Queue

Hsuan-Tien Lin

Dept. of CSIE, NTU

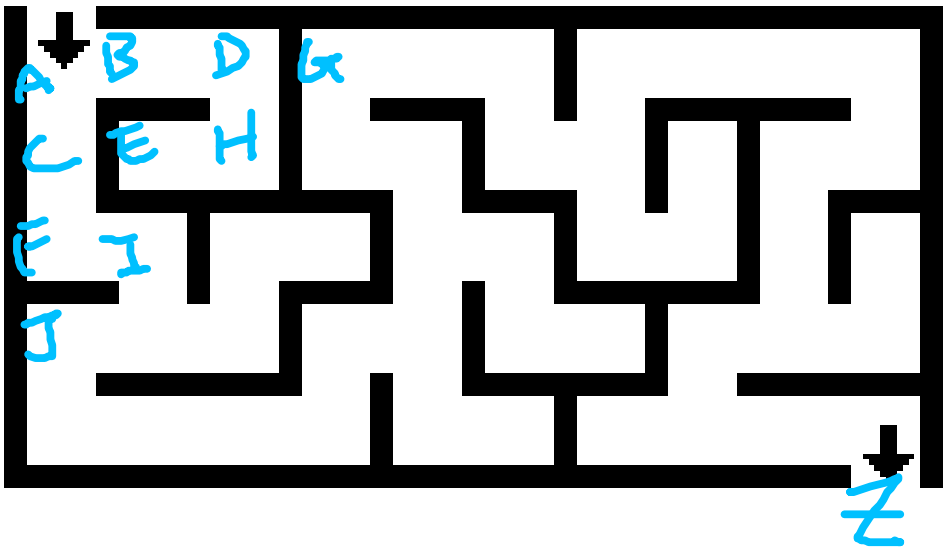
March 23, 2021

What We Have Done

algorithm	data structure
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 postifix	stack

next: another algorithm with stack (and more)

The Maze Problem



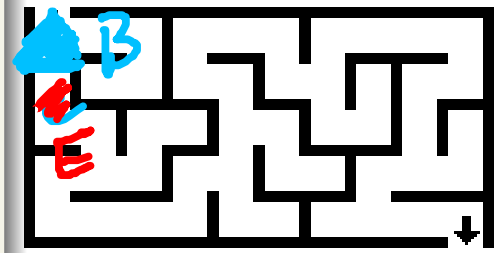
http://commons.wikimedia.org/wiki/File:Maze01-01.png

given a (2D) maze, is there a way out?

A General Maze Algorithm

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

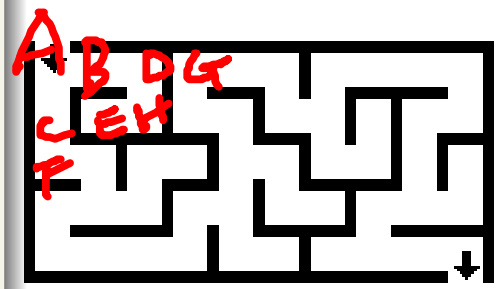


- if "random" remove from container: "random walk" to exit

Getting Out of Maze by Stack

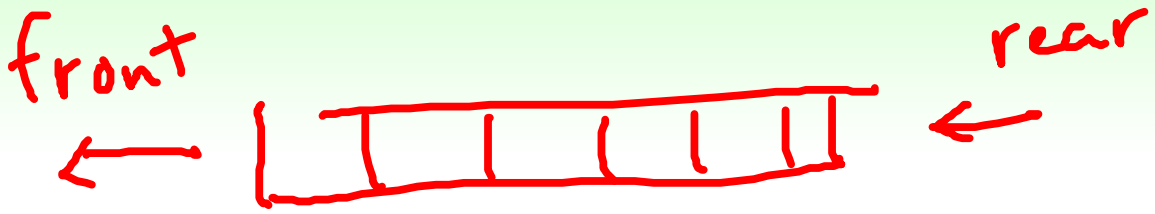
GET-OUT-STACK(Maze m, Postion (i, j))

while stack not empty **do**
  $(i, j) \leftarrow$ pop from stack
  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
    push (k, ℓ) to stack [and mark (k, ℓ) as todo]
  end for
end while
return FALSE



other containers?

