Graph Traversal Algorithms III

Instructor: Meng-Fen Chiang

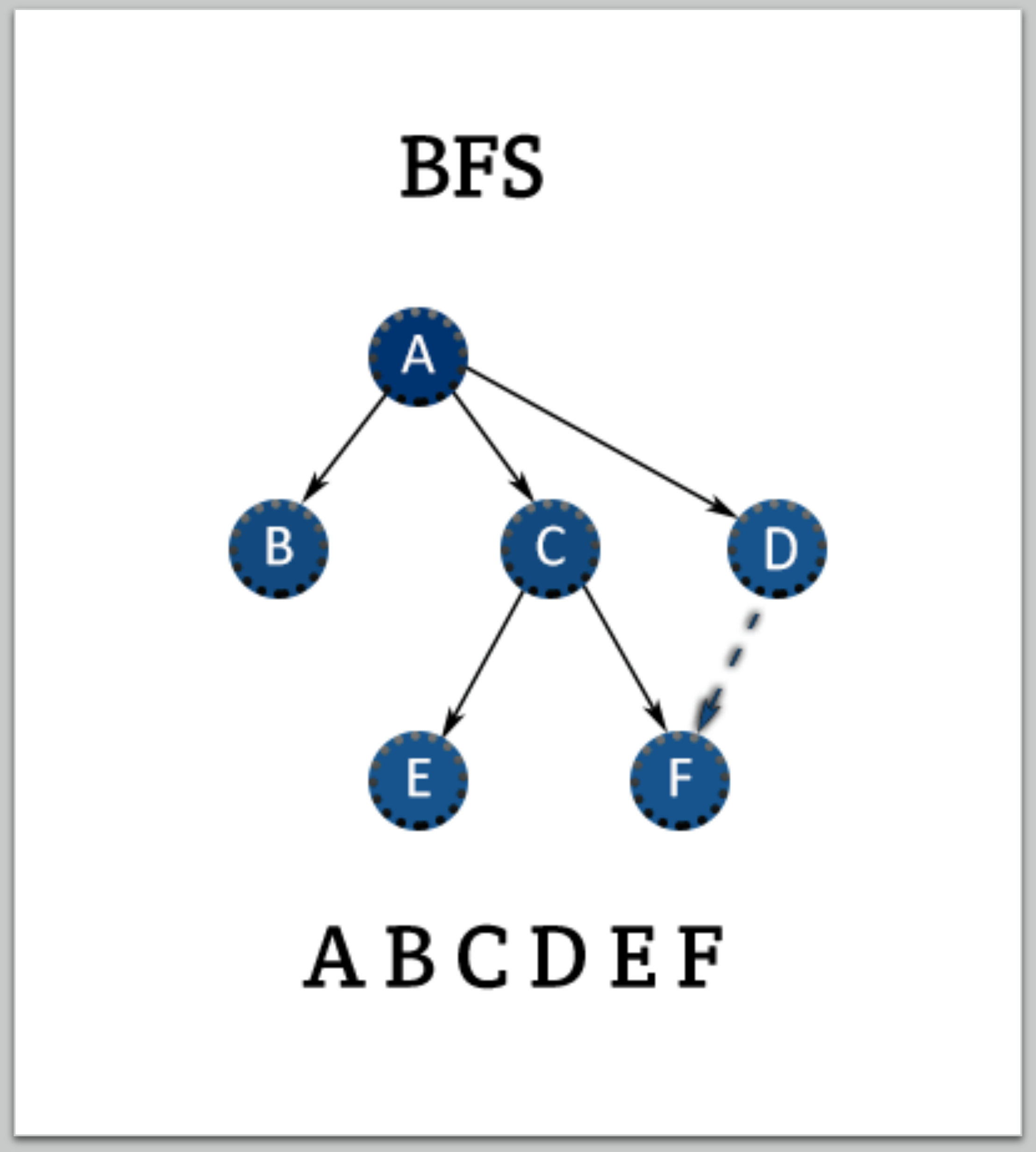
COMPSCI: WEEK 9.5-6

Slides adapted from Mark Wilson, Georgy Gimel'farb, Simone Linz and Tanya Gvozdeva

OUTLINE

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)

- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS



Breadth-first Search Algorithm (BFS)

- BFS is also a specific implementation of our fundamental graph traversal algorithm (and also known as breadth-first traversal)

- It specifies that we select the next grey vertex to pick as the oldest remaining grey vertex.

