# طراحی و تحلیل الگوریتم


استاد:

**دکتر زاهد رحمتی**

تدریسیاران:

**داریوش کاظمی**

**اشکان ودادی**


ترم دوم ۱۴۰۰

# جلسه چهارم
گراف

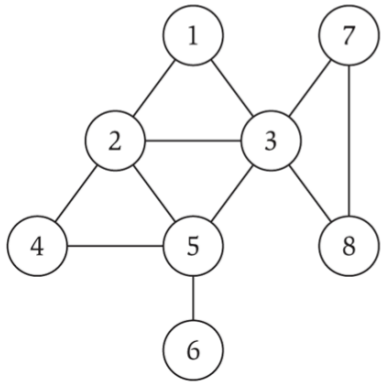
# گراف

1. تعاریف اولیه گراف
2. BFS و DFS
3. DAG و Topological Order
4. دایجسترا و الگوریتم‌های پیشرفته‌تر (بعدا)

# انواع گراف از نظر نوع یال

Undirected graph $G = (V, E)$

- V = nodes.
- E = edges between pairs of nodes.
- Captures pairwise relationship between objects.
- Graph size parameters: $n = |V|$, $m = |E|$.




V = { 1, 2, 3, 4, 5, 6, 7, 8 }

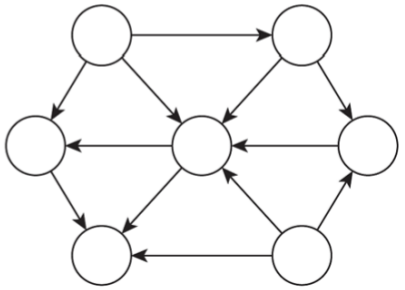
E = { 1-2, 1-3, 2-3, 2-4, 2-5, 3-5, 3-7, 3-8, 4-5, 5-6 }

n = 8

m = 11

Directed graph $G = (V, E)$

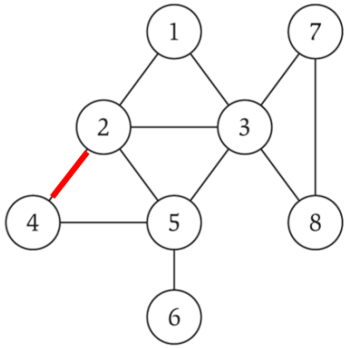
- Edge (u, v) goes from node u to node v.



# نمایش گراف

An $n \times n$ matrix with $A_{uv} = 1$ if (u, v) is an edge.

- Two representations of each edge.
- Space proportional to $n^2$.
- Time to check if (u, v) is an edge: $\Theta(1)$.
- Identifying all edges takes $\Theta(n^2)$ time.

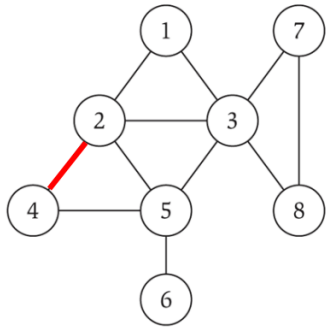


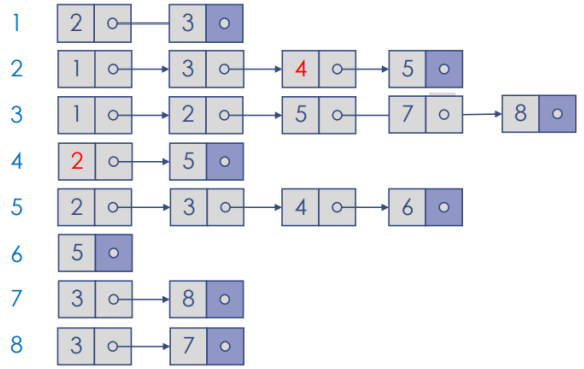

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 3 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |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

Node indexed array of lists.

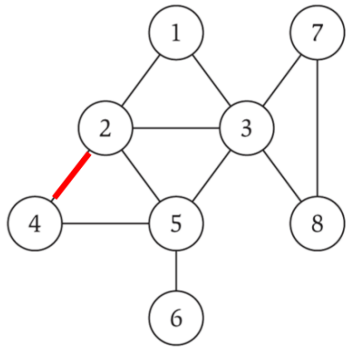
- Two representations of each edge.
- Space proportional to m + n.
- Time to check if (u, v) is an edge: O(deg(u)).
- Identifying all edges takes $\Theta(m + n)$ time.

number of neighbors of u

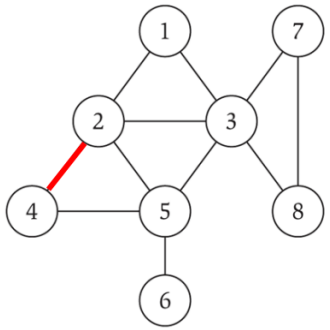


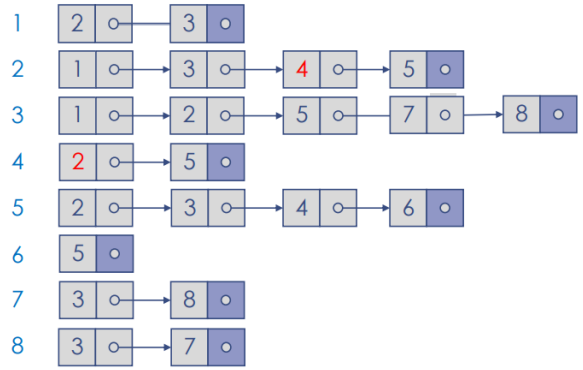



# نمایش گراف




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




**سوال:** اگر گراف وزن دار باشد به چه صورتی باید ذخیره کنیم؟

# سایر تعاریف

1. مسیر
2. دور
3. درخت
4. همبندی و ناهمبندی
5. کوتاه ترین مسیر
6. گراف دوبخشی
7. گراف‌های چندبخشی
8. اویلر و همیلتون
9. مجموعه‌های مستقل راسی و یالی و ...

به طور کلی درس مبانی ترکیبیات و نظریه گراف در دوران کارشناسی از درس‌های زیبایی است که برای جا افتادن مفاهیم گراف خیلی میتواند به شما کمک کند.

# BFS

- الگوریتم پیمایش اول سطح یا جستجوی اول سطح (Breadth First Search - BFS) از جمله الگوریتم‌های مشهور پیمایش و جستجوی گراف است که در حل مسائل الگوریتمی و هوش مصنوعی کاربرد دارد.

- الگوریتم برای پیمایش و جستجوی گراف از **یک صف** برای نگهداری ترتیب جستجو استفاده می‌کند.

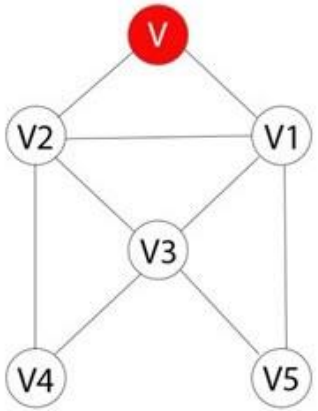
- الگوریتم BFS با وارد کردن گره مبدأ به صف پردازش شروع شده و تا خالی نشدن این صف مراحل زیر را تکرار می‌شود:
  ۱. عنصر جلوی صف را به عنوان گره جاری انتخاب و از صف حذف کن.
  ۲. گره جاری را پردازش کن.
  ۳. گره‌های مجاور گره جاری که پردازش نشده و در صف پردازش نیز قرار ندارند به این صف اضافه کن.

# مرتبه‌ی زمانی الگوریتم جستجوی اول سطح

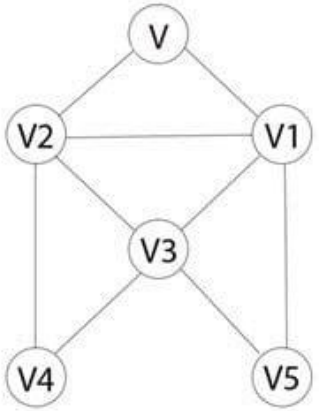
- در گراف G = (V, E) مرتبه‌ی زمانی الگوریتم جستجوی اول سطح **O(|V| + |E|)** است؛
- چرا که این الگوریتم در بزرگترین حالت تمامی گره‌ها را پیمایش کرده و نیاز به بررسی تمامی یال‌ها دارد.

- **تمرین:** این مرتبه‌ی اجرایی در یک گراف همبند به صورت O(|E|)بوده و در حالت کلی متناسب با تعداد یال‌ها حداکثر از مرتبه‌ی $O(n^2)$است. (چرا؟)

