COP 3330 4/3/26

Last Built-In Data Structure Lecture!

Priority Queue (Heap)

Operations

1) insert element
2) delete minimum item

} about it! O(lg n)

n = # items in priority queue

Note: Can have duplicates

Roughly anything this does a TreeSet (to handle duplicates) could do too → PriorityQueue if preferable since the hidden constant in runtime is lower.

USE

same:
- add(E e) O(lg n)
- clear()
- offer(E e) O(lg n)

E peek() ~~O(lg n)~~ O(1)

E poll() O(lg n)

int size() O(1)

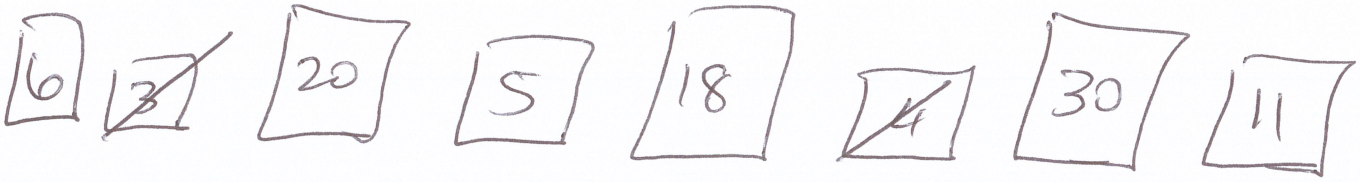
DON'T USE

contains(Object o) O(n)

remove(Object o) O(n)

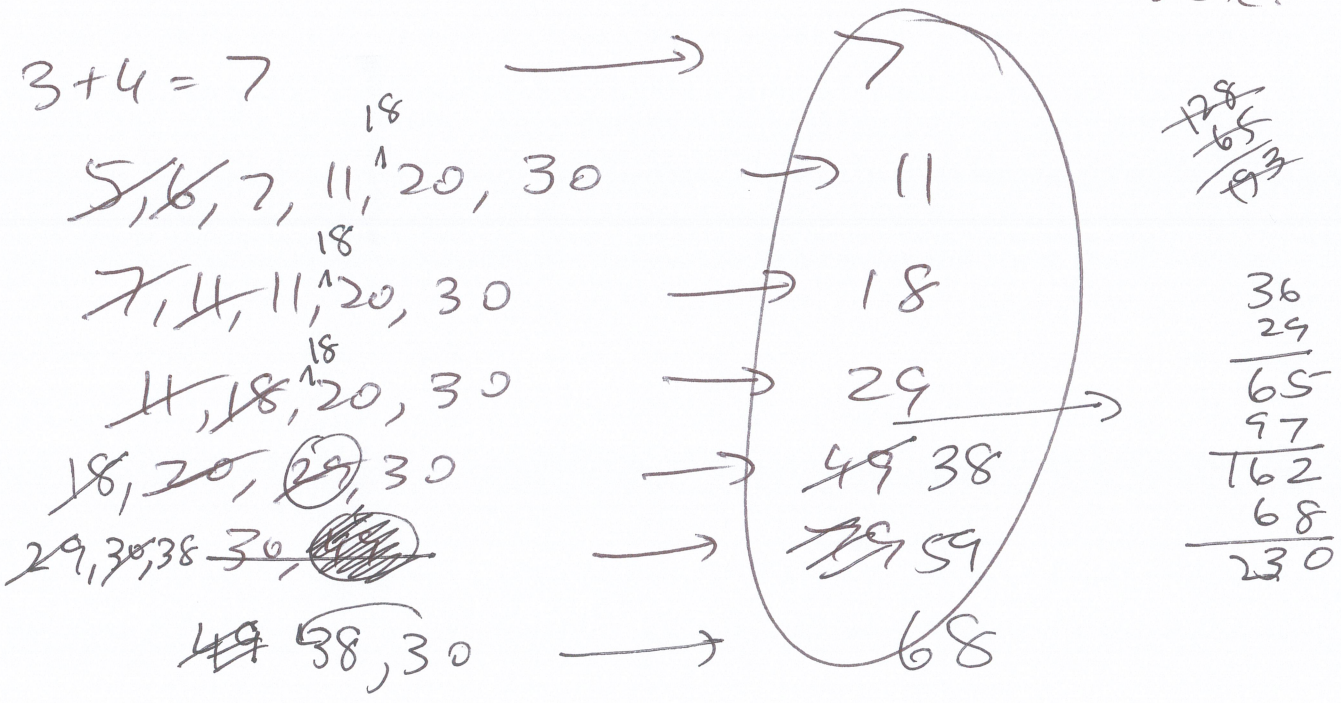
AddAll

Stacks of papers in sorted alpha order



cost of merging stacks of size a, b is a+b.

Problem: Determine minimum cost of merging all the stacks into 1 sorted stack.



		Priority	Time
t = 0	Homework	15	30
t = 7	Clean	30	20
t = 12	Watch TV	10	45
t = 20	Lunch	1	30
t = 30	Team Mtg	5	60

, etc.

1. Add request to PQ
2. If you're free start on the ~~best~~ most important task available.

Priority → break ties by time of task (shorter) cores first

PQ used in CS2

Dijkstra's Alg. (single source shortest path)

Huffman Coding (greedy algorithm)

High Level Design

1) Load PQ w/all items at 1st valid start time

2) Main Loop { while (PQ.size() > 0)

   2a) get ~~next~~ next item from PQ

   2b) DO THIS TASK

   2c) Updak current time

   2d) Add all items from list that ~~where~~ were requested before or at current time

}