The Abstract Data Type: Queue

- Special list in which insertion occurs at one end (tail) and deletion occurs at the other end (head) (first in first out principle).
- Add an element to the list (INSERT or PUSH or ENQUEUE).
- Delete an element (DELETE or POP or DEQUEUE).
- Return the head element without deleting it (GETHEAD or PEEK).

Breadth-first Search Algorithm (BFS)

Algorithm 4 Breadth-first search algorithm

```
1: **function** BFS(digraph $G$)
2:     queue $Q$
3:     array $colour[0..n-1], pred[0..n-1], d[0..n-1]$
4:     **for** $u \in V(G)$ **do**
5:         $colour[u] \leftarrow$ WHITE
6:         $pred[u] \leftarrow$ null
7:     **for** $s \in V(G)$ **do**
8:         **if** $colour[s] =$ WHITE **then**
9:             BFSVISIT($s$)
10:    **return** $pred, d$
```

Iterative View of BFSVISIT

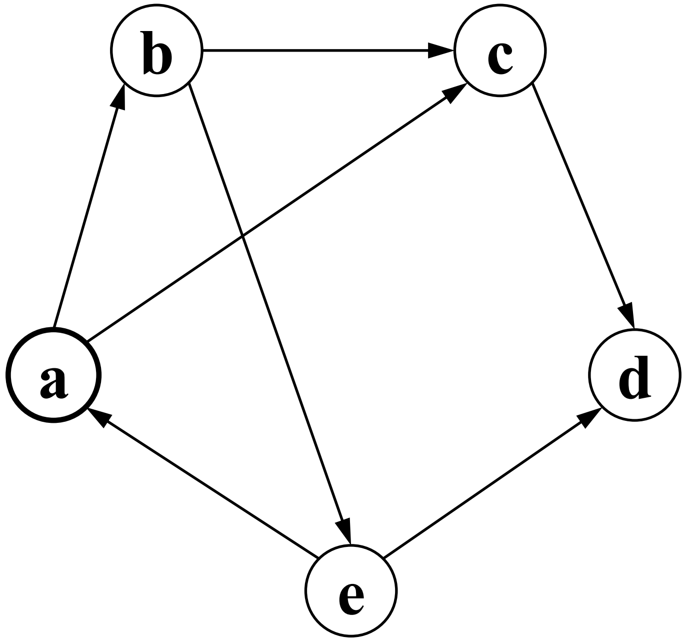
Algorithm 5 Breadth-first visit algorithm.

```
1: function BFSVISIT(node $s$)
2:     $color[s] \leftarrow$ GREY
3:     $d[s] \leftarrow 0$
4:     $Q.insert(s)$
5:     while not $Q.isEmpty()$ do
6:         $u \leftarrow Q.peek()$
7:         for each $v$ adjacent to $u$ do
8:             if $colour[v] =$ WHITE then
9:                 $colour[v] \leftarrow$ GREY; $pred[v] \leftarrow u; d[v] \leftarrow d[u] + 1$
10:                $Q.insert(v)$
12:        $Q.delete()$
13:        $colour[u] \leftarrow$ BLACK
```

A BFS example (1)