# اجرای الگوریتم BFS مرحله به مرحله

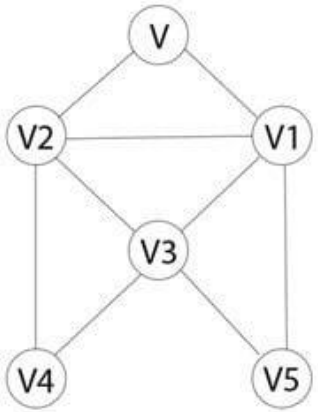



Root Node = V



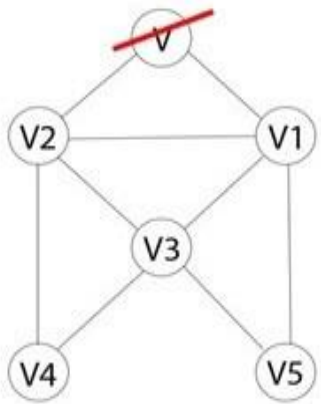

In case V is not visited add it to BFS queue

Queue = | V | |

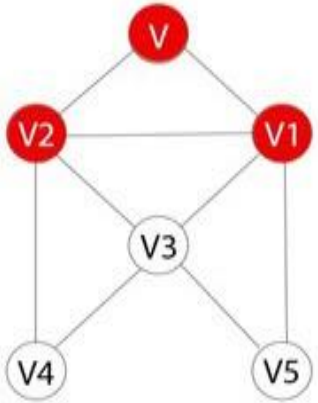



Delete Vertex V from the quoue

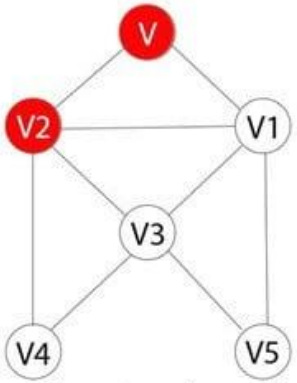
~~V~~




1. Start the BFS search
2. After completion, mark V as completed

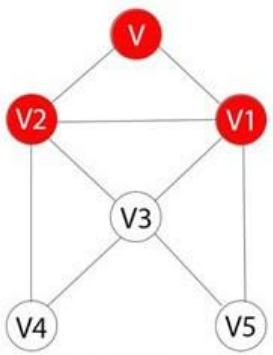



Visit and retrieve all the adjacent and un-visited nodes from the node V



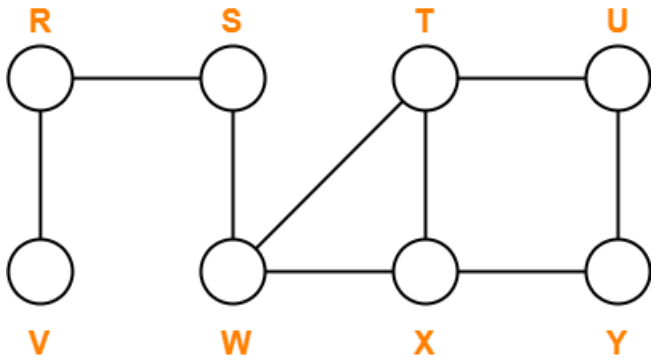

For adjacent and un-visited vertex say V1, add it to BFS queue

Queue = | ~~V~~ | V1 |



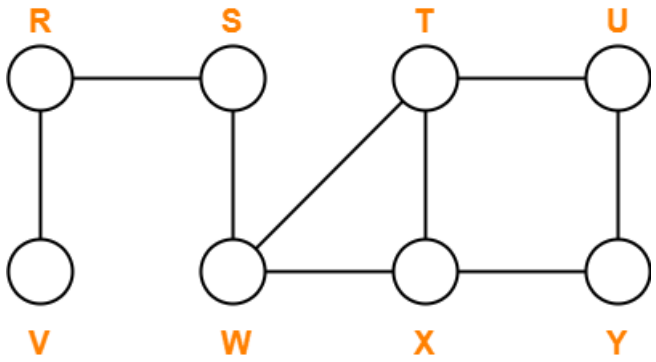

BFS will visit V1, mark it as visited and delete it from the queue

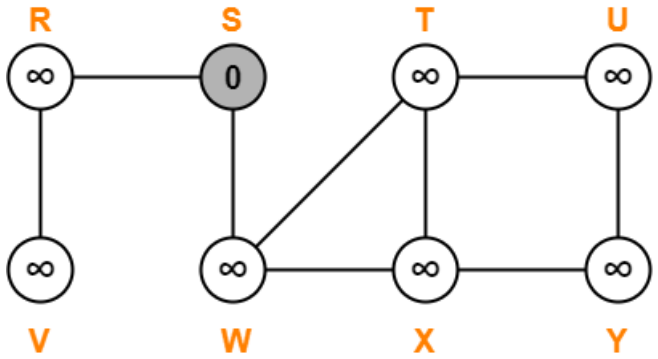
Queue = | ~~V~~ | ~~V1~~ |



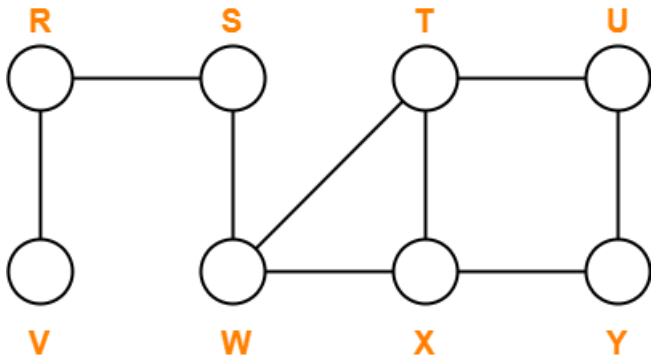

# اجرای الگوریتم BFS مرحله به مرحله با شروع از S

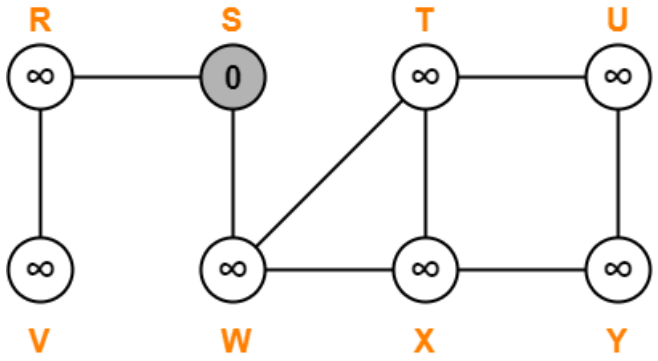
# اجرای الگوریتم BFS مرحله به مرحله با شروع از S



