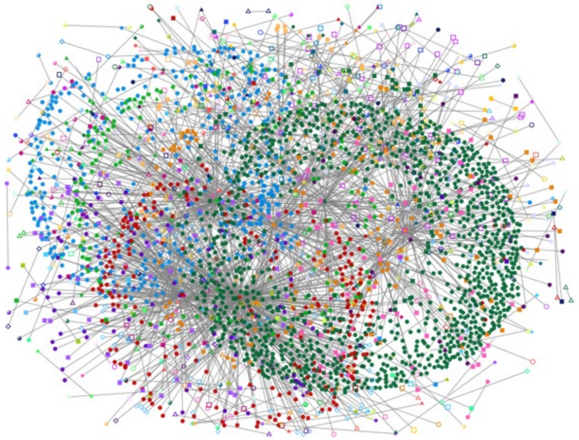
[CSED233-01] Data Structure

Graph Traversals

Jaesik Park



POSTECH

Graph Traversal/Search

- Visiting the vertices of a graph in some specific order (in some organized manner)
  - Similar in concept to a tree traversal

- Many graph problems can be solved using a search method

- Two commonly used search methods
  - DFS (Depth-First Search)
  - BFS (Breadth-First Search)

Graph Traversal/Search

- Troublesome cases
  - If the graph is not connected
    → unreachable to all vertices
  - If the graph contains cycles
    → infinite loop

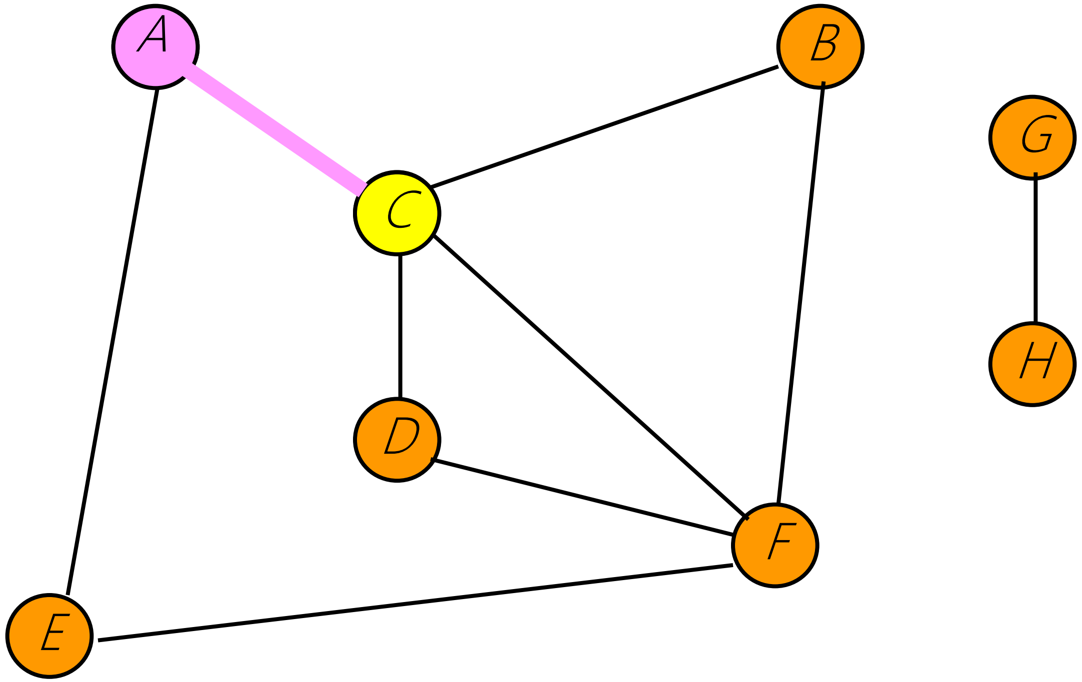
- Maintain a mark bit for each vertex to solve the problems

Depth-First Search (DFS)

- Going as deep as possible on each child before going to the next sibling
  - Generalization of the preorder traversal of a tree to a graph
- Recursive implementation:
  - Using Recursion Stack

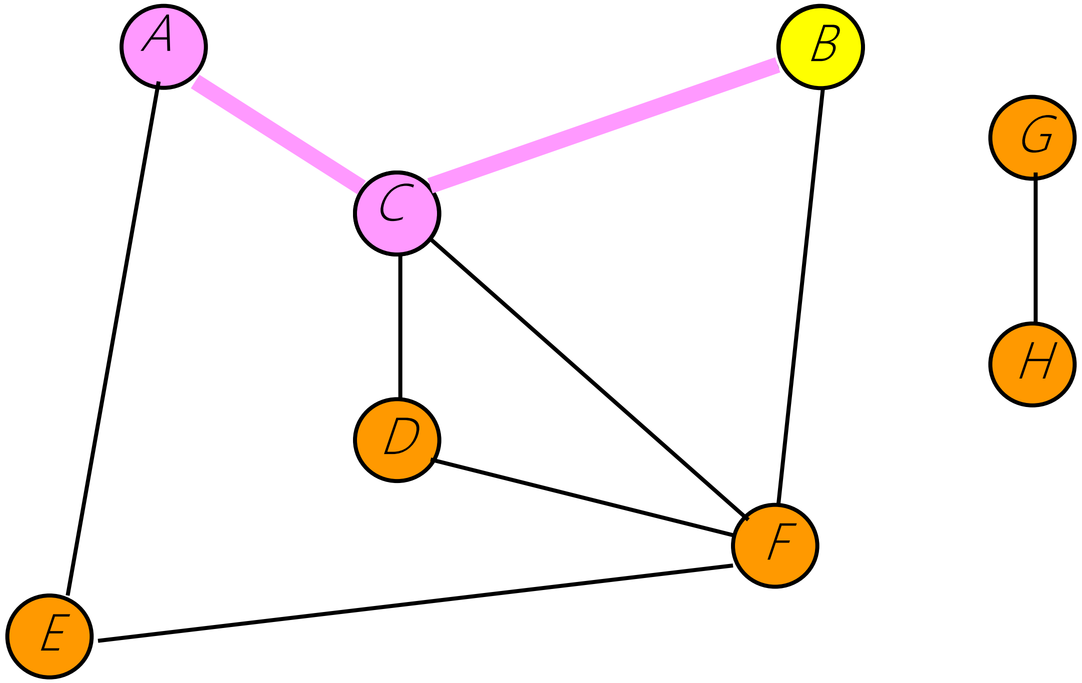
```
DFS (v) {
    mark[v] := visited;
    for (each unvisited vertex w adjacent from v)
        DFS(w);
}
```

Depth-First Search (DFS)



