CSE 250 Recitation

October 20 - 21: Stacks/Queues, Graph Representations, PA2

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) *

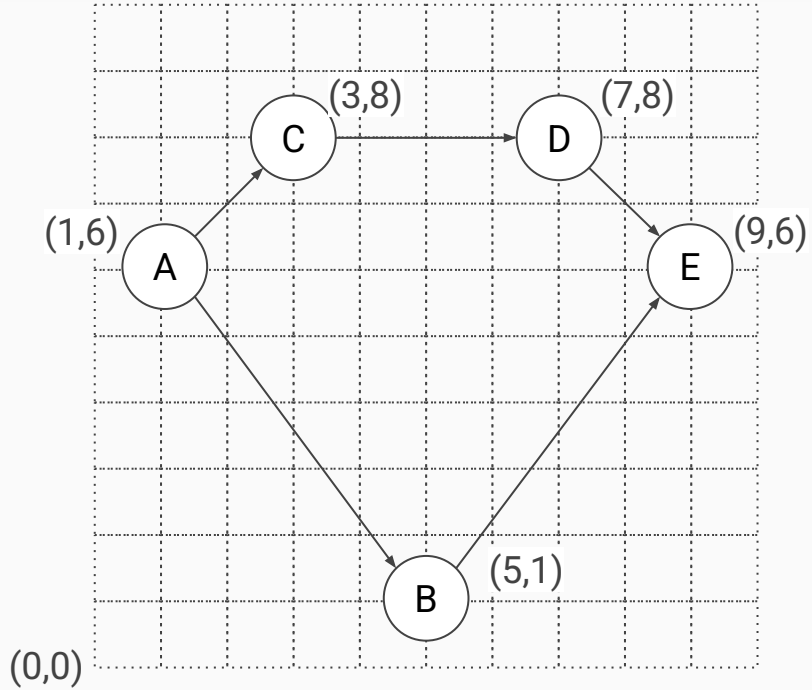
**** amortized O(1) for enqueue with array-based implementation)***

```
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 Representation

Exercise:

Write out an edge list implementation of this graph. Do the same for an adjacency list and adjacency matrix.