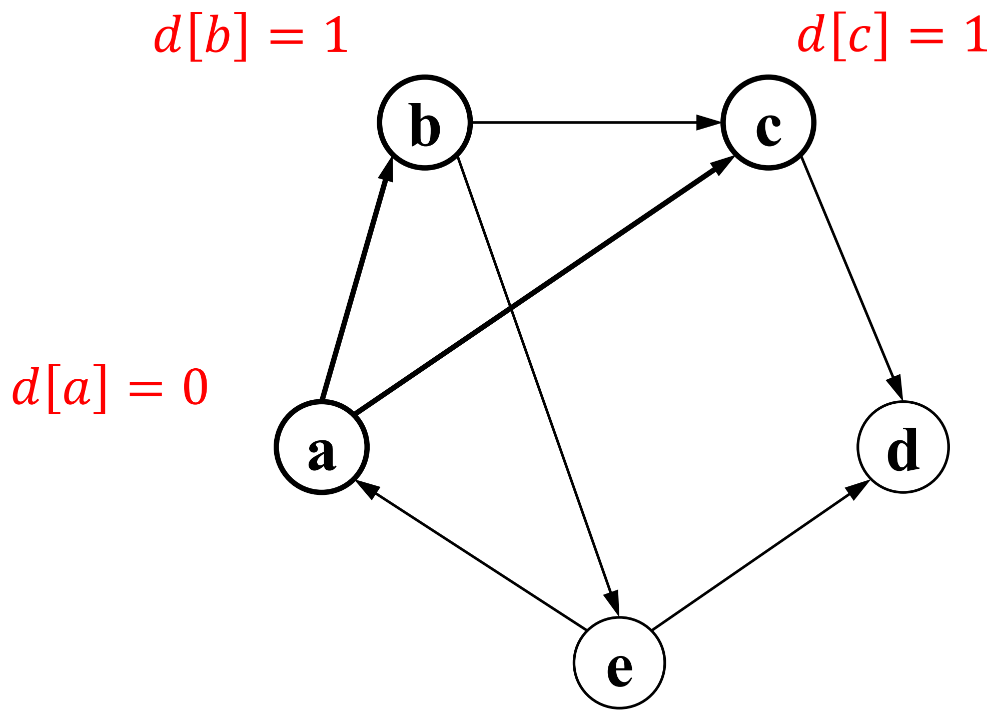
Queue: a

$d[a] = 0$



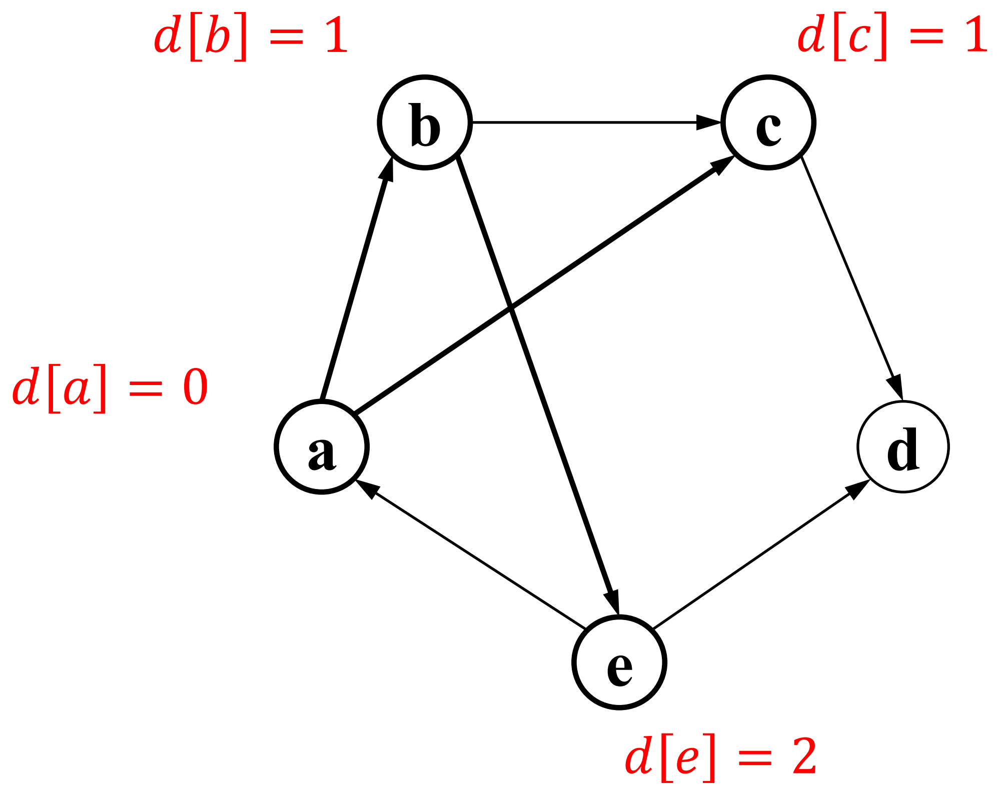
A BFS example (1)