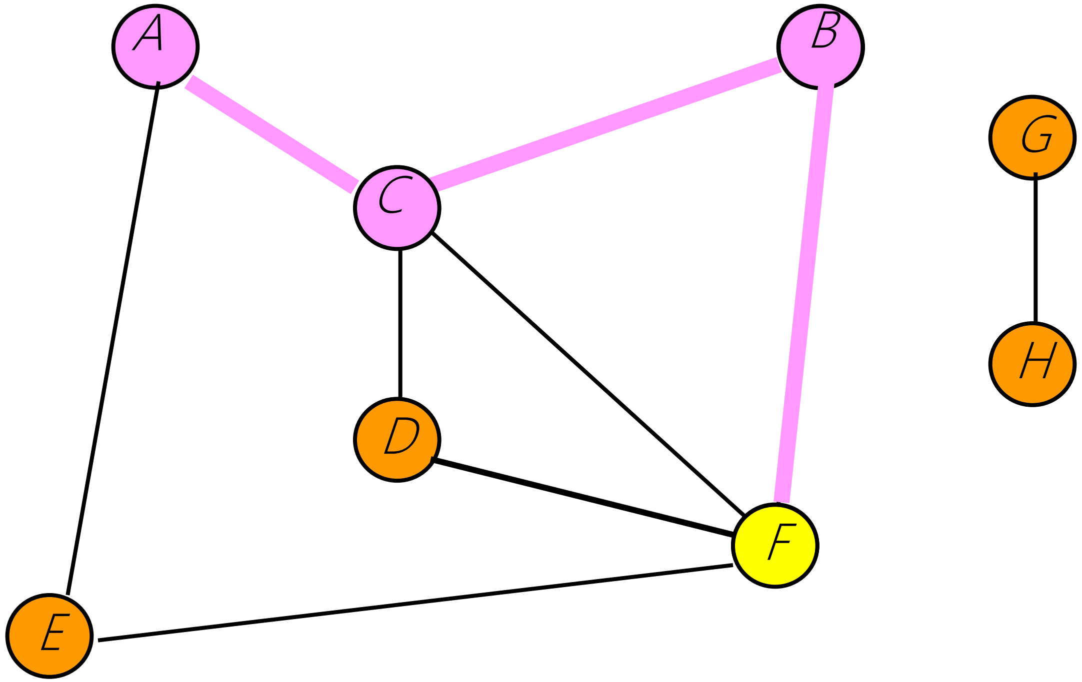
- Start at vertex *A*
- Mark vertex *A* and do DFS from either *C* or *E*

Depth-First Search (DFS)



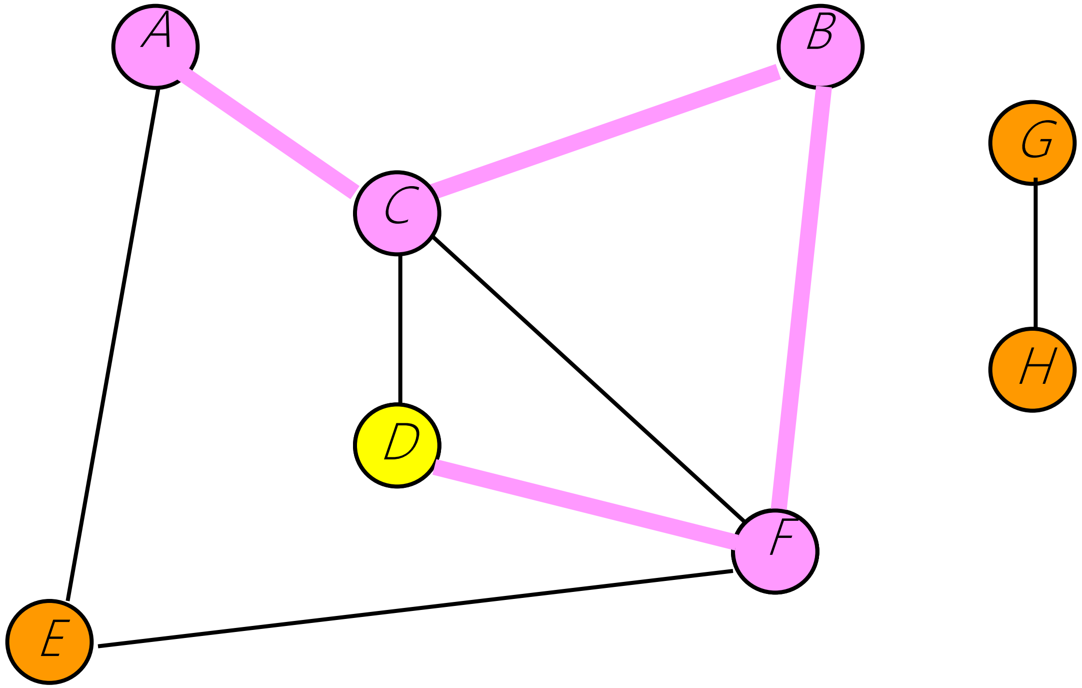
- Mark vertex *C* and do DFS from either *B*, *D* or *F*

Depth-First Search (DFS)



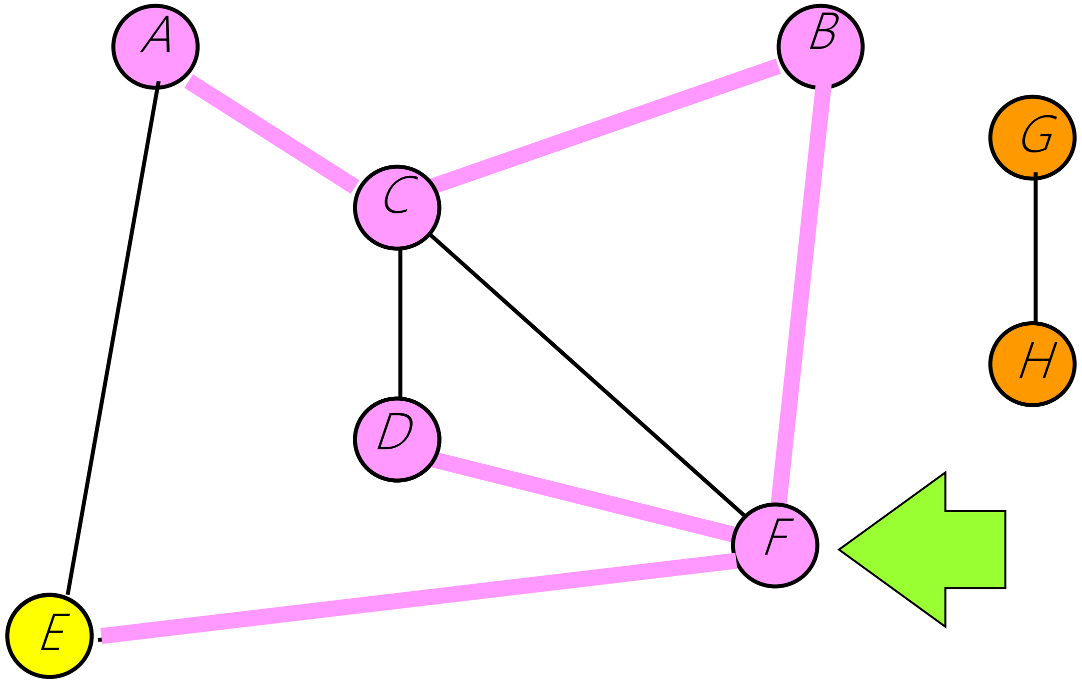
- Mark vertex *B* and do DFS from *F*

Depth-First Search (DFS)



