CSE 250 Recitation

March 27 - 28: Stacks, Queues, Graph Traversals

Stacks vs Queues

Exercise:
What does the following code print when MysterySequence is a Stack? Queue?

What are the relevant operations for each?

What are their runtimes for different backing data structures?

```
MysterySequence seq = new MysterySequence()
seq.addSomething("A")
seq.addSomething("B")
seq.addSomething("C")
seq.addSomething("D")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
seq.addSomething("E")
print(seq.removeSomething())
seq.addSomething("F")
print(seq.removeSomething())
seq.addSomething("G")
seq.addSomething("H")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
```

Stacks vs Queues

Stack (LIFO)

==========

Prints: DCBEFHGA

Operations: push, pop

Runtimes: O(1)

Queue (FIFO)

==========

Prints: ABCDEFGH

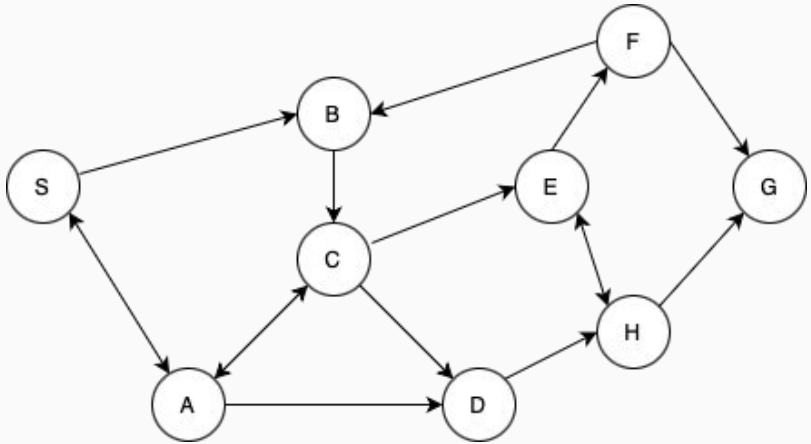
Ops: enqueue,dequeue

Runtimes: O(1) *

**** amortized O(1) for enqueue with array-based implementation)***

```
MysterySequence seq = new MysterySequence()
seq.addSomething("A")
seq.addSomething("B")
seq.addSomething("C")
seq.addSomething("D")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
seq.addSomething("E")
print(seq.removeSomething())
seq.addSomething("F")
print(seq.removeSomething())
seq.addSomething("G")
seq.addSomething("H")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
```

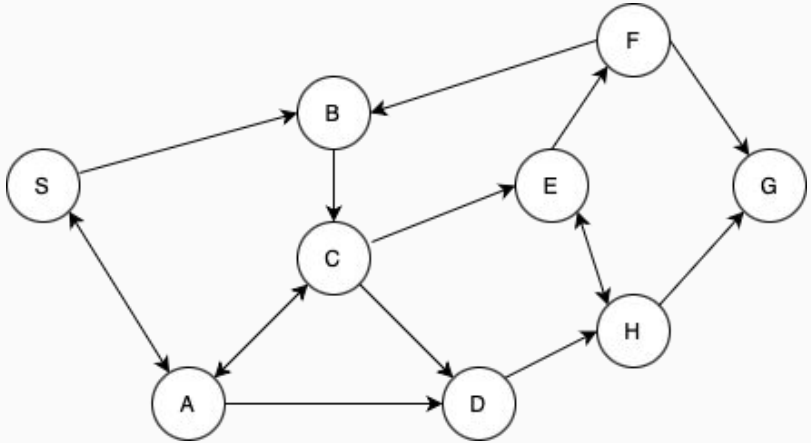
Graph Traversal - DFS



1. Insert the starting node into the [TODO]
2. While the [TODO] is not empty:
   a. Remove a node from the [TODO]
   b. For each of that nodes unvisited neighbors:
      i. Mark the neighbor as visited
      ii. Add it to our [TODO]

Example:
How does DFS starting at **S** progress?
How does the stack change at each step?
What does our edgeTo map look like at each step?
After the traversal, how can we find a path from **S** to **H**?

Graph Traversal Exercise - BFS

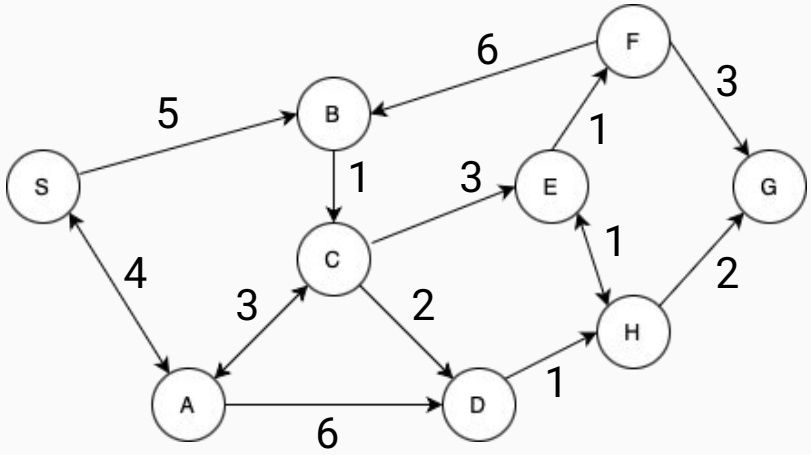


1. Insert the starting node into the [TODO]
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. For each of that nodes unvisited neighbors:
        i. Mark the neighbor as visited
        ii. Add it to our [TODO]

Exercise:
Perform on your paper the BFS traversal of this graph.

- Write out what the [TODO] looks like at each step.
- Write out what the `edgeTo` map looks like.
- Construct a path from S to H from the `edgeTo` map

Graph Traversal Exercise - Djikstra's



1. Insert the starting node into the [TODO]
2. While the [TODO] is not empty:
    a. Remove a node from the [TODO]
    b. If it not VISITED
        i. Mark it as visited
        ii. Add its unvisited neighbors to [TODO]

Exercise:
Perform on your paper the Djikstra's traversal of this graph.

- Write out what the [TODO] looks like at each step.
- Write out what the `edgeTo` map looks like.
- Construct a path from S to H from the `edgeTo` map
- What would happen if you mark a vertex as VISITED when you add it to the [TODO] instead (like BFS did)?

Wrap Up Discussion

Complexity of rebuilding a path from an edgeTo map (assuming O(1) lookup time)?

Complexity of BFS?

Complexity of BFS and then rebuilding a path?

Wrap-Up Discussion

Complexity of rebuilding a path from an edgeTo map (assuming O(1) lookup time)? O(n)

Complexity of BFS? O(n + m)

Complexity of BFS and then rebuilding a path? O(n + m)