Queue: b c



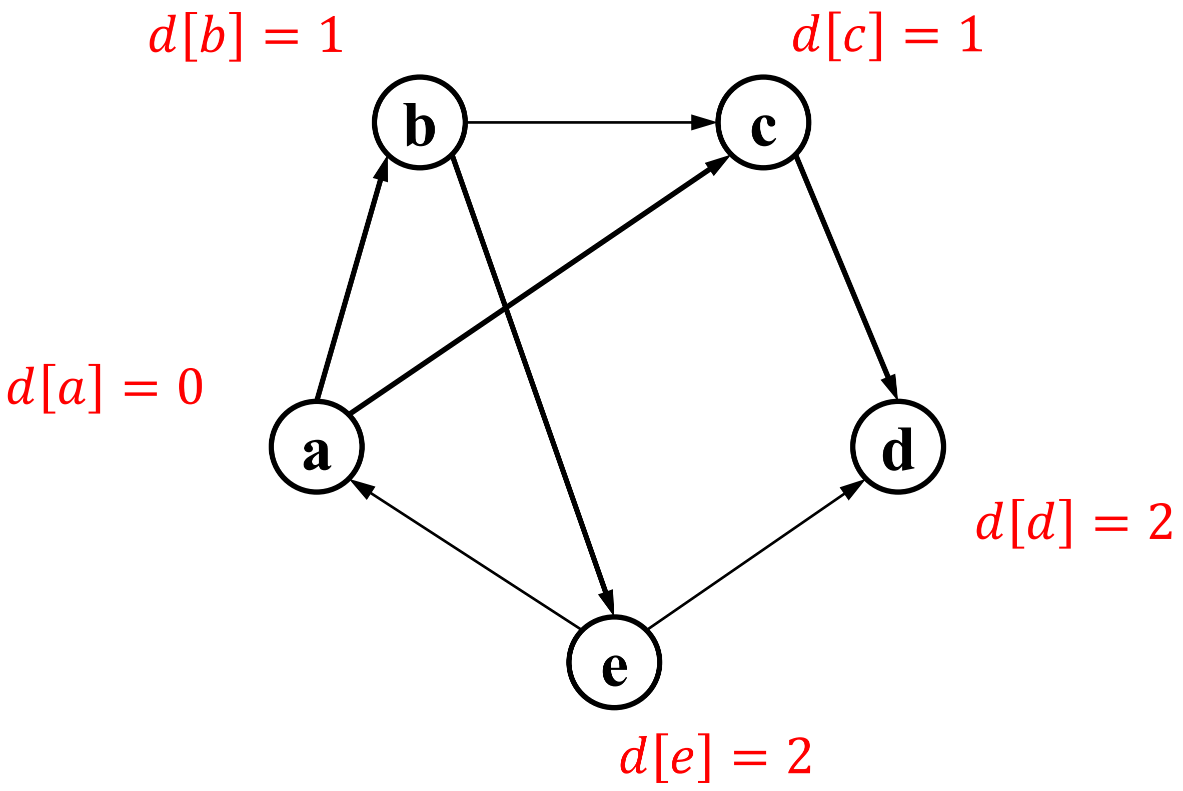
A BFS example (1)