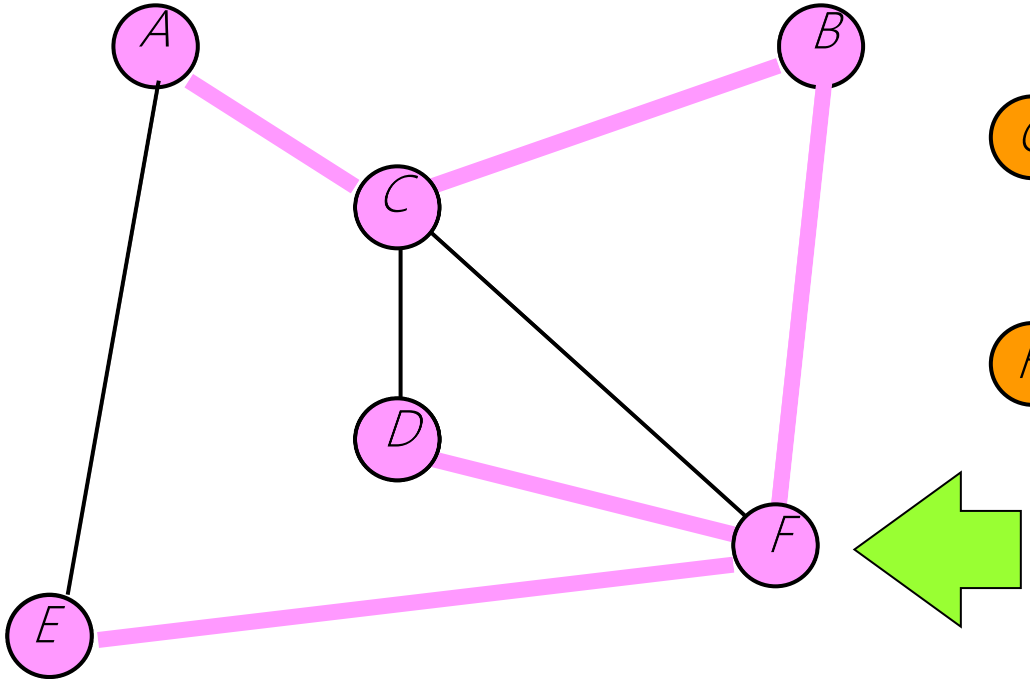
- Mark vertex *F* and do DFS from either *D* or *E*

Depth-First Search (DFS)



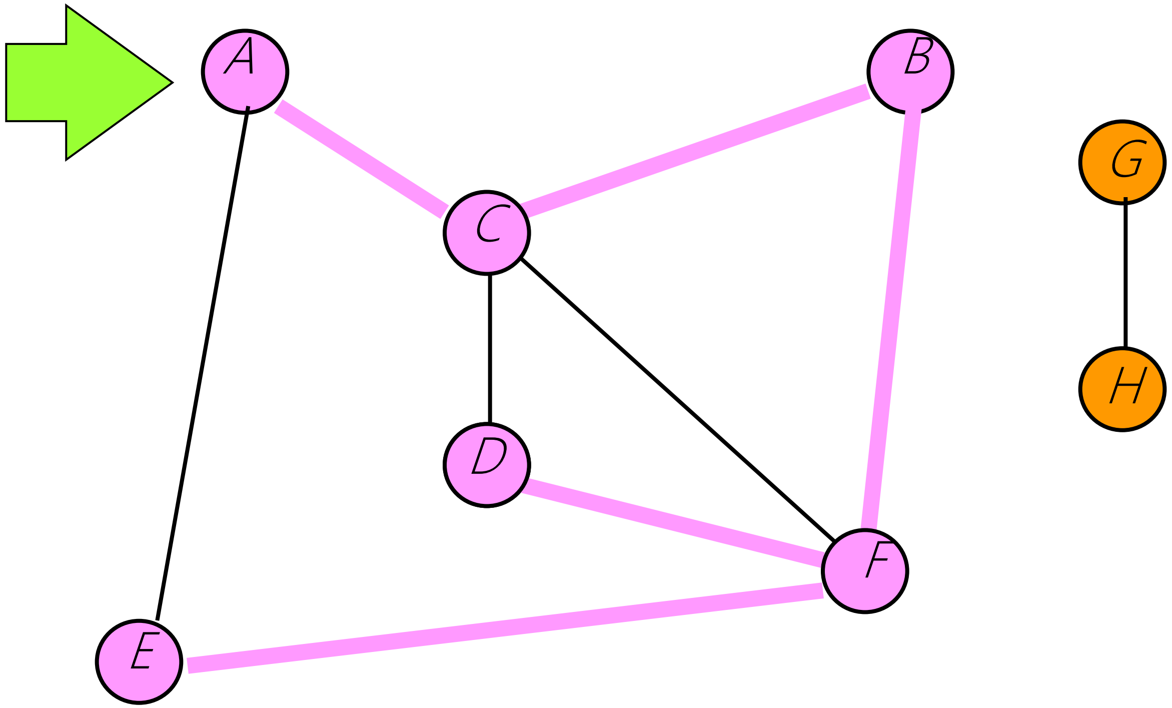
- Mark vertex *D* and return (backtrack) to vertex *F*
- From F, do DFS(*E*)

Depth-First Search (DFS)



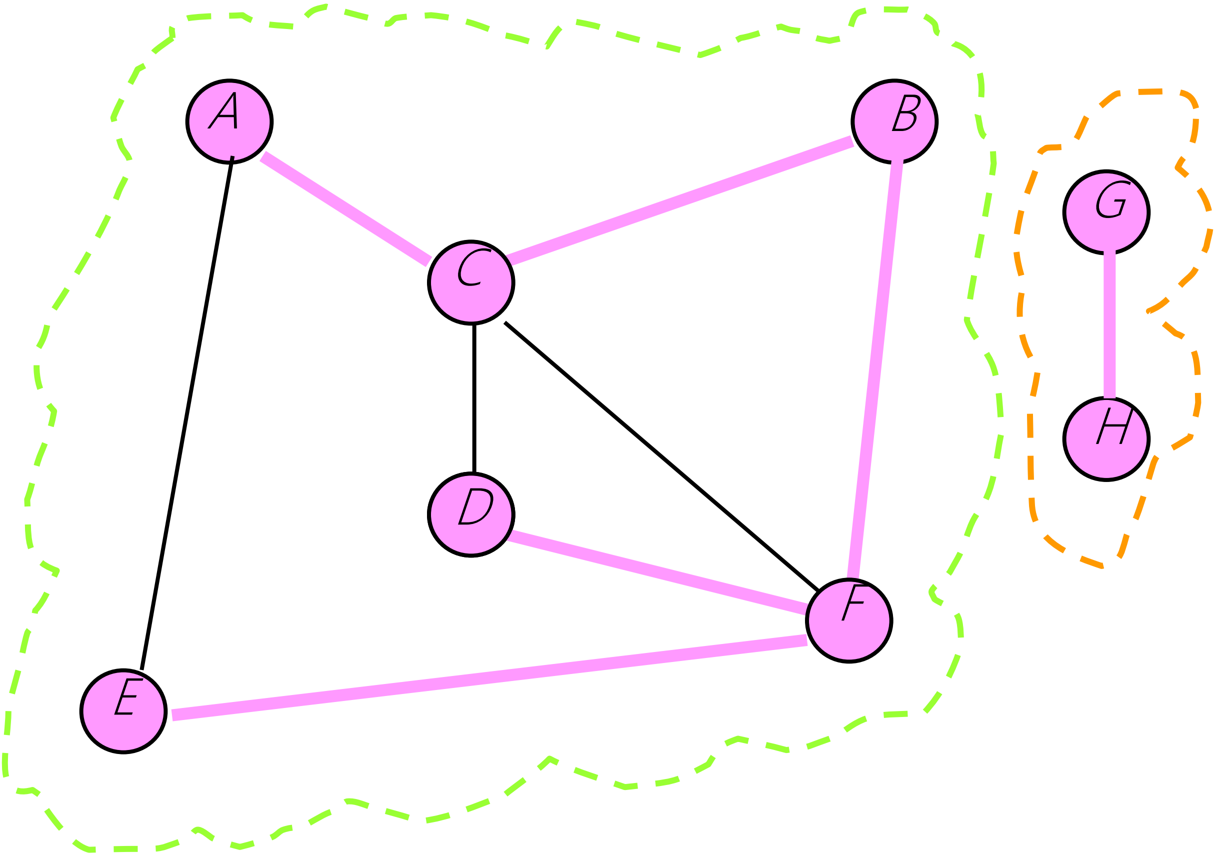
- Mark vertex *E* and return (backtrack) to vertex *F*
- Return to *B*, *C*, and *A* in turn

