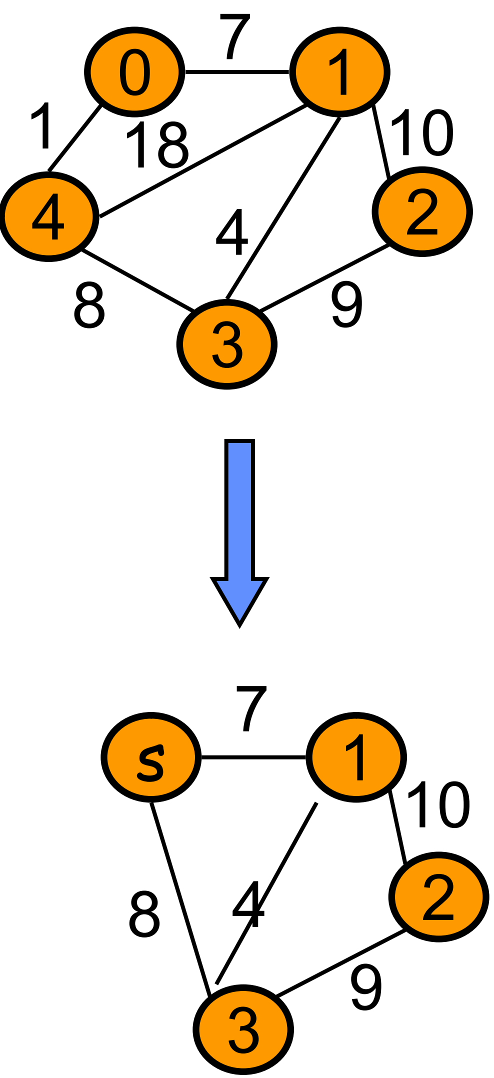
זמן $O(n)$ .

סילוק צומת i: ע"י מיזוג שורה i עם השורה של צומת s. זמן $O(n)$ .

סה"כ $O(n^2)$.

# מימוש עבור גרף נתון ברשימת סמיכויות

נניח שרשימות הסמיכויות ממוינות. או נמיינן בזמן $O(n+m)$.




<u>מציאת קשת קלה ביותר היוצאת מצומת $s$</u>: ע"י מעבר על רשימת הסמיכויות המתאימה לצומת $s$.

<u>סילוק צומת $i$</u>: ע"י מיזוג רשימת הסמיכויות של צומת $i$ עם רשימת הסמיכויות של צומת $s$.

<u>ניתוח זמנים</u>

בכל שלב מוצאים קשת קלה ביותר בזמן $O(n)$(אורך מקסימלי של רשימת סמיכויות של $s$).

בכל שלב משלבים שתי רשימות ממוינות- זו המתאימה לצומת $i$ וזו המתאימה לצומת $s$.

הזמן הנדרש הוא אורך הרשימות, כלומר $O(n)$.

סכ"ה $O(n^2)$.

# שיפורים

נשמור את הקשתות היוצאות מצומת $s$ <u>בערימה</u>. זמן מציאת קשת קלה ביותר היוצאת מצומת $s$ יהיה $O(1)$.

נשמור את רשימת הסמיכויות של צומת $s$ <u>במערך</u> $A[i]=w(s,i)$. זמן עדכון למזוג צומת $i$ עם צומת $s$ יהיה $O(degree(i))$ כיוון שפשוט נעבור על רשימת הסמיכויות של צומת $i$ ולכל צומת $j$ המופיע בה נעדכן את $A[j]$.

כמו כן נשמור מערך מצביעים $B[i]$ לקשתות שבערימה כדי לאפשר הוצאתם.

# ניתוח זמנים

מהן הפעולות המתבצעות על הערימה ?

| בכל שלב: | זמן הפעולה |
| --- | --- |
| • מוצאים מינימום | $O(1)$ |
| • קשתות נוספות לערימה | $O(\log n)$ |
| • קשתות מוצאות מהערימה | $O(\log n)$ |
| • משנים משקל של קשת בערימה | $O(\log n)$ |
| כמו כן בכל שלב מעדכנים את $A$ | $O(degree(j))$ |

# ניתוח זמנים

מהן הפעולות המתבצעות על הערימה ?

| בכל שלב: | זמן הפעולה |
|---|---|
| • מוצאים מינימום | $O(1)$ |
| • קשתות נוספות לערימה | $O(\log n)$ |
| • קשתות מוצאות מהערימה | $O(\log n)$ |
| • משנים משקל של קשת בערימה | $O(\log n)$ |
| כמו כן בכל שלב מעדכנים את $A$ | $O(degree(j))$ |

כל קשת נכנסת לערימה או גורמת עדכון פעם אחת לכל היותר.

לכן פעולות הערימה מתבצעות בזמן $O(m \log n)$.

פעולות העדכון של $A$ מתבצעות בזמן כולל של $O(m)$ (סכום הדרגות).

# ניתוח זמנים

מהן הפעולות המתבצעות על הערימה ?

| בכל שלב: | זמן הפעולה |
| --- | --- |
| • מוצאים מינימום | $O(1)$ |
| • קשתות נוספות לערימה | $O(\log n)$ |
| • קשתות מוצאות מהערימה | $O(\log n)$ |
| • משנים משקל של קשת בערימה | $O(\log n)$ |
| כמו כן בכל שלב מעדכנים את $A$ | $O(degree(j))$ |

כל קשת נכנסת לערימה או גורמת עדכון פעם אחת לכל היותר.

לכן פעולות הערימה מתבצעות בזמן $O(m \log n)$.

פעולות העדכון של $A$ מתבצעות בזמן כולל של $O(m)$ (סכום הדרגות).

מתי השיפור עוזר (יחסית לזמן $O(n^2)$)?

# שיפור נוסף אפשרי

בעזרת ערימות פיבונצי ניתן לשפר לזמן $O(m + n \log n)$.

לגרפים דלילים ניתן להגיע אף לסיבוכיות זמן $O(m \log^* n)$.

פרטים בספר הלימוד פרק 24.

Graphs