Concept Challenge: DFS and BFS

Concept Challenge: Procedure

- **Pause** Try to solve the problem yourself
- **Discuss** with other learners (if you can)
- **Watch** the UC San Diego learners video
- **Answer** the question again
- **Confirm** your understanding with our explanation

```
DFS(S, G):
    Initialization of structures
    Push S on stack and add to visited
    while stack is not empty:
        pop node curr from top of stack
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            push n to top of stack
    // If we get here then there's no path
```

```
BFS(S, G):
    Initialization of structures
    Enqueue S in queue and add to visited
    while queue is not empty:
        dequeue node curr from front of queue
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            enqueue n to back of queue
    // If we get here then there's no path
```

Which algorithm is asymptotically faster in the worst case?

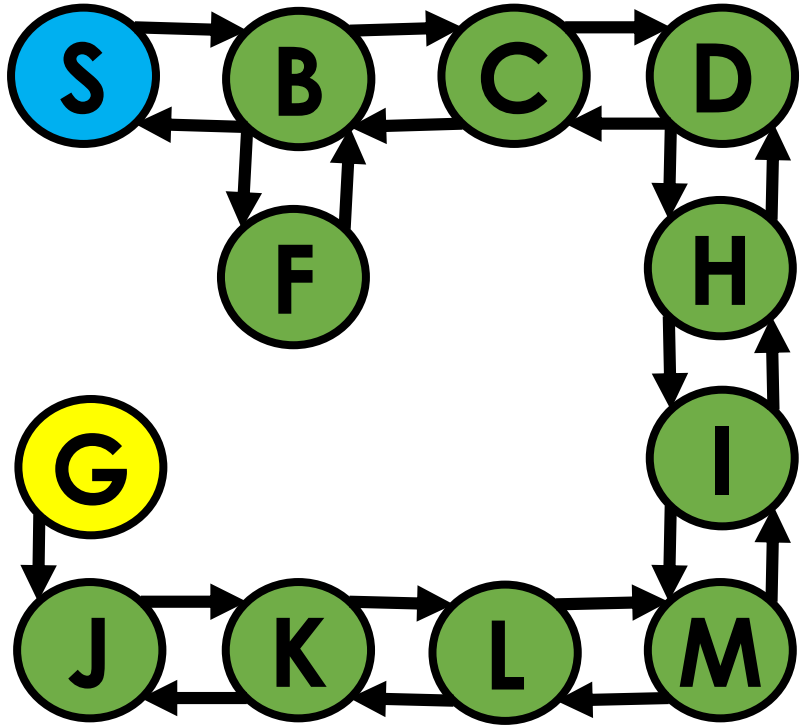
```
DFS(S, G):
    Initialization of structures
    Push S on stack and add to visited
    while stack is not empty:
        pop node curr from top of stack
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            push n to top of stack
    // If we get here then there's no path
```

```
BFS(S, G):
    Initialization of structures
    Enqueue S in queue and add to visited
    while queue is not empty:
        dequeue node curr from front of queue
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            enqueue n to back of queue
    // If we get here then there's no path
```

