CS 313
- Kangmei Yang
Priority Queue

It’s a queue, but instead of FIFO, top priority out first.
Each element has a priority value. Dequeue the element has the top priority first, not the one in the queue longest (timewise).

Useful in scheduling.

How to store such queue?
- efficiently perform all basic operation enqueue, dequeue, peek
  (Smaller value represent higher priority)
Add priority 4,6,2,7,1,3

header
  - unsorted: dequeue and peek is costly.
  - sorted: enqueue is costly

trailer
Implement with Heap

Binary heap – elements are comparable
- is a complete tree (every level except maybe last level is full. Fill up the tree from top level down, and left to right, no gap.
- every node’s key has a higher priority or equal priority than children’s. Highest priority element is at the root of the tree.
- allow duplicate elements

Min-heap: smaller value represent a higher priority
Max-heap: larger value represent a higher priority

Given the property of a complete tree, we may represent it using an array to save space.
Min Heap

- represent in an array, space efficiency
- store from the root, then level by level, no gap
- easy to calculate where parent and children are

-easier if index starts from 1
- for any value at index i
  - it’s parent value (if exists) stores at index i / 2
  - it’s left child value (if exists) stores at i * 2
  - it’s right child value (if exists) stores at i * 2 + 1
Min Heap Priority Queue

- Efficient basic operations (Constant or Logarithmic time)
- peek(), return the root, a[1]
- enqueue()
  - add to the end
  - move up to a valid spot
    - as long as current position violates the heap property, swap with it’s parent until it’s at a good spot.
      (bubble up)
Enqueue

Add 3

- Add to the first empty node of the tree
- Compare with it’s parent
- Swap if violate the property
- Repeats with the new value until the spot is good

Time complexity $O(\log n)$
Dequeue

remove top minimum value only

- To maintain a complete tree. Replace it with last value in the tree.
- Compare with it’s children
- Swap with smaller child if violate the property (bubble down)
- Repeats with the new value until the spot is good

Time complexity $O(\log n)$
MinHeap

Add 2, 1

remove, remove