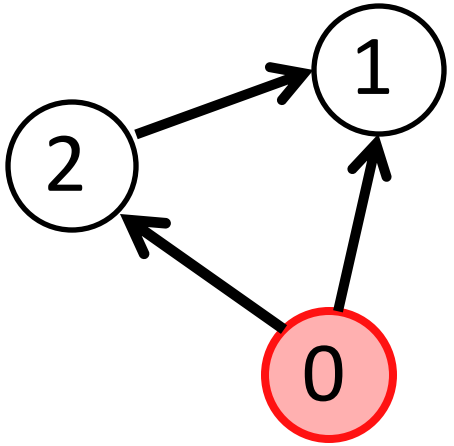
COMPLEXITY ANALYSIS OF OF GRAPH SEARCH

BFS: Running Time Complexity



Algo exploreBFS (Graph G, vertex s):

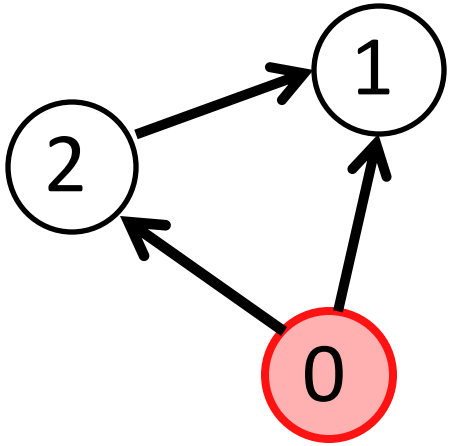
- Mark all the vertices as “not visited”
- Mark ***s*** as visited
- push ***s*** into a queue
- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***’s neighbor (***v***)
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited
      - Push ***v*** in the queue

n: number of vertices
m: number of edges

How many times does the while loop run?

A. n
B. m
C. n + m
D. nm
E. None of the above

BFS: Running Time Complexity



Algo exploreBFS (Graph G, vertex s):

For each iteration of the while loop, the for loop runs a variable number of times. How should we proceed to analyze the Big-O running time?

- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***'s neighbor (***v***):
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited
      - Push ***v*** in the queue

A. Bound the maximum number of times the for loop runs **per iteration** of the while loop

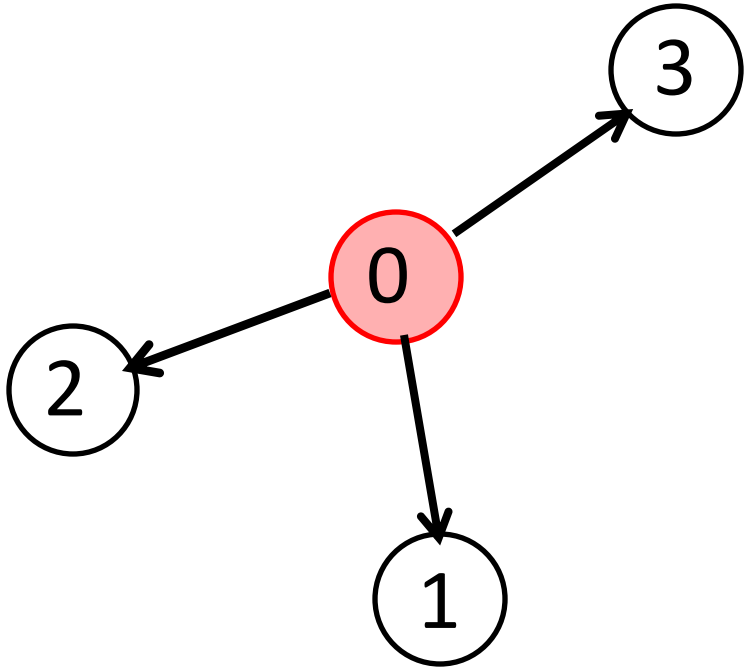
B. Compute the total number of times the for loop runs over **the entire run of exploreBFS**

