

Queue

Hsuan-Tien Lin

Dept. of CSIE, NTU

April 7, 2020

What We Have Done

algorithm	data structure
<i>Get Min Index</i> sequential search	array (or linked list) ✓
selection sort	array (or linked list) ✓
insertion sort	linked list (or array) ✓
binary search	ordered array ✓
polynomial merge	sparse array on linked list ✓
parenthesis matching	stack ✓
postfix evaluation	stack)
infix to postfix	stack)

next: another algorithm with stack (and more)

Recursive Algorithm

GET-OUT-RECURSIVE(m , $(0, 0)$)

Getting Out of Maze Recursively

GET-OUT-RECURSIVE(Maze m , Position (i, j))

mark (i, j) as **visited**

for each unmarked (k, ℓ) reachable from (i, j) **do**

if (k, ℓ) is an exit

return TRUE

end if

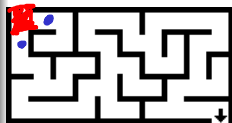
if GET-OUT-RECURSIVE(m , (k, ℓ))

return TRUE

end if

end for

return FALSE ←



Recursion (Reading Assignment: Section 3.5, Remember?)

- a function call to itself
- be ware of **terminating conditions**
- can represent programming intentions clearly
- at the expense of **“space”** (why?)

A General Maze Algorithm

General

Getting Out of Maze by **Container**

GET-OUT-CONTAINER(Maze m , Position (i, j))

while container not empty **do**

$(i, j) \leftarrow$ remove from container

 mark (i, j) as visited

for each unmarked (k, ℓ) reachable from (i, j) **do**

if (k, ℓ) is an exit

return TRUE

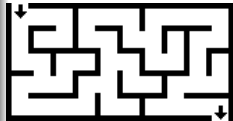
end if

 insert (k, ℓ) to container [and mark (k, ℓ) as todo]

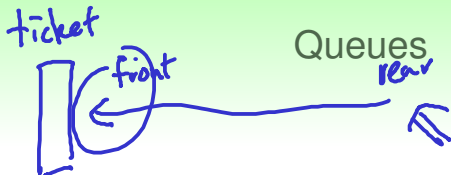
end for

end while

return FALSE



- if “random” remove from container: “random walk” to exit



Queue

- object: a container that holds some elements
- action: [constant-time] enqueue (to the rear), dequeue (from the front)

insert At Rear

remove from

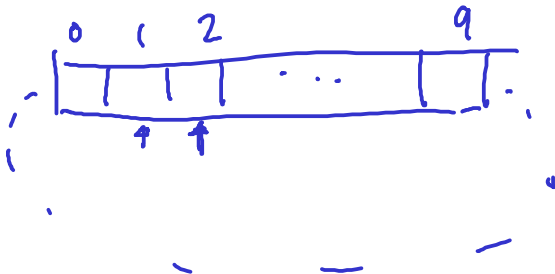
- first-in-first-out (FIFO): 買票, 印表機
- also very restricted data structure, but also important for computers

stack: LIFO

Queues Implemented on Circular Array (5.2.4)

Reading Assignment

be sure to go ask the TAs or me if you are still confused



Reading Assignment

be sure to go ask the TAs or me if you are still confused

Maze From Stack to Queue

Getting Out of Maze by Queue

GET-OUT-QUEUE(Maze m , Position (i, j))

while queue not empty **do**

$(i, j) \leftarrow$ dequeue from queue \leftarrow

mark (i, j) as visited

for each unmarked (k, ℓ) reachable from (i, j) **do**

if (k, ℓ) is an exit

return TRUE

end if

enqueue (k, ℓ) to queue [and mark (k, ℓ) as todo]

end for

end while

return FALSE

- use of stack/queue: store the yet-to-be-explored positions
- stack version : first (lexicographically) way out (explore deeply)
—depth-first search DFS
- queue version : shortest way out (explore broadly) —breadth-first search BFS