Queues

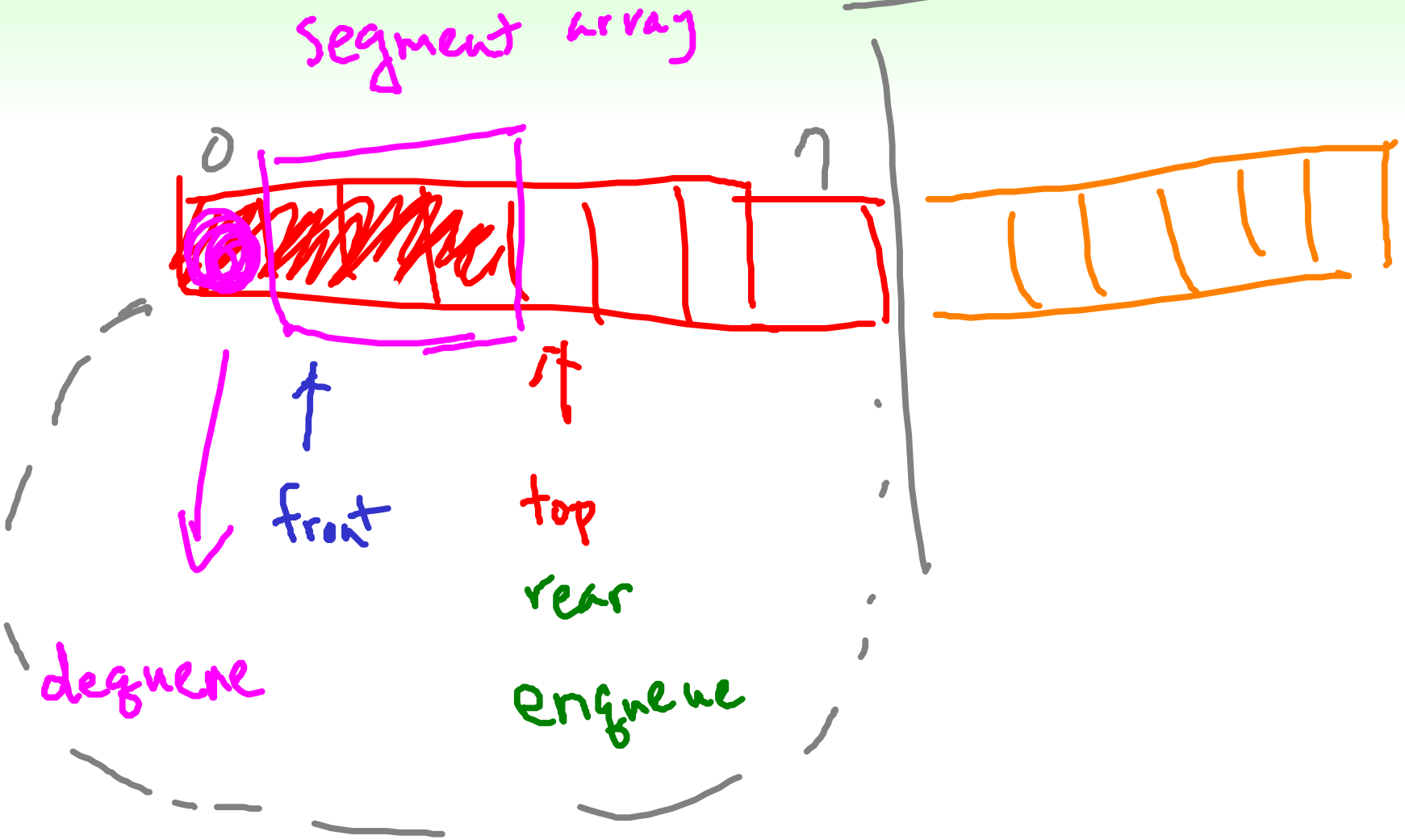


Queue

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

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

Queues Implemented on Circular Array

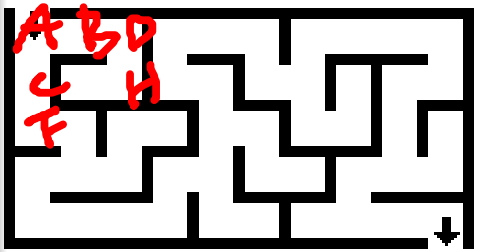


Maze From Stack to Queue

Getting Out of Maze by Queue

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

```
while queue not empty do
    (i, j) ← dequeue from queue
    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
- queue version : shortest way out (explore broadly) —breadth-first search

Deques

Deque = Stack + Queue + push_front

- object: a container that holds some elements
- action: [constant-time] push_back (like push and enqueue), pop_back (like pop), pop_front (like dequeue), push_front

- application: job scheduling

can be implemented by circular array or doubly-linked list