Exam 1 → 6pm due Sunday @ 8am
- no clarifications
- autograder → must compile
- video → (mp4) → must run/play in gradescope

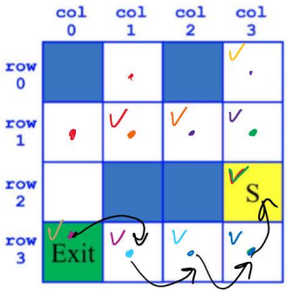
PA3 released

**Breadth-First Search (BFS)** Queue

Guaranteed shortest path

```
class Square {
  boolean visited;
  Square previous;
}
```

class Node {
  boolean visited
  Next next
}




```
SearchForTheExit

Initialize a Queue to hold Squares as we search
Mark starting square as visited
Enqueue starting square on Queue
While Queue is not empty
    Dequeue square sq from Queue
    Mark sq as visited
    If sq is the Exit, we're done!
    For each of square's unvisited neighbors (S, W, N, E):
        Set neighbor's previous to sq
        Enqueue neighbor to Queue
```

path?

Run through the SearchForTheExit algorithm. Draw the queue.

front → (2,3) (3,3) (1,3) (3,2) (1,2) (0,3) (3,1) (1,1) (3,0) (1,0) (0,1)

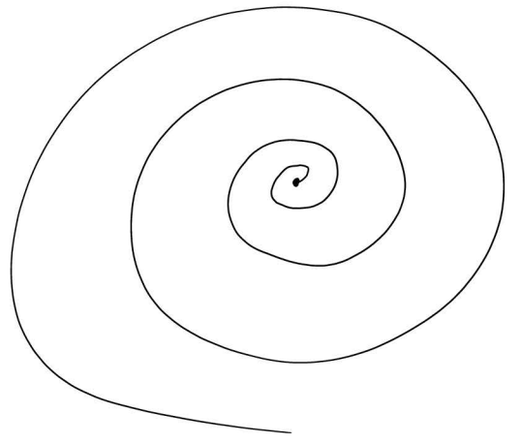
(3,0) → (3,1) → (3,2) → (3,3) → (2,3)

exit start

How many nodes were visited? 9

How many total squares were added to the queue? 11

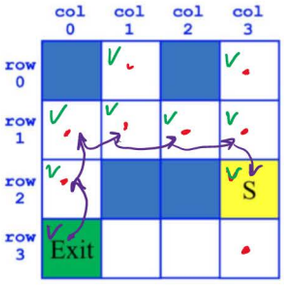
Was this the shortest path? true



**Depth-First Search (DFS)** stack

Will always find a path. Possibly faster.

```
class Square {
  boolean visited;
  Square previous;
}
```




```
SearchForTheExit

Initialize a Stack to hold Squares as we search
Mark starting square as visited
Push starting square on Stack
While Stack is not empty
    Pop square sq from Stack
    Mark sq as visited
    If sq is the Exit, we're done!
    For each of square's unvisited neighbors (S, W, N, E):
        Set neighbor's previous to sq
        Push neighbor to Stack
```

Run through the SearchForTheExit algorithm. Draw the stack.

top → (2,3)
→ (1,3) (3,3)
→ (0,3) (1,2) (3,3)
→ (1,2) (3,3)
→ (1,1) (3,3)
→ (0,1) (1,0) (3,3)
→ (2,0) (3,3)
→ (3,0) (3,3)

How many nodes were visited? 9

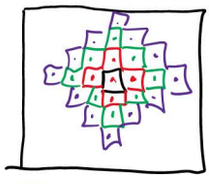
How many total squares were added to the stack? 10

Was this the shortest path? NOpe

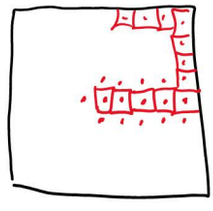
Exit (3,0)
↓ (2,0)
↓ (1,0)
↓ (1,1)
↓ (1,2)
↓ (1,3)
↓ start (2,3)