Depth-First Search (DFS)



- There are still unvisited vertices G and H
- So, we invoke DFS(G) again

Depth-First Search (DFS)



- Depth-first spanning forest (pink edges)
- Two connected components

Time Complexity of DFS

- All *n* vertices are visited only once
- When a vertex is visited, we examine its adjacent vertices
  - *O(vertex degree)* if *adjacency lists* are used
  - *O(n)* if *adjacency matrix* is used
- Total time – requires multiple parameters
  - *O(n + e) (adjacency lists)*
    - $T = (c+d_1) + (c+d_2) + \ldots + (c+d_n) = c*n + e$
    - Aggregate analysis
  - *O(n*n)* (*adjacency matrix*)
    - $T = n*(c_1 + c_2*n)$

Applications of DFS (1)

- How to identify a spanning tree
  - Start DFS at any unvisited vertex
  - If G is connected, the *n-1* edges used to reach unvisited vertices defines a spanning tree

- How to find a path from vertex *v* to vertex *w*
  - Start DFS at vertex *v*
  - Terminate when vertex *w* is visited or when there are no more unvisited vertices (whichever occurs first)

Applications of DFS (2)

- Is the graph is connected?
  - Start DFS at any vertex
  - G is connected if all vertices get visited

- How to identify connected components
  - Start DFS at any unvisited vertex
  - Newly visited vertices (plus edges between them) define a component
  - Repeat above until all vertices are visited

Applicable to digraphs (directed graphs)

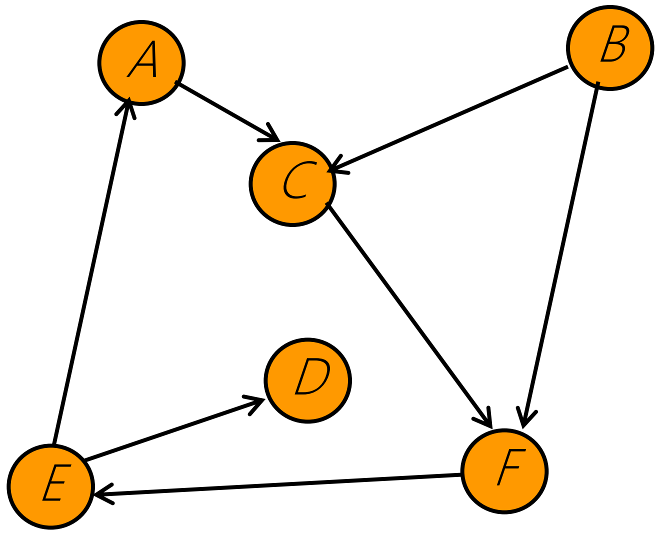
How to Find the Strong Components

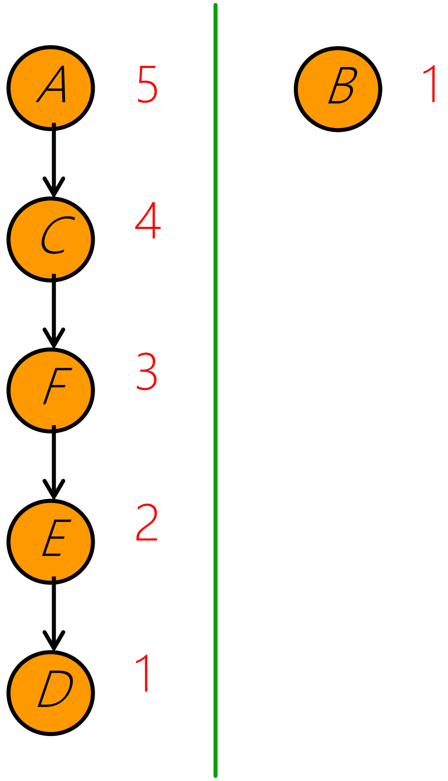
- Input: a digraph G
- Output: strong components
- Algorithm

① Do DFS of G (numbering the vertices in order of completion of the recursive calls)
② Construct a new digraph G_r by reversing the arc directions in G
③ Do DFS on G_r, starting at the highest numbered vertex according to the numbering at step ①