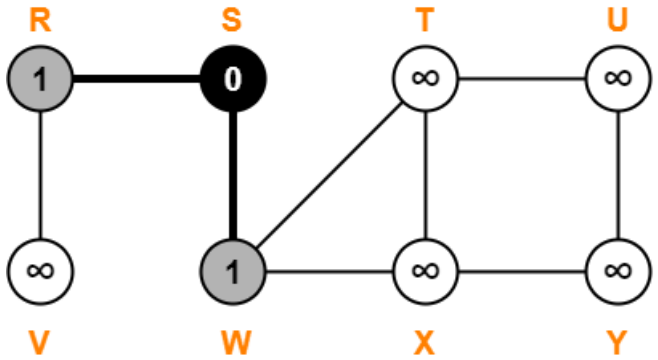




# اجرای الگوریتم BFS مرحله به مرحله با شروع از S



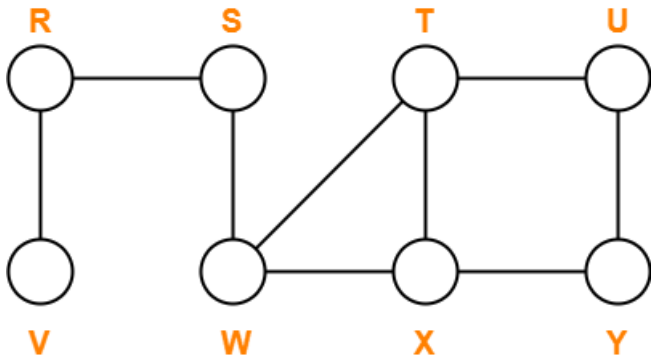



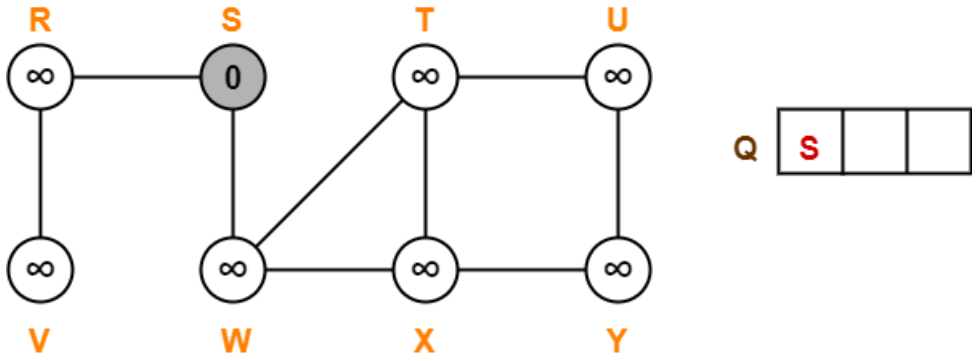

Q | S | | |

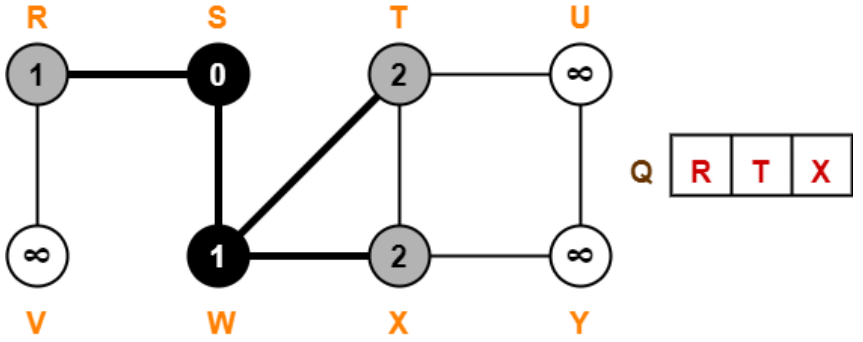


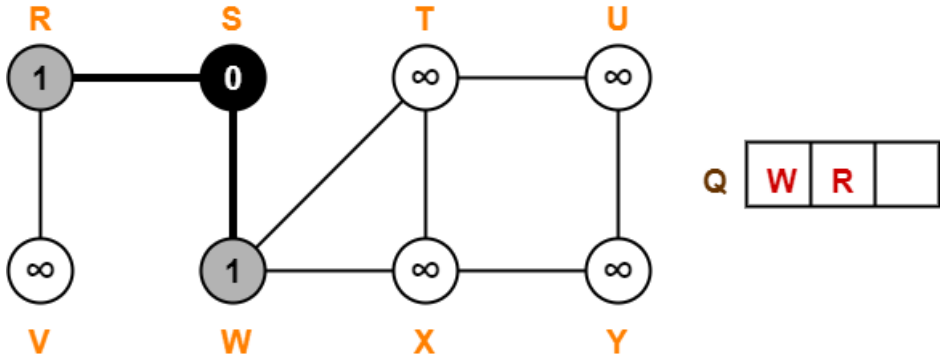

Q | W | R | |

# اجرای الگوریتم BFS مرحله به مرحله با شروع از S

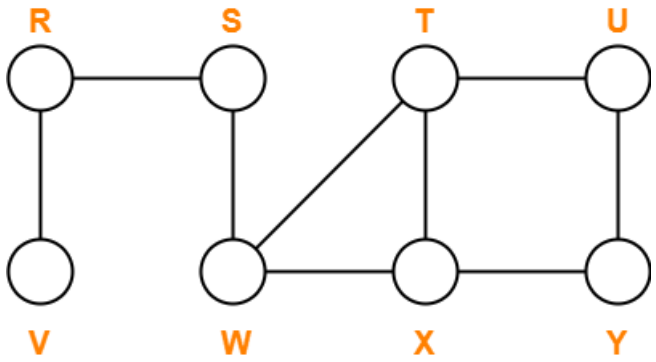


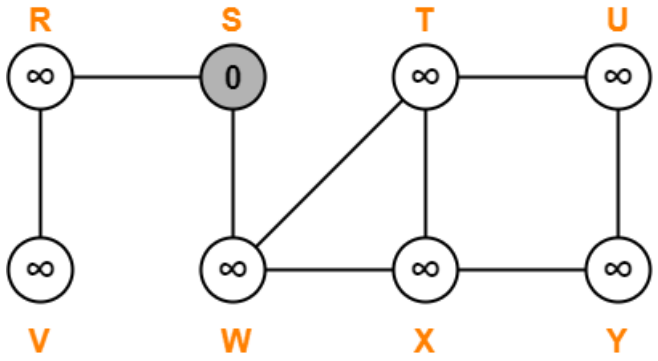



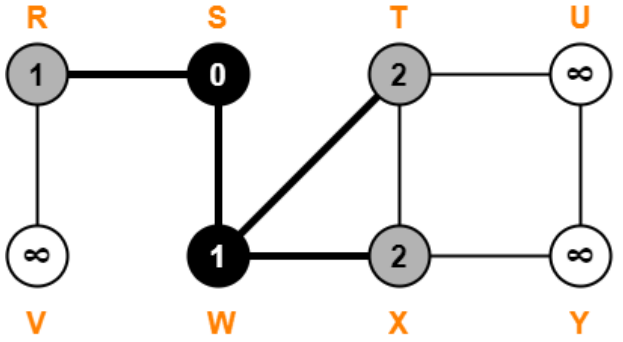



# اجرای الگوریتم BFS مرحله به مرحله با شروع از S

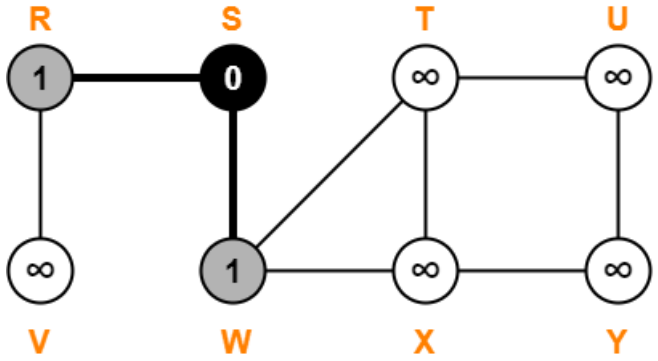


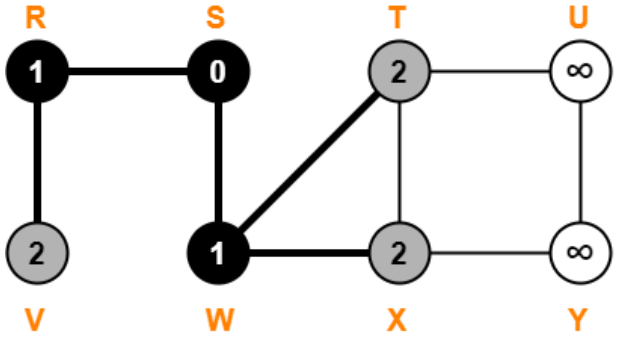

Q | S | | |




Q | R | T | X |

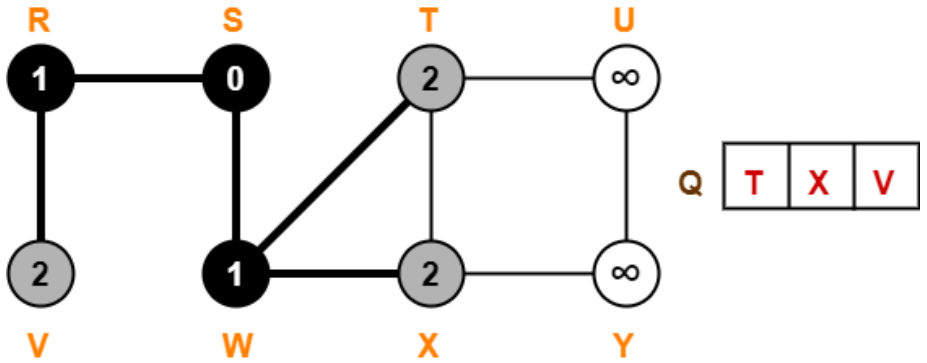



Q | W | R | |

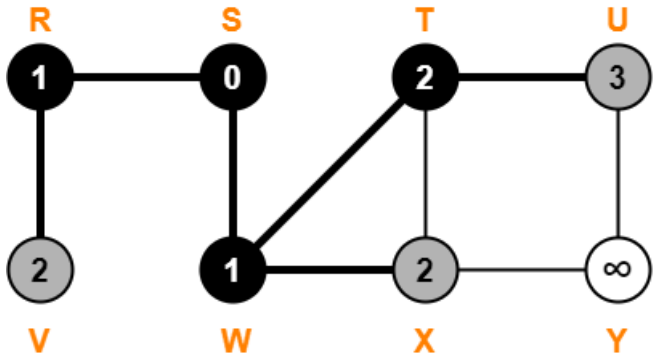



Q | T | X | V |

# اجرای الگوریتم BFS مرحله به مرحله با شروع از S

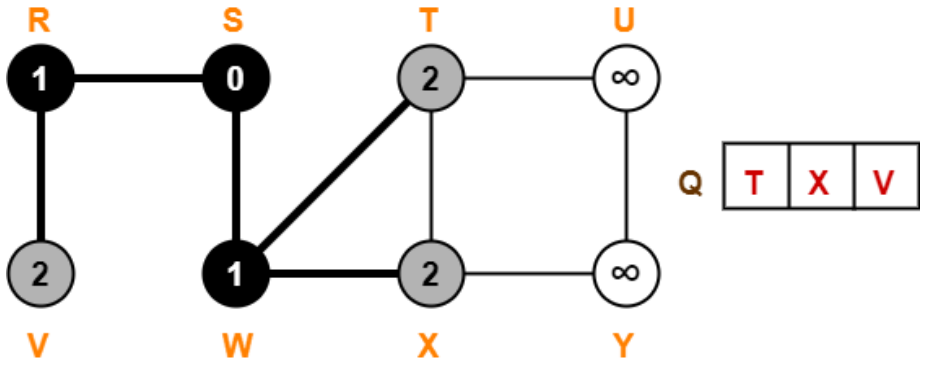



Q | T | X | V

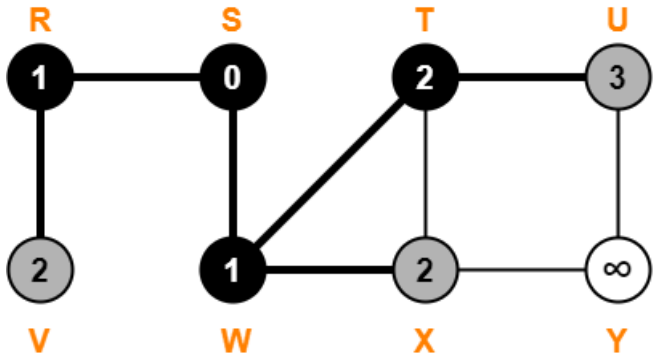



Q | X | V | U

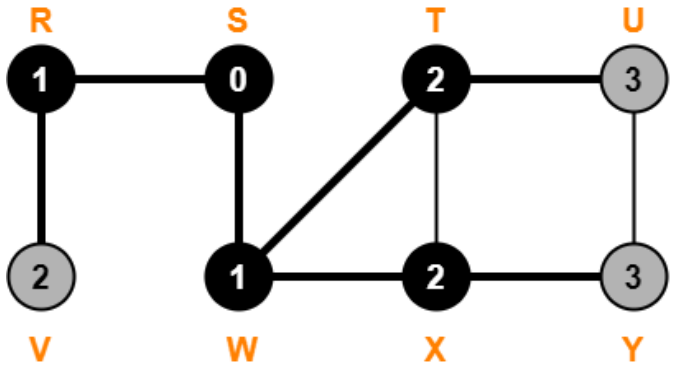
# اجرای الگوریتم BFS مرحله به مرحله با شروع از S




Q | T | X | V

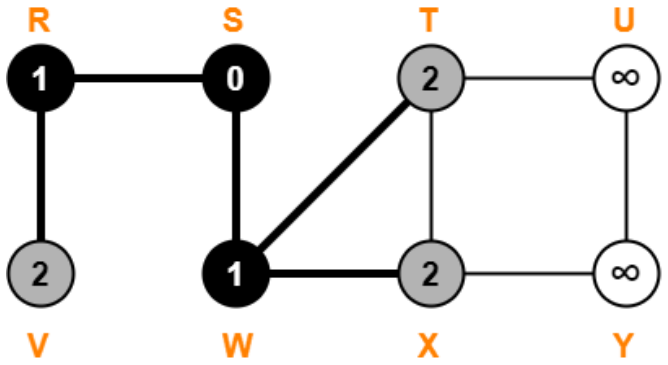



Q | X | V | U

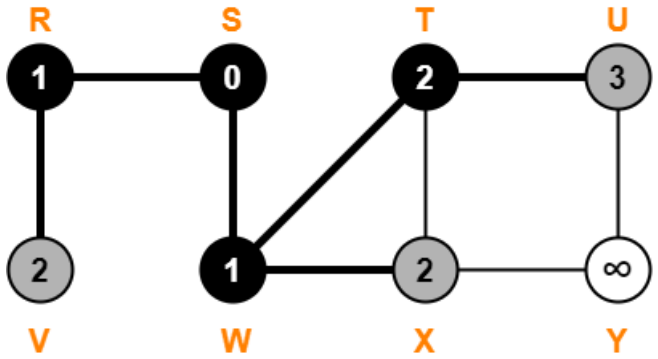



Q | V | U | Y

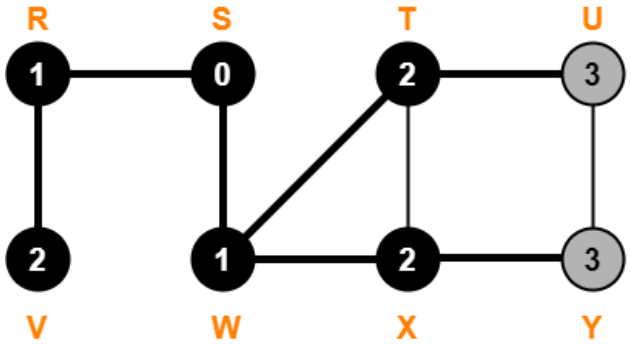
# اجرای الگوریتم BFS مرحله به مرحله با شروع از S



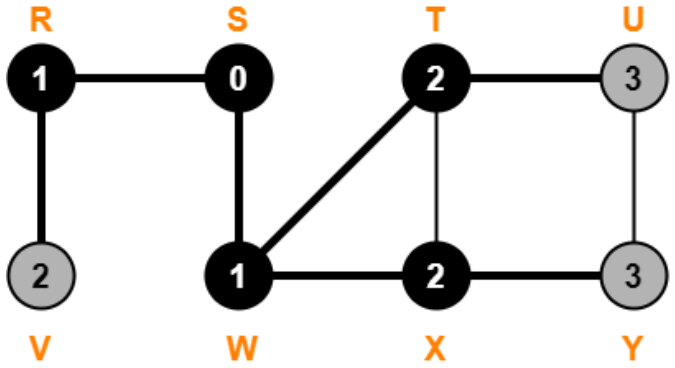

Q | T | X | V




Q | X | V | U

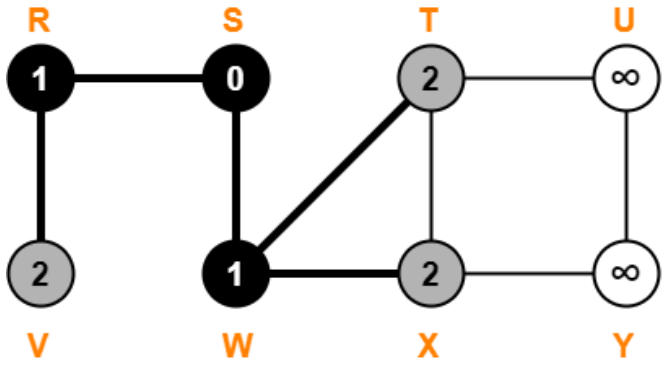



Q | U | Y |

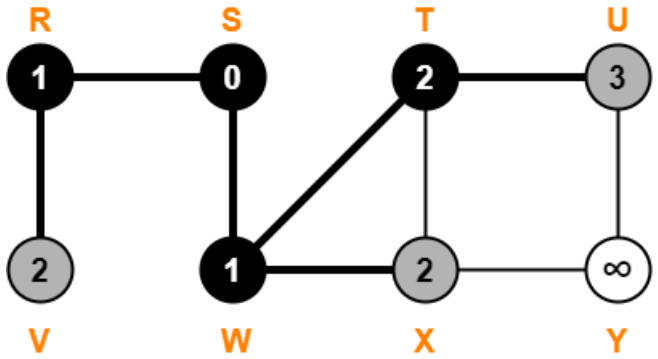



Q | V | U | Y

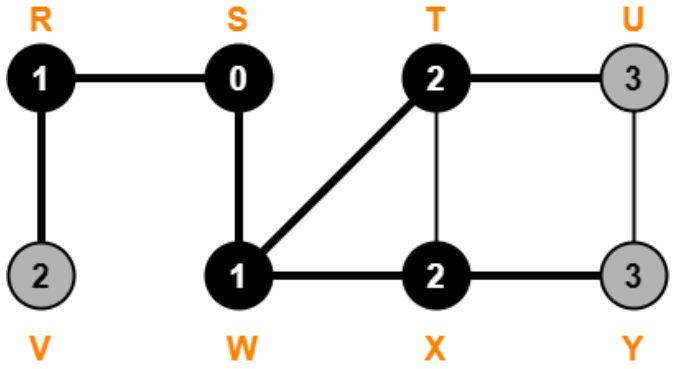
# اجرای الگوریتم BFS مرحله به مرحله با شروع از S