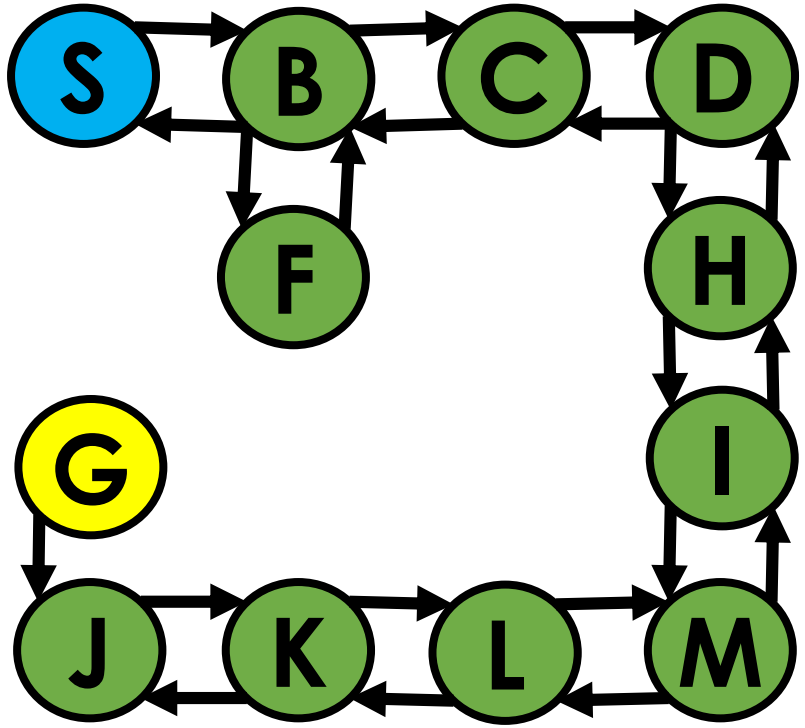
Which algorithm is asymptotically faster in the worst case?

A. BFS is faster
B. DFS is faster
C. They are both the same

Give tght big O bounds for each algorithm.

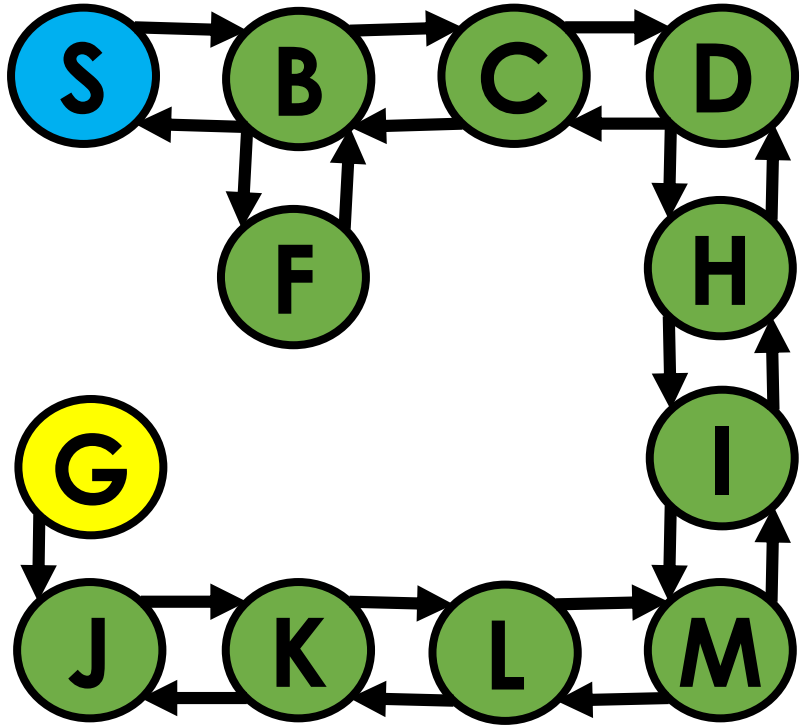


What does the worst case look like?



How many nodes will each algorithm visit?