Example: Strong Components

- A given digraph G

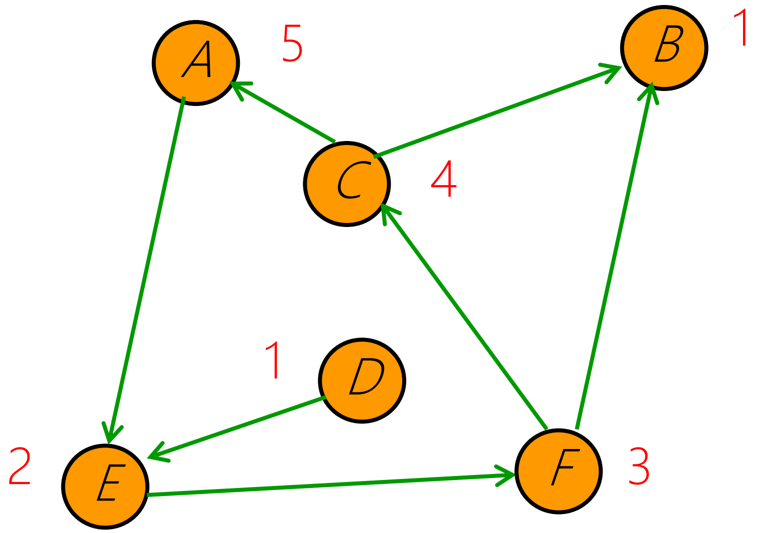




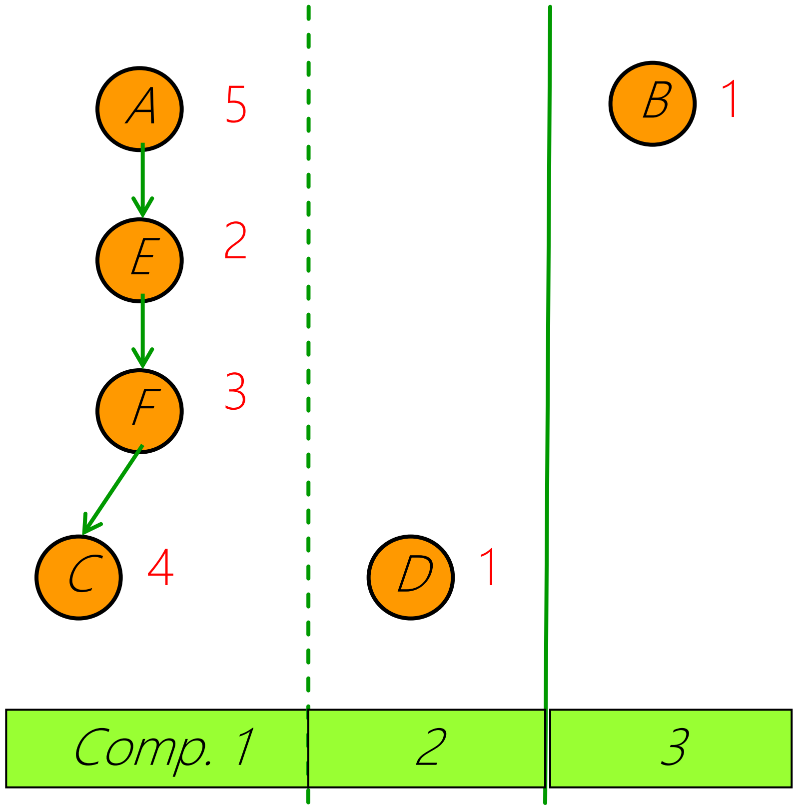
- After step (1) – DFS(G) with numbering

Example: Strong Components

- After step (2) – Construct a new digraph G_r

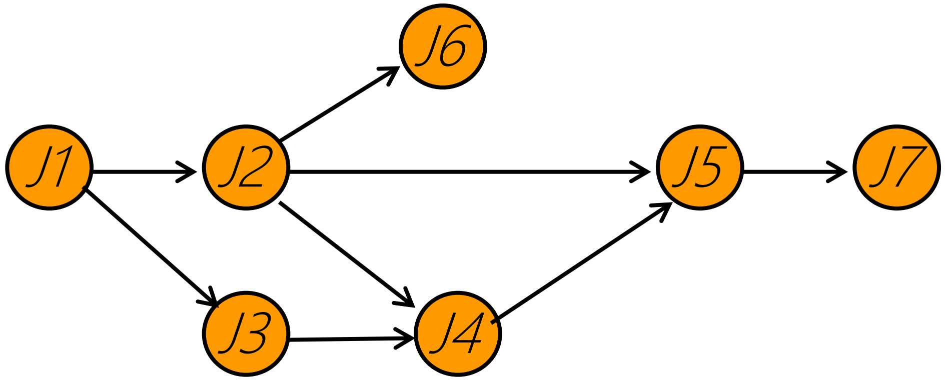


- After step (3)
  - DFS(G_r)



Topological Sort

- A linear ordering of the vertices of a DAG s.t.
  - if there is a path from *v* to *w*, then *v* appears before *w* in the ordering



  - Acceptable topological sorts ?
    - J1, J2, J3, J4, J5, J6, J7
    - J1, J3, J2, J6, J4, J5, J7 etc.

Stack-Based Topological Sort

- DFS-based algorithm
  - Do a DFS on the graph (starting with any vertex)
  - When a vertex is visited, no action
  - When the recursion pops back to the vertex, print the vertex

    → Topological sort in reverse order

```
TOPSORT (v) {
    mark[v] := visited;
    for (each unvisited vertex w adjacent from v)
        TOPSORT(w);
    print(v)
}
```

Queue-Based Topological Sort

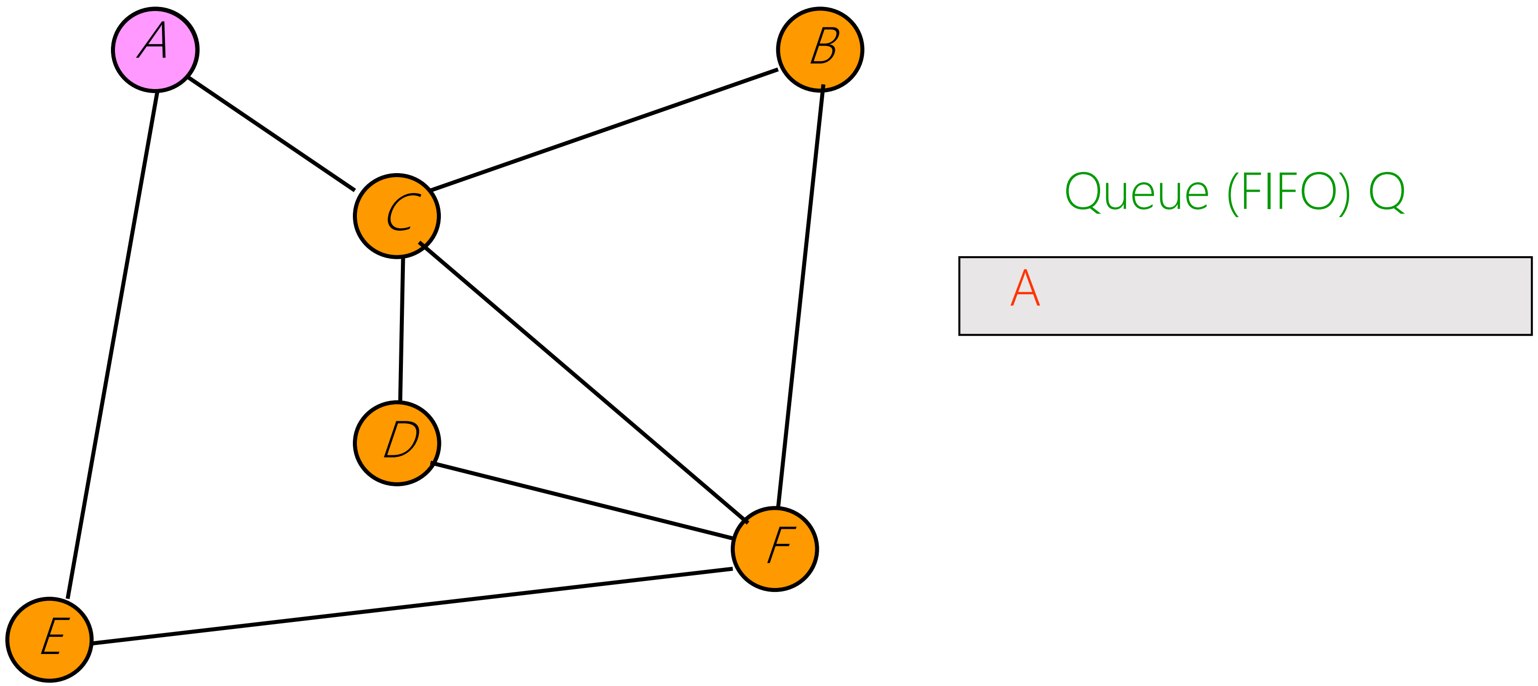
Breadth-First Search (BFS)

- Going as broad as possible on each depth before going to the next depth

- Iterative implementation:
  - Similar to DFS except that a *queue* replaces the *recursion stack*

