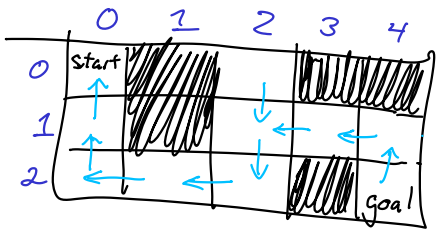
Searching

Solving a maze:



Collection of places we know we can reach but haven't explored

List

- take a coord out: O(?)
- put a coord in: O(?)

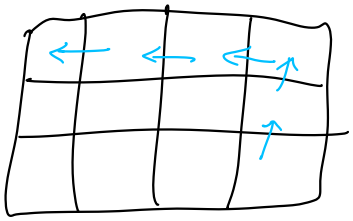
~~0,0~~
~~0,1~~
~~0,2~~
~~1,2~~
~~2,2~~
~~2,1~~
2,0
~~3,1~~
~~4,1~~
~~4,2~~ ✓

Where do I know about?

2D Array

- Get a coord obj O(1)

0,0
0,1
0,2
1,2
2,2
2,1
2,0
3,1
4,1
4,2



Initially

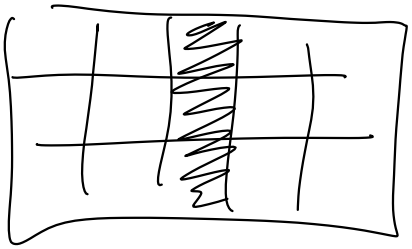
Until I reach my goal or I run out of places to look

Take a place I haven't explored

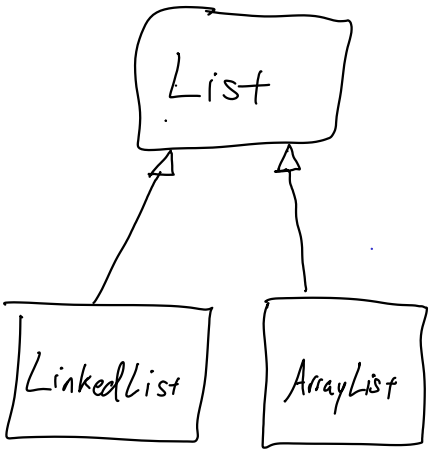
Figure out where I can go (observe walls, etc.)

For each new place:

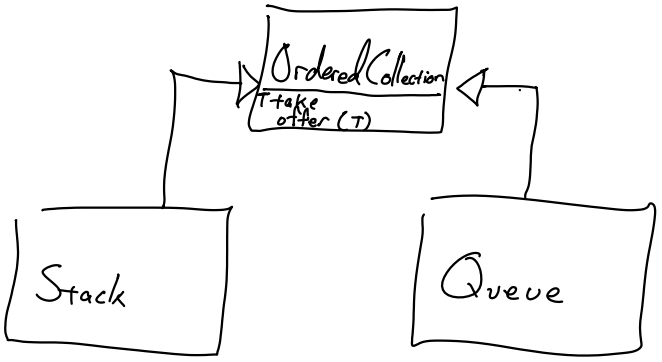
Add to both my explore coll. and my know-about coll. and remember where I was when I found the new place



ADT
abstract data type



Data Structures



Stack ADT

- virtual void push (T element) = 0; ← put something on top
- virtual T pop () = 0; ← take something off of the top
- virtual int getSize () = 0;

LIFO
last in first out

Which is better for searching?

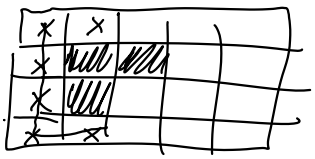
No.

Queue ADT

- virtual void enqueue (T element) = 0;
- virtual T dequeue () = 0;
- virtual int getSize () = 0;

FIFO
first in first out

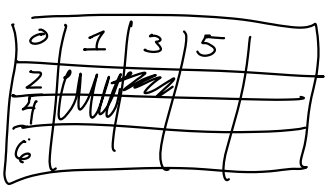
Search w/ Stack: Depth-first Search (DFS)

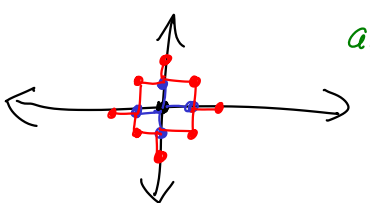


recency

less memory

Search w/ Queue: Breadth-first Search (BFS)





always find shortest path first