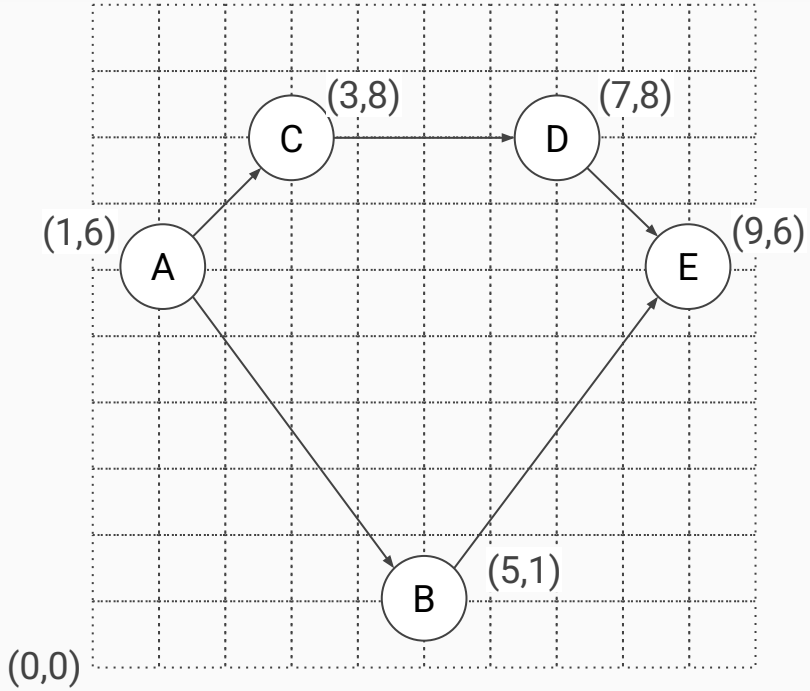
Graph Representation

Exercise:

Edge List:

A → B → C → D → E

(A,C) → (C,D) → (D,E) → (A,B) → (B,E)

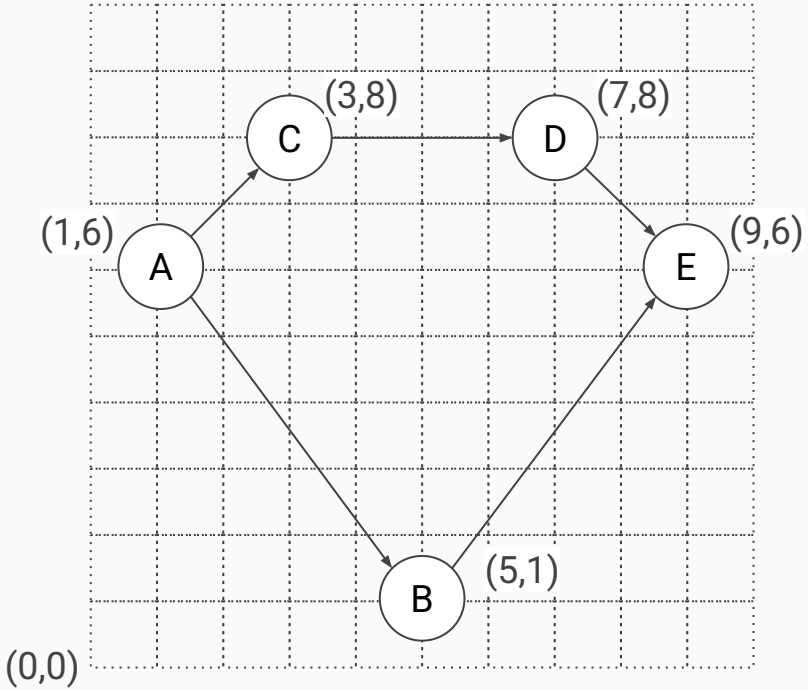


Graph Representation

Exercise:

Adjacency List

A: (A,C) → (A,B)
B: (B,E)
C: (C,D)
D: (D,E)

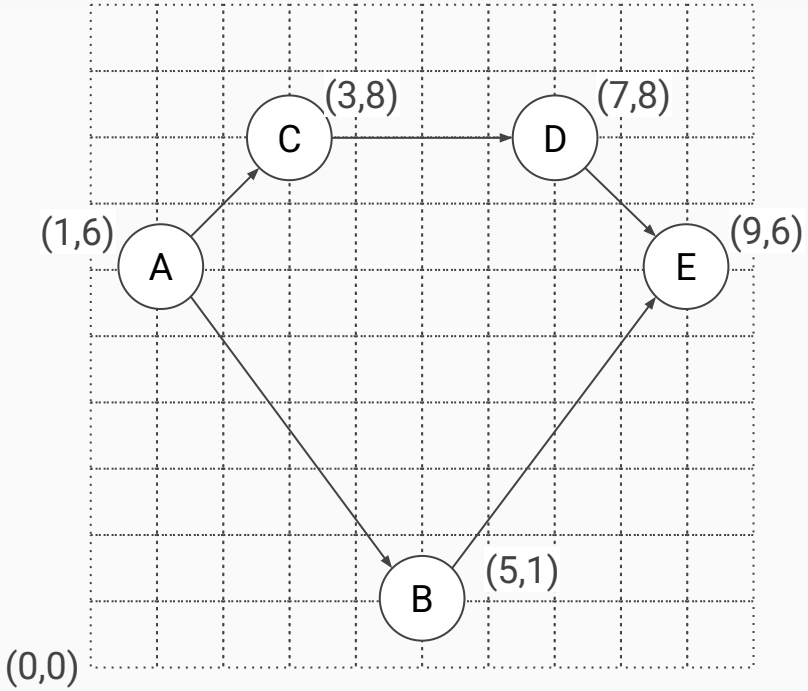


Graph Representation

Exercise:

Adjacency Matrix

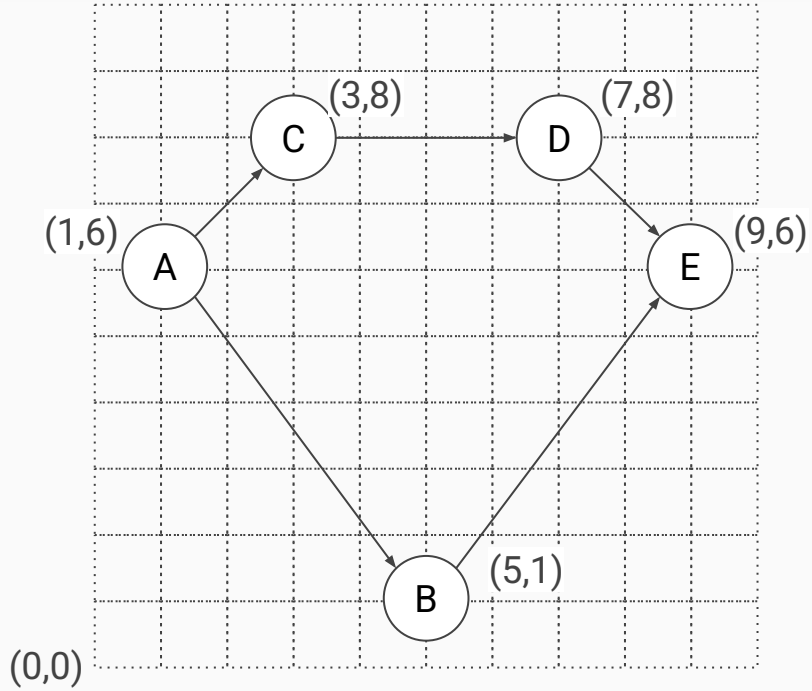
	A	B	C	D	E
A		(A,B)	(A,C)		
B					(B,E)
C				(C,D)	
D					(D,E)
E					



Traversal Discussion

Consider finding a path from A to E:

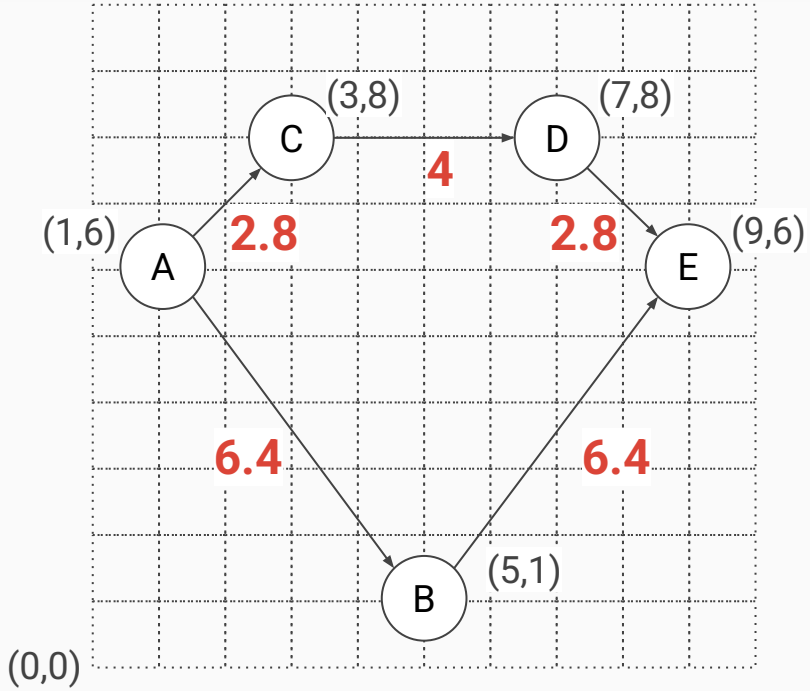
1. How many unique paths are there?
2. Which has the fewest edges?
   a. Which algo finds this path?
3. Which has the shortest distance?
   a. Which algo finds this path?