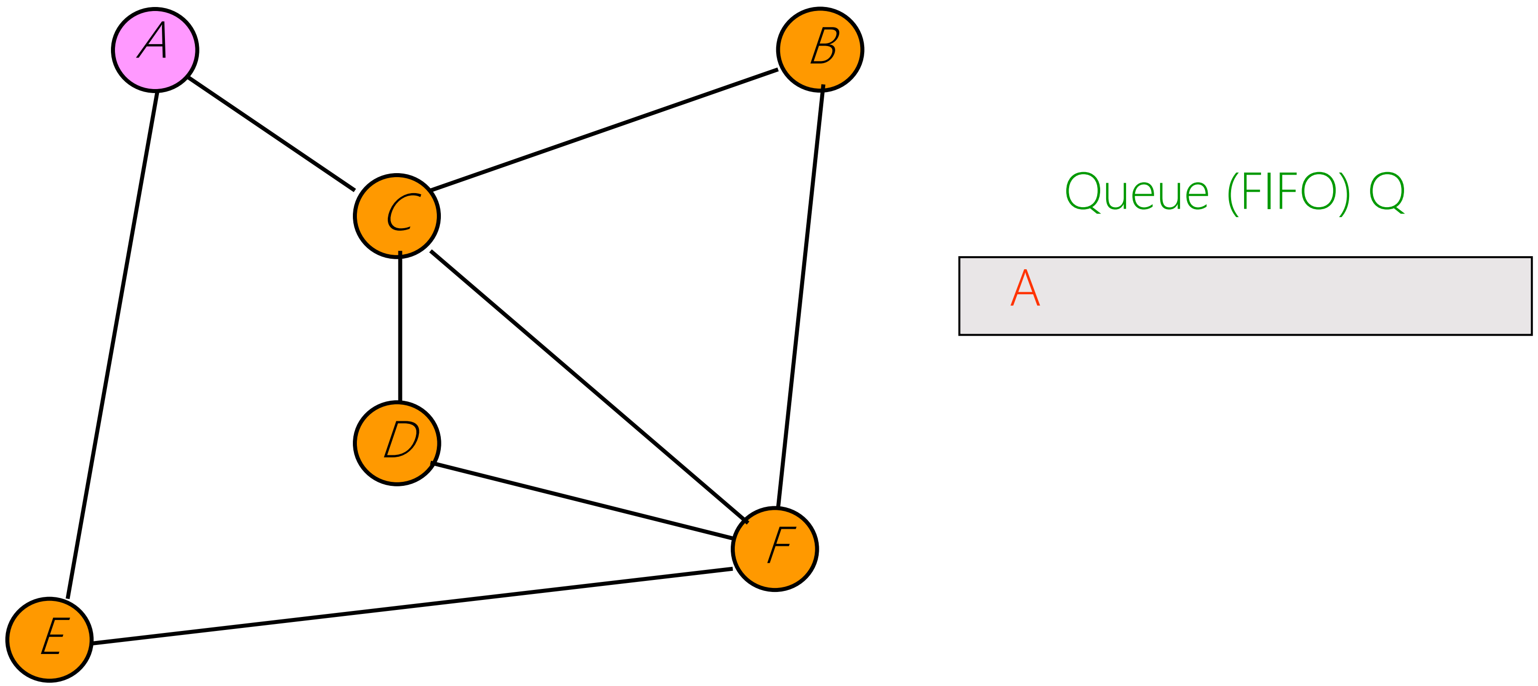
```
Visit/Mark a start vertex & put into the queue;
Repeat until the queue is empty {
    Remove a vertex from the queue;
    Visit/Mark its unvisited adjacent vertices;
    Put the newly visited vertices into the queue
}
```

Breadth-First Search (BFS)



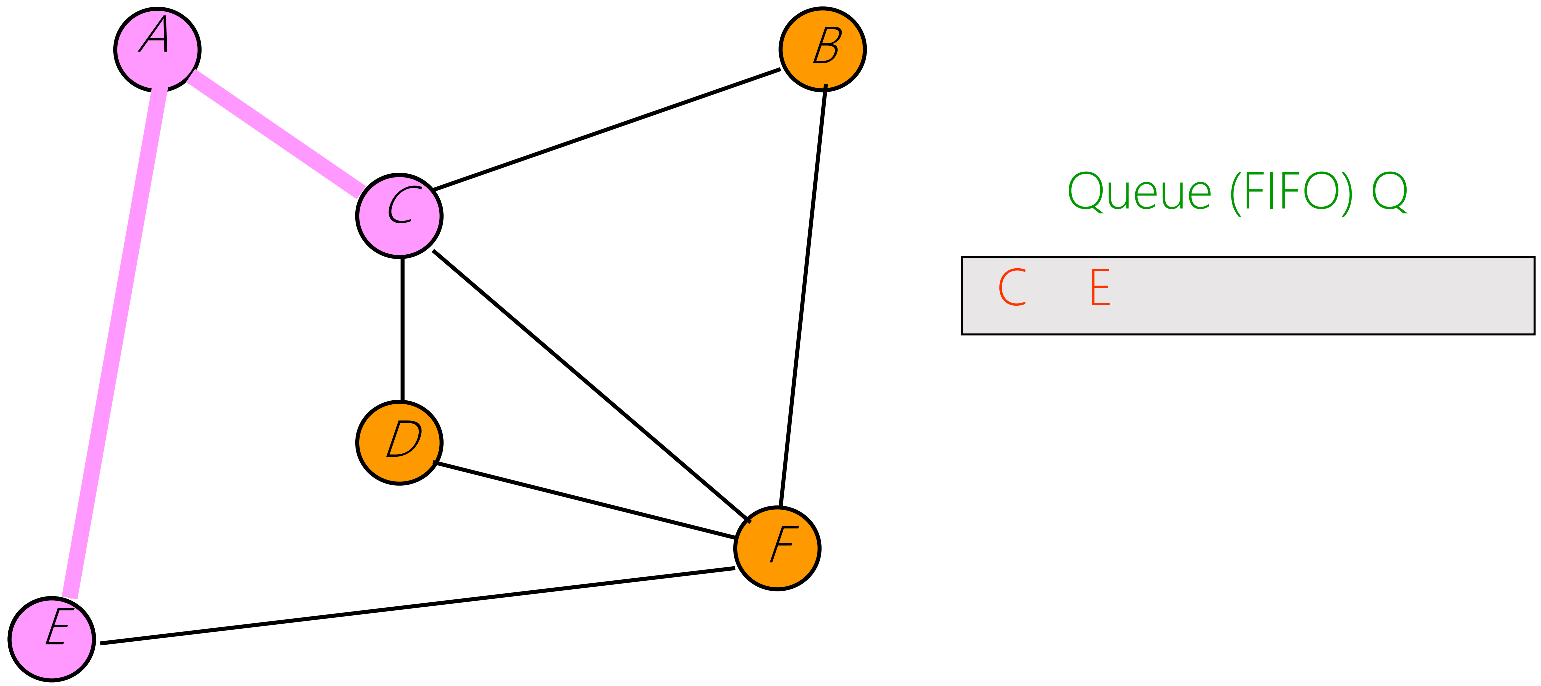
- Start at vertex *A*
- Mark vertex *A* & put it into queue Q

Breadth-First Search (BFS)



