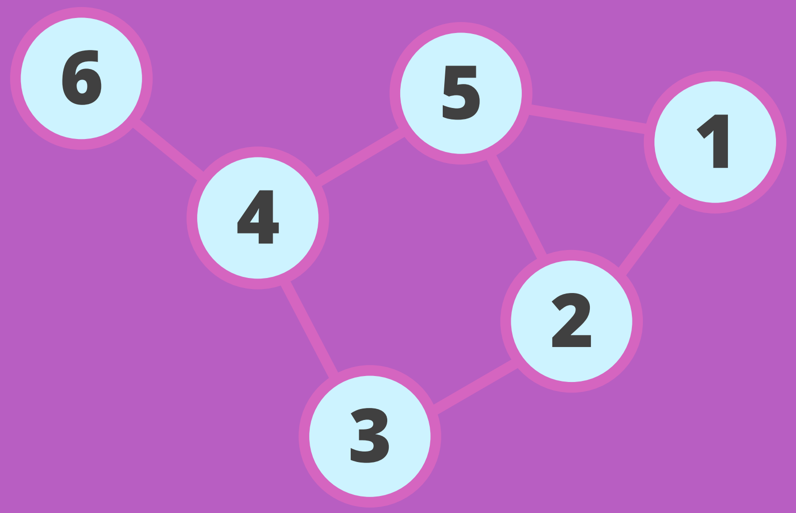
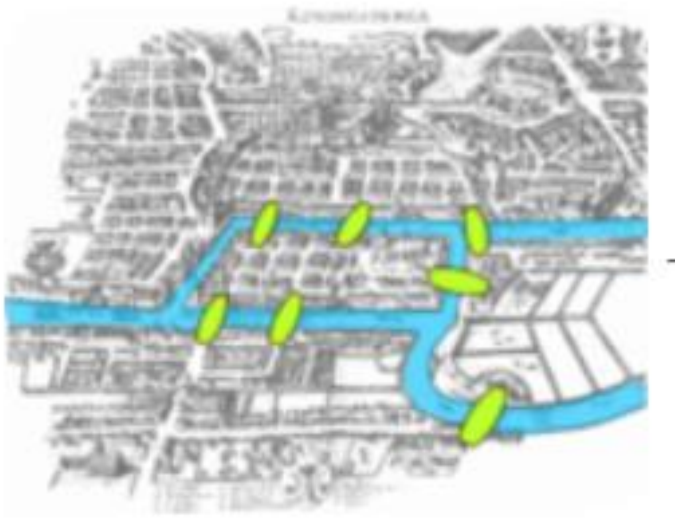
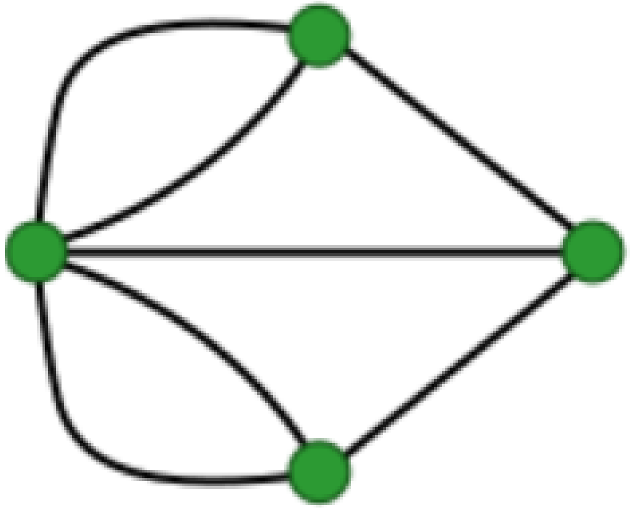
גרפים
Graphs



Dr. Irina Rabaev
SHAMOON COLLEGE OF ENGINEERING

הגשרים של קניגסברג

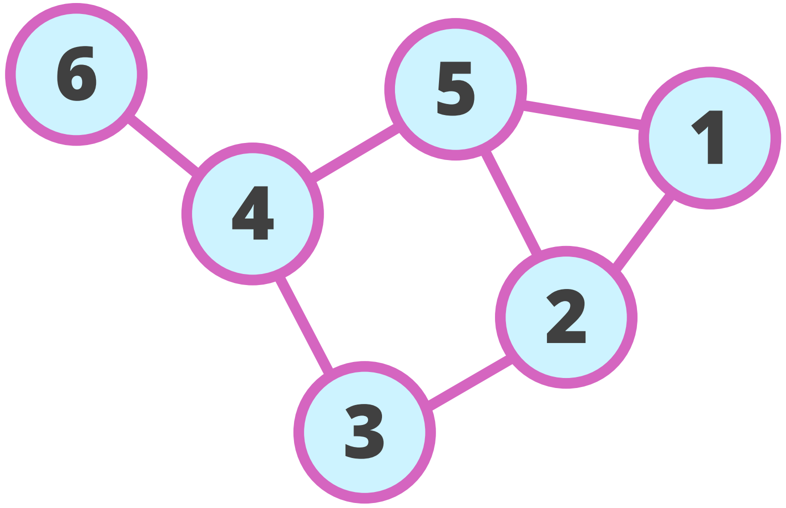




Leonhard Euler

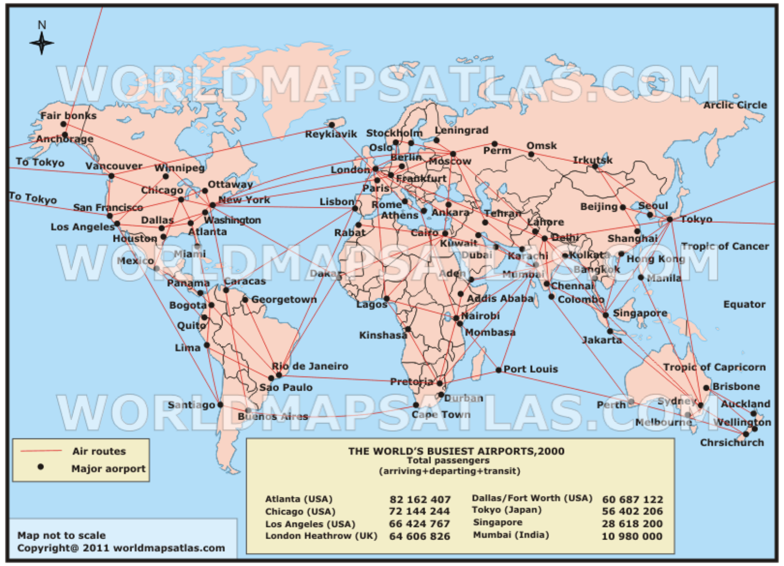
מבוא

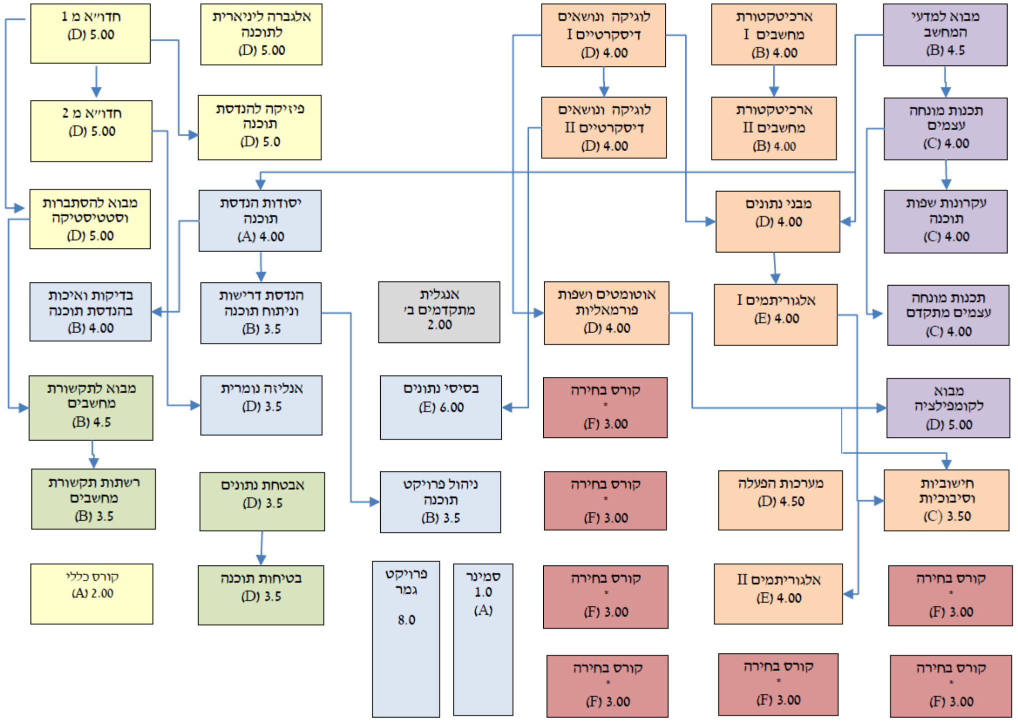
- גרפים הם מבנה נתונים שהשימוש בו חודר יותר ויותר למדעי המחשב, ואלגוריתמים לעבודה עם גרפים הם מיסודות התחום.
- קיימות מאות בעיות מעניינות המוגדרות במונחים של גרפים.




Metropolitana di Milano
M1
M2
M3

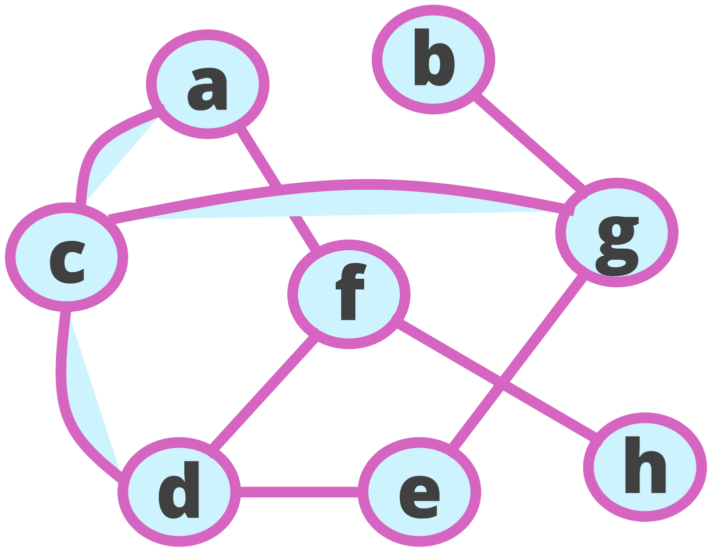

SOCIAL NETWORK


N
WORLDMAPSATLAS.COM
Arctic Circle
Tropic of Cancer
Equator
Tropic of Capricorn
Fair banks
Anchorage
To Tokyo
Vancouver
Winnipeg
Ottaway
Chicago
New York
San Francisco
Washington
Los Angeles
Dallas
Atlanta
Houston
Miami
Mexico
Panama
Caracas
Bogota
Georgetown
Quito
Lima
Rio de Janeiro
Sao Paulo
Santiago
Buenos Aires
Reykiavik
Stockholm
Oslo
Leningrad
London
Berlin
Moscow
Frankfurt
Paris
Perm
Omsk
Irkutsk
Lisbon
Rome
Athens
Ankara
Tehran
Rabat
Cairo
Kuwait
Dubai
Karachi
Lahore
Delhi
Kolkata
Mumbai
Dakar
Aden
Lagos
Addis Ababa
Nairobi
Mombasa
Kinshasa
Chennai
Colombo
Bangkok
Beijing
Seoul
Tokyo
Shanghai
Hong Kong
Manila
Singapore
Jakarta
Port Louis
Pretoria
Durban
Cape Town
Perth
Sydney
Melbourne
Brisbone
Auckland
Wellington
Chrsichurch
Air routes
Major aorport
Map not to scale
Copyright@ 2011 worldmapsatlas.com
THE WORLD'S BUSIEST AIRPORTS,2000
Total passengers
(arriving+departing+transit)
Atlanta (USA) 82 162 407
Chicago (USA) 72 144 244
Los Angeles (USA) 66 424 767
London Heathrow (UK) 64 606 826
Dallas/Fort Worth (USA) 60 687 122
Tokyo (Japan) 56 402 206
Singapore 28 618 200
Mumbai (India) 10 980 000


מבוא למדעי המחשב (B) 4.5
תכנות מונחה עצמים (C) 4.00
עקרונות שפות תוכנה (C) 4.00
תכנות מונחה עצמים מתקדם (C) 4.00
מבוא לקומפילציה (D) 5.00
חישוביות וסיבוכיות (C) 3.50
קורס בחירה * (F) 3.00
קורס בחירה * (F) 3.00
ארכיטקטורת מחשבים I (B) 4.00
ארכיטקטורת מחשבים II (B) 4.00
מבני נתונים (D) 4.00
אלגוריתמים I (E) 4.00
מערכות הפעלה (D) 4.50
אלגוריתמים II (E) 4.00
קורס בחירה * (F) 3.00
לוגיקה ונושאים דיסקרטיים I (D) 4.00
לוגיקה ונושאים דיסקרטיים II (D) 4.00
אוטומטים ושפות פורמאליות (D) 4.00
קורס בחירה * (F) 3.00
קורס בחירה * (F) 3.00
קורס בחירה * (F) 3.00
קורס בחירה * (F) 3.00
אנגלית מתקדמים ב׳ 2.00
בסיסי נתונים (E) 6.00
ניהול פרויקט תוכנה (B) 3.5
פרויקט גמר 8.0
סמינר 1.0 (A)
אלגברה ליניארית לתוכנה (D) 5.00
פיזיקה להנדסת תוכנה (D) 5.0
יסודות הנדסת תוכנה (A) 4.00
הנדסת דרישות וניתוח תוכנה (B) 3.5
אנליזה נומרית (D) 3.5
אבטחת נתונים (D) 3.5
בטיחות תוכנה (D) 3.5
חדו״א מ 1 (D) 5.00
חדו״א מ 2 (D) 5.00
מבוא להסתברות וסטטיסטיקה (D) 5.00
בדיקות ואיכות בהנדסת תוכנה (B) 4.00
מבוא לתקשורת מחשבים (B) 4.5
רשתות תקשורת מחשבים (B) 3.5
קורס כללי (A) 2.00