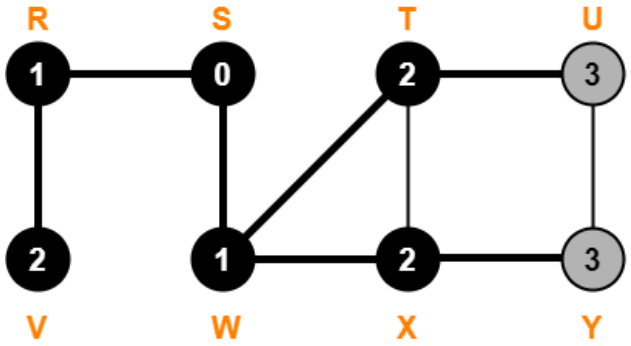

Q | T | X | V |



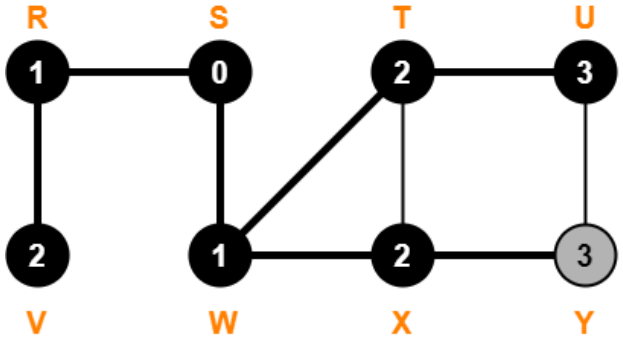

Q | X | V | U |




Q | V | U | Y |

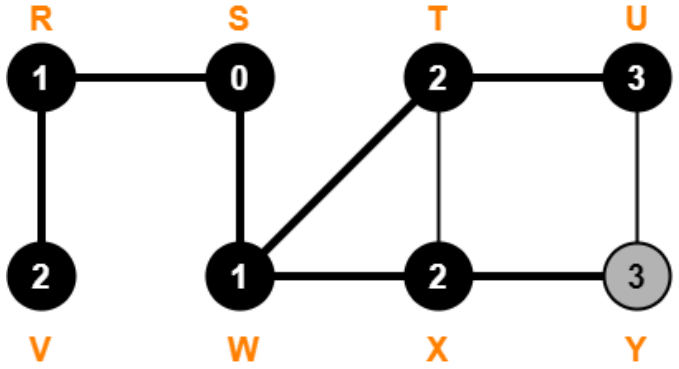



Q | U | Y | |

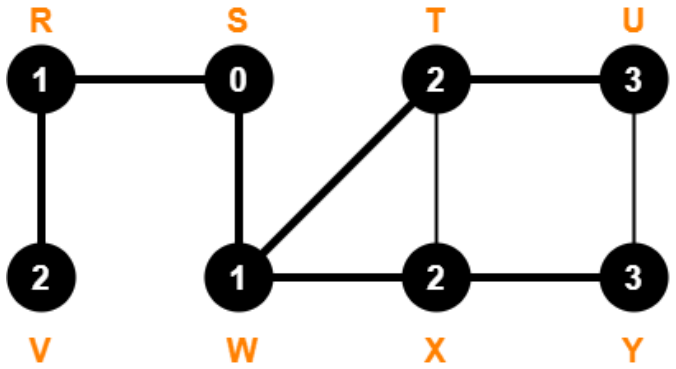



Q | Y | | |

# اجرای الگوریتم BFS مرحله به مرحله با شروع از S








# DFS

- در هر تکرار الگوریتم DFS تنها یکی از گره‌های مجاور گره پردازش شده برای مرحله‌ی بعد انتخاب می‌شود.
- الگوریتم DFSبه جای صف از **یک پشته** برای مشخص کردن مسیر پیمایش استفاده می‌کند.

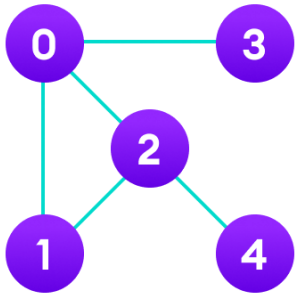
- الگوریتم DFSبا فرض انتخاب گره مبدأ به عنوان گره جاری از مراحل زیر تشکیل یافته است:
  ۱. گره جاری را به پشته اضافه کن.
  ۲. گره جاری را پردازش کن.
  ۳. از گره‌های مجاور گره جاری یک گره پیمایش نشده را به عنوان گره جاری انتخاب کرده و برو به مرحله‌ی ۱.
  ۴. اگر همه‌ی گره‌های مجاور گره جاری پیمایش شده‌اند، گره بالای پشته را به عنوان گره جاری از پشته حذف کرده و برو به مرحله‌ی ۳.
  ۵. اگر گرهی در پشته وجود ندارد، اجرای الگوریتم را متوقف کن.

# مرتبه‌ی زمانی الگوریتم جستجوی اول عمق

- در گراف G = (V, E) مرتبه‌ی زمانی الگوریتم جستجوی اول سطح **O(|V| + |E|)** است؛
- چرا که این الگوریتم در بزرگترین حالت تمامی گره‌ها را پیمایش کرده و نیاز به بررسی تمامی یال‌ها دارد.

- **تمرین:** این مرتبه‌ی اجرایی در یک گراف همبند به صورت O(|E|)بوده و در حالت کلی متناسب با تعداد یال‌ها حداکثر از مرتبه‌ی $O(n^2)$است. (چرا؟)

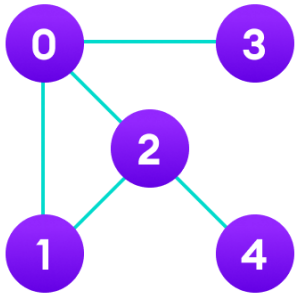
# اجرای الگوریتم DFS
# مرحله به مرحله



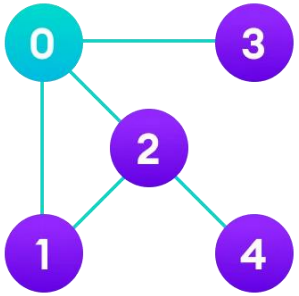

Visited

Stack

# اجرای الگوریتم DFS
# مرحله به مرحله

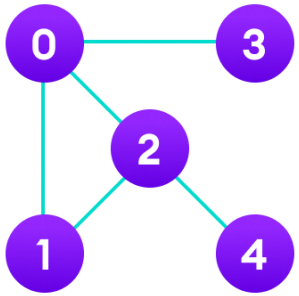



| | | | | | |
|---|---|---|---|---|---|
| | | | | | Visited |
| | | | | | Stack |




| | | | | | |
|---|---|---|---|---|---|
| 0 | | | | | Visited |
| 1 | 2 | 3 | | | Stack |

# اجرای الگوریتم DFS
# مرحله به مرحله

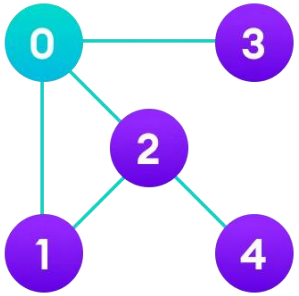



| | | | | |
|---|---|---|---|---|
| | | | | |

Visited

| | | | | |
|---|---|---|---|---|
| | | | | |

Stack

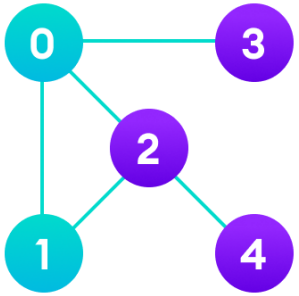



| | | | | |
|---|---|---|---|---|
| 0 | | | | |

Visited

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | | |

Stack



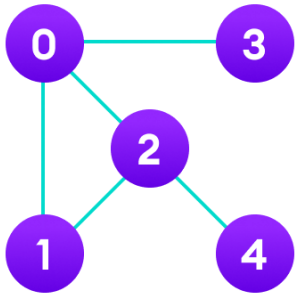

| | | | | |
|---|---|---|---|---|
| 0 | 1 | | | |

Visited

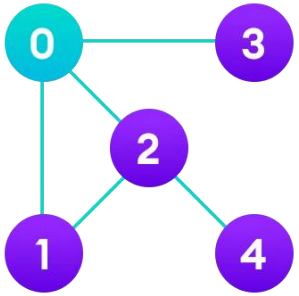
| | | | | |
|---|---|---|---|---|
| 2 | 3 | | | |

Stack

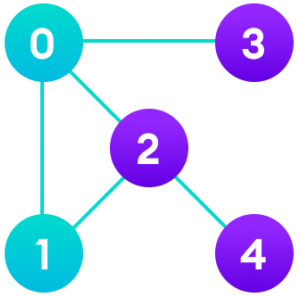
# اجرای الگوریتم DFS
# مرحله به مرحله



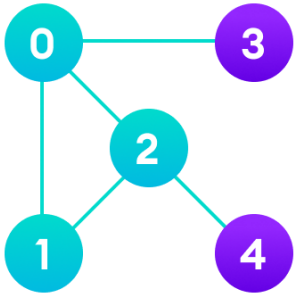

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Visited |
| | | | | | Stack |




| | | | | | |
|---|---|---|---|---|---|
| 0 | | | | | Visited |
| 1 | 2 | 3 | | | Stack |

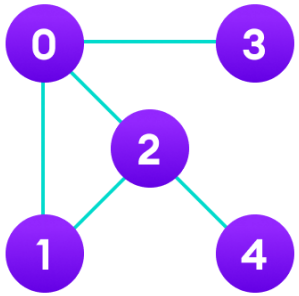



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | | | | Visited |
| 2 | 3 | | | | Stack |

