COMPLEXITY ANALYSIS REVISITED

Analyze the running time of mystery

```
void mystery(int n) {
    for (int i = 0; i < n; i++){
        cout << i; // statement A
        for (int j = i; j < n; j++){
            cout << j; // statement B
        }
    }
}
```

Which statement dominates the running time of mystery?

General Insight

```
void mystery(int n) {
    for (int i = 0; i < n; i++){
        cout << i; // A: Outer loop operation
        for (int j = i; j < n; j++){
            cout << j; // B: Inner loop operation
        }
    }
}
```

When two operations have the same complexity, the one that runs more times dominates.

So total cost = count total executions of the dominant operation.

BFS: Running Time Complexity

Algo exploreBFS (Graph G, vertex s):

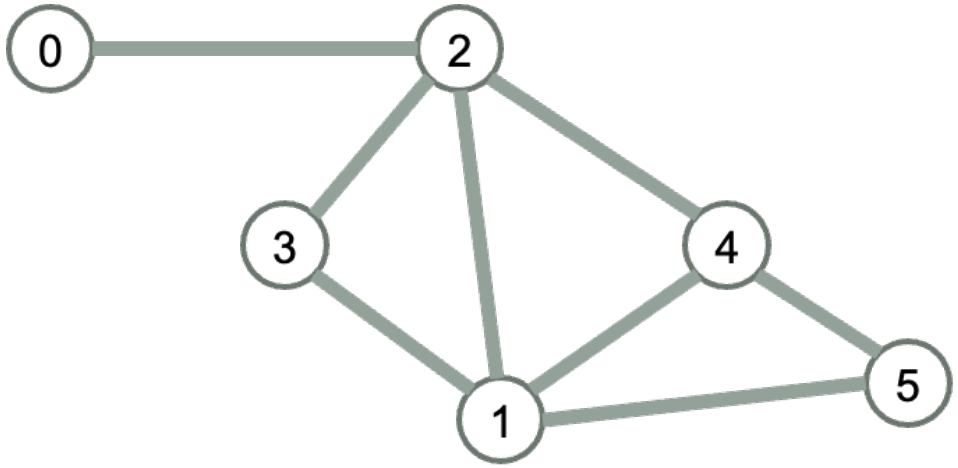
- Mark all the vertices as “not visited”
- Mark ***s*** as visited
- push ***s*** into a queue
- while the queue is not empty:
  - pop the vertex ***u*** from the front of the queue
  - for each of ***u***’s neighbor (***v***)
    - If ***v*** has not yet been visited:
      - Mark ***v*** as visited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

Which statement dominates the running time of exploreBFS?

Total cost of while loop = Total executions of neighbor check

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



BFS: Space Complexity

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

n: number of vertices
m: number of edges

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

exploreDFS(G, s): Time Complexity

```
exploreDFS(v, visited)
  visited[v] = true
  For each edge (v,w):
    If not w.visited
      exploreDFS(w)

exploreDFS(Graph G, vertex s):
      Mark all vertices as "not visited"
      exploreDFS(s, visited)
```

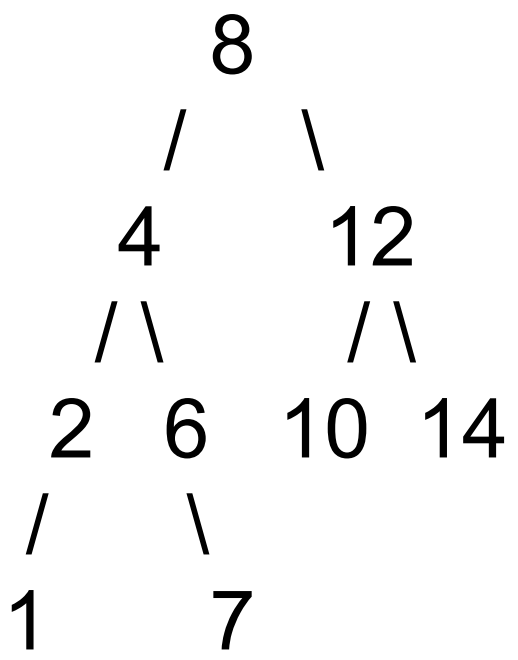
exploreDFS: Space Complexity

```
exploreDFS(v, visited)
  visited[v] = true
  For each edge (v,w):
    If not w.visited
      exploreDFS(w)

exploreDFS(Graph G, vertex s):
      Mark all vertices as "not visited"
      exploreDFS(s, visited)
```

What is the output of this code for the given BST?

```
void printSetValues(const std::set<int>& s){
    for (int value : s) {
        std::cout << value << " ";
    }
}
```



What is the running time complexity for a set with n keys?

```
void printSetValues(const std::set<int>& s){
    for (int value : s) {
        std::cout << value << " ";
    }
}
```

A. O(1) B.O(log n) C. O(n) D. O(nlogn)