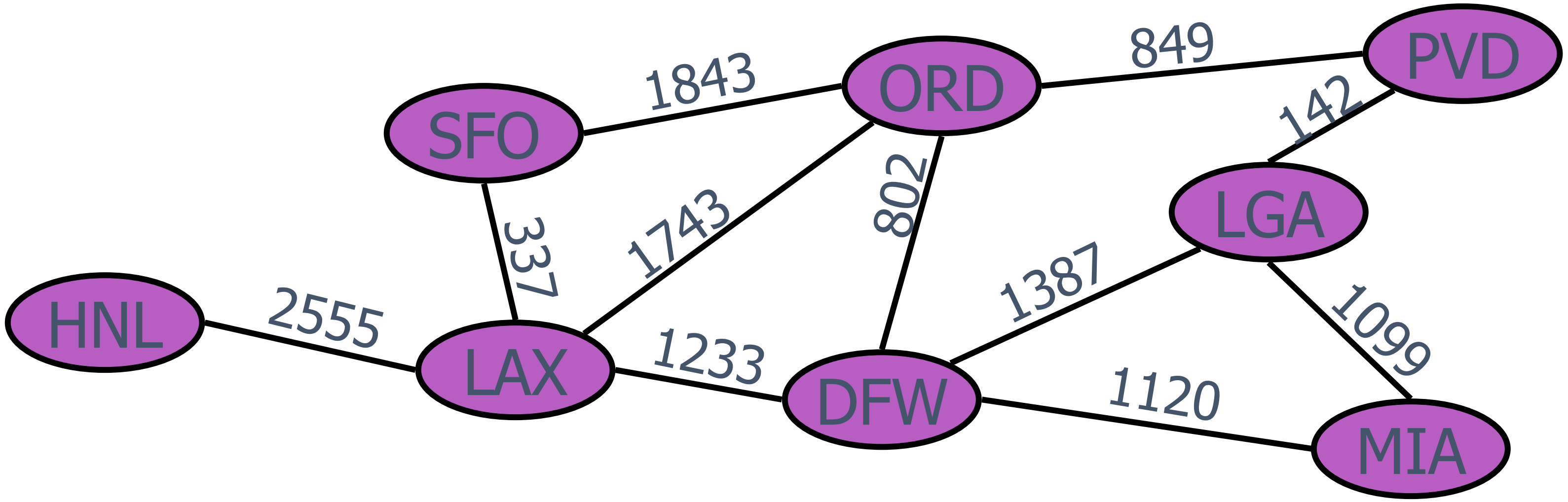
הגדרה

- גרף G=(V, E) הוא זוג (V, E) כאשר
  - V – קבוצת **הקודקודים**
  - E - קבוצת הזוגות הקודקודים הנקראים **קשתות**
- גרף הוא הדרך לייצג יחסים (קשרים)
- דוגמה:
  - V={a, b, c, d, e, f, g, h}
  - E = {(a,c), (a,f), (b,g), (c,d), (c,g),(d,f), (d,e),(e,g),(f, h)}



דוגמה לשימוש

- קודקוד מייצג נמל תעופה ומסומן ע"י שלוש אותיות
- הקשת מייצגת דרך טיסה בין שני נמלי תעופה ומאחסנת את מרחק ביניהם



קשתות

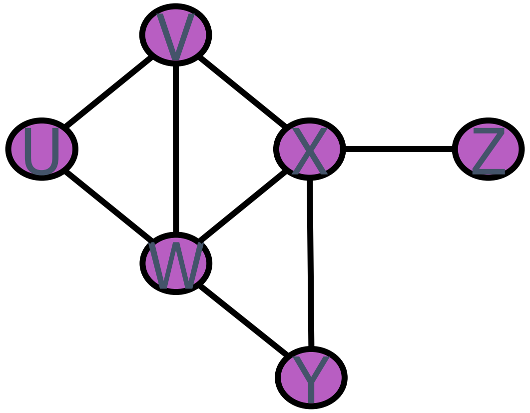
- **קשת מכוונת (u, v) – זוג סדור**
  - u מקור
  - v יעד
  - למשל, כיוון הטיסה

- **קשת לא מכוונת (u, v) – זוג לא סדור**
  - למשל, מרחק בין שני נמלי תעופה

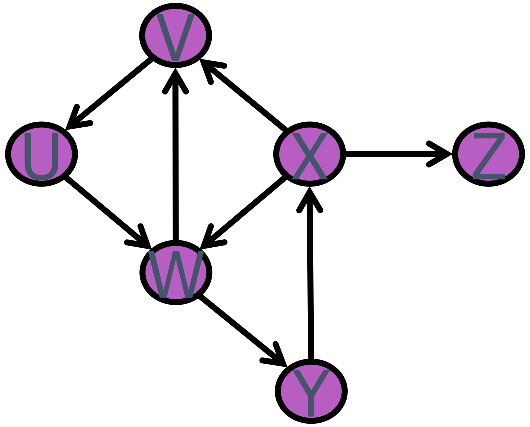
- **גרף מכוון – כל הקשתות מכוונות**
- **גרף לא מכוון – כל הקשתות לא מכוונות**

קודקודים סמוכים, דרגת הקודקוד

- אם (u, v) קשת בגרף G = (V, E), אז v סמוך (adjacent) ל-u
- דרגה של קודקוד בגרף בלתי מכוון – מספר הקשתות המחברות אותו עם קודקודים אחרים
  - degree(x) = 4

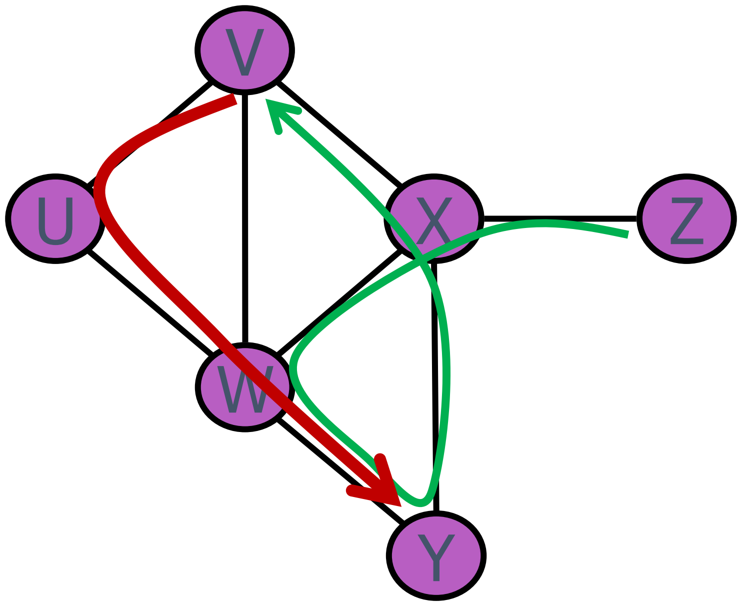


- בגרף מכוון
  - **דרגת הכניסה** in-degree
  - **דרגת היציאה** out-degree
    - in-degree(x) = 1
    - out-degree(x) = 3



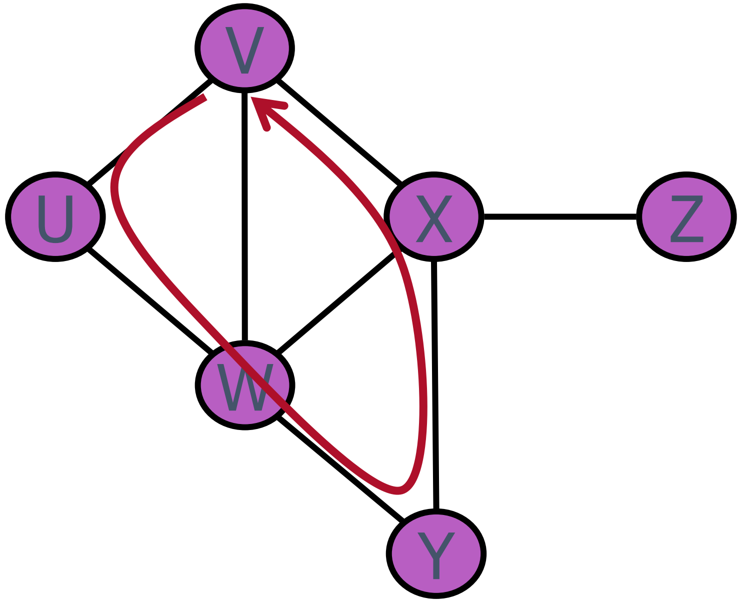
מסלול

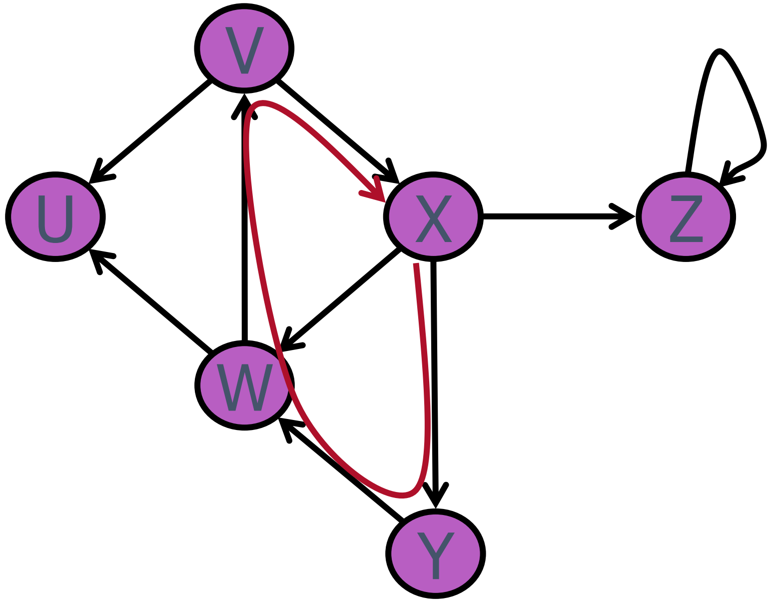
- בגרף $G=(V,E)$, **מסלול באורך k** מקודקוד u לקודקוד u' הוא סדרת קודקודים $(v_0, v_1, v_2, \ldots, v_k)$ כך ש-$u=v_0$ ו-$u'=v_k$ ו- $(v_{i-1}, v_i)\in E$.
- **אורך המסלול** – מספר הקשתות במסלול
  - (v, u, w, y)
  - אורך המסלול = 3
- **מסלול פשוט** – כל הקודקודים שונים
  - (v, u, w, y) – פשוט
  - (z, x, y, x, v) – לא פשוט



מעגל

- מסלול $(v_0, v_1, v_2, \ldots, v_k)$ יוצר **מעגל** אם $v_0 = v_k$
- <u>בגרף מכוון</u> – מעגל מכיל לפחות קשת אחת
- <u>בגרף בלתי מכוון</u> - מעגל מכיל לפחות 3 קשתות

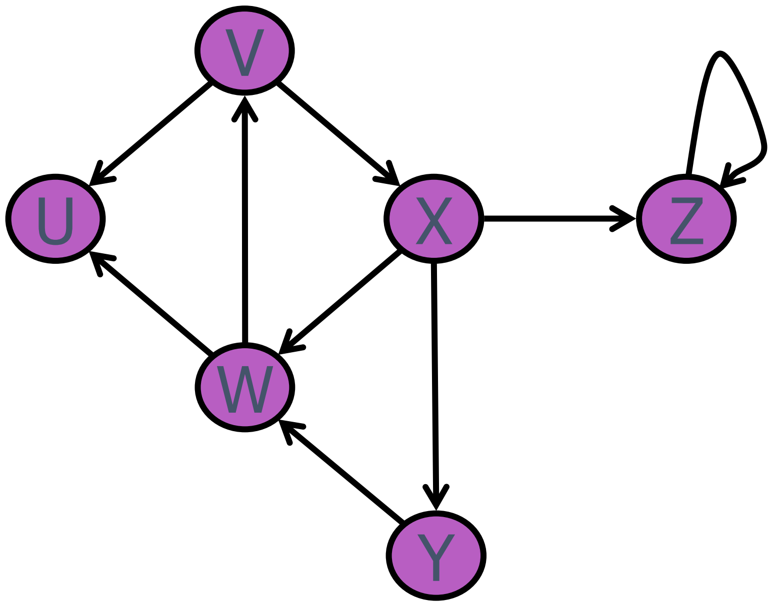




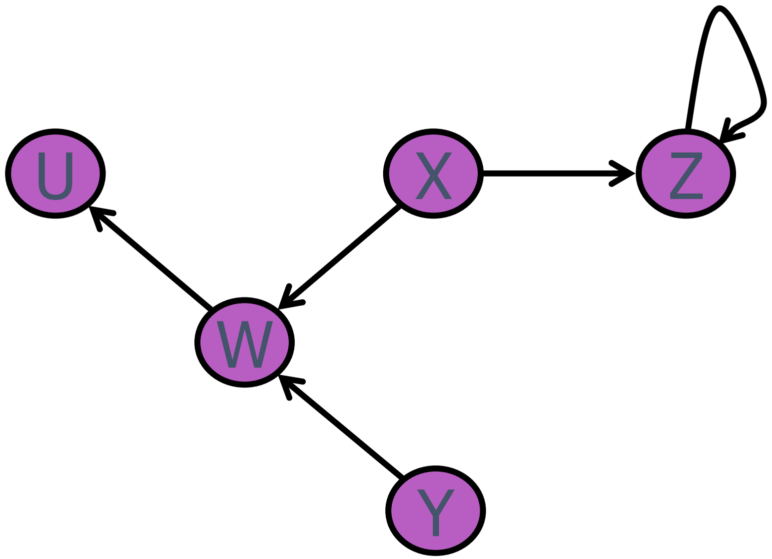
(z,z) – לולאה עצמית

תת-גרף (Subgraph)

• גרף $G'=(V', E')$ הוא תת-גרף של $G=(V,E)$ אם $V' \subseteq V$ ו- $E' \subseteq E$



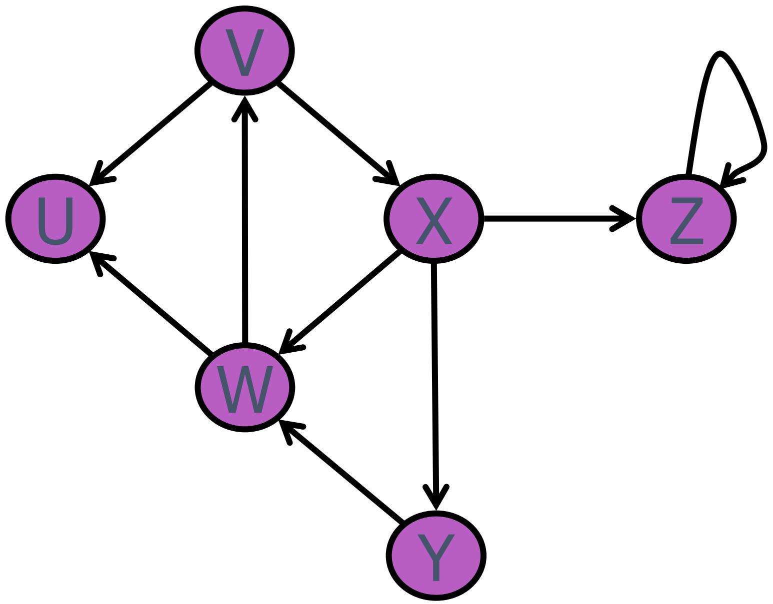
G=(V, E)



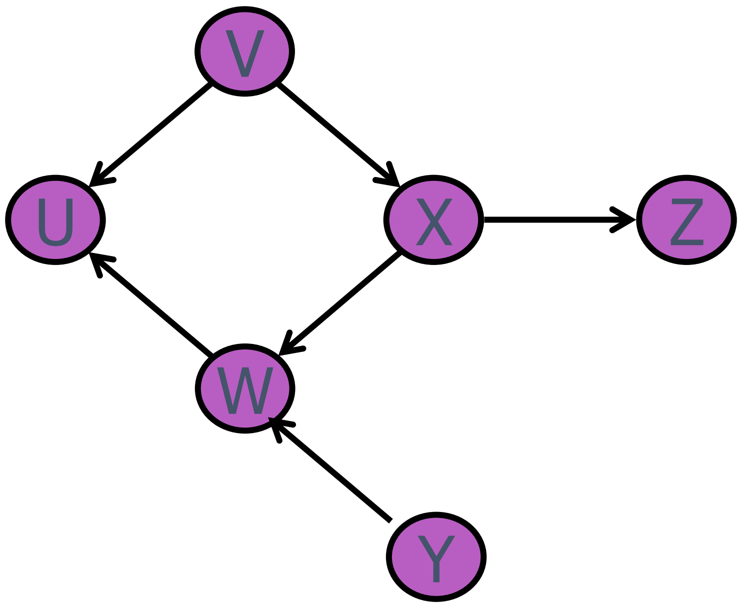
G'=(V', E')