Queue: c e

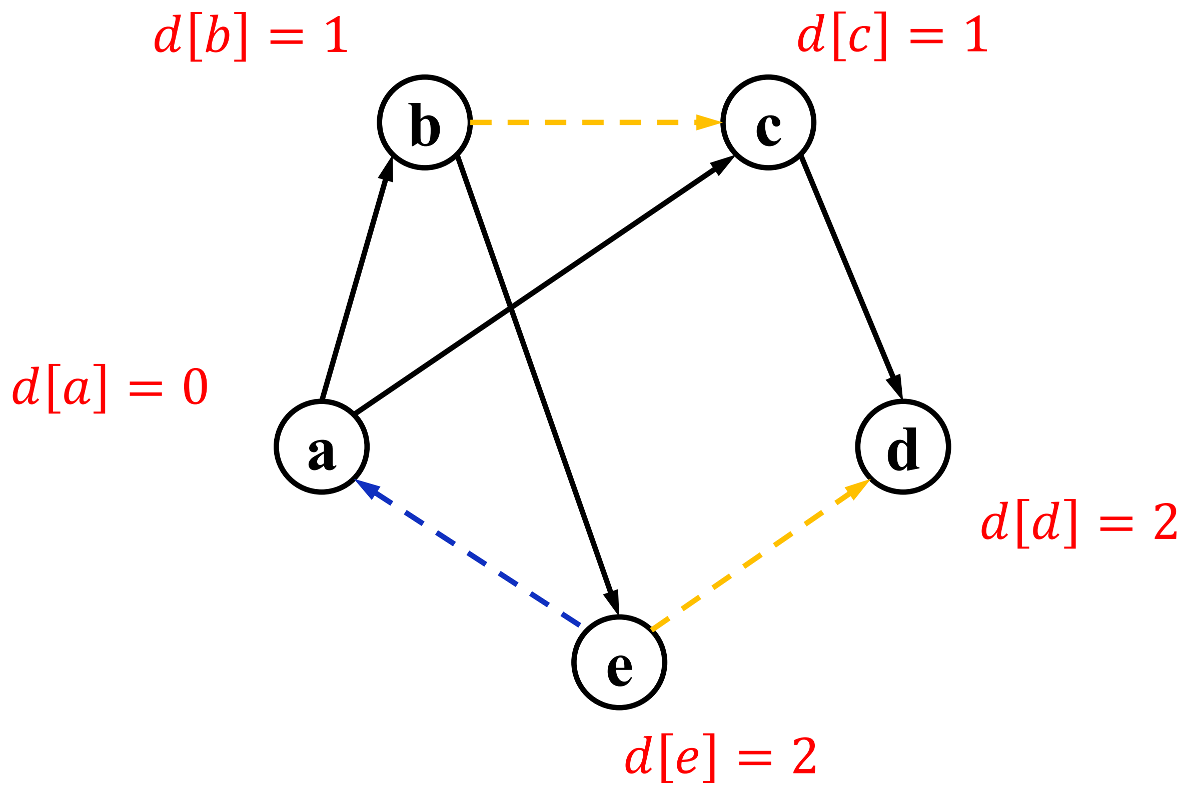


A BFS example (1)

Queue: e d



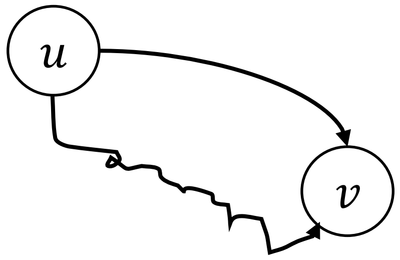
A BFS example (1)



Basic properties of Breadth-first Search

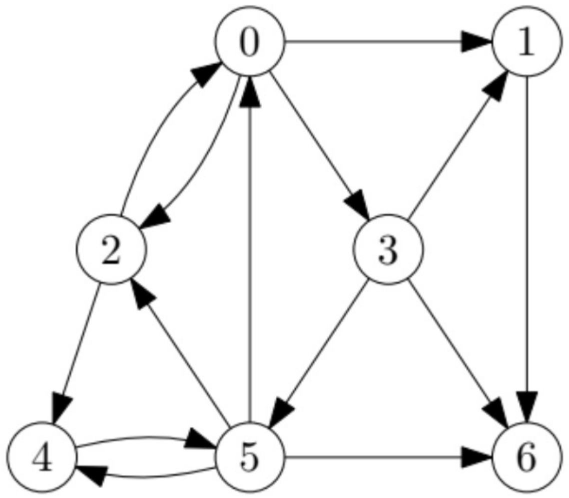
- BFS Property: There are no forward arcs when performing BFS on a digraph. Why?

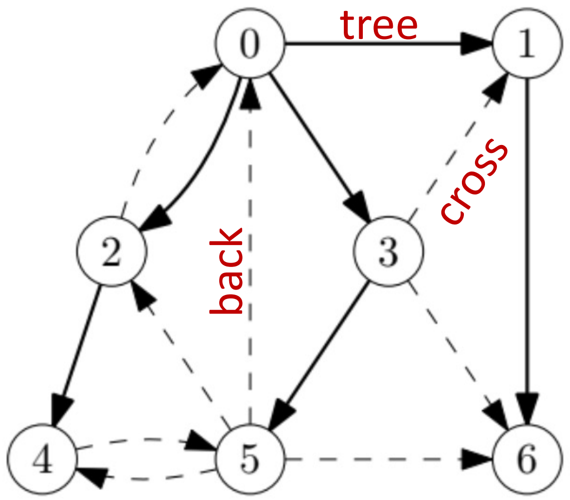
- **Proof**:
- Suppose there exist a forward arc. Then we have the following in the search forest.
- There is a directed path of tree arcs(at least with two arcs) from u to v, and there is an arc (u,v) that is not in any tree of the search forest. But then BFS explores v as a neighbour of u when v is white. So (u,v) is a tree arc, a contradiction.