4. Which path would DFS find?



Traversal Discussion

Consider finding a path from A to E:

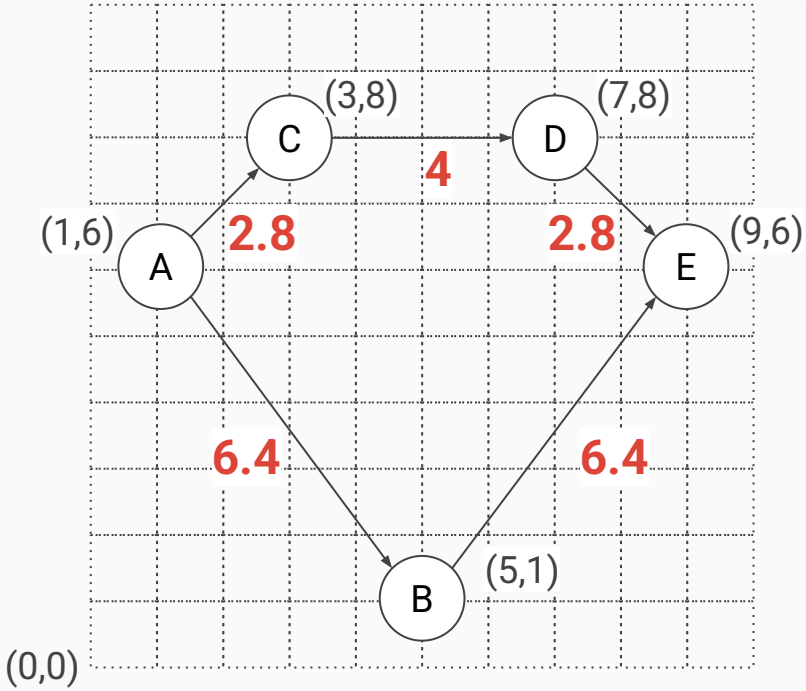
1. How many unique paths are there?
2. Which has the fewest edges?
   a. Which algo finds this path?
3. Which has the shortest distance?
   a. Which algo finds this path?
4. Which path would DFS find?



Traversal Discussion

Consider finding a path from A to E:

1. How many unique paths are there? **2**
2. Which has the fewest edges? **ABE**
   a. Which algo finds this path? **BFS**
3. Which has the shortest distance? **ACDE**
   a. Which algo finds this path? **Dijkstra's**
4. Which path would DFS find? **Either**



PA2: Testing Exercise

1. Draw out a graph (on a grid) containing *at least* 8 labeled vertices.
2. Label one of the vertices the starting vertex, and one the ending vertex.
3. Trade papers with a neighbor, and answer the following about their graph:
    a. What is the path from start to finish with the fewest edges? Is it a unique path?
    b. What is the path from start to finish of the shortest distance? Is it a unique path?
    c. Are your answer to a and b the same? Why would it be problematic if they are?
    d. Are the paths in a and b the **only** paths? Why would it be problematic if they are?
    e. Name features that might show up in a map of Buffalo that are not present in this graph.
    f. How would you change the graph to include some of these features?