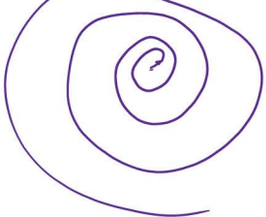
S W N E

BFS



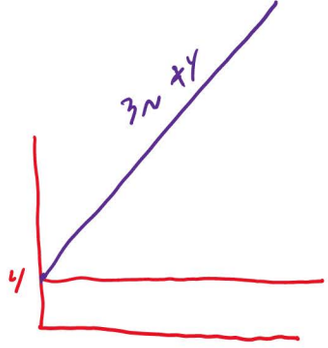
DFS

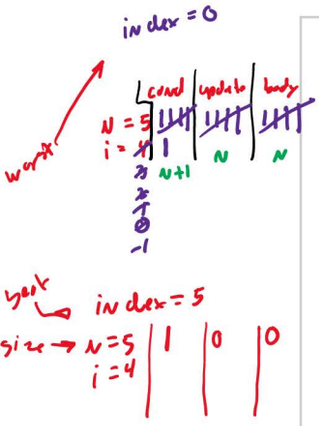




Pathfinding

usefulness ↓
- DFS
- BFS
- Djikstra
- A* (game & simulation)







**Counting Steps**

ArrayList Insert - ignore ExpandCapacity

| Best Case | Worst Case | Avg Case |
|---|---|---|
| 0 | 0 | 0 |
| 1+1+0 | 1+(N+1)+N | 1+(N/2+1)+N/2 |
| 0 | N | N/2 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 4 | 3N+4 | 3/2N+4 |

```
public void insert(int index, String s) {
  //expandCapacity();   //ignore
  for (int i = size - 1; i >= index ; i--) {
    this.elements[i+1] = this.elements[i];
  }
  this.elements[index] = s;
  this.size += 1;
}
```

ArrayList ExpandCapacity

| Best Case | Worst Case | ~~Avg Case~~ |
|---|---|---|
| 1 | 1 | |
| 1+1 | 1+0 | |
| 0 | 2N+2N+1 | |
| 0 | 1+(N+1)+N | |
| 0 | N | |
| 3 | 7N+6 | |

allocate + set default

```
private void expandCapacity() {
  int currentCapacity = this.elements.length;
  if(this.size < currentCapacity) { return; }
  String[] expanded = new String[currentCapacity * 2];
  for(int i = 0; i < this.size; i += 1) {
    expanded[i] = this.elements[i];
  }
  this.elements = expanded;
}
```

ArrayList Insert - with ExpandCapacity → week 7, 8?

| Best Case | Worst Case | ~~Avg Case~~ |
|---|---|---|
| 3 | 7N+6 | |
| 1+1+0 | 1+(N+1)+N | |
| 0 | N | |
| 1 | 1 | |
| 1 | 1 | |
| 7 | 10N+10 | |
| 2 | 7N+8 | |

```
public void insert(int index, String s) {
  expandCapacity();
  for (int i = size - 1; i >= index ; i--) {
    this.elements[i+1] = this.elements[i];
  }
  this.elements[index] = s;
  this.size += 1;
}
```

add

**Counting Steps - where size of the contents is *n***

LinkedList Add

| Best Case | Worst Case | Avg Case |
|---|---|---|

```
public void add(String s) {
  Node current = this.front;
  while(current.next != null) {
    current = current.next;
  }
  current.next = new Node(s, null);
  this.size += 1;
}
```

LinkedList Insert

| Best Case | Worst Case | Avg Case |
|---|---|---|
| 1 | 1 | |
| 1+1+0 | 1+(N+1)+0 | |
| 0 | N | |
| 1 | 1 | |
| 1 | 1 | |
| 5 | 3N+5 | |

```
public void insert(int index, String s) {
  Node current = this.front;
  for(int i = 0; i < index; i += 1) {
    current = current.next;
  }
  current.next = new Node(s, current.next);
  this.size += 1;
}
```

LinkedList Get

| Best Case | Worst Case | Avg Case |
|---|---|---|
| 1 | 1 | 1 |
| 1+1+0 | 1+(N+1)+N | 1+(N/2+1)+N/2 |
| 0 | N | N/2 |
| 1 | 1 | 1 |
| 4 | 3N+4 | 3/2N+4 |

```
public String get(int index) {
  Node current = this.front.next;
  for(int i = 0; i < index; i += 1) {
    current = current.next;
  }
  return current.value;
}
```

ArrayList Get

| Best Case | Worst Case | Avg Case |
|---|---|---|
| 1 | 1 | 1 |

```
public String get(int index) {
  return this.elements[index];
}
```