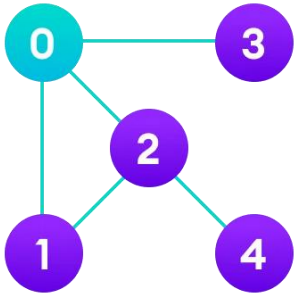



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | | | Visited |
| 4 | 3 | | | | Stack |

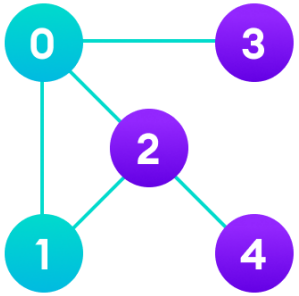
# اجرای الگوریتم DFS
# مرحله به مرحله



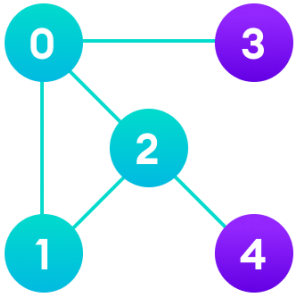

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Visited |
| | | | | | Stack |



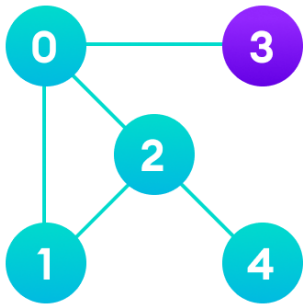

| | | | | | |
|---|---|---|---|---|---|
| 0 | | | | | Visited |
| 1 | 2 | 3 | | | Stack |




| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | | | | Visited |
| 2 | 3 | | | | Stack |

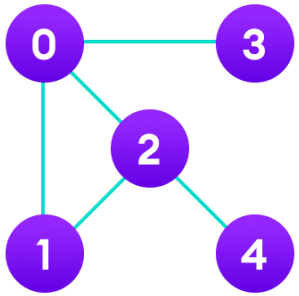



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | | | Visited |
| 4 | 3 | | | | Stack |

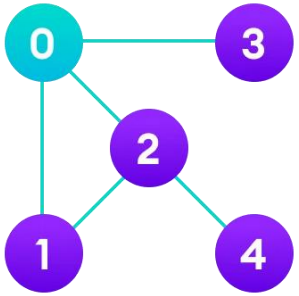



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 4 | | Visited |
| 3 | | | | | Stack |

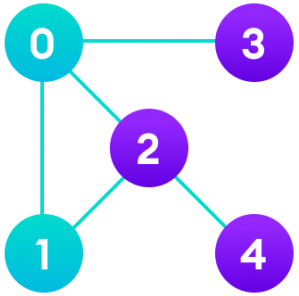
# اجرای الگوریتم DFS
# مرحله به مرحله



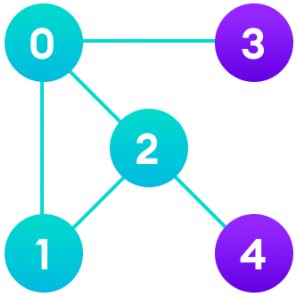

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Visited |
| | | | | | Stack |



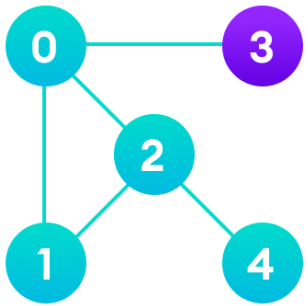

| | | | | | |
|---|---|---|---|---|---|
| 0 | | | | | Visited |
| 1 | 2 | 3 | | | Stack |



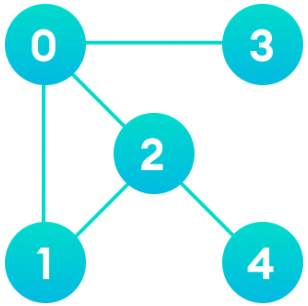

| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | | | | Visited |
| 2 | 3 | | | | Stack |




| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | | | Visited |
| 4 | 3 | | | | Stack |

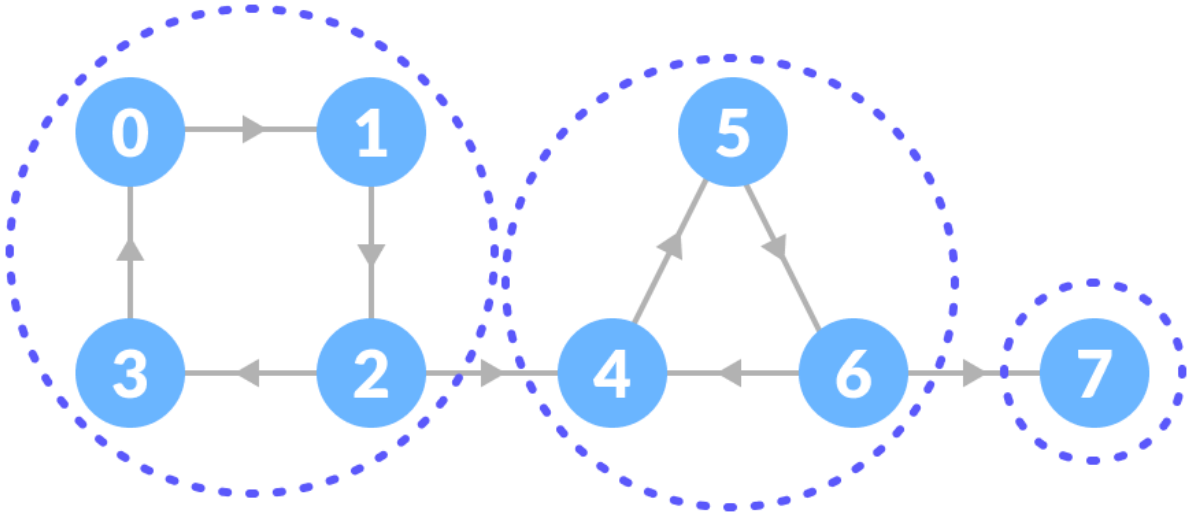



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 4 | | Visited |
| 3 | | | | | Stack |

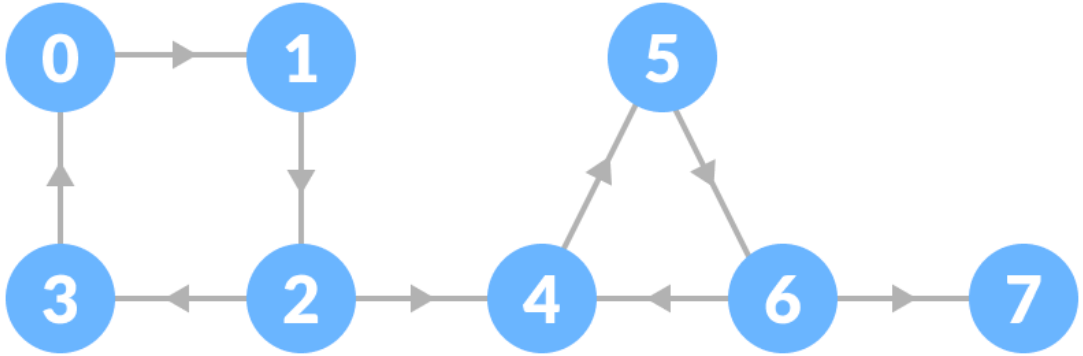



| | | | | | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 4 | 3 | Visited |
| | | | | | Stack |

# گراف‌های همبند قوی




مولفه های همبند قوی




گراف اولیه

# گراف‌های همبند قوی

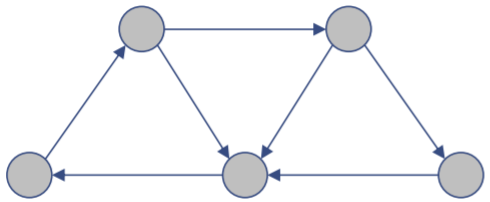
دو نوع مسئله داریم:

1. نشان دهید گراف داده شده همبند قوی است یا نه
2. مولفه های همبند قوی گراف داده شده را بیایید.

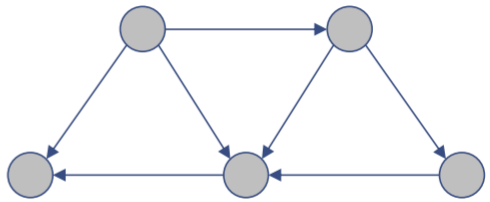
# گراف‌های همبند قوی

1. نشان دهید گراف داده شده همبند قوی است یا نه:

- Pick any node s.
- Run BFS from s in G.
- Run BFS from s in $G^{rev}$.
- Return true iff all nodes reached in both BFS executions.
- Correctness follows immediately from previous lemma.



strongly connected



not strongly connected

# گراف‌های همبند قوی

**2. مولفه های همبند قوی گراف داده شده را بیابید: (Kosaraju's Algorithm):**

We have two Stacks : Visited and Stack

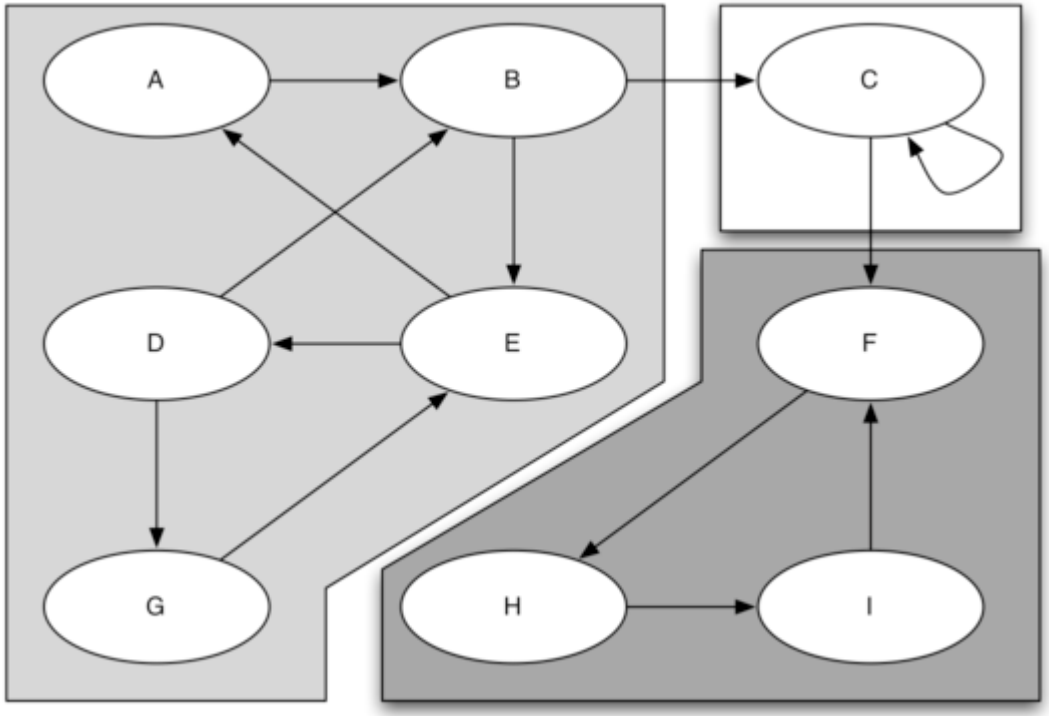
1. **Perform a depth first search on the whole graph.** Let us start from vertex–0, visit all of its child vertices, and mark the visited vertices as done. If a vertex leads to an already visited vertex, then push this vertex to the stack.
2. **Reverse the original graph.**
3. **Perform depth–first search on the reversed graph.** Start from the top vertex of the stack. Traverse through all of its child vertices. Once the already visited vertex is reached, one strongly connected component is formed.