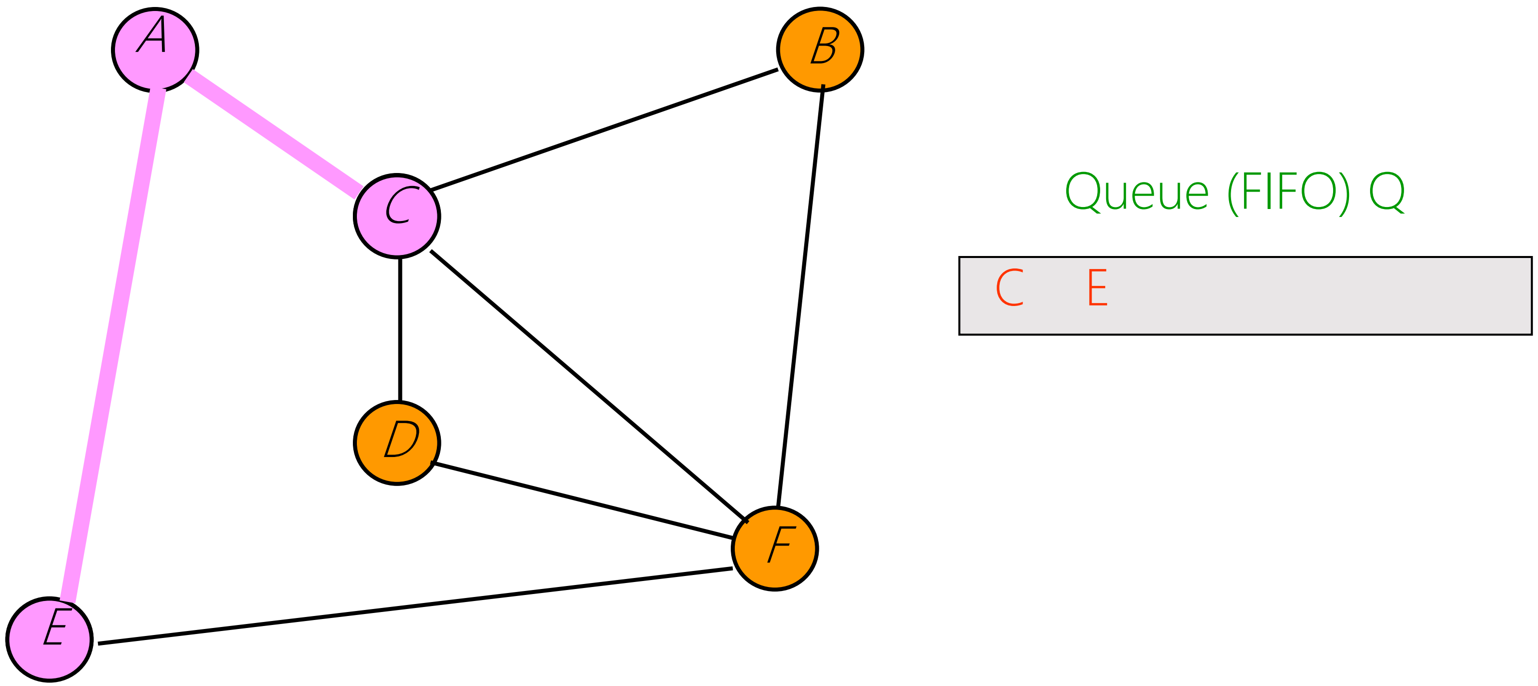
- Remove *A* from Q; Visit all unvisited adjacent vertices; Put them into Q

Breadth-First Search (BFS)



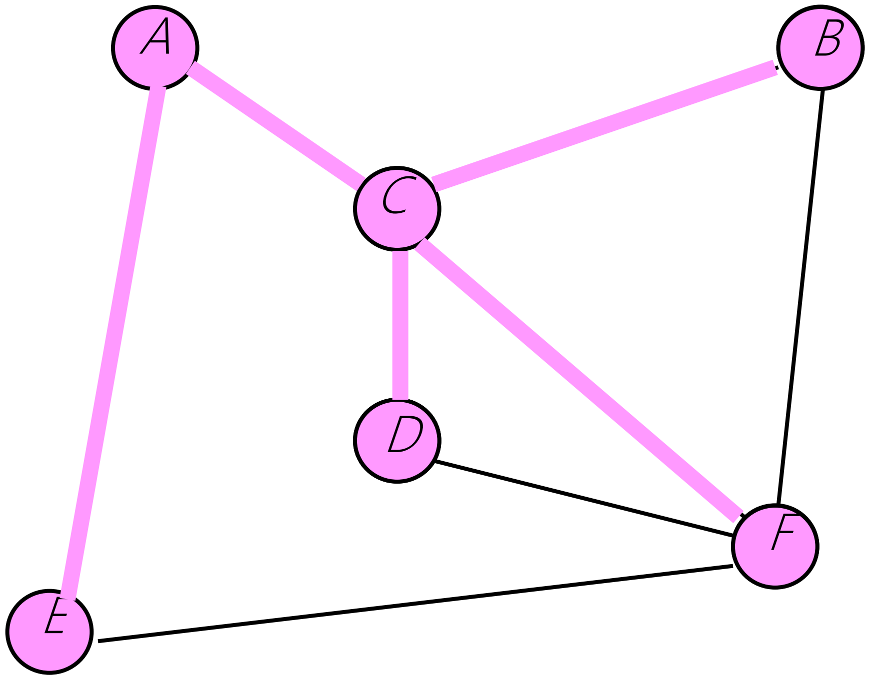
- Remove *A* from Q; Visit all unvisited adjacent vertices; Put them into Q

Breadth-First Search (BFS)



- Remove *C* from Q; Visit all unvisited adjacent vertices; Put them into Q

Breadth-First Search (BFS)

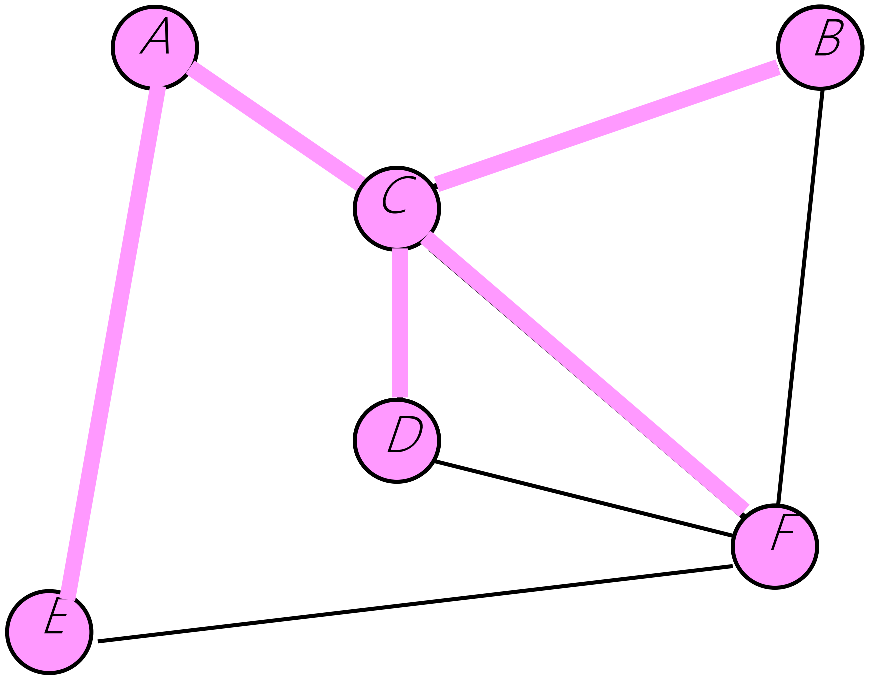


Queue (FIFO) Q

E	B	D	F

- Remove *C* from Q; Visit all unvisited adjacent vertices; Put them into Q

Breadth-First Search (BFS)