תת-גרף פורש
(spanning subgraph)

- גרף $G'=(V', E')$ הוא תת-גרף פורש של $G=(V,E)$ אם $V' = V$



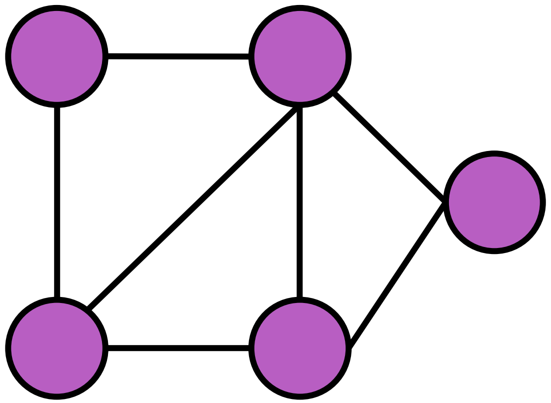
$G=(V, E)$



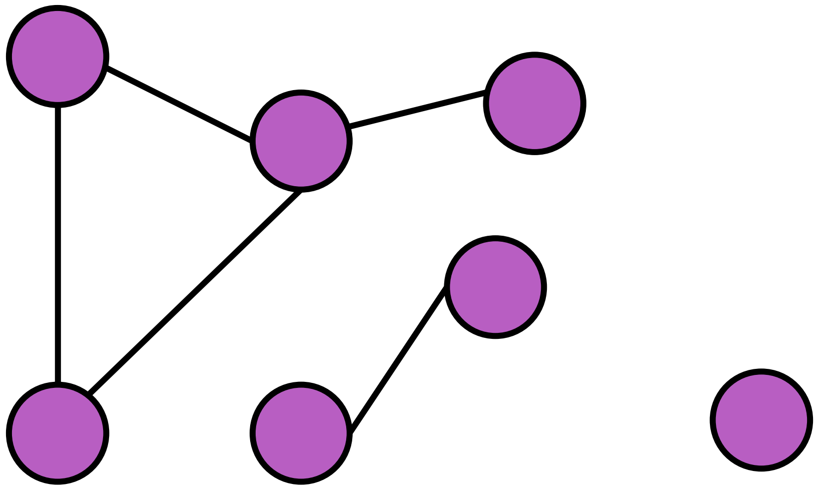
$G'=(V', E')$ תת גרף פורש של G

גרף קשיר (connected graph)

- גרף לא מכוון נקרא **קשיר** (connected) אם כל זוג קודקודים מקושר ע"י מסלול (כלומר קיים מסלול מכל קודקוד לכל קודקוד אחר)
- **רכיב קשירות** (connected component) של גרף לא מכוון G הוא תת גרף קשיר מקסימאלי של G (כלומר, כל תת-גרף של G שיכיל אותו כבר לא יהיה קשיר).



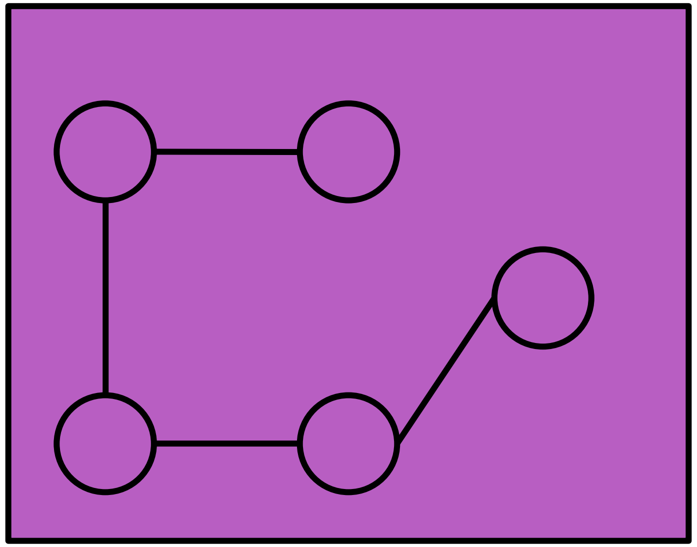
גרף קשיר



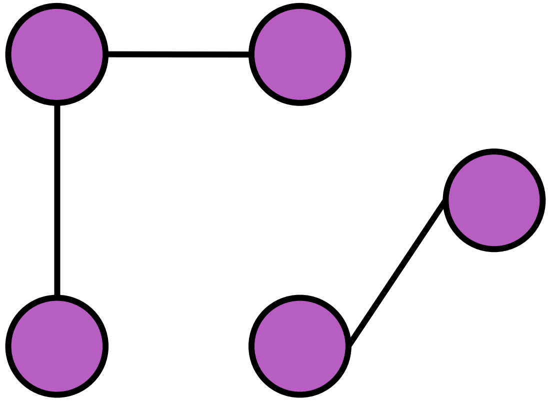
גרף לא קשיר, שלושה רכיבי קשירות

עץ ויער (tree and forest)

- עץ (tree)– גרף בלתי מכוון קשיר חסר מעגלים.
- יער (forest) – גרף בלתי מכוון חסר מעגלים (ייתכן שאינו קשיר).



עץ



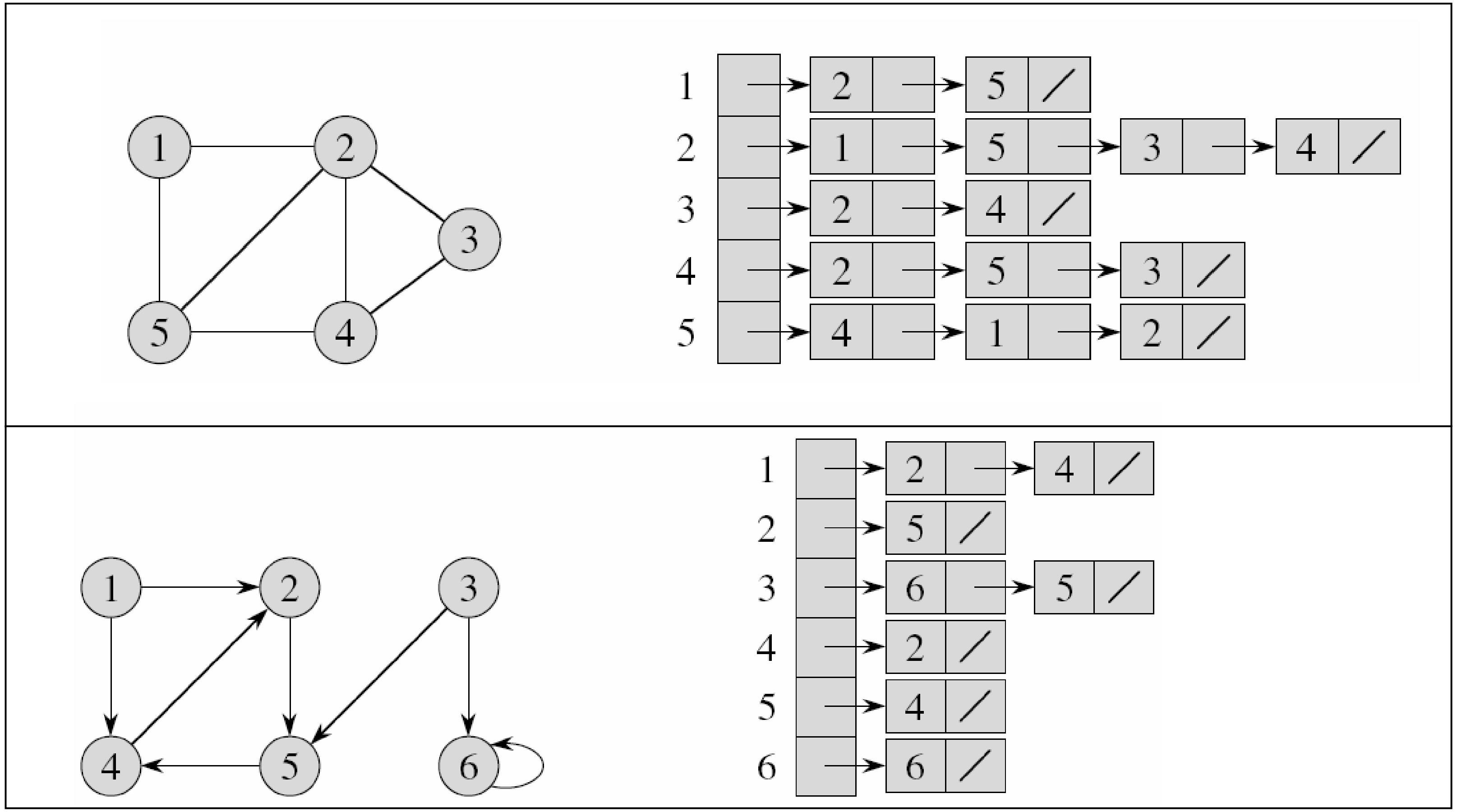
יער

ייצוג של גרפים

- קיימות שתי דרכים מקובלות לייצוג של גרפים
  - רשימות סמיכות (adjacency-list representation)
  - מטריצת סמיכויות (adjacency-matrix representation)

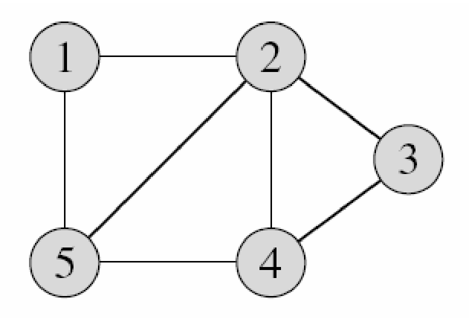
ייצוג של גרפים – רשימות סמיכות

- מערך Adj של $|V|$ רשימות, אחת עבור כל קודקוד ב-V.
- עבור כל $u \in V$, רשימת הסמיכות Adj[u] מכילה את כל הקודקודים הסמוכים ל-u

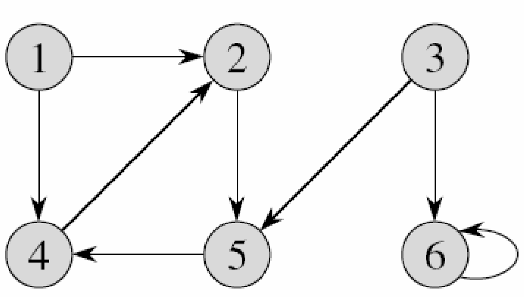


ייצוג של גרפים – מטריצת סמיכויות

- מניחים שקודקודים ממוספרים |V|,...,1,2 בסדר שרירותי
- מטריצה $A=(a_{ij})$ שמימדיה |V|x|V|



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

$$a_{ij} = \begin{cases} 1 & (i, j) \in E \\ 0 & otherwise \end{cases}$$

השוואה בין רשימות סמיכות ומטריצת סמיכויות

- **רשימות סמיכות**
  - לייצוג גרפים דלילים (sparse) – אלה שעבורם $|E|$ קטן בהרבה מ-$|V^2|$
- **מטריצת סמיכויות**
  - לייצוג גרפים צפופים (dense) – $|E|$ קרוב ל-$|V^2|$
  - יש צורך לבדוק במהירות האם קיימת קשת בין שני קודקודים

השוואה בין רשימות סמיכות ומטריצת סמיכויות

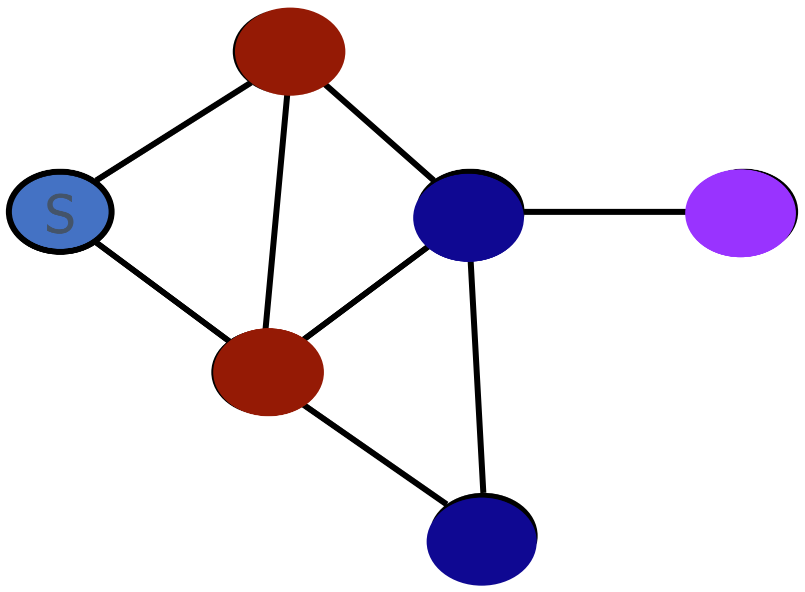
$G=(V,E)$ $\|V\|=n,\ \|E\|=m$	רשימות סמיכות	מטריצת סמיכויות
מקום	$O(n+m)$	$O(n^2)$
בדיקה האם $(v,u)\in E$	$O(degree(v))$	$O(1)$
מעבר על כל הקשתות הסמוכות לקודקוד v	$O(degree(v))$	$O(n)$
הכנסת קשת חדשה (u, v)	$O(1)$	$O(1)$
מחיקת קשת (u,v)	$O(degree(v))$	$O(1)$

סריקה של גרף

- סריקת גרף - מעבר שיטתי על קשתות הגרף לצורך ביקור בקדקודיו
- אלגוריתם הסורק גרף יכול לגלות דברים רבים על מבנהו

חיפוש לרוחב
(BFS) Breadth-First Search

- שלבי עבודה של האלגוריתם:
  1. מגלה את כל הקודקודים הנמצאים במרחק 1 מ-s
  2. אחר כך את כל הקודקודים הנמצאים במרחק 2 מ-s
  3. אחר כך את כל הקודקודים הנמצאים במרחק 3 מ-s
  4. ...