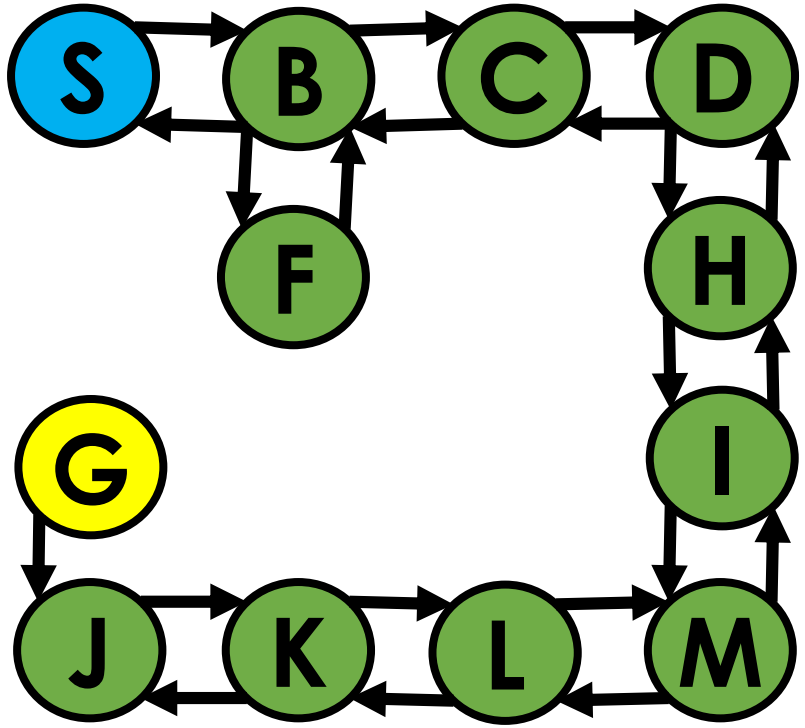
```
Search(S, G):
    Initialization of structures
    Add S to [stack/queue] and add to visited
    while [stack/queue] is not empty:
        remove node curr
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            add n to [stack/queue]
```



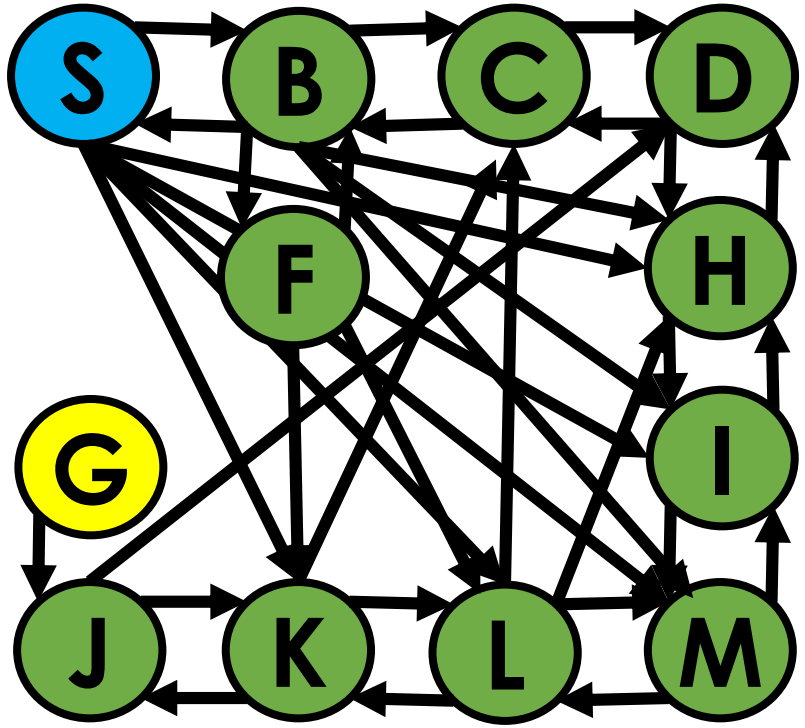
How many nodes will each algorithm visit?