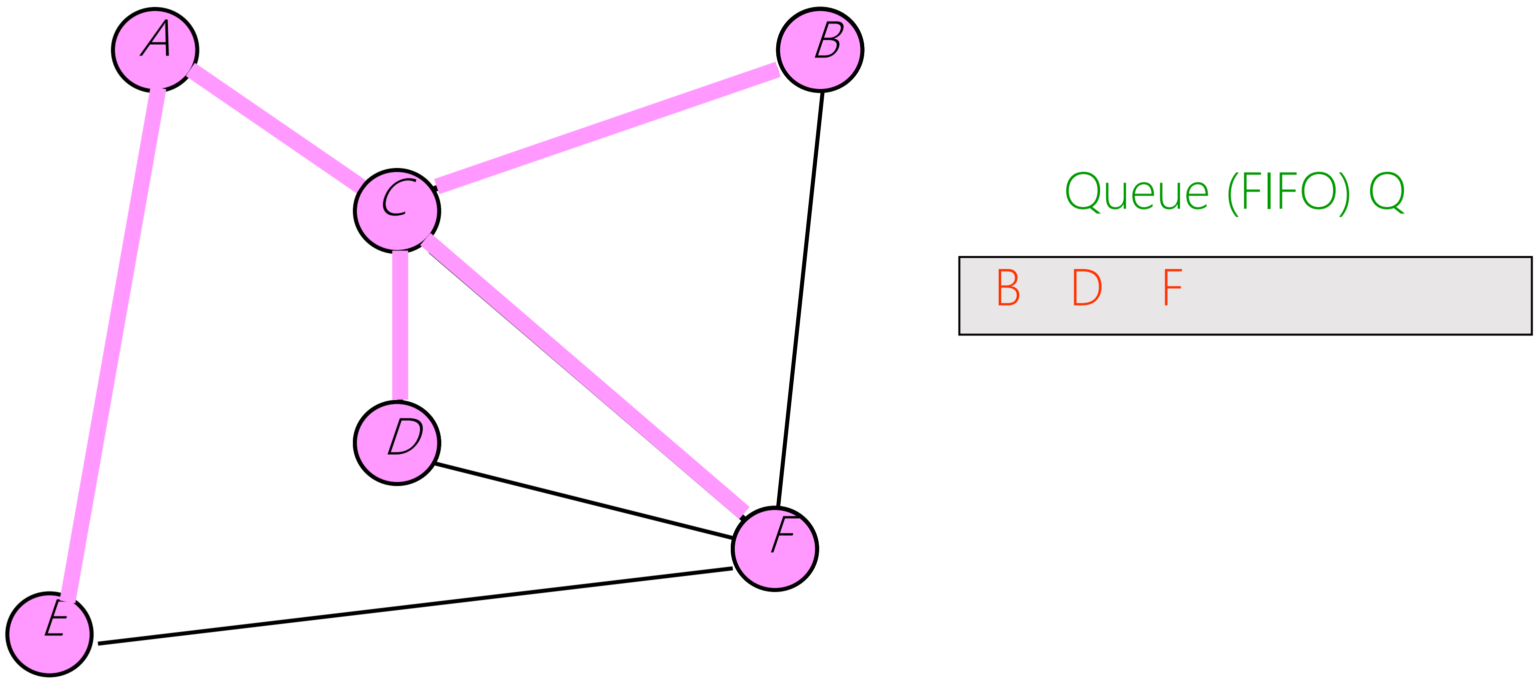


Queue (FIFO) Q

E B D F

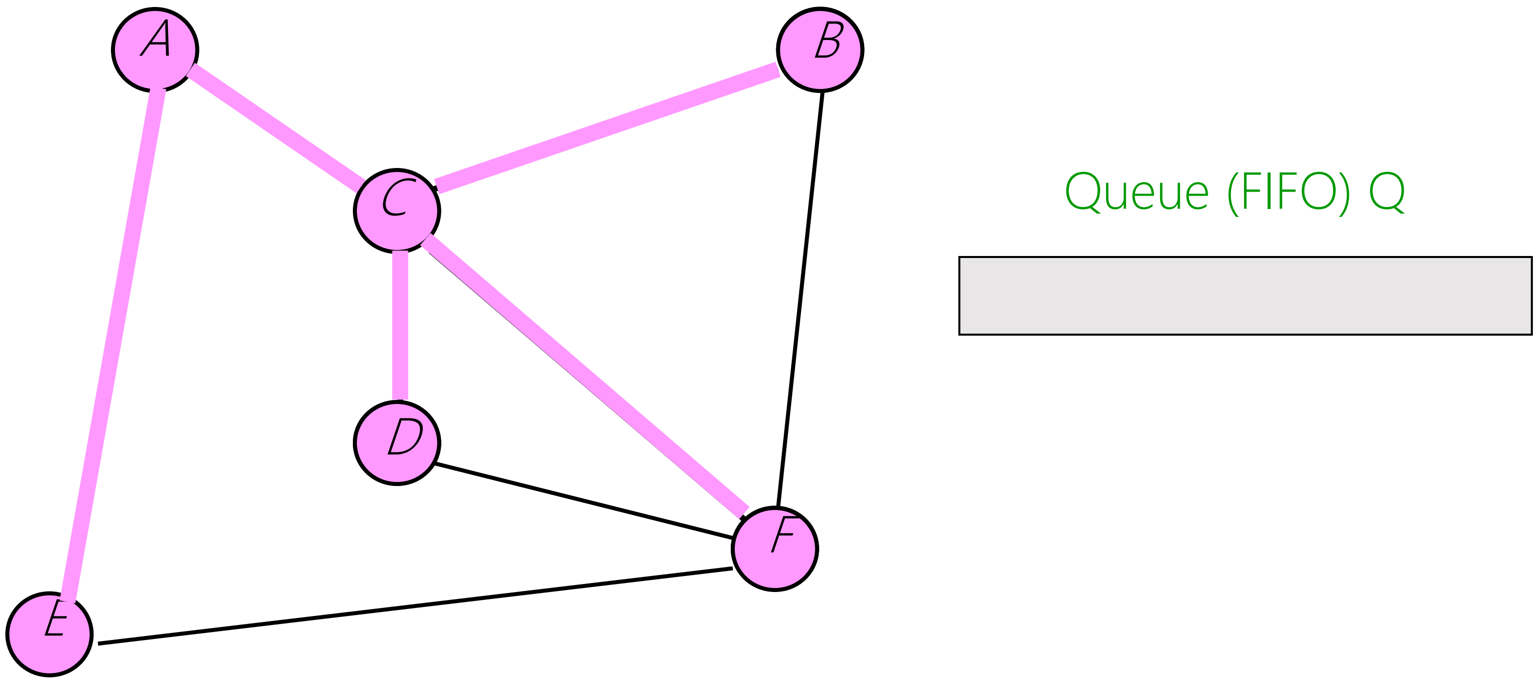
- Remove *E* from Q; Visit all unvisited adjacent vertices; Put them into Q

Breadth-First Search (BFS)



- Remove *E* from Q; Visit all unvisited adjacent vertices; Put them into Q
- Similarly, *B*, *D*, and *F* can be removed from Q

Breadth-First Search (BFS)



- Queue is empty
- Search terminates

Breadth-First Search

- Same time complexity as DFS
- Same properties with respect to graph connectivity, connected components, spanning trees, path finding
- There are problems for which BFS is better than DFS, and vice versa

References

- Further reading list and references
  - https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/

- Slide credit
  - Jaesik Park
  - Seung-Hwan Baek
  - Jong-Hyeok Lee