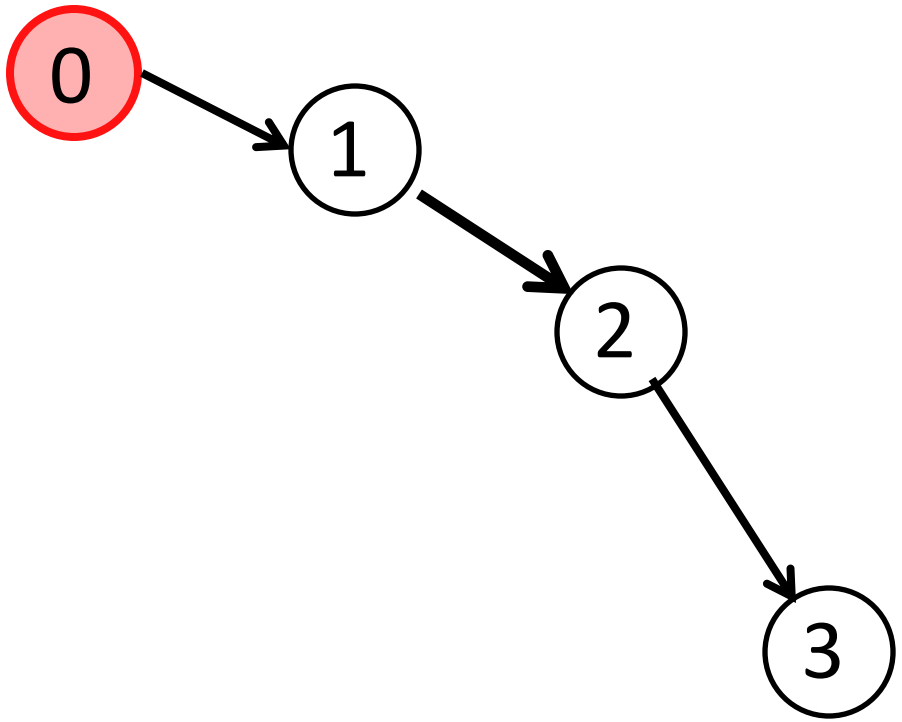
C. Cannot compute Big-O because running time depends on two parameters (n, m)

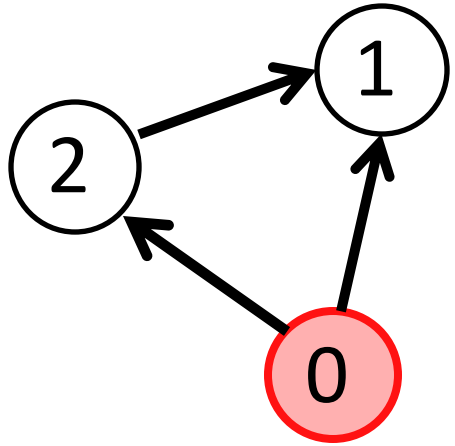
BFS: Running Time Complexity

Total number of times the for loop runs over **the entire run of exploreBFS**

Total number of times each neighbor (u) is checked over **the entire run of exploreBFS**







BFS: Time Complexity

Algo exploreBFS (Graph G, vertex s):

- Mark all the vertices as “not visited”
- Mark ***s*** as visited
- push ***s*** into a queue
- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***’s neighbor (***v***)
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited
      - Push ***v*** in the queue

n: number of vertices
m: number of edges

What is the time complexity of exploreBFS?

A. O(n)
B. O(m)
C. O(n + m)
D. O(nm)
E. None of the above

BFS Traverse: Space Complexity

Algo exploreBFS (Graph G, vertex s):

- Mark all the vertices as “not visited”
- Mark ***s*** as visited
- push ***s*** into a queue
- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***’s neighbor (***v***)
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited
      - Push ***v*** in the queue

n: number of vertices
m: number of edges

What is the Big -O auxiliary space complexity of exploreBFS?

A. O(n)
B. O(m)
C. O(n + m)
D. O(n^2)
E. None of the above

- Auxiliary Space complexity: Additional space usage (not including input and output)

exploreDFS: Time Complexity

```
exploreDFS(v)

  v.visited ← true

  For each edge (v, w):

    If not w.visited

      exploreDFS(w)
```

n: number of vertices
m: number of edges

What is the time complexity of exploreDFS?

A. O(n)
B. O(m)
C. O(n + m)
D. O(n^2)
E. None of the above

exploreDFS: Space Complexity

```
exploreDFS(v)

  v.visited ← true

  For each edge (v,w)

    If not w.visited

      exploreDFS(w)
```

n: number of vertices
m: number of edges

What is the worst-case space complexity of exploreDFS?

A. O(n)
B. O(m)
C. O(n + m)
D. O(n^2+ n.m)
E. None of the above