חיפוש לרוחב
(BFS) Breadth-First Search

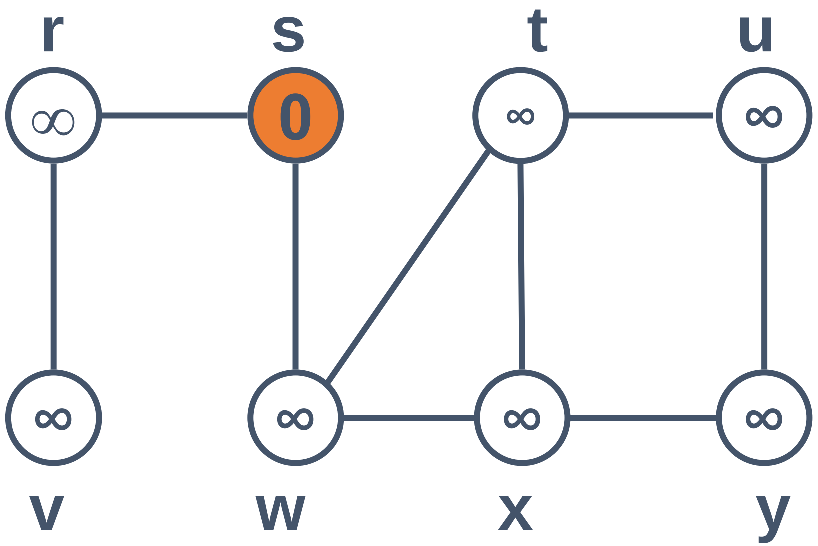
- **פועל גם על גרפים מכוונים וגם על בלתי מכוונים**
- **בהינתן גרף G=(V,E) וקדקוד מסוים s המשמש כמקור (source), אלגוריתם BFS**
  - מגלה את כל הקודקודים שניתן להגיע אליהם מ-s
  - מחשב מסלול קצר ביותר (מספר המינימאלי של קשתות) מ-s לכל הקודקודים שניתן להגיע אליהם מ-s
  - בונה "עץ רוחב" ששורשו s

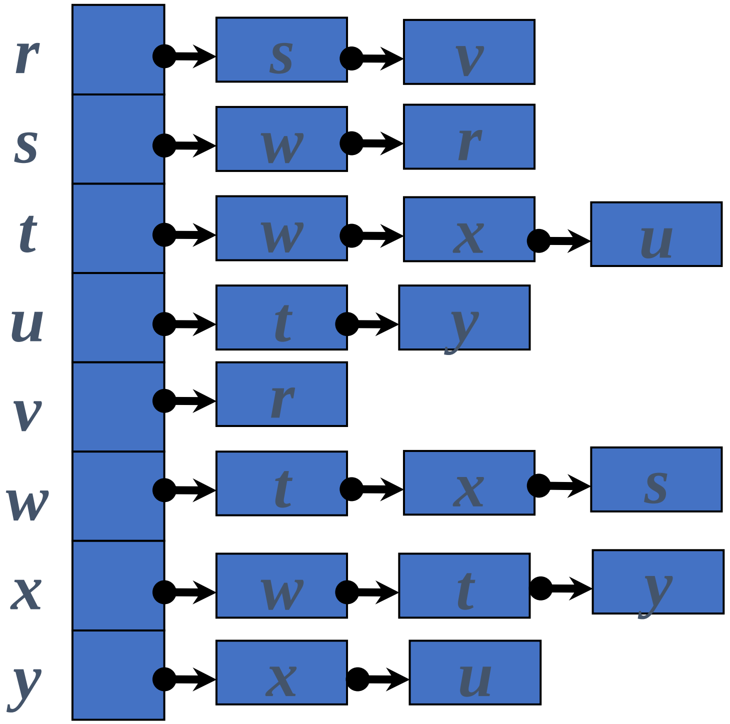
סיווג קדקודים – BFS

- קדקוד שטרם התגלה
- קדקוד שהתגלה אבל לא סיימנו טיפול בו
- קדקוד שהתגלה וסיימנו טיפול בו