# گراف‌های همبند قوی

**2. مولفه های همبند قوی گراف داده شده را بیابید: (Kosaraju's Algorithm)**
**شرح فارسی الگوریتم:**
این الگوریتم با انجام دو بار پیمایش عمق‌اول روی گراف مولفه‌های همبندی را تشخیص می‌دهد.
ابتدا روی گراف پیمایش عمق‌اول می‌زنیم. **از هر راسی که خارج شدیم(کارمان با آن راس تمام شد) آنرا در یک پشته می‌ریزیم. به این ترتیب راس‌ها به ترتیب زمان خروج در پشته ریخته می‌شوند. حال این‌بار روی گراف معکوس پیمایش عمق‌اول می‌زنیم. اما این‌بار از راسی شروع می کنیم که بین راس‌هایی که هنوز دیده نشده، دیرترین زمان خروج را در پیمایش اولیه داشته‌باشد.** یعنی ابتدا روی بالاترین راسی که در پشته قرار دارد، در گراف معکوس پیمایش عمق اول می‌زنیم. تمام راس‌هایی که دیده می‌شوند با این راس در یک مولفه قرار دارند. تمام این راس‌ها را از پشته حذف می‌کنیم(یا علامتی روی آنها می‌زنیم به این معنی که دیده شده‌اند). حال بین راس هایی که دیده نشده‌اند راس با دیرترین زمان خروج (بالاترین راس پشته که دیده نشده) را انتخاب می‌کنیم و با شروع از آن روی گراف معکوس پیمایش عمق‌اول می‌زنیم و این روند را تا پیدا شدن تمام مولفه‌ها ادامه می‌دهیم.

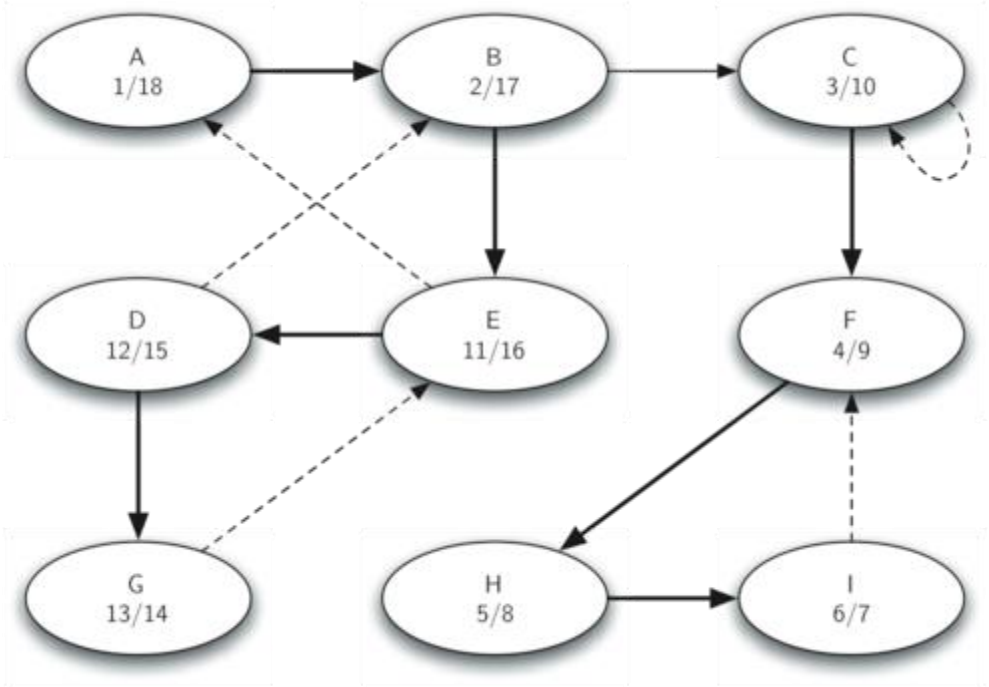
# گراف‌های همبند قوی (مثال نوع 2)

2. مولفه های همبند قوی گراف داده شده را بیایید.



# گراف‌های همبند قوی (مثال نوع 2)

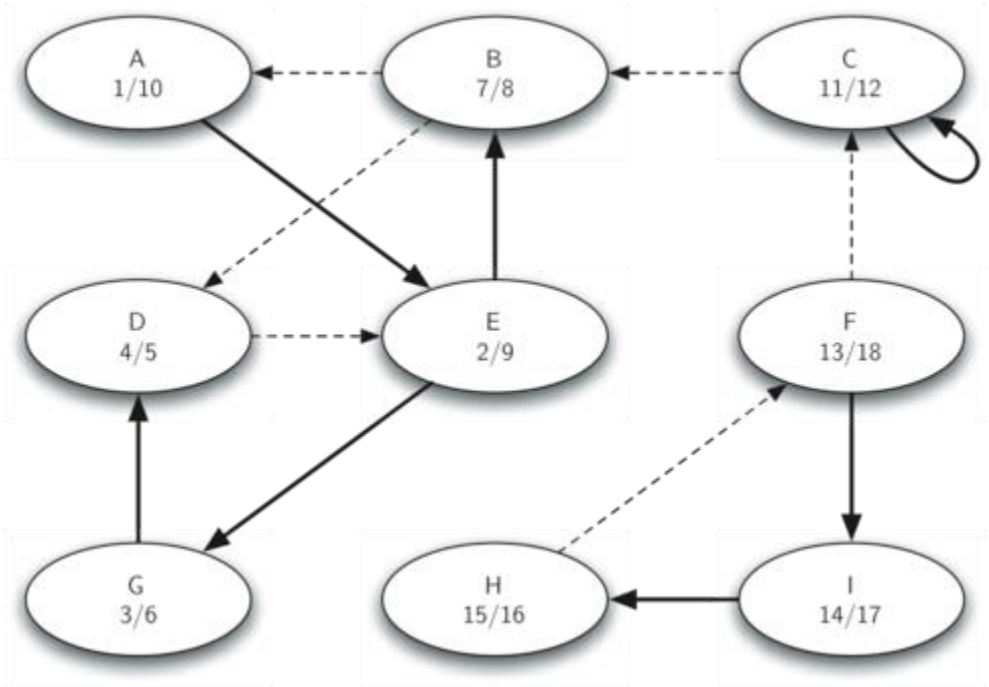
**2. مولفه های همبند قوی گراف داده شده را بیایید.**
ابتدا روی گراف اصلی DFS میزنیم. به جای استک از زمان خروج از هر راس استفاده میکنیم.



# گراف‌های همبند قوی (مثال نوع 2)

**2. مولفه های همبند قوی گراف داده شده را بیابید.**

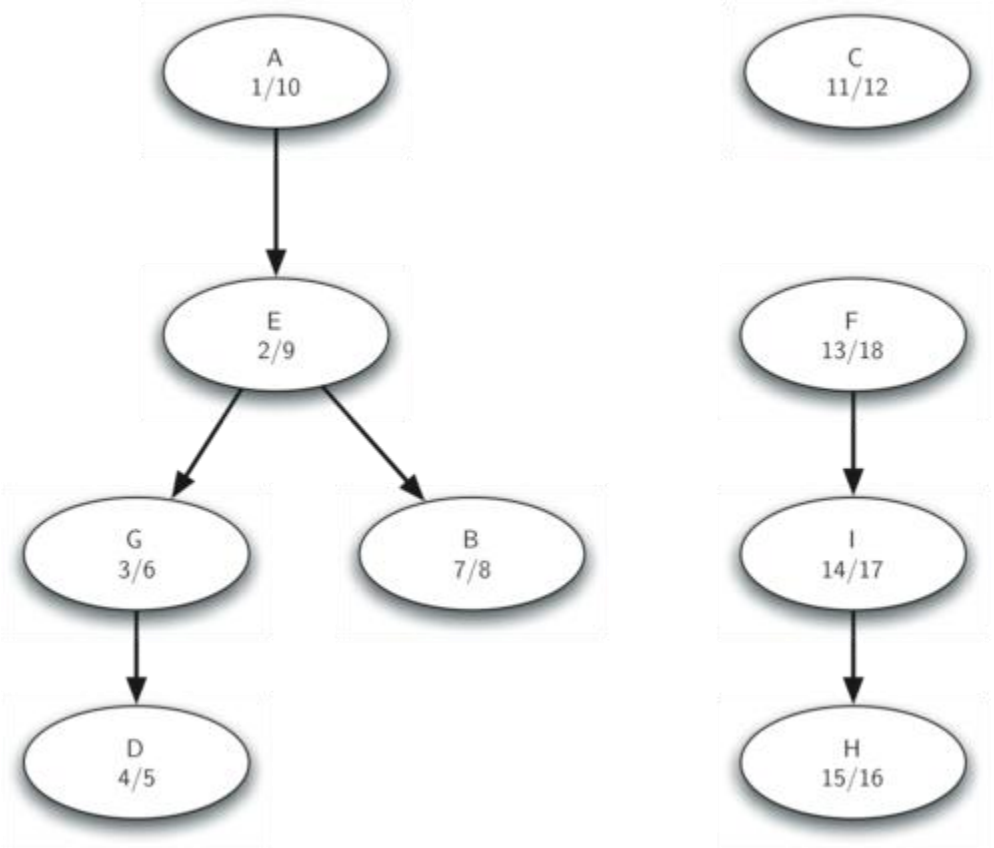
حال روی گراف برعکس شده عمل DFS را انجام میدهیم ولی اولویت زمان خاتمه آن ها ست. (از کم به زیاد)



# گراف‌های همبند قوی (مثال نوع 2)

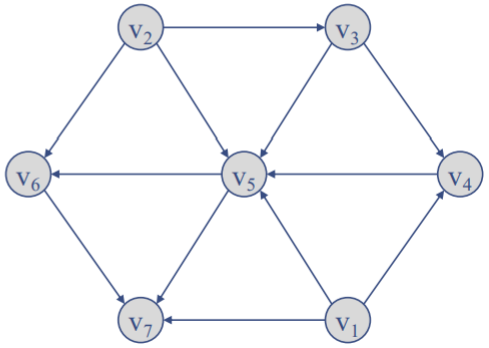
**2. مولفه های همبند قوی گراف داده شده را بیایید.**

هردنباله ای از اعداد که داریم یک مولفه همبندی را تشکیل میدهد.



