CSE12 - Lecture 8 - B00

Monday, October 10, 2022 9:00 AM

PA2 due tomorrow

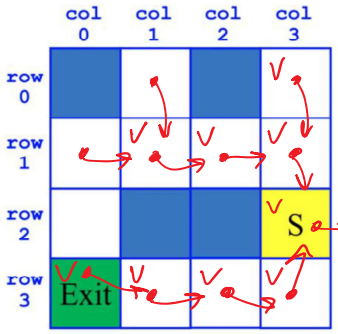
2D Search

Breadth-First Search (BFS)

Guaranteed shortest path

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

class Node {
boolean visited;
Node next;
}



Worklist

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
```

null

S W N E
1 2 3 4

Run through the SearchForTheExit algorithm. Draw the queue.

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

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

5 4 3 2 1

exit start

use stack for reverse

How many nodes were visited?

9

How many total squares were added to the queue?

11

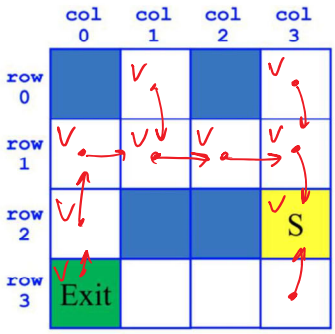
Was this the shortest path?

Yes

Depth-First Search (DFS)

Will always find a path. Possibly faster.

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



Work it!

```
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
```

S W N E
4 3 2 1

Run through the SearchForTheExit algorithm. Draw the stack.

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

exit → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (1,3) → (2,3) → null
7 6 5 4 3 2 1

How many nodes were visited?

9

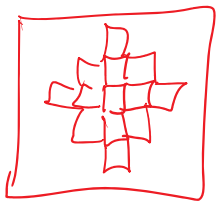
How many total squares were added to the stack?

10

Was this the shortest path?

NO

BFS



DFS