Stacks and Queues
Problem Solving Club Oct 19 2016
Stacks

- A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle.
- Only two operations are allowed: push the item into the stack, and pop the item out of the stack.
Usage of stack

- Undo mechanism
- Function call stack
- Reverse a string
- Depth first search (DFS)
Stack implementation

- Array stack implementation
- Java ArrayList/Stack
- C++ std::vector/stack

- Linked list stack implementation
- Java LinkedList
- C++ std::list
Queues

- A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle.
- An excellent example of a queue is a line of students in the food court.
Usage of queues

- Job processing / scheduling
- Breadth first search (BFS)
  - single source shortest paths in an undirected graph
Queue implementation

- Array-based double ended queue
- Java ArrayDeque
- C++ std::deque/queue
- Linked list based queue
- Java LinkedList
- C++ std::list
Priority queues

- A priority queue is like a regular queue or stack data structure.
- But additionally each element has a "priority" associated with it.
- In a priority queue, an element with high priority is served before an element with low priority.
Usage of priority queues

- Sorting (heapsort)
- Caching
- Dijkstra’s algorithm – singles source shortest paths in a directed graph
Priority queue implementation

- Binary heap based priority queue
  - Java PriorityQueue
  - C++ std::priority_queue
- Self-balancing binary search tree based priority queue
  - Java TreeSet
  - C++ std::set
Recap

● Stack – last-in first-out (LIFO).

● What is the complexity of push/pop?

● Answer: O(1) – constant time

● What is the preferred data structure for implementation?

● Answer: Array – faster and uses less memory than linked list

● Queue - first-in first-out (FIFO)

● What is the complexity of enqueue/dequeue?

● Answer: O(1)