- האלגוריתם משתמש בתור FIFO לניהול קבוצת הקודקודים האפורים

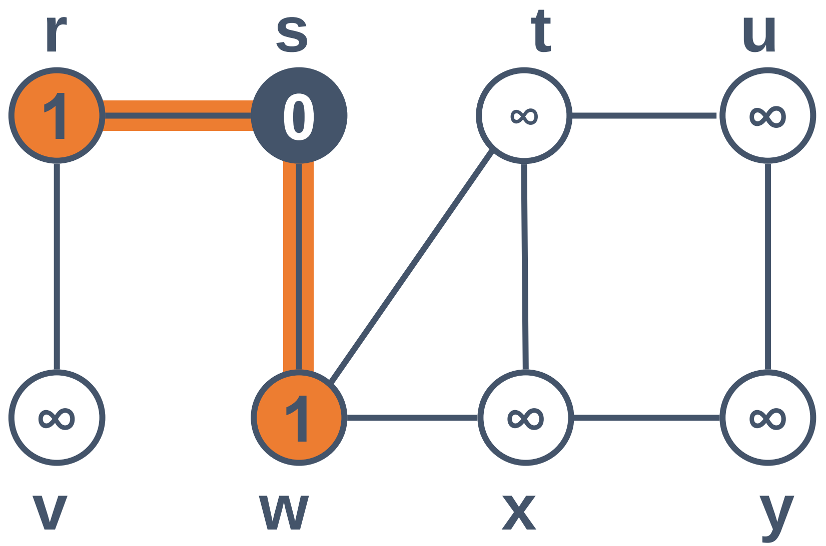
דוגמה

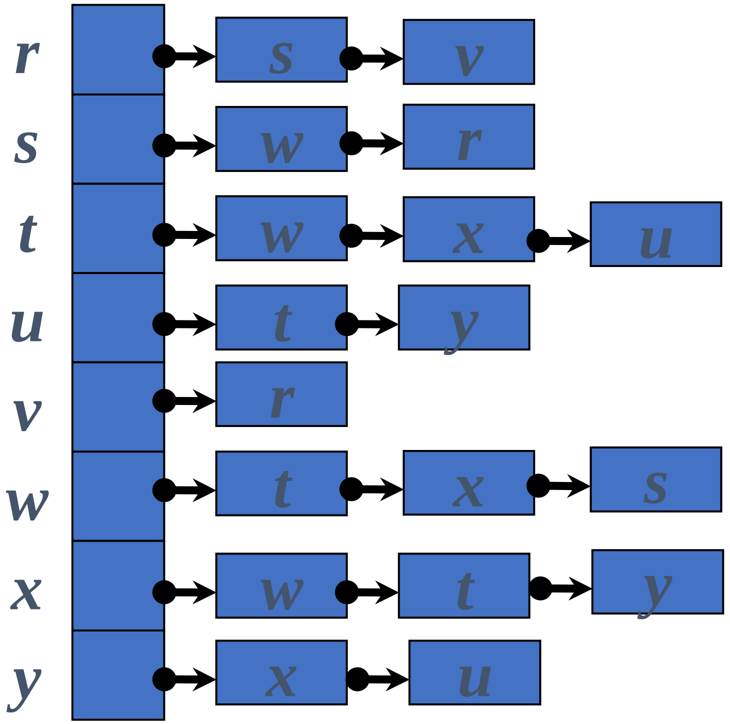




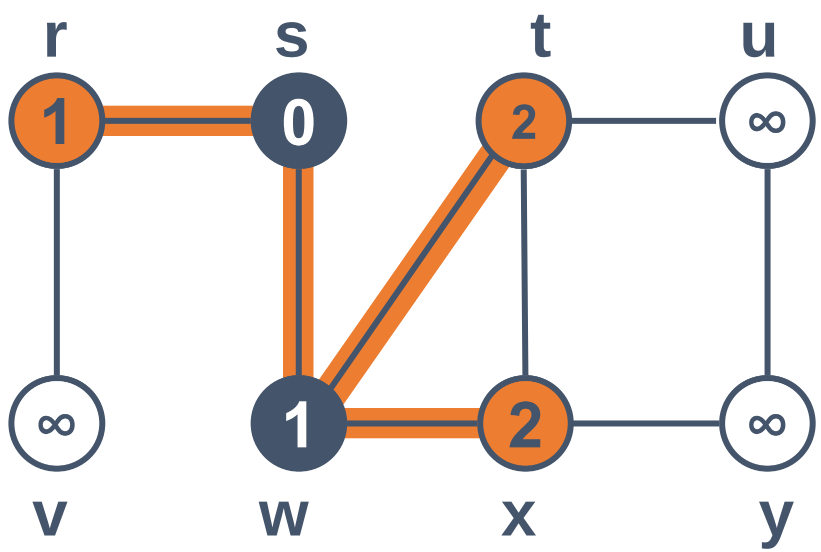
Q

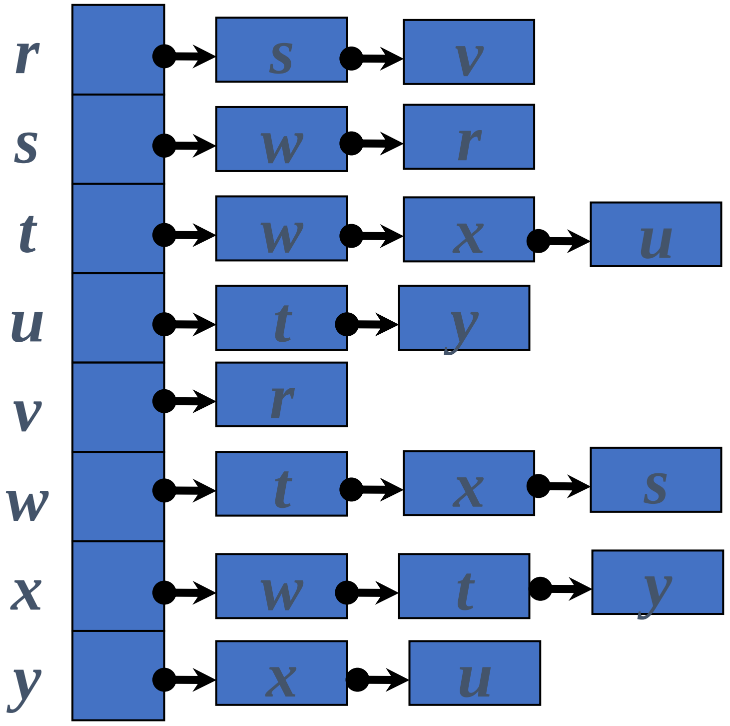
דוגמה



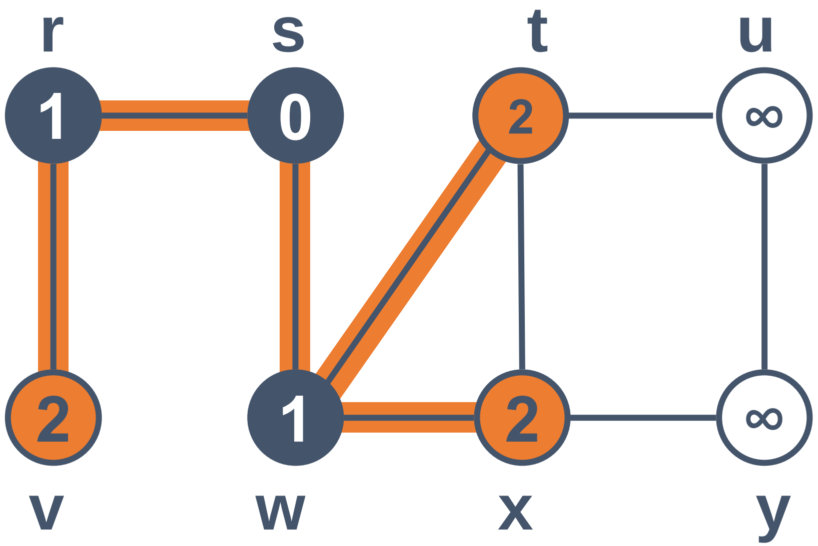


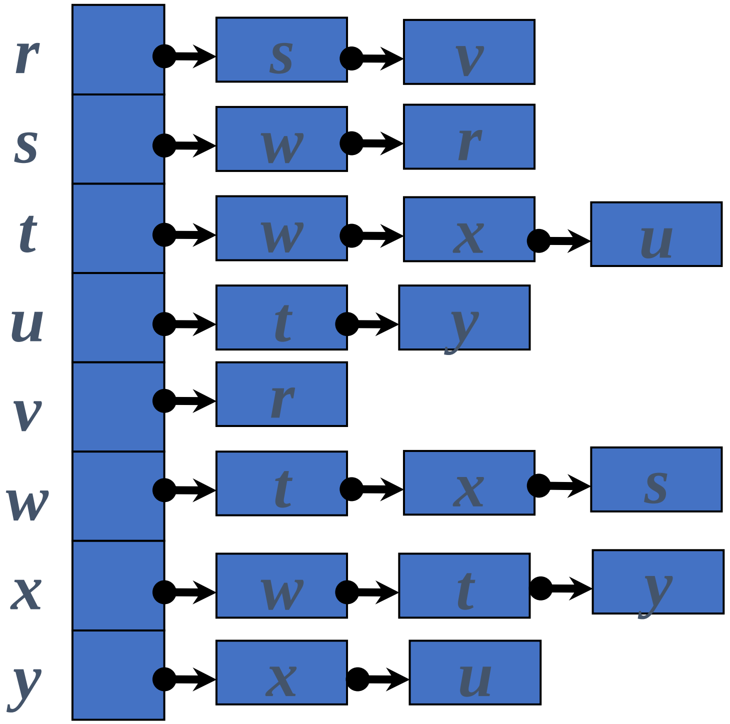
דוגמה



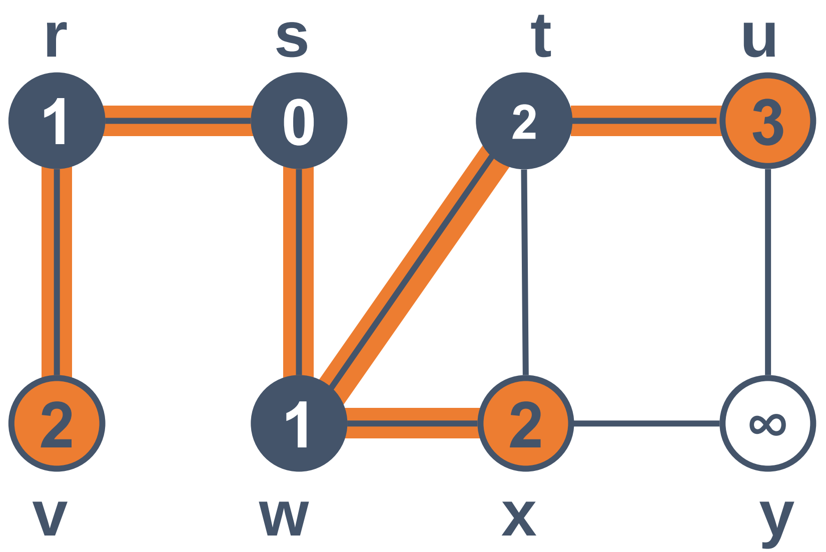


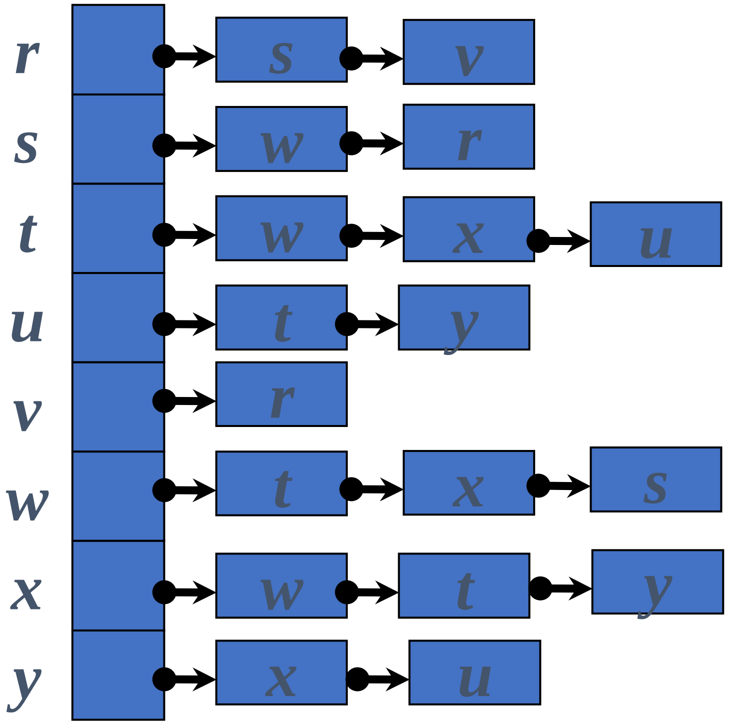
דוגמה





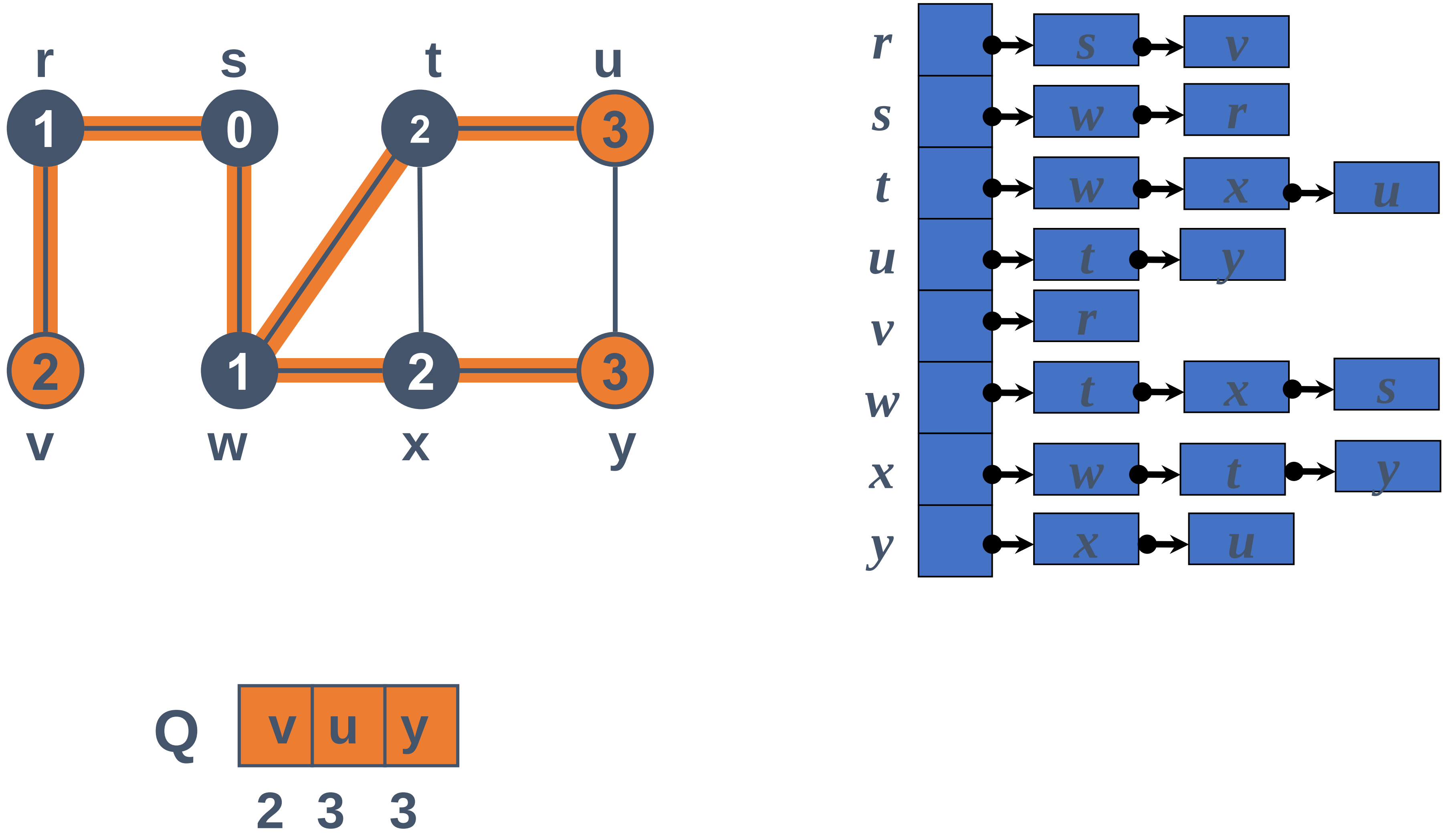
דוגמה



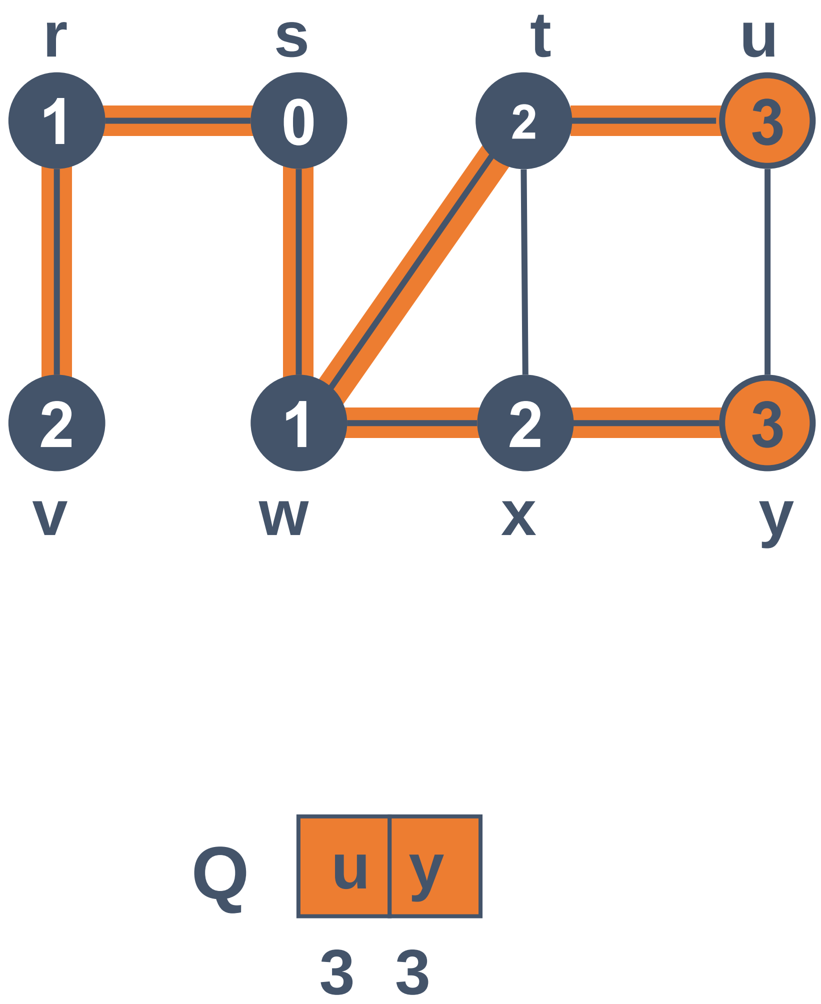


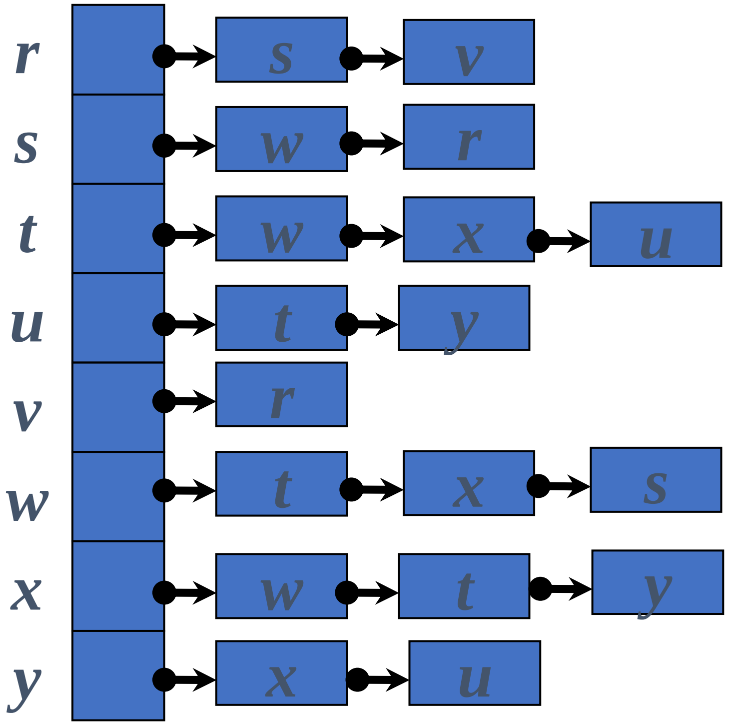
Q	x	v	u
	2	2	3

דוגמה

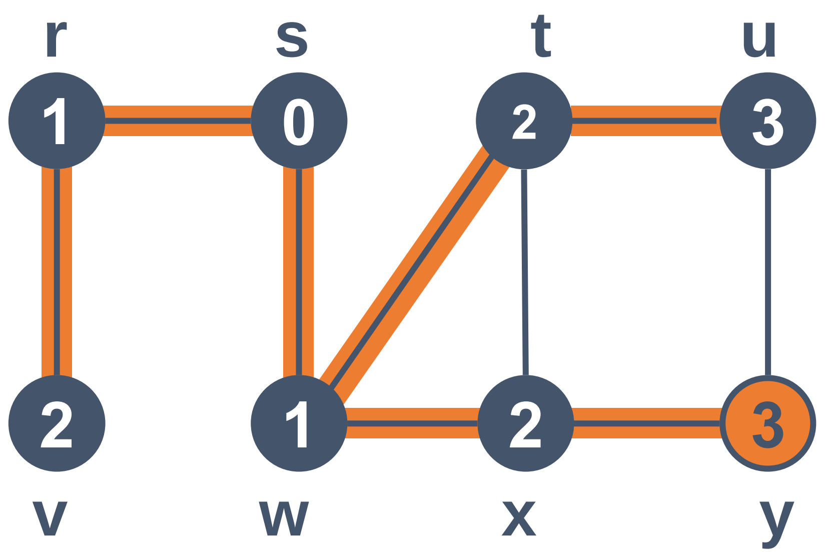


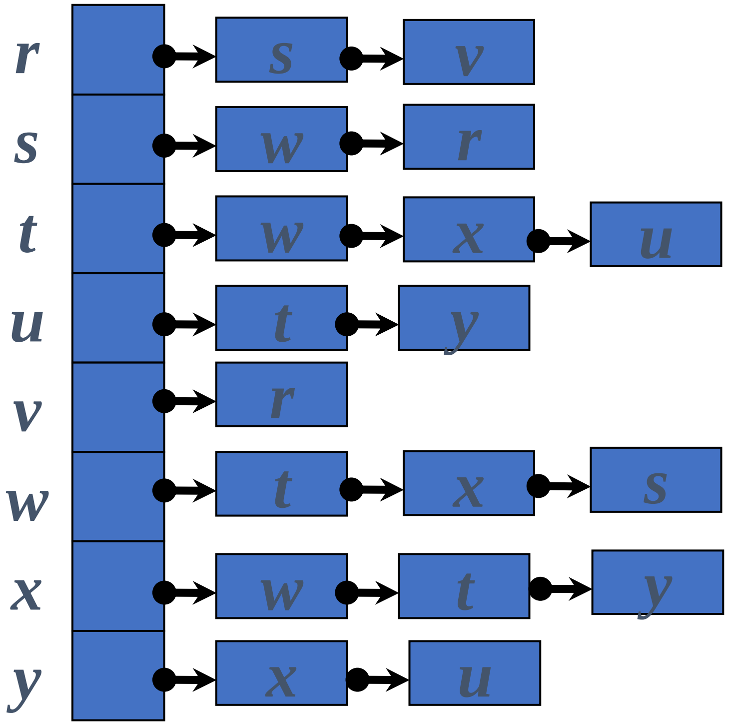
דוגמה



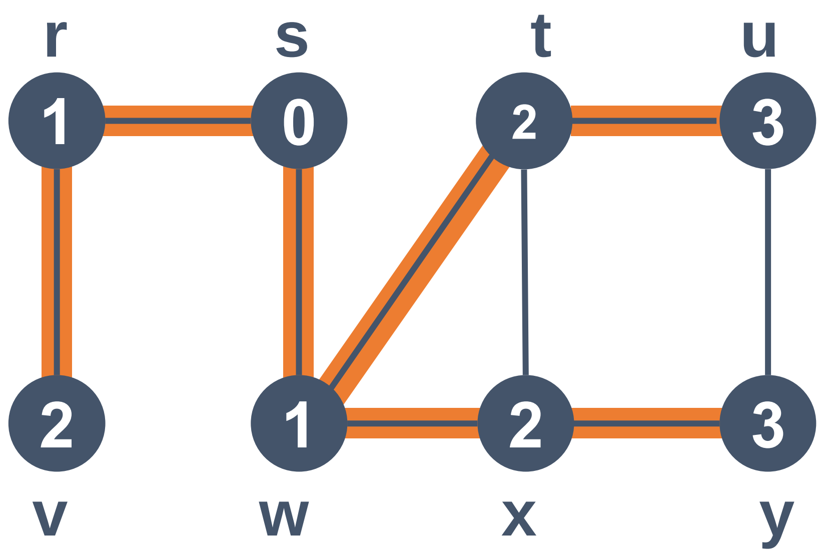


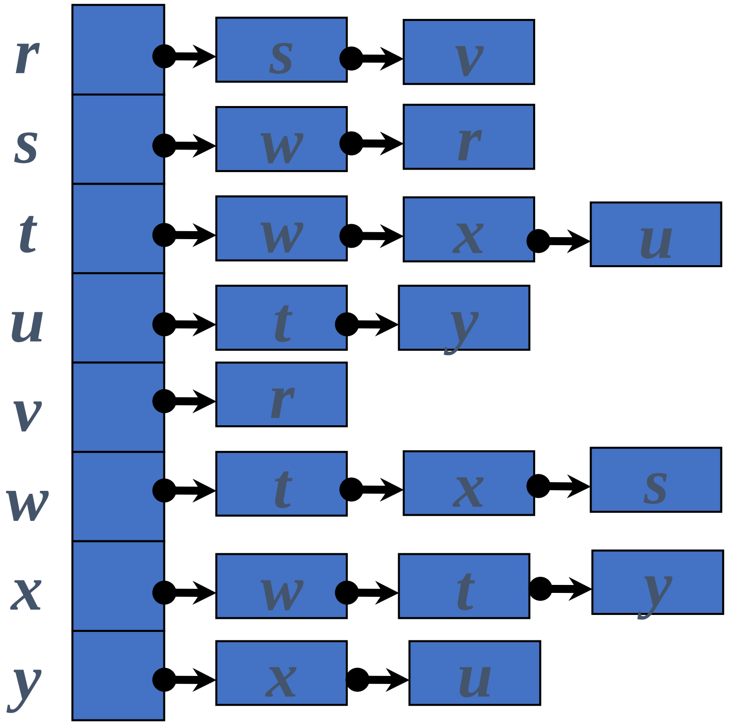
דוגמה





דוגמה





Q $\emptyset$

BFS

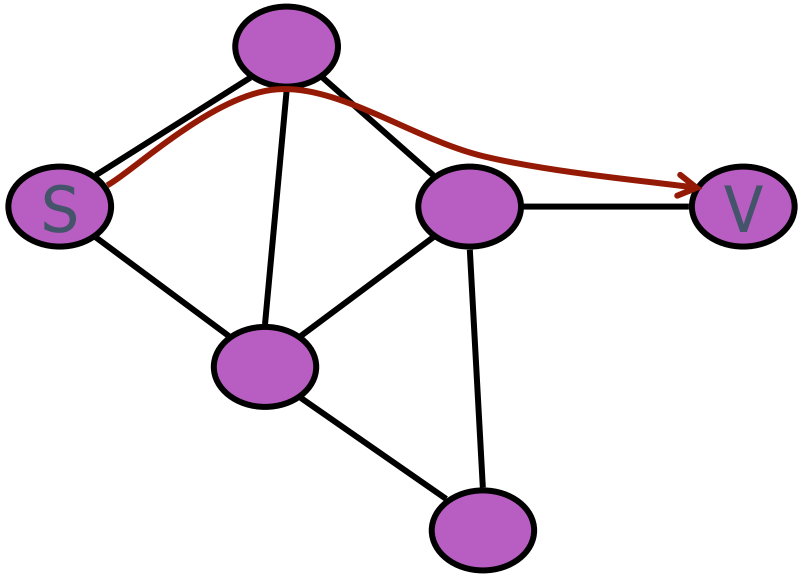
```
BFS(G=(V, E), s)
// d[u] – distance from s to u
// π[u] - predecessor of u
for each vertex u∈V–{s}
        color[u] ← WHITE
        d[u] ← ∞
        π[u] ← NULL
color[s] ← GRAY
d[s] ← 0
π[s] ← NULL
Q ← Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    u ← DEQUEUE(Q)
    for each v ∈ Adj[u]
        if color[v] = WHITE
            color[v] ← GRAY
            d[v] ← d[u] + 1
            π[v] ← u
            ENQUEUE(Q, v)
    color[u] ← BLACK
```