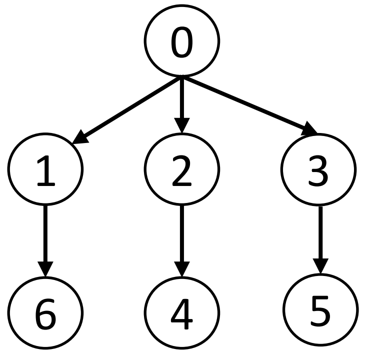


Example 24.2

- A digraph and its BFS search tree, rooted at node 0. The dashed arcs indicate the original arcs that are not part of the BFS search tree.



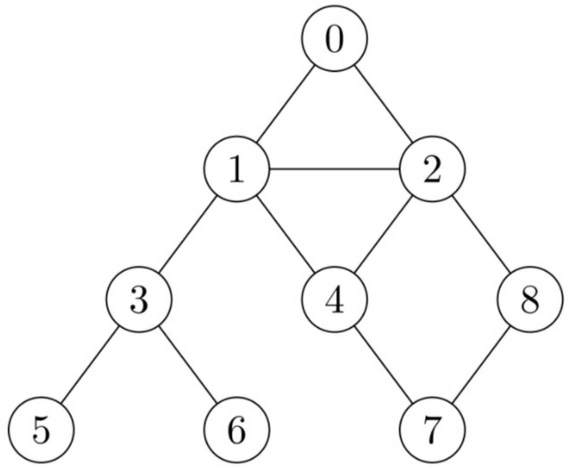


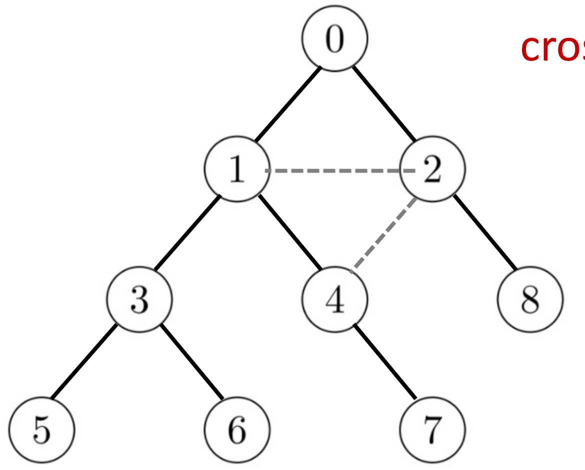


- Find a tree arc, cross arc, forward arc and back arc (or say if that type of arc does not exist for that traversal).

Example 24.3

- Use the nodes on the right to draw the search tree you obtain by running BFS on the graph on the left, starting at vertex 0. Use dashed edges to indicate edges that are not arcs in the search tree.





cross arcs

- Find a tree arc, cross arc, forward arc and back arc (or say if that type of arc does not exist for that traversal).

Forward arcs = backward arcs (no directions), but there are no forward arcs
→ all dotted arcs are cross arcs.

Time Complexity: DFS v.s. BFS

- Both BFS and DFS have same complexity
    - the next grey/white node has constant complexity

- With adjacency matrix: Need to find out-neighbours for n vertices. With $\Theta(n)$ per vertex, this means $\Theta(n^2)$.

- With adjacency list: the sequence for each node is only visited once and we need to visit all the edges. So the complexity is $\Theta(n + m)$.