# DAG

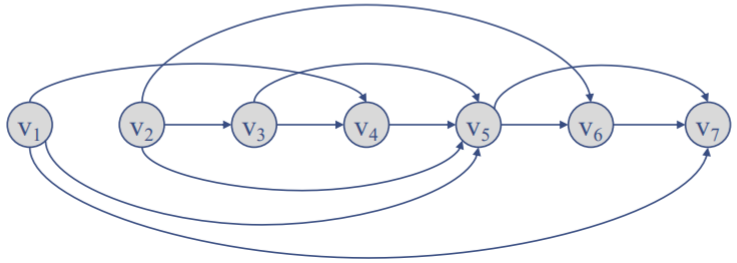
- **Directed Acyclic graph (DAG)**: A directed graph that contains no directed cycles.




- Example: Precedence constraints
  - Edge ($v_i$, $v_j$) means task $v_i$ must occur before $v_j$.

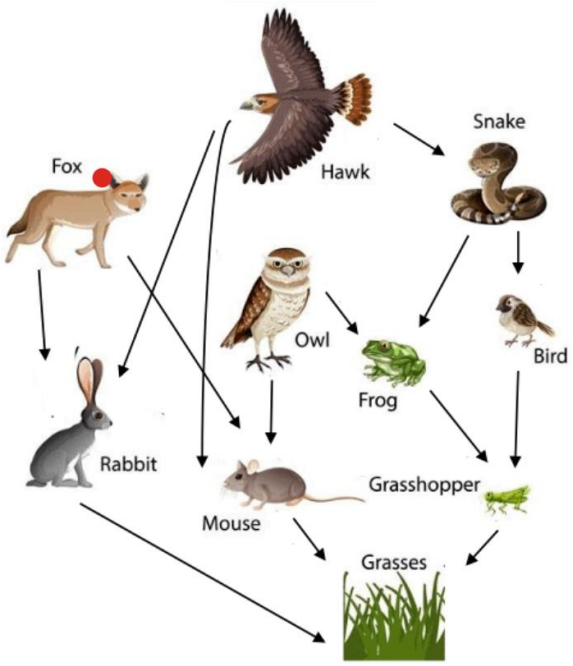
# Topological Order

**Topological order:** An ordering $v_1, v_2, \ldots, v_n$ of the vertices of a directed graph G = (V, E) so that **for every edge $(v_i, v_j)$ we have i < j**.

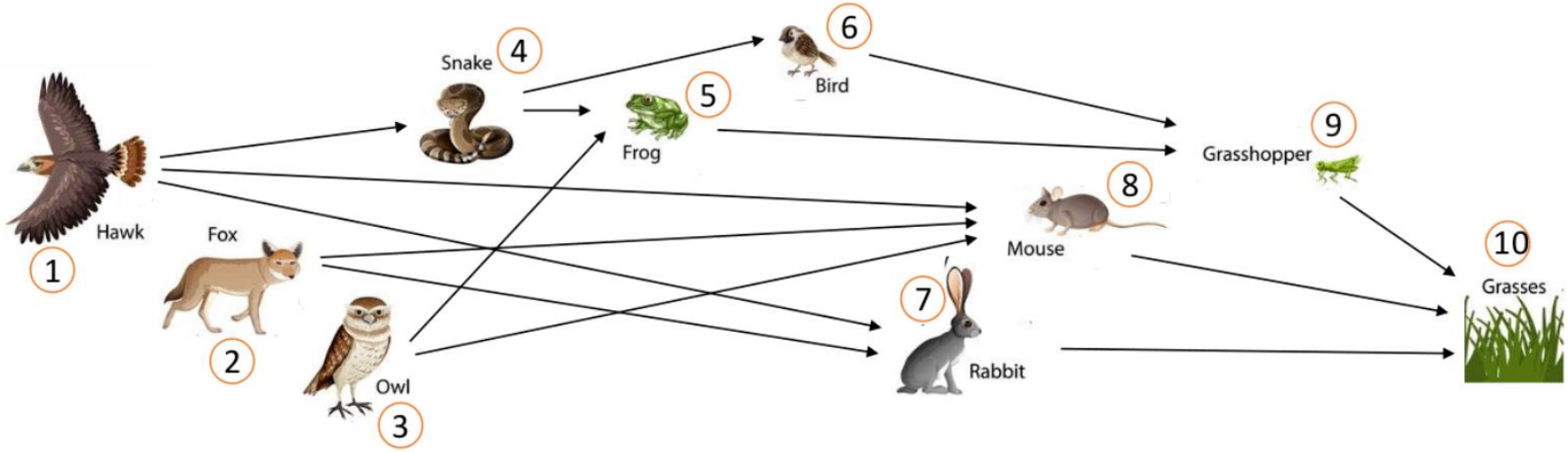



- **Algorithm** for computing a topological ordering of $G(V, E)$:
  - Find a node $v$ with no incoming edges and order it first.
  - Delete $v$ from $G$.
  - Recursively compute a topological ordering of $G(V - \{v\}, E)$ and append this order after v.
- This algorithm finds a topological order in O(m + n) time.

# Topological Order (Example)

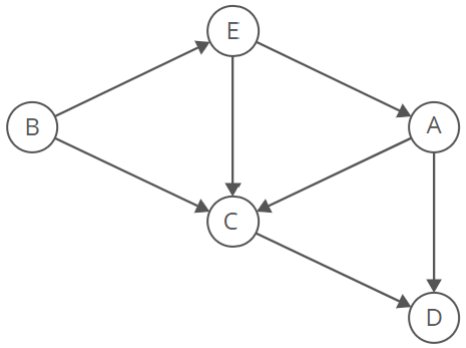



گراف اولیه زنجیره غذایی

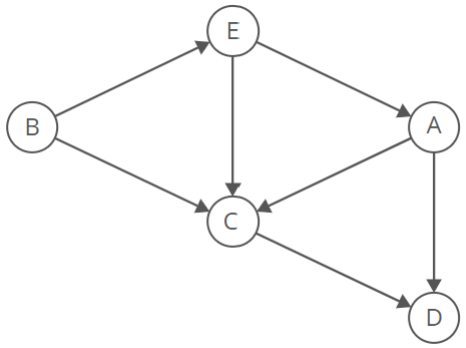


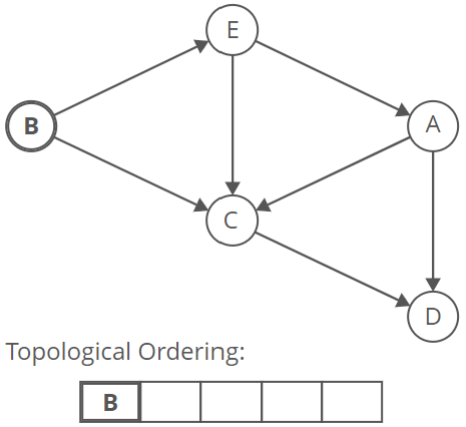

گراف مرتب شده topological زنجیره غذایی

# Topological Order (Example)



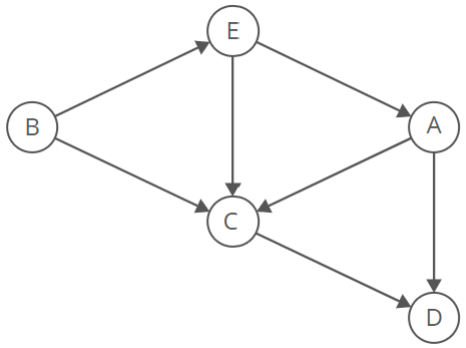
# Topological Order (Example)

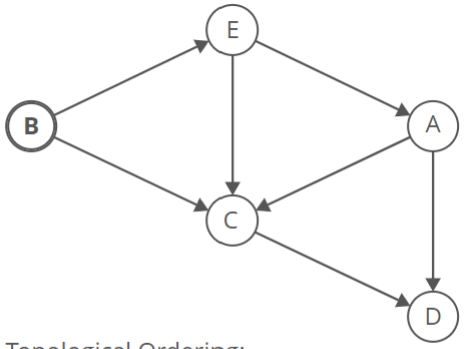


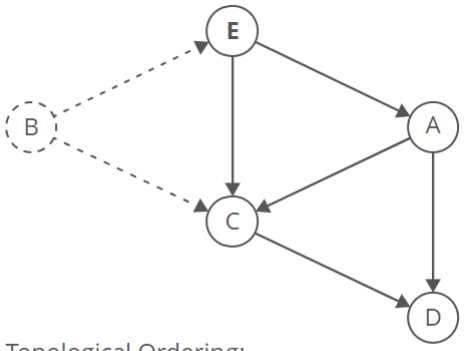

Topological Ordering:

# Topological Order (Example)



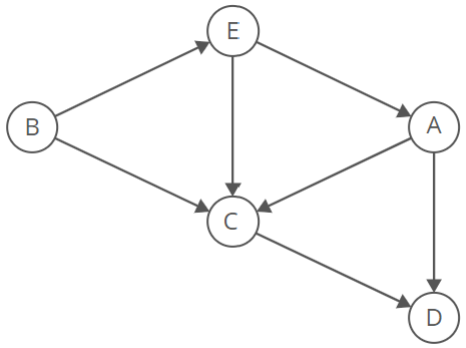



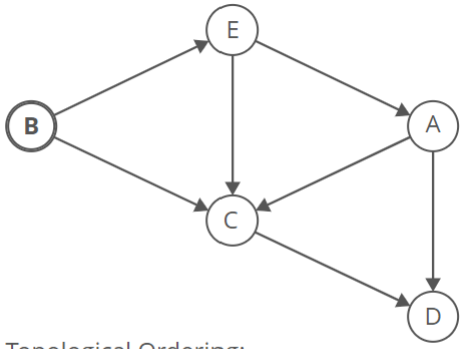

Topological Ordering:


Topological Ordering:

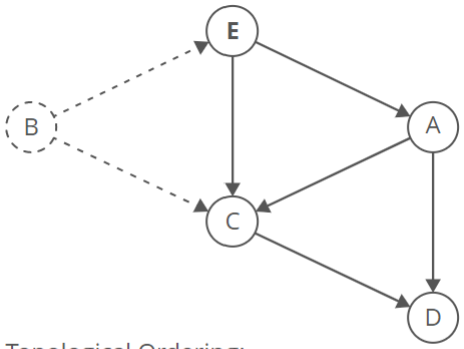
# Topological Order (Example)



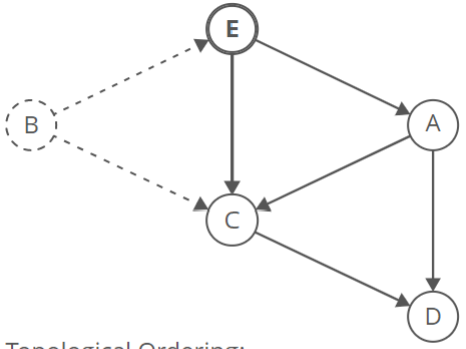

Topological Ordering:

| B | | | | |
|---|---|---|---|---|




Topological Ordering:

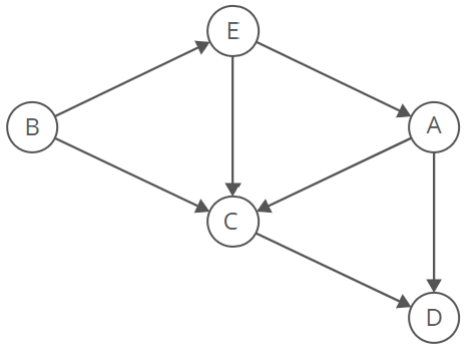
| B | | | | |
|---|---|---|---|---|

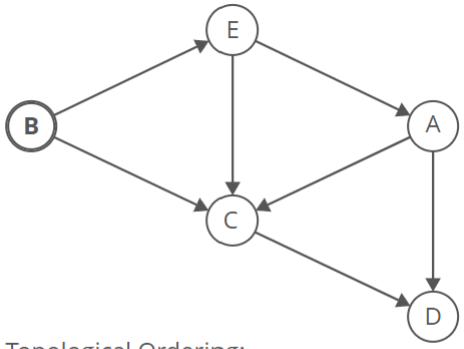



Topological Ordering:

| B | E | | | |
|---|---|---|---|---|

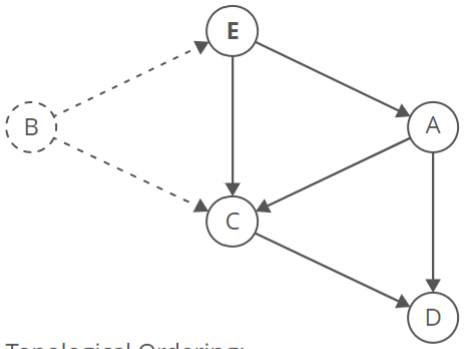
# Topological Order (Example)



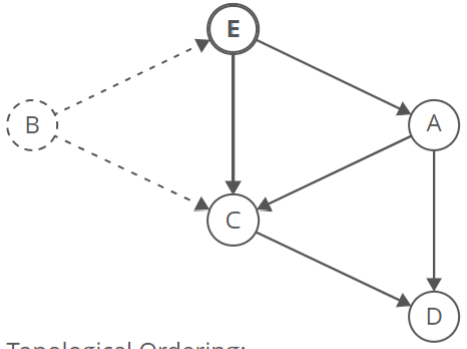

Topological Ordering:

| B | | | | |
|---|---|---|---|---|



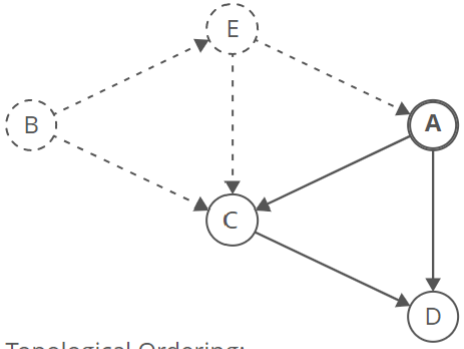

Topological Ordering:

| B | | | | |
|---|---|---|---|---|




Topological Ordering:

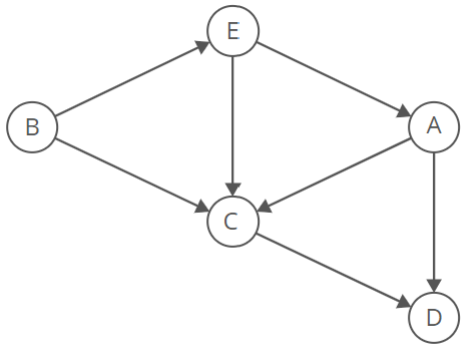
| B | E | | | |
|---|---|---|---|---|

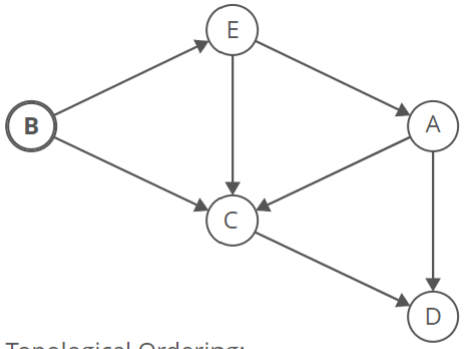



Topological Ordering:

| B | E | A | | |
|---|---|---|---|---|

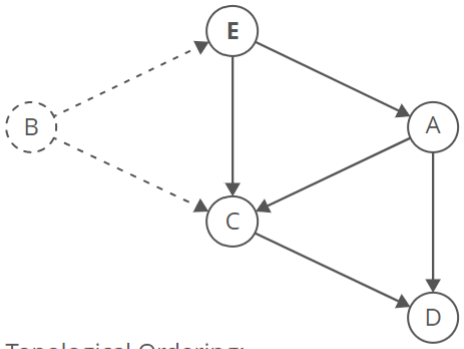
# Topological Order (Example)



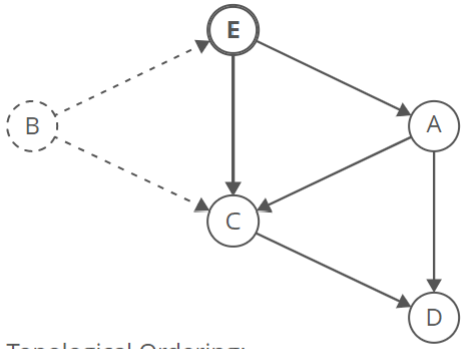

Topological Ordering:

| B | | | | |
|---|---|---|---|---|



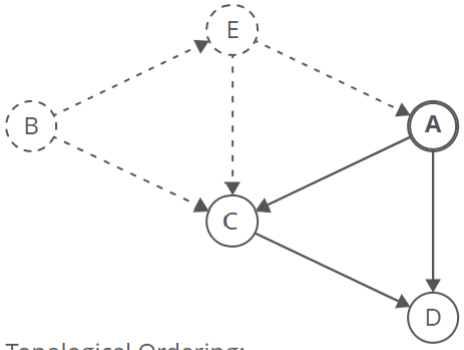

Topological Ordering:

| B | | | | |
|---|---|---|---|---|



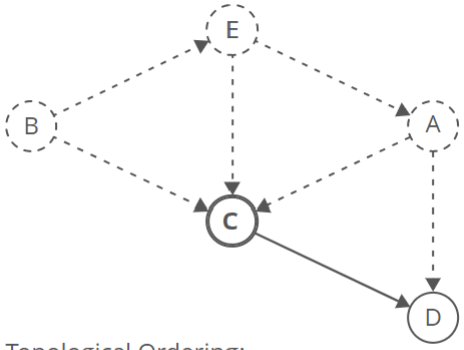

Topological Ordering:

| B | E | | | |
|---|---|---|---|---|




Topological Ordering:

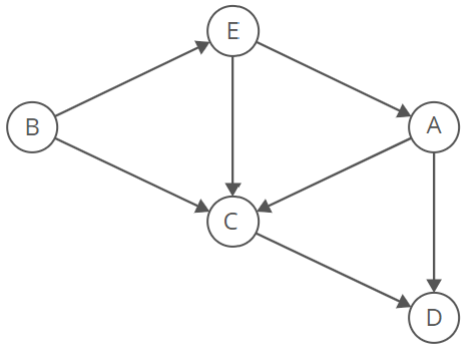
| B | E | A | | |
|---|---|---|---|---|

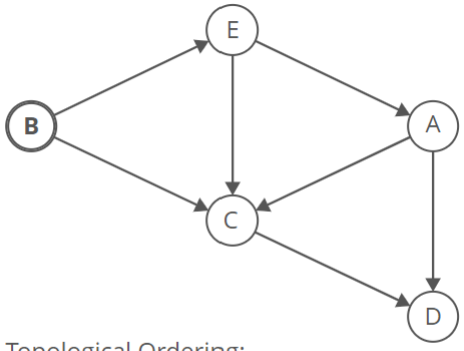



Topological Ordering:

| B | E | A | C | |
|---|---|---|---|---|

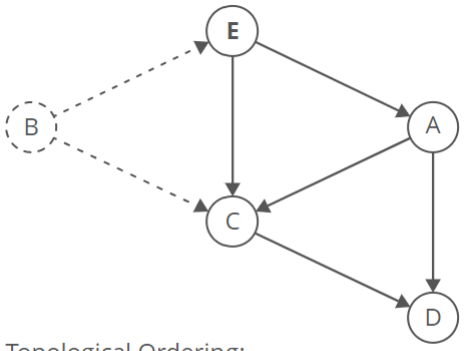
# Topological Order (Example)



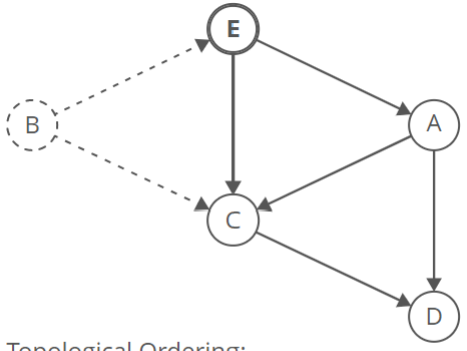

Topological Ordering:

| B | | | | |
|---|---|---|---|---|



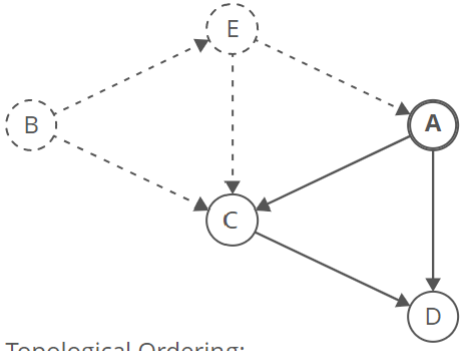

Topological Ordering:

| B | | | | |
|---|---|---|---|---|



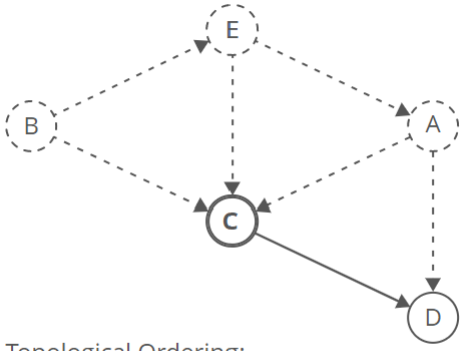

Topological Ordering:

| B | E | | | |
|---|---|---|---|---|



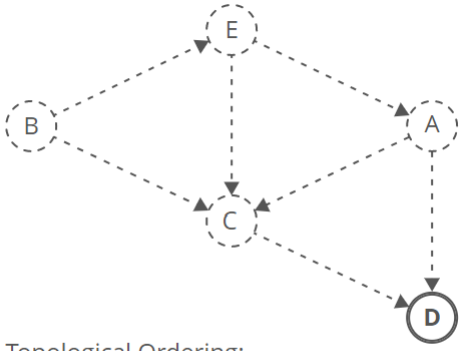

Topological Ordering:

| B | E | A | | |
|---|---|---|---|---|




Topological Ordering:

| B | E | A | C | |
|---|---|---|---|---|




Topological Ordering:

| B | E | A | C | D |
|---|---|---|---|---|

# خسته نباشید!

داریوش کاظمی - اشکان ودادی