זמן ריצה O(V + E)

אם גרף מיוצג על ידי רשימות סמיכות

מסלולים קצרים ביותר

- נגדיר $\delta(s, v)$
  - **אורך המסלול הקצר ביותר** מ-s ל-v (מספר מינימאלי של קשתות)
  - ∞ אם לא קיים מסלול מ-s ל-v
- **מסלול באורך $\delta(s, v)$ מ-s ל-v נקרא מסלול קצר ביותר**

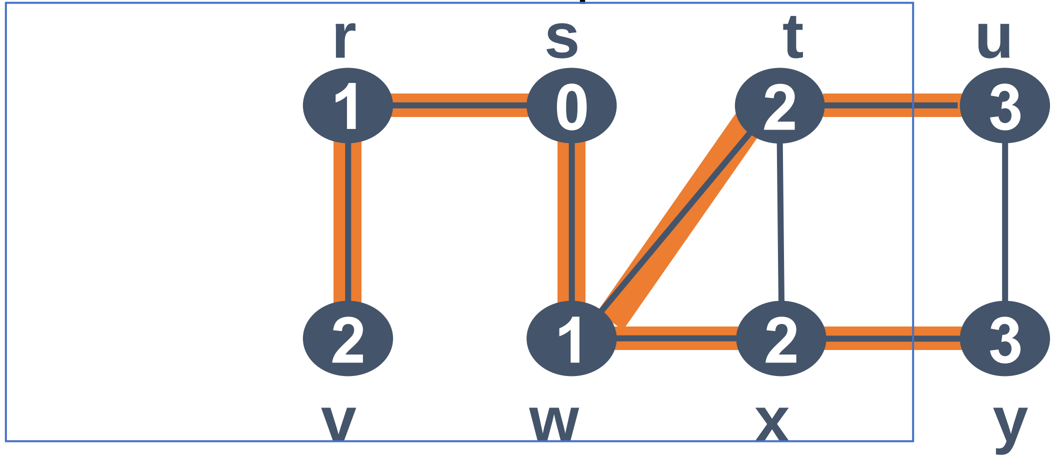


$\delta(s, v) = 3$

מסלולים קצרים ביותר

משפט: בהינתן גרף G=(V,E) וקדקוד מקור s, BFS

1. מגלה את כל הקודקודים שניתן להגיע אליהם מ-s
2. מחשב מסלול קצר ביותר מ-s לכל הקודקודים שניתן להגיע אליהם מ-s, כלומר $d[v] = \delta(s, v)$ לכל קודקוד v.
3. בונה "עץ רוחב" ששורשו s, המכיל את כל הקדקודים שניתן להגיע אליהם מ-s. המסלול מ-s ל-v בעץ הרוחב הוא מסלול באורך $\delta(s, v)$.



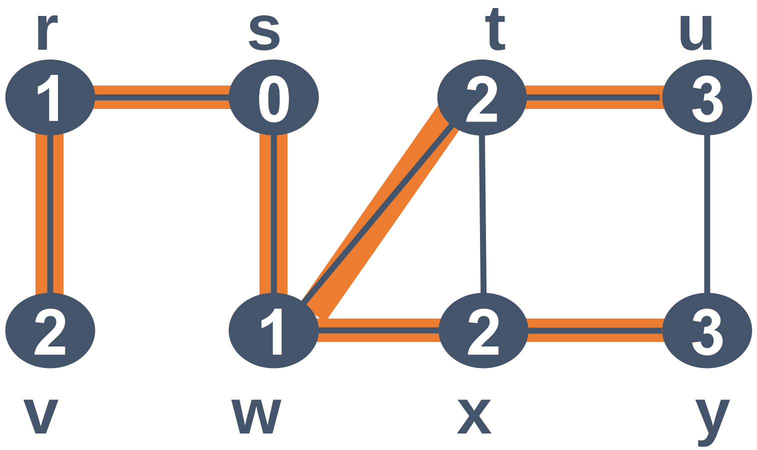
עץ רוחב

- נגדיר $G_\pi=(V_\pi, E_\pi)$
  תת-גרף הקודמים (predecessor subgraph) של G:

$$V_\pi=\{v\in V \mid \pi[v] \neq NULL\} \cup \{s\}$$

$$E_\pi=\{(\pi[v], v)\in E \mid v \in V_\pi-\{s\} \}$$

- כאשר BFS מופעל על גרף $G = (V, E)$, הוא בונה את השדות π כך שתת-גרף הקודמים $G_\pi=(V_\pi, E_\pi)$ הוא עץ רוחב



מסלול קצר ביותר מ-s ל-v

```
PRINT-PATH(G, s, v)
if v = s
    print s
else if π[v] = NULL
    print "no path from" s "to" v "exists"
else PRINT-PATH(G, s, π[v])
     print v
```

זמן ריצה ?

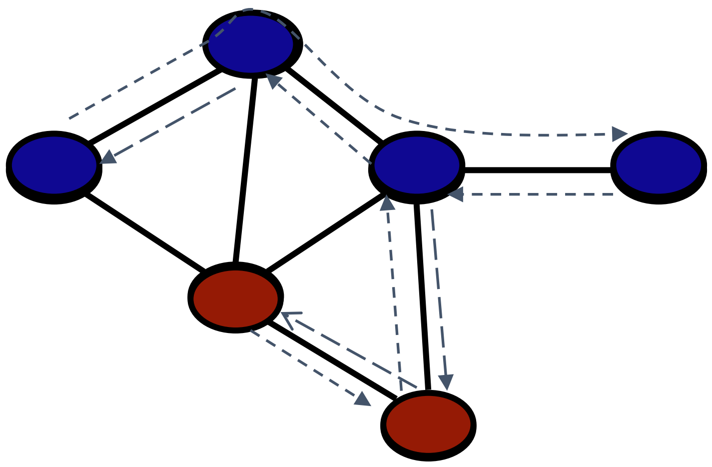
חיפוש לעומק (Depth-First Search) DFS

- **חיפוש לעומק (DFS) הוא אלגוריתם לסריקת הגרפים.**
- **פועל גם על גרפים מכוונים וגם על בלתי מכוונים**
- **בהינתן גרף G=(V,E), אלגוריתם DFS**
  - מבקר בכל הצמתים וקשתות של G
  - בודק האם G קשיר
  - מחשב רכיבי קשירות של G
  - מחשב "יער פורש" של G

האסטרטגיה

- **לחפש "עמוק יותר" בגרף ככל שהדבר אפשרי**

1. נבדקות קשתות של הקדקוד v שהוא הקדקוד האחרון שהתגלה עד עכשיו
2. לאחר שנבדקו כל הקשתות היוצאות מ-v, החיפוש "נסוג" וממשיך בבדיקת הקשתות היוצאות מקדקוד שממנו התגלה v
3. אם נותרו קודקודים שטרם התגלו חוזרים על התהליך



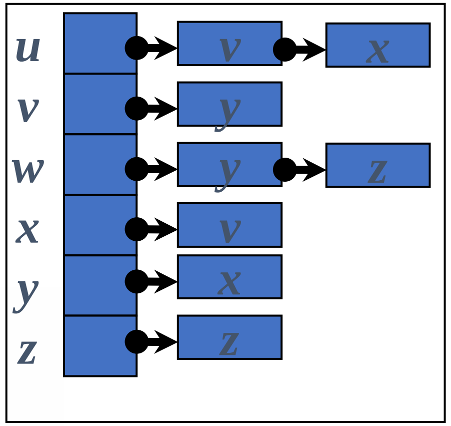
סיווג קדקודים – DFS

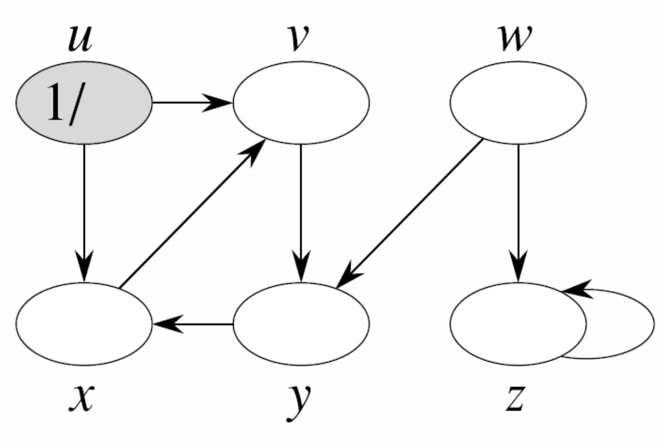
- קדקוד שטרם התגלה
- קדקוד שהתגלה אבל לא סיימנו טיפול בו
- קדקוד שהתגלה וסיימנו טיפול בו

בכל קודקוד שומרים **חותמות הזמן (timestamps)**

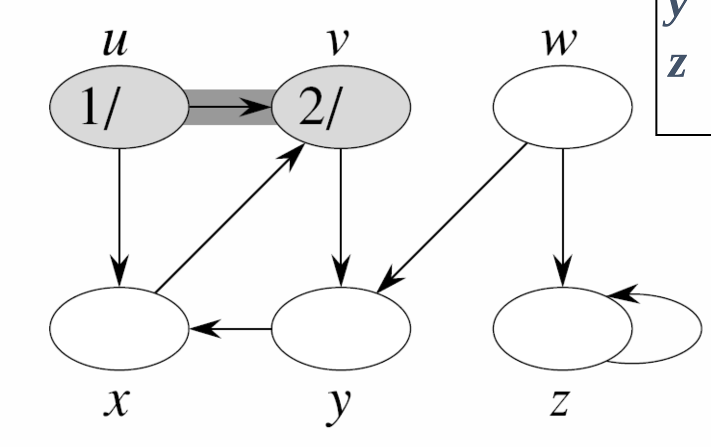
- $d[v]$ – מועד גילוי של v
- $f[v]$ – מועד סיום הטיפול ב-v
- $d[v], f[v]$ הם ערכים שלמים בין 1 ל-$2|V|$
- $d[v] < f[v]$ לכל קודקוד v

דוגמה

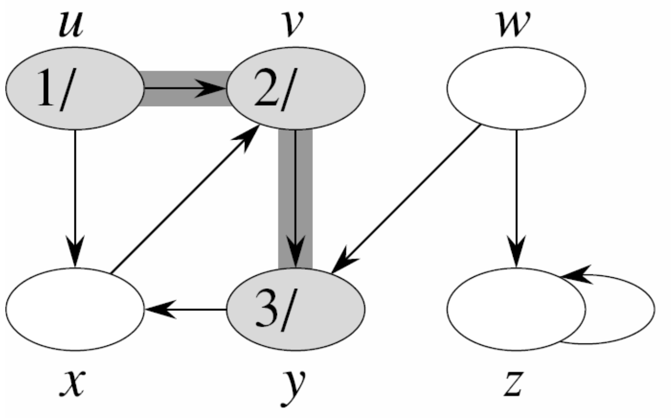




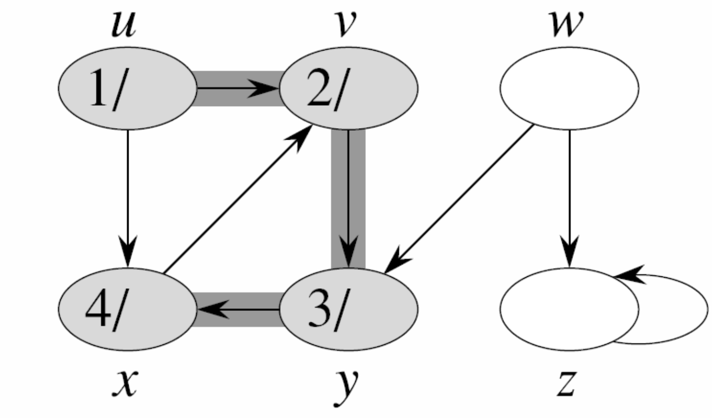
(a)



(b)

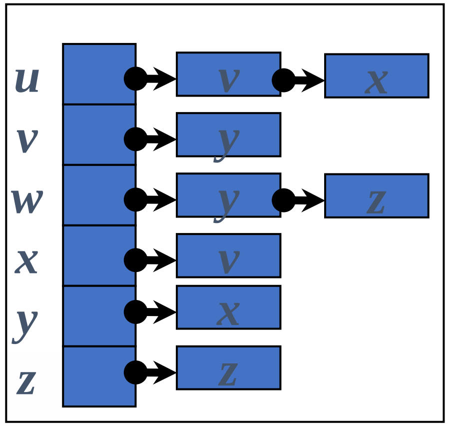


(c)

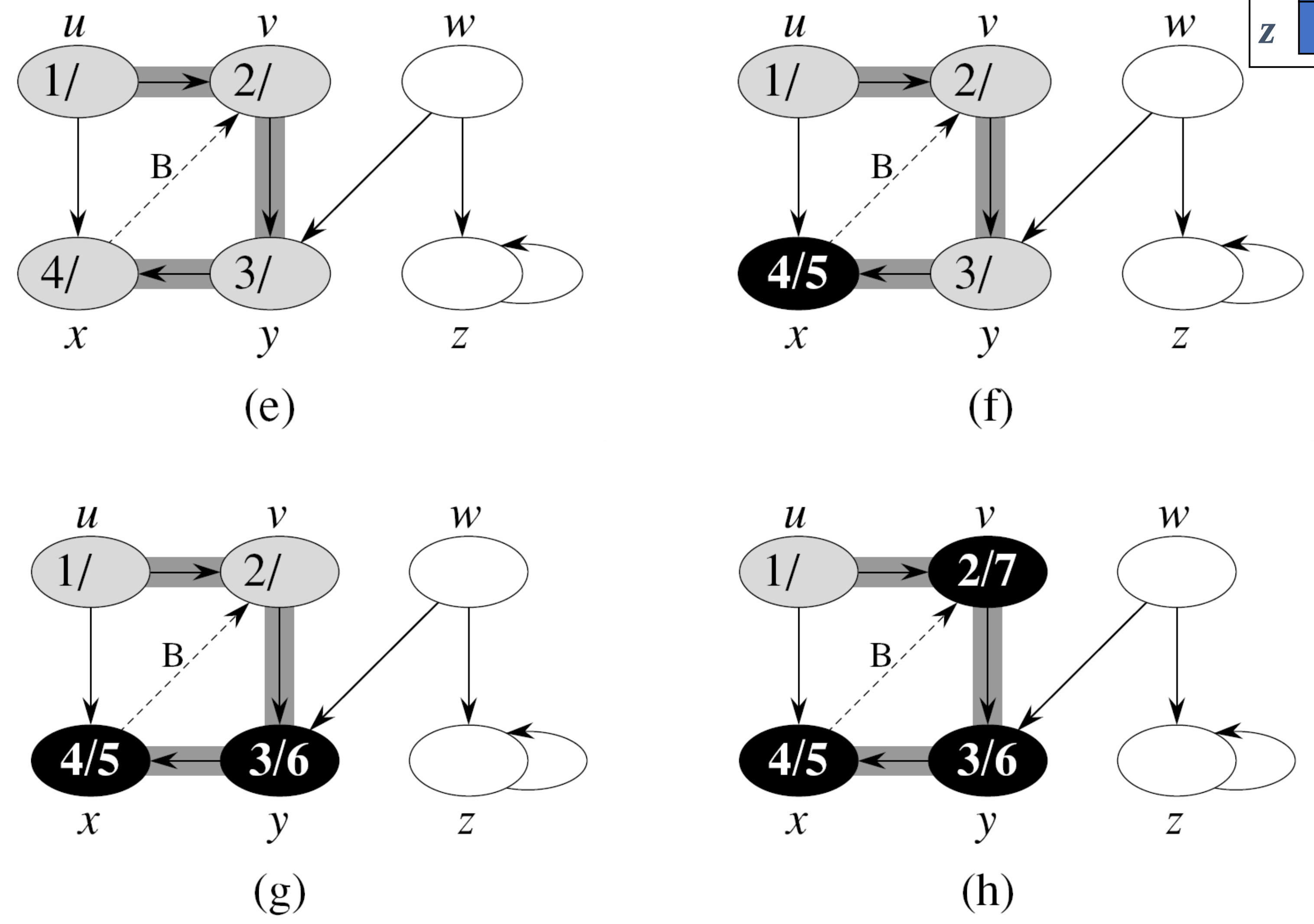


(d)