OUTLINE

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)
- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS

Priority-first Search (PFS)

- In PFS, the next grey vertex is selected according to a priority value
- Typically, this is an integer, such that the grey vertex with the lowest value is selected first
- Priority value is assigned (often computed) no later than when the vertex turns grey

Priority-first-search (PFS) Algorithm

Algorithm 6 Priority-first search algorithm

```
1: function PFS(digraph $G$)
2:     priority queue $Q$
3:     array $colour[0..n-1], pred[0..n-1]$
4:     for $u \in V(G)$ do
5:         $colour[u] \leftarrow$ WHITE
6:         $pred[u] \leftarrow$ null
7:     for $s \in V(G)$ do
8:         if $colour[s] =$ WHITE then
9:             PFSVISIT($s$)
10:    return $pred$
```

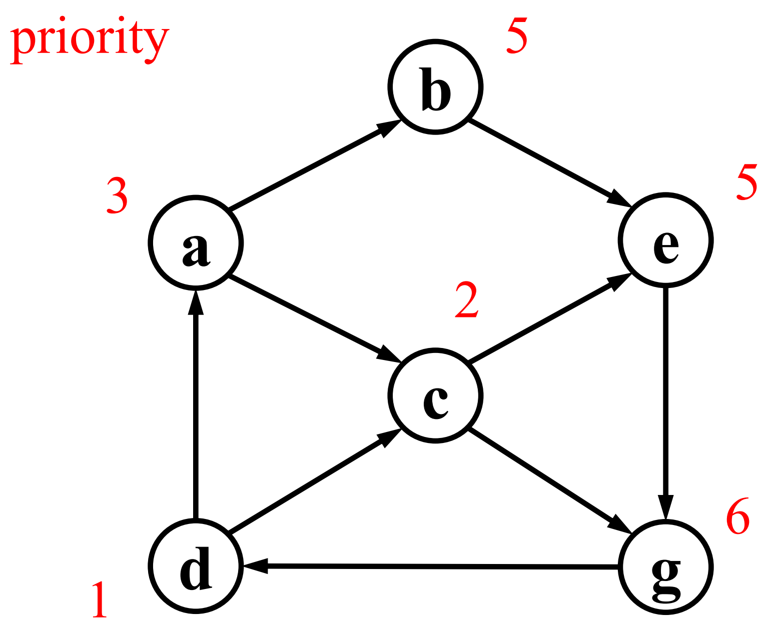
Iterative View of PFSVISIT

Algorithm 7 Priority-first visit algorithm.

```
1: function PFSVISIT(node $s$)
2:     $color[s] \leftarrow$ GREY
3:     $Q.insert(s, setKey(s))$
4:     while not $Q.isEmpty()$ do
5:         $u \leftarrow Q.peek()$
6:         if $u$ has a neighbour $v$ with $colour[v] =$ WHITE then
7:             $colour[v] \leftarrow$ GREY
8:             $Q.insert(v, setKey(v))$
9:         else
10:            $Q.delete()$
11:            $colour[u] \leftarrow$ BLACK
```

A PFS example

- Start at a



(a,3)

(a,3) (b,5)

(a,3) (b,5) (c,2)

(a,3) (b,5) (c,2) (e,5)

(a,3) (b,5) (c,2) (e,5) (g,6)

(a,3) (b,5) (e,5) (g,6)

(b,5) (e,5) (g,6)

(e,5) (g,6)

(g,6)

(g,6) (d,1)

(g,6)

Priority-first Search (PFS)

- In simple PFS, the priority value of a vertex does not change.
- In advanced PFS, the priority value of a vertex may be updated again later.

- BFS and DFS are special cases of simple PFS.
  - In BFS, the priority values are the order in which the vertices turn grey (1, 2, 3, …).
  - In DFS, the priority values are the negative order in which the vertices turn grey (-1, -2, -3, …).

