Stack — ADT

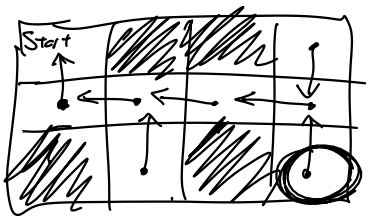
Array Stack
Linked Stack
— concrete data structure

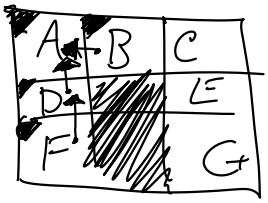
Maze solving

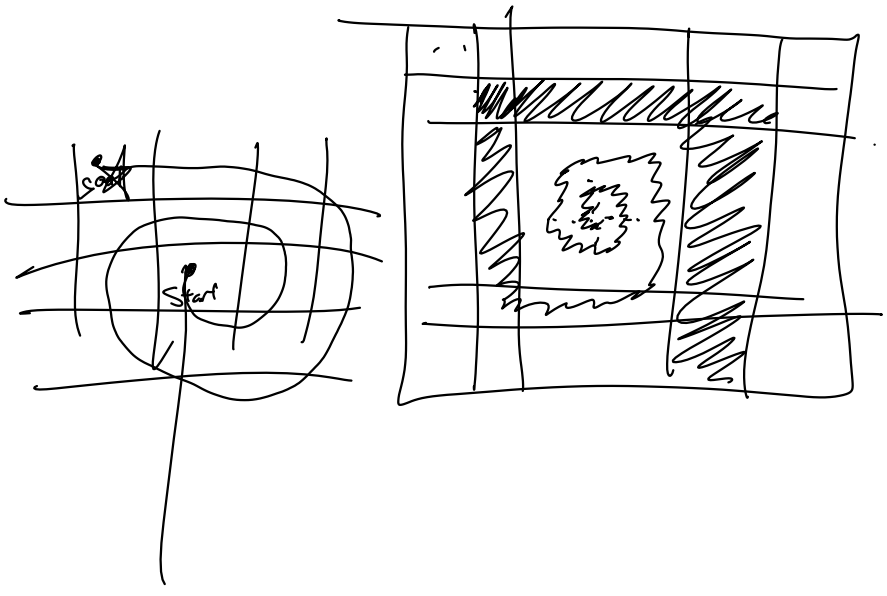
where have I been?
where haven't I been?
what locations are adjacent?

Function Search(grid, frontier) // assume start in pos (0,0)
// assume end in lower right

mark start Location as visited
frontier ← new LinkedStack() Queue
frontier → enqueue push (startLocation) insert
While frontier → getSize() > 0:
current ← frontier → dequeue pop()
if current is goal: remove
return the path I used to get here
for each neighbor of current:
if neighbor has not been visited:
frontier → enqueue push(neighbor) insert
mark neighbor as visited
set neighbor's previous to current
EndWhile
Return no path





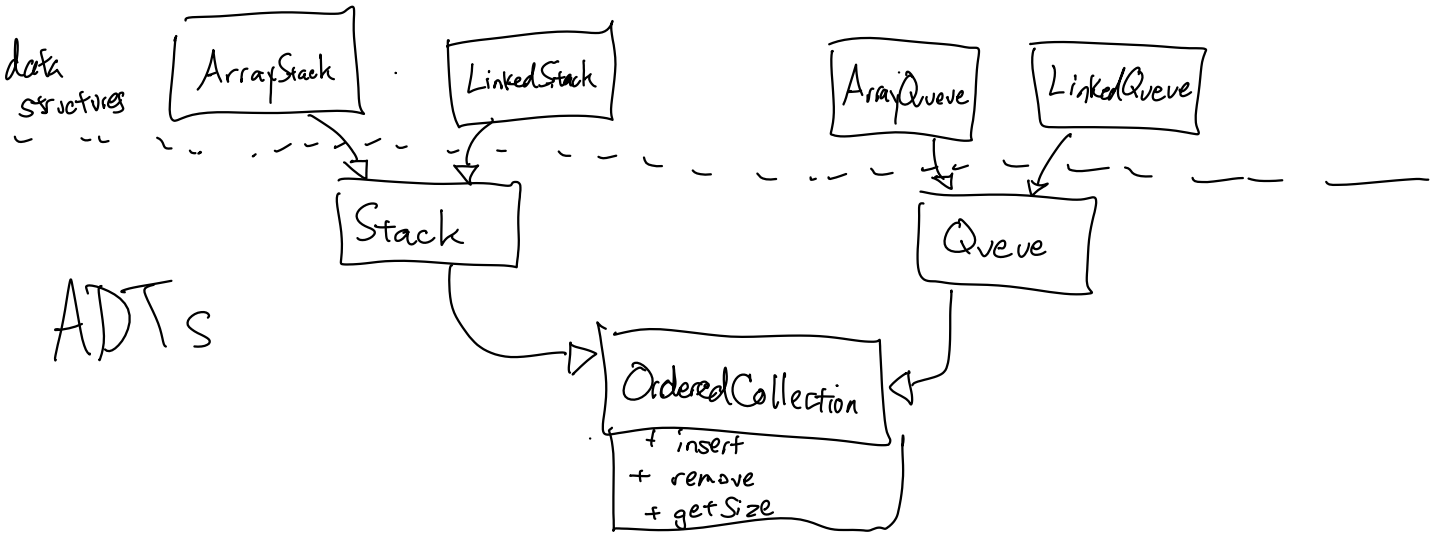


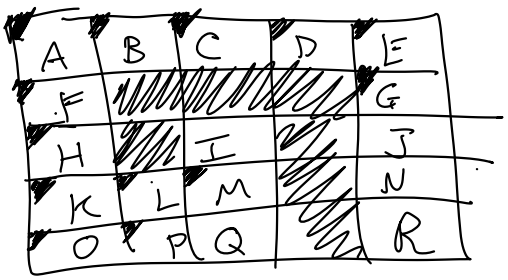
Queue — ADT

enqueue — add an element
dequeue — remove the least-recently added (but not yet removed) element
getSize — # of items

Queue: FIFO — first in, first out
Stack: LIFO — last in, first out

LinkedQueue
ArrayQueue



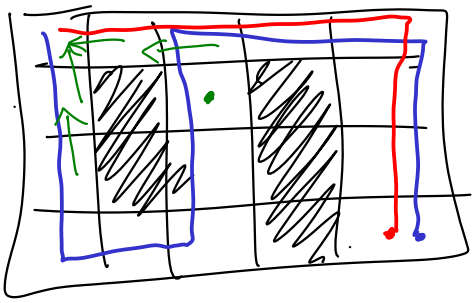


frontier

G, M, P

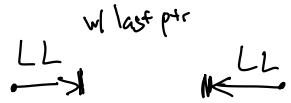
Depth-First Search (Stack) goes in a direction completely before trying another path. Good memory usage, bad in that unlucky searches are expensive.

Breadth-First Search (Queue) goes in all directions simultaneously.
More expensive w.r.t. memory, but limits luck.
All spaces which are n distance from start are explored before any spaces at distance n+1.



DFS
BFS DFS

Linked Queue



enqueue	O(1)	O(1)
dequeue	O(n)	O(1)