דוגמה

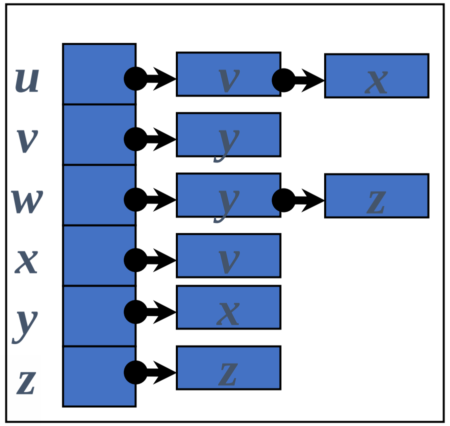


u
v
w
x
y
z
v
x
y
y
z
v
x
z

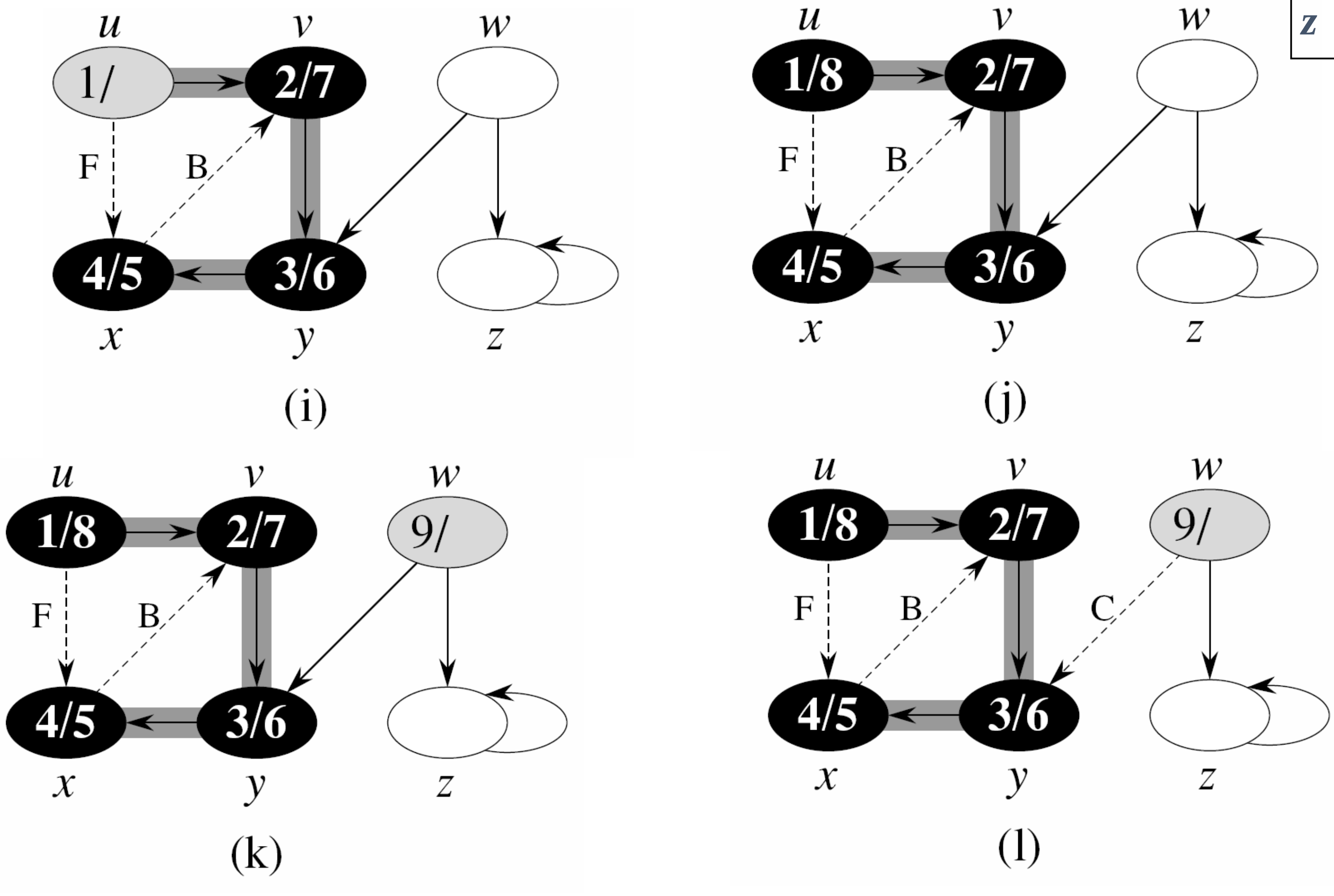


u
v
w
1/
2/
B
4/
3/
x
y
z
(e)
u
v
w
1/
2/
B
4/5
3/
x
y
z
(f)
u
v
w
1/
2/
B
4/5
3/6
x
y
z
(g)
u
v
w
1/
2/7
B
4/5
3/6
x
y
z
(h)

דוגמה

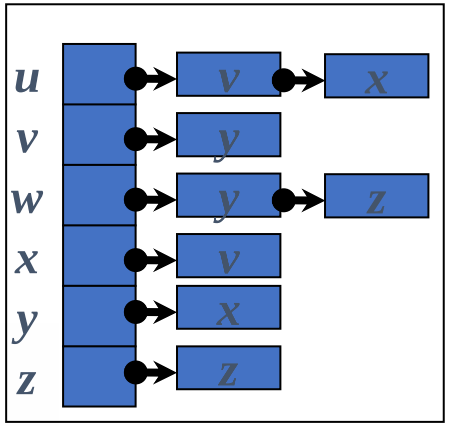


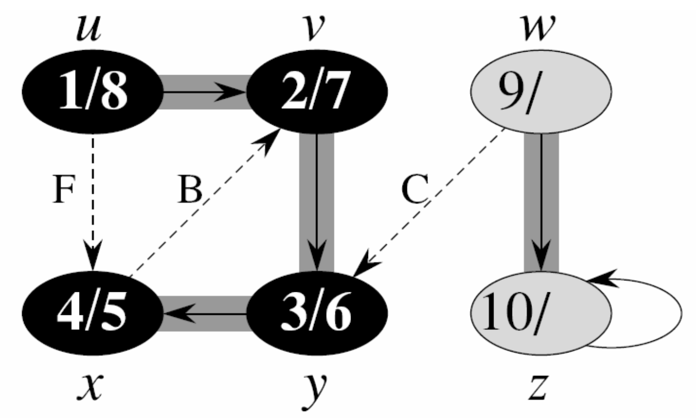
u
v
w
x
y
z
v
x
y
y
z
v
x
z



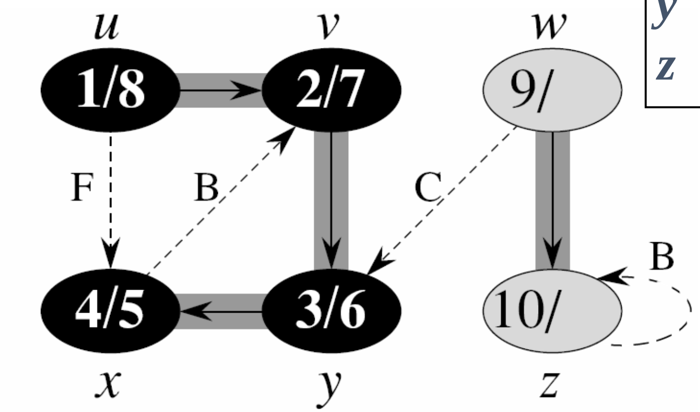
u
v
w
1/
2/7
F
B
4/5
3/6
x
y
z
(i)
1/8
2/7
F
B
4/5
3/6
(j)
1/8
2/7
9/
F
B
4/5
3/6
(k)
1/8
2/7
9/
F
B
C
4/5
3/6
(l)

דוגמה

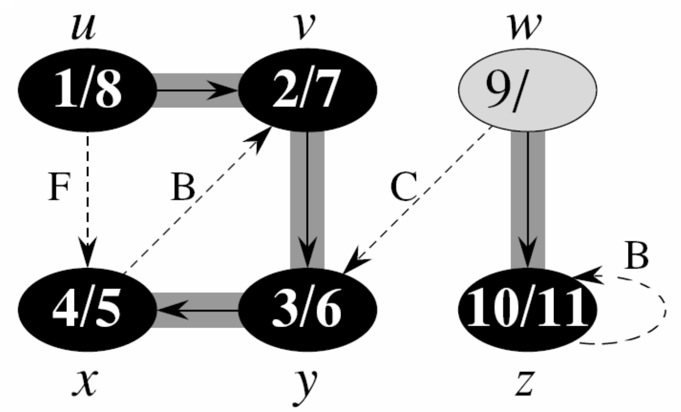

u
v
w
x
y
z
v
x
y
y
z
v
x
z


u
v
w
1/8
2/7
9/
F
B
C
4/5
3/6
10/
x
y
z

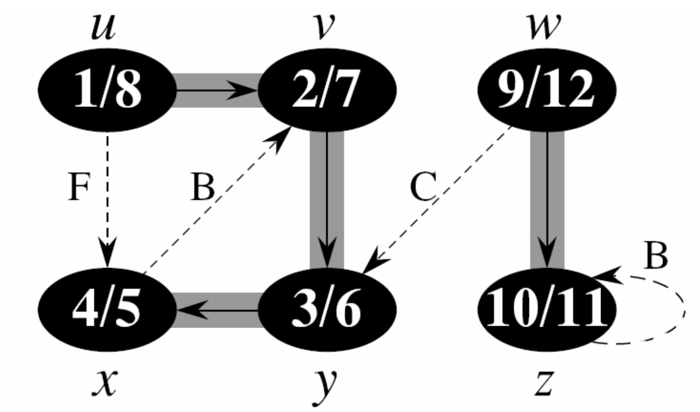
(m)


u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/
x
y
z

(n)


u
v
w
1/8
2/7
9/
F
B
C
B
4/5
3/6
10/11
x
y
z

(o)


u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z

(p)

DFS

DFS(*G=(V,E)*)

// $\pi[u]$ - predecessor of *u*

for each vertex $u \in V$
    $color[u] \leftarrow$ WHITE
    $\pi[u] \leftarrow$ NULL
$time \leftarrow 0$
for each vertex $u \in V$
    if $color[u] =$ WHITE
        DFS-VISIT(u)

זמן ריצה O(V + E)
אם גרף מיוצג על ידי רשימות סמיכות

DFS-VISIT(*u*)

// white vertex *u* has just been discovered

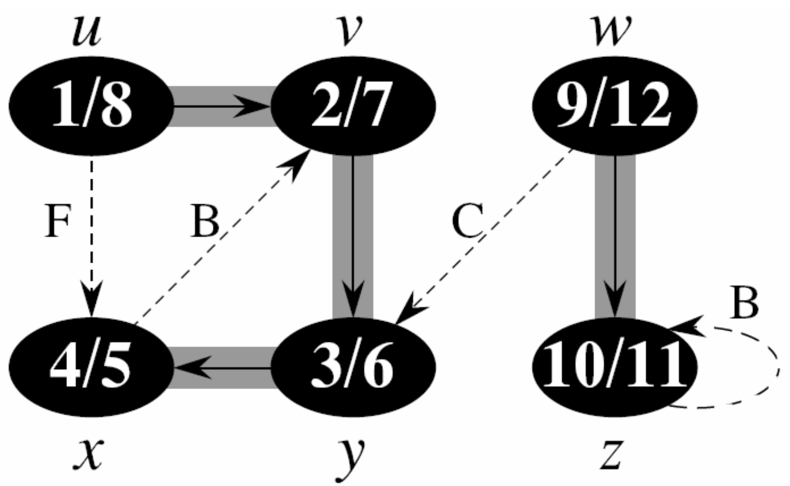
$color[u] \leftarrow$ GRAY
$time \leftarrow time+1$
$d[u] \leftarrow time$
for each $v \in Adj[u]$ // explore edge (u, v)
    if $color[v] =$ WHITE
        $\pi[v] \leftarrow u$
        DFS-VISIT(v)
$color[u] \leftarrow$ BLACK
// blacken u; it is finished.
$f[u] \leftarrow time \leftarrow time+1$

תכונות של חיפוש לעומק

- נגדיר $G_\pi=(V, E_\pi)$ תת-גרף הקודמים (predecessor subgraph) של G:

$$E_\pi=\{ (\pi[v], v) \in E \mid \pi[v] \neq NULL \text{ and } v \in V \}$$

- כאשר DFS מופעל על גרף G = (V, E), הוא בונה יער עומק המורכב מכמה עצי עומק (depth-first forest)



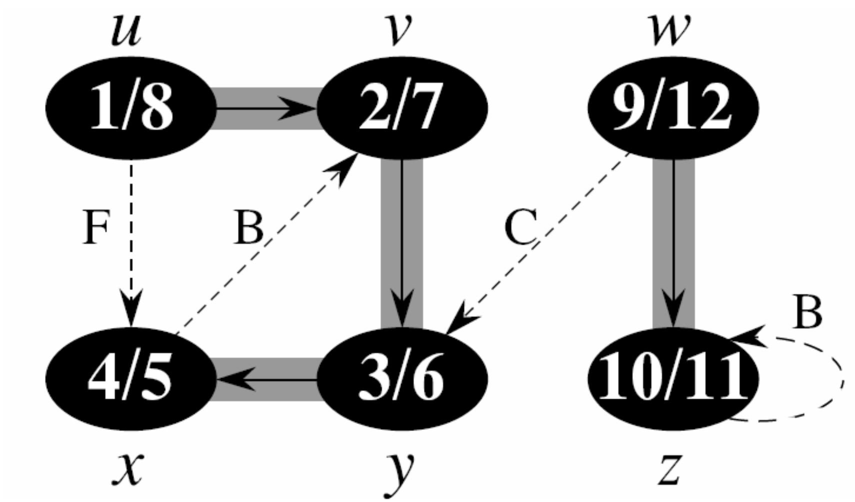
סיווג קשתות

1. קשת עץ (tree edge)- קשת השייכת ליער G_π
2. קשת אחורה (back edge) – מחברת קודקוד לאב קדמון שלו ביער העומק G_π
3. קשת קדימה (forward edge) – רק בגרף מכוון.