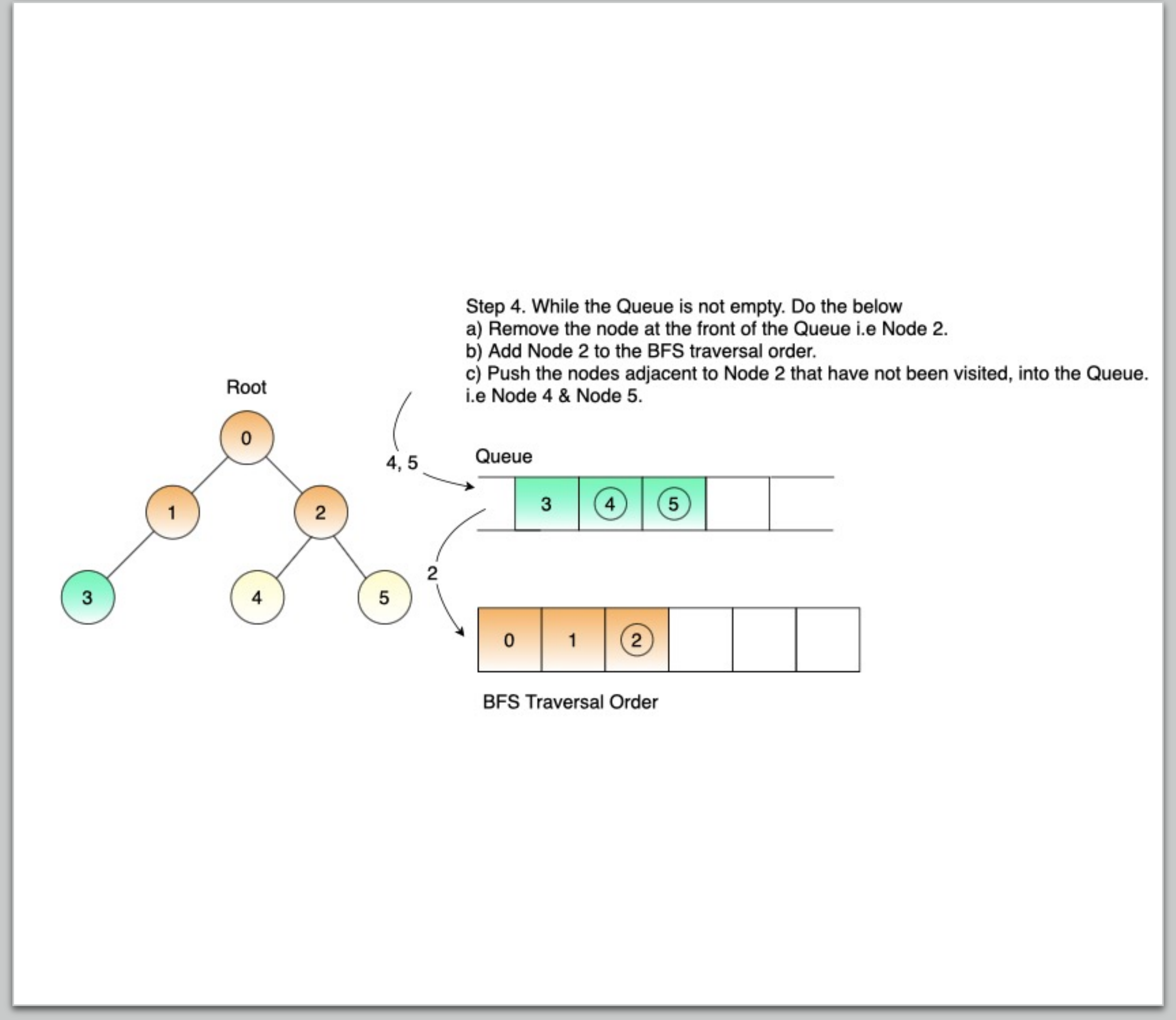
Time Complexity: PFS

- General time complexity is not very good: Need to find minimum priority value each time we need to select the next grey node
- If we search up to n vertices (say, in an array) for the minimum priority value, we need $\Omega(n^2)$
- This can be improved on a little by using a priority queue (e.g., a heap) for the grey vertices, but it's still slower than pure DFS or BFS
- Use PFS when there is an advantage in processing high priority vertices first (e.g., when this allows us to remove other vertices or edges from the (di)graph to be traversed, thereby reducing the search space)

SUMMARY

- Graph Traversal Algorithms
  - Depth-first Search (DFS)
  - Breadth-first Search (BFS)
  - Priority-first Search (PFS)

- Implementation
  - Stack – DFS
  - Queue – BFS
  - Priority Queues – PFS


Step 4. While the Queue is not empty. Do the below
a) Remove the node at the front of the Queue i.e Node 2.
b) Add Node 2 to the BFS traversal order.
c) Push the nodes adjacent to Node 2 that have not been visited, into the Queue. i.e Node 4 & Node 5.
Root
0
1
2
3
4
5
4, 5
Queue
3
4
5
2
0
1
2
BFS Traversal Order

Source: https://www.algotree.org/algorithms/tree_graph_traversal/breadth_first_search/