Week 2 - Stacks & Queues

AD 325 - Brenden West

Contents

Learning Outcomes

- Stacks
- Queues
- Iterators

Reading & Videos

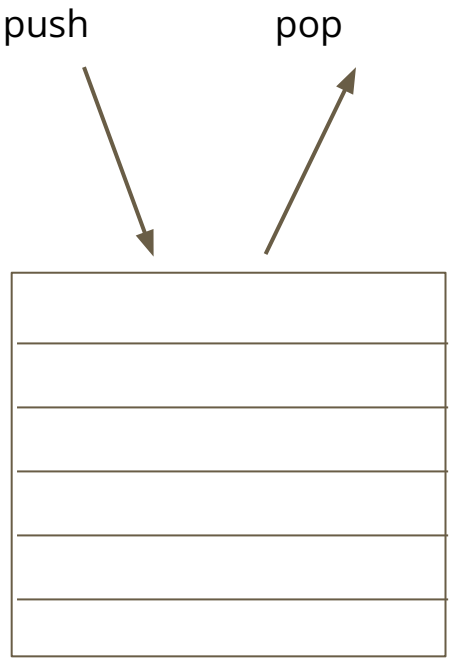
- Carrano & Henry, Chapters 5 - 8, Interlude 4 Iterators
- https://www.coursera.org/learn/algorithms-part1/home/week/2
- https://algs4.cs.princeton.edu/13stacks/ (review)
- https://www.geeksforgeeks.org/stack-data-structure/ (review)
- https://www.geeksforgeeks.org/queue-data-structure/ (review)

Stacks

A data collection based on last-in, first-out (LIFO) principle.

Insertion and deletion both happen at the “top” of the stack.

- Stack items are accessed in reverse order of being added
- Useful for reversing items in a collection without knowing total count
- Common stack use cases:
  a. browser history
  b. mobile application screens
  c. evaluating arithmetic expressions



Stack operations

- isEmpty()
- size() - return count of items in stack
- peek() - return item at top of stack
- push() - add item to top of stack
- pop() - remove item from top of stack

Stack implementation

Stacks can use a linked-list or an array for data storage.

- For linked-list implementation, it's most efficient to treat first node as 'top' of stack. Stack operations for linked-list implementation are O(1) in worst case.
- For array implementation, it's most efficient to treat last occupied element as top of stack. Stack operations for array implementation are O(1), except for resizing the array.

Stack implementation - Array

Array implementation of a stack.

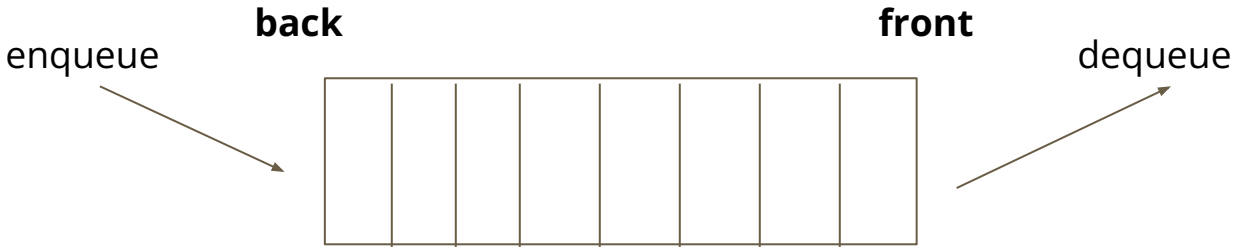
- Use array s[] to store N items on stack.
- push(): add new item at s[N].
- pop(): remove item from s[N-1]
- Resize array when full or if size falls below ¼ full

Queues

A data collection based on first-in, first-out (**FIFO**) principle.

Similar to stack, but operations happen at both ends of the collection.

- Data items are organized in the order received - earliest item is at the front and most recently added item is at the back
- Double-ended queue (Deque) is similar to a queue, but items can be added/removed from either end
- Priority queue orders items by importance rather than arrival time. Requires that items be Comparable



Queue operations

- isEmpty()
- size() - return count of items in queue
- peek() - return item at front of queue
- enqueue() - add item to back of queue
- dequeue() - remove item from front of queue

Queue Implementation - Linked List

- Maintain pointers to **first** and **last** nodes in linked list
- insert/remove from opposite ends

Queue Implementation - Array

- Use array q[] to store items in queue.
- enqueue(): add new item at q[tail].
- dequeue(): remove item from q[head].
- Update head and tail modulo the capacity.
- Add resizing array.