קשת שאינה קשת עץ, מחברת קודקוד לצאצא שלו ביער G_π

4. קשת חוצה (cross edge) – רק בגרף מכוון. כל קשת אחרת.

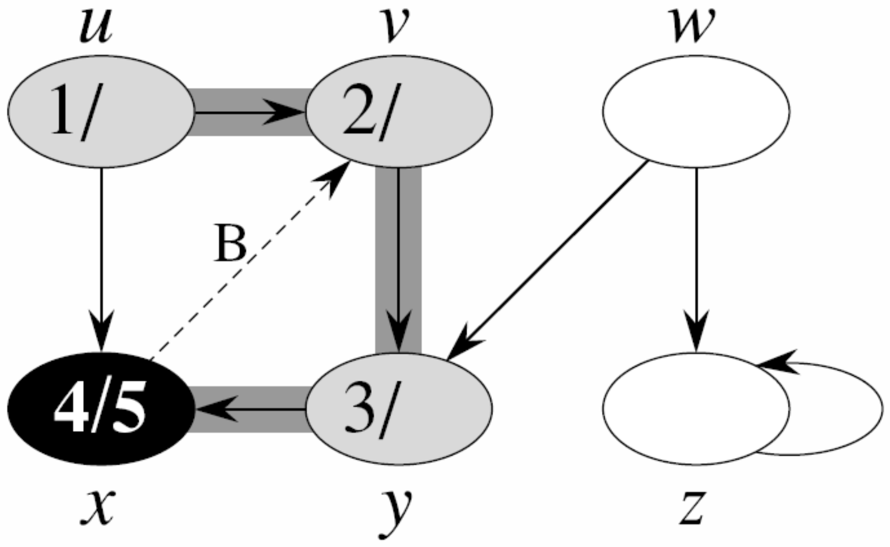
- קשת המחברת בין שני עצי עומק שונים או
- קשת בין שני קדקודים שונים באותו עץ שאינם ביחס "אב קדמון-צאצא"



סיווג קשתות

- **ניתן לשנות את DFS כך שיסווג כל קשת (u, v) במהלך הריצה לפי צבעו של הקדקוד v.**

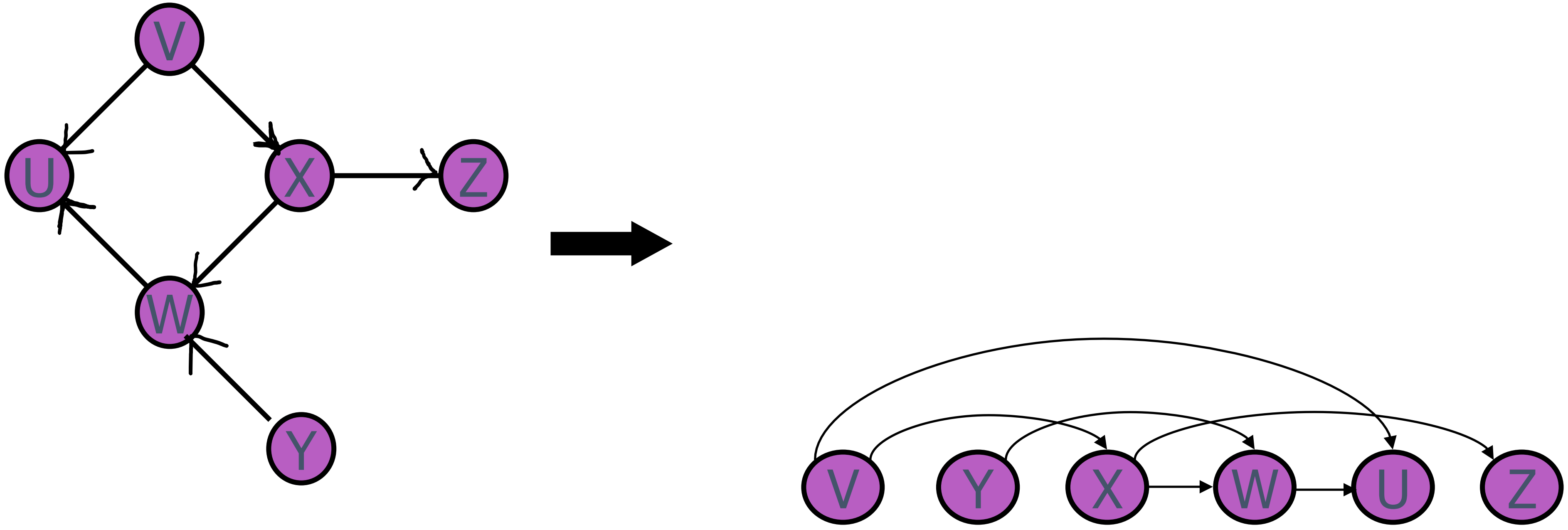
1. לבן – קשת עץ
2. אפור – קשת אחורה
3. שחור – אם d[v] > d[u] קשת קדימה, אם d[v] < d[u] קשת חוצה



Topological Sort– מיון טופולוגי

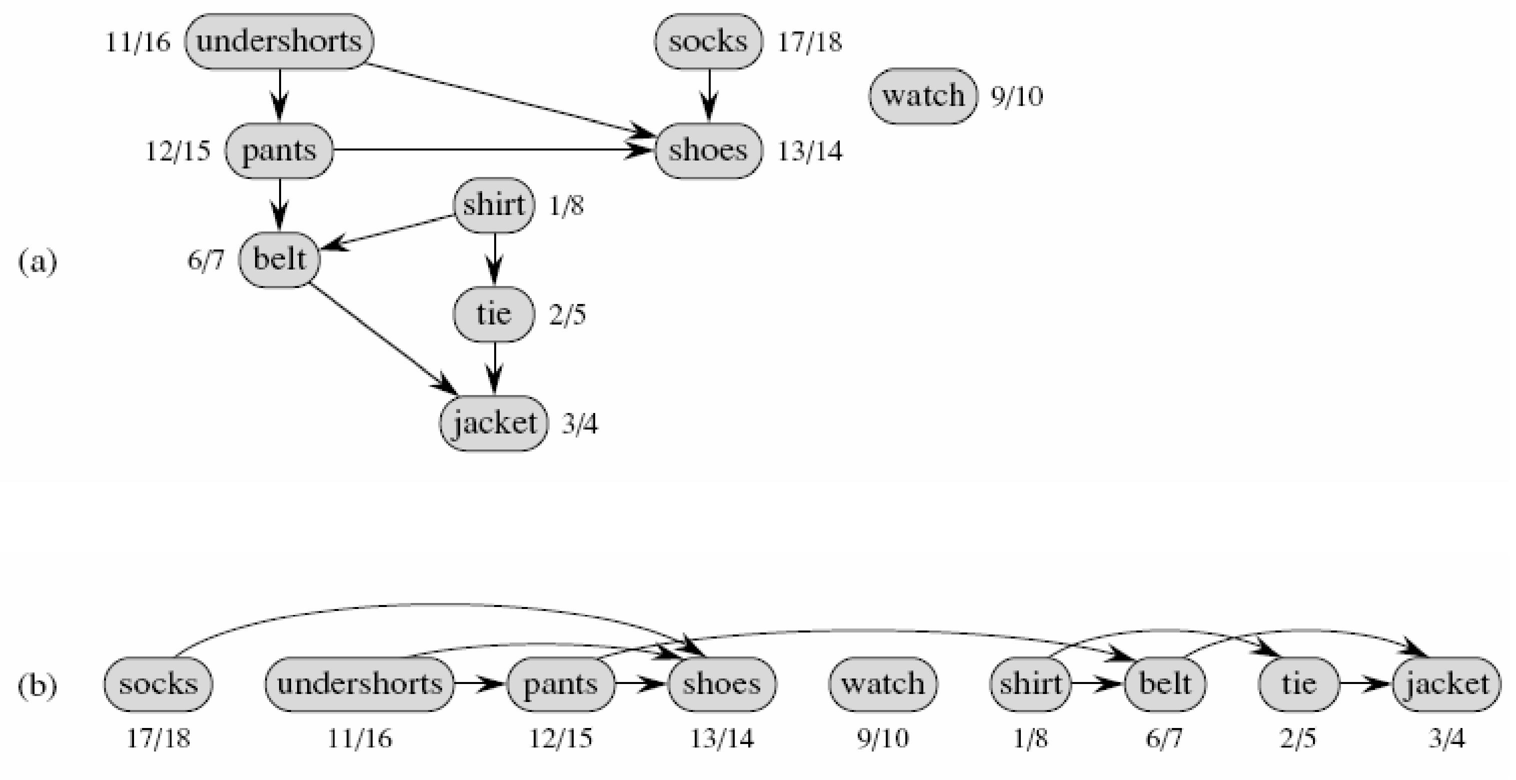
- Direct Acyclic Graph (DAG) – גרף מכוון חסר מעגלים

מיון טופולוגי של DAG הוא סידור ליניארי של קדקודים כך שאם גרף מכיל קשת (u, v), אז u מופיע לפני v בסידור



מיון טופולוגי

- ל-DAG שימושים רבים לציון קדימויות בתוך קבוצת המאורעות
- איך מתלבש פרופסור בומסטד בבוקר:
  - יש ללבוש פריטי לבוש מסוימים לפני האחרים (גרביים לפני נעליים)
  - יש פריטי לבוש אותם אפשר ללבוש בסדר כלשהו (גרביים ומכנסיים)



מיון טופולוגי

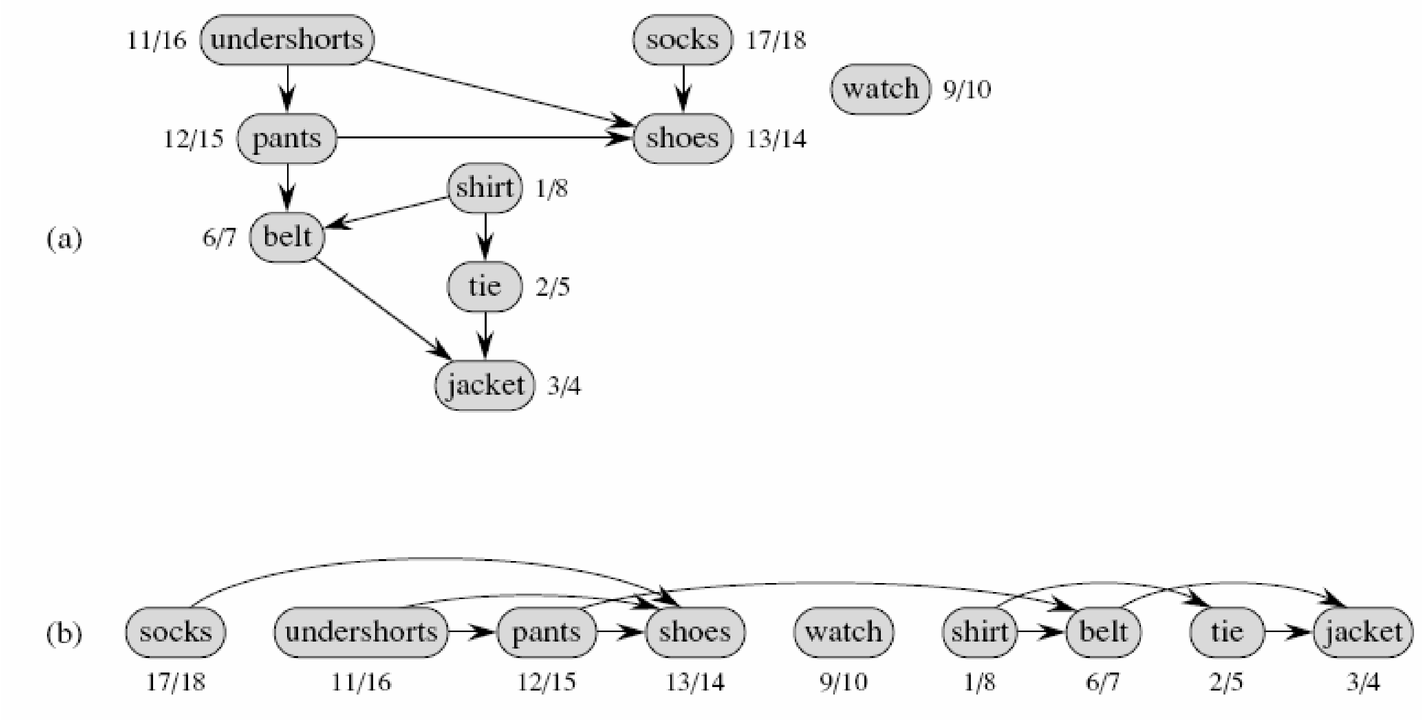
TOPOLOGICAL-SORT*(G)*

1 call DFS*(G)* to compute finishing times *f* [*v*] for each vertex *v*

2 as each vertex is finished, insert it onto the front of a linked list

3 return the linked list of vertices

זמן ריצה O(V + E)



מתי גרף הוא DAG

<u>משפט:</u> גרף מכוון G=(V, E) אינו מכיל מעגל אם ורק אם DFS אינו מניב קשתות אחורה

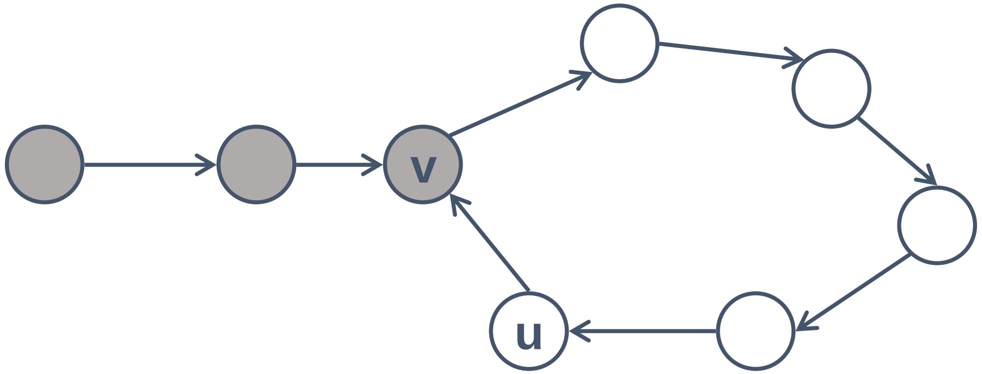
<u>הוכחה:</u>

- (כיוון ראשון) נניח (u, v) – קשת אחורה, אז v אב קדמון של u. לכן קיים מסלול מ-v ל-u בעץ העומק וקשת (u, v) סוגרת מעגל.

מתי גרף הוא DAG

- (כיוון שני) נניח G מכיל מעגל c. יהי v הקדקוד הראשון המתגלה ב-c, ויהי (u,v) הקשת הקודמת ב-c.

  בזמן d[v] קיים מסלול של קדקודים לבנים מ-v ל-u <=
  u צאצא של v <= (u,v) קשת אחורה.



מיון טופולוגי - נכונות האלגוריתם

<u>משפט:</u> אלגוריתם Topological-Sort(G) יוצר מיון טופולוגי של גרף מכוון חסר מעגלים G.

<u>הוכחה:</u>

די להראות שעבור כל קשת (u, v) ב-DAG מתקיים f[u] >f[v].

כאשר קשת (u, v) נבדקת נתבונן בצבעים של u ו-v.

- צבע של u אפור
- צבע של v לא יכול להיות אפור (כי אז (u,v) קשת אחורה)
- v לבן, אז v צאצא של u ואז f[u] > f[v].
- v שחור, אז הטיפול ב-v הסתיים ו- f[u] > f[v]