$$|V| - 1 = O(|V|)$$

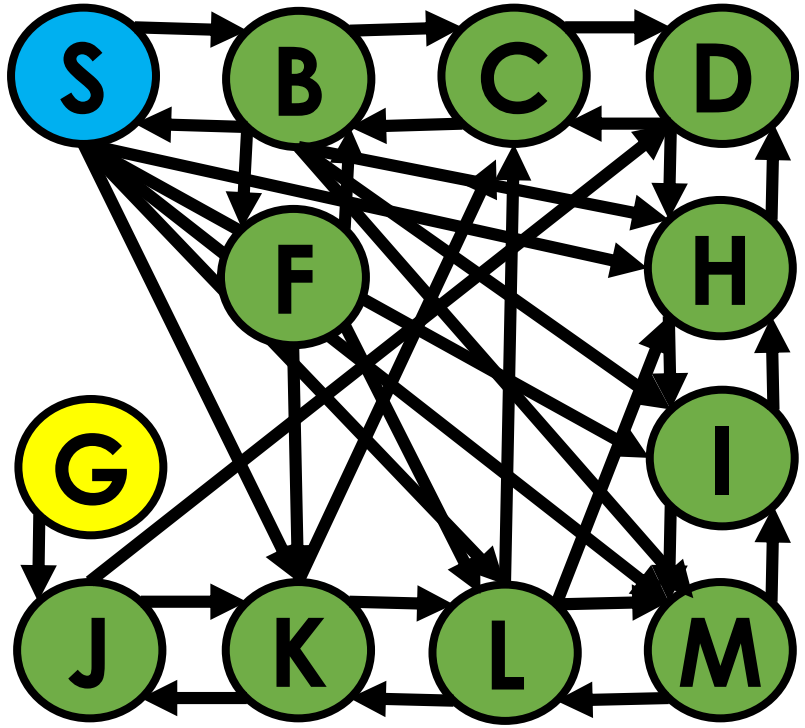
So is that it?




```
Search(S, G):
    Initialization of structures
    Add S to [stack/queue] and add to visited
    while [stack/queue] is not empty:
        remove node curr
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            add n to [stack/queue]
```




```
Search(S, G):
    Initialization of structures
    Add S to [stack/queue] and add to visited
    while [stack/queue] is not empty:
        remove node curr
        if curr == G return parent map
        for each of curr's unvisited neighbors, n:
            add n to visited set
            add curr as n's parent in parent map
            add n to [stack/queue]
```

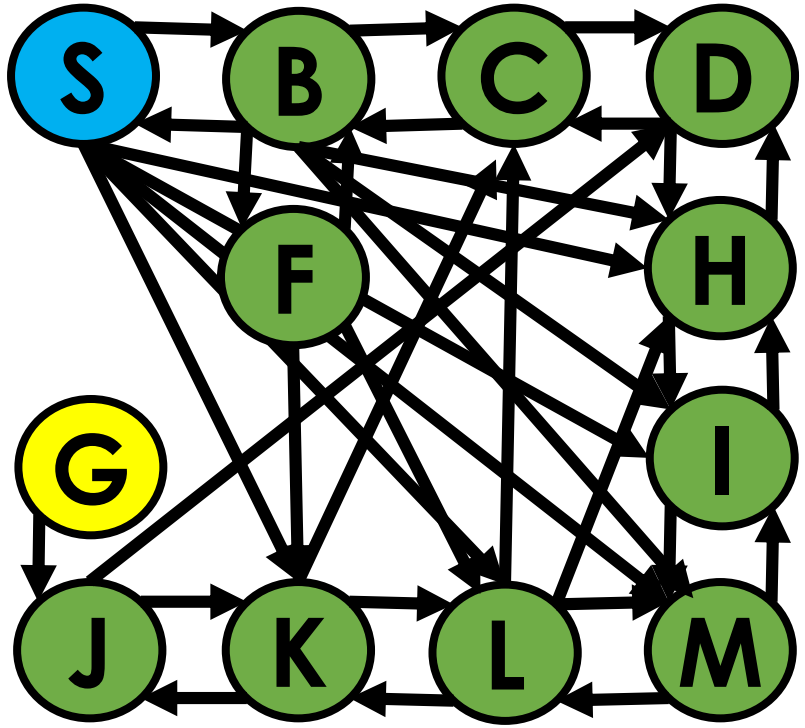


How many nodes will each algorithm visit? O(|V|)

But DFS and BFS will traverse each edge too!

How many edges can there be in a graph?

|E| can be up to $|V|^2$



How many nodes will each algorithm visit? O(|V|)

How many edges will each algorithm traverse? O(|E|)

Worst case running time for both → O(|V| + |E|)