PA2: Testing Discussion

Did you need to know how to implement BFS/Djikstra's to determine which paths should be found in your neighbors graph? **No!**

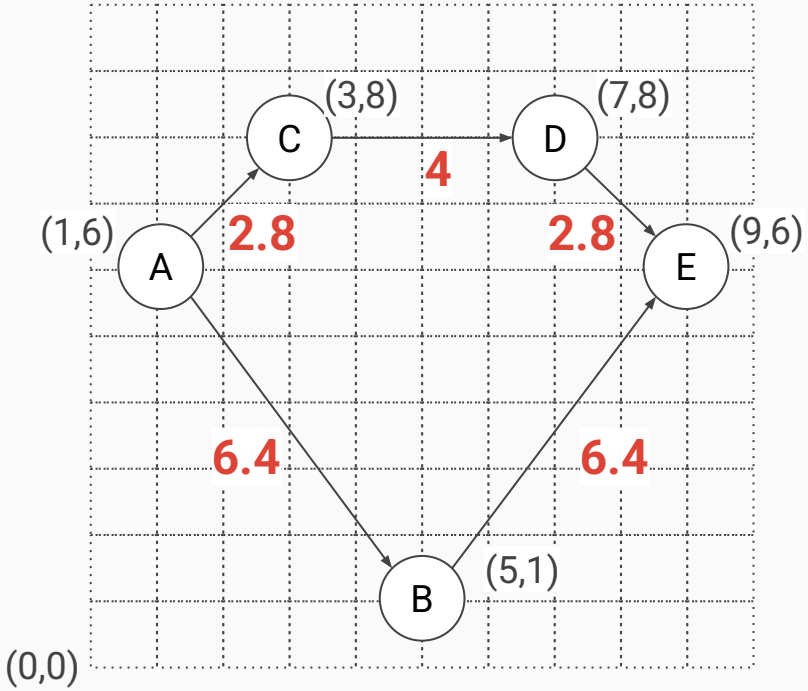
It's ok to have some very simple tests…but you should have complex ones too:

- If there's only one path…even a buggy implementation will probably find it
- If both BFS and Djikstra's return the same path it may hide some bugs
- Buffalo streets have: cycles, two way streets, one way streets, dead ends
    - Could these features expose bugs in your search? Your tests should have these features!

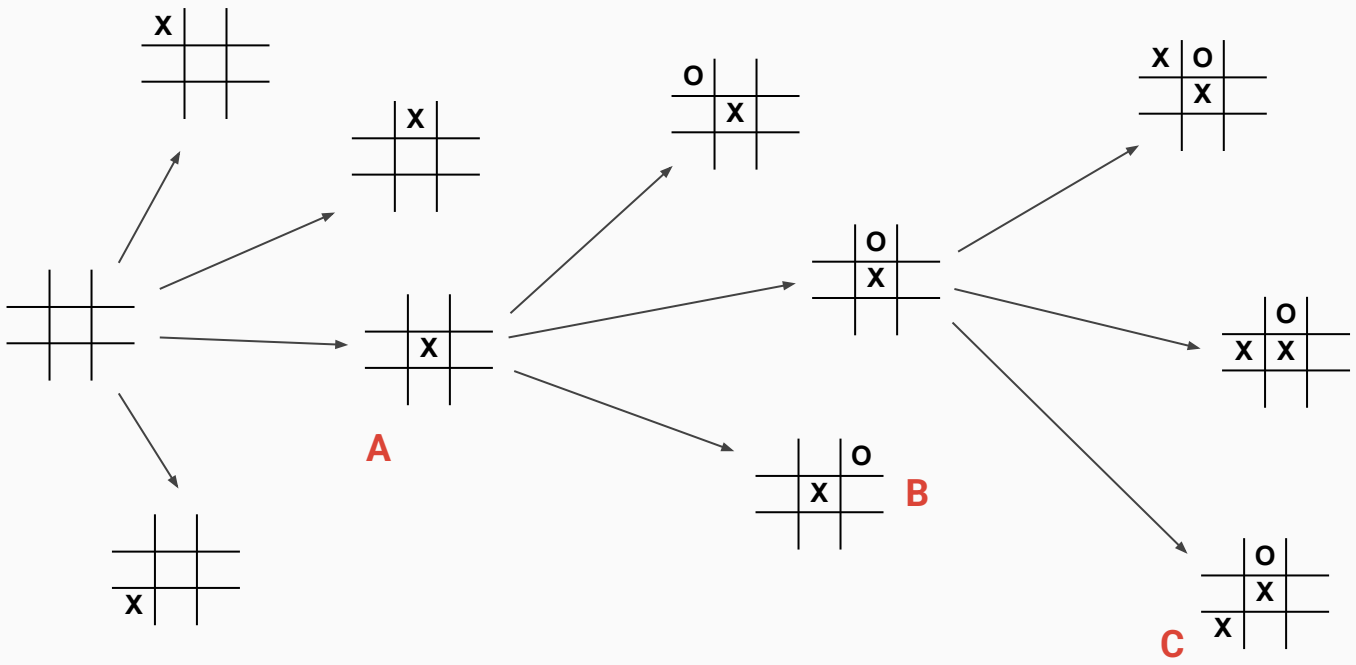
Traversal Discussion Revisited

Based on the previous exercise:

1. What characteristics of this graph make it a good test?
2. What characteristics does it lack?
3. What could you add/change/include in a different test to improve your tests?



Bonus Content: Tic Tac Toe Example



Note: This does not show all edges / vertices…

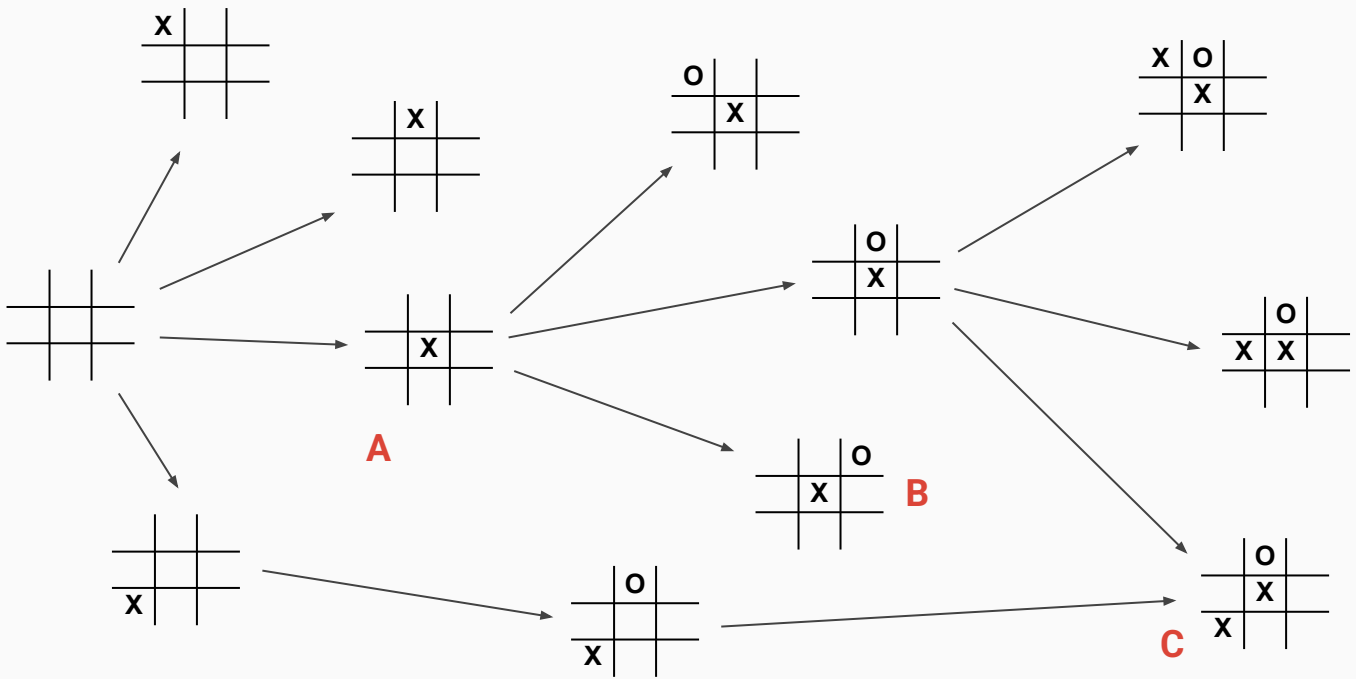
What is the out degree of the vertex for the empty board? What about the in degree?

What is the out degree of the vertex labeled A? B?

How many edges are in the full graph?

Is the in degree of every non-starting node 1?

Bonus Content: Tic Tac Toe Example



Note: This does not show all edges / vertices…

What is the out degree of the vertex for the empty board? 9
What about the in degree? 0

What is the out degree of the vertex labeled A? B? 8, 7

How many edges are in the full graph? ***9!***

Is the in degree of every non-starting node 1? No ie C