Priority queues

Pending jobs list (Job, priority)

→ new jobs may join list at any time

→ Extract job with highest priority in list to be executed

How to maintain list of pending jobs and priorities so that jobs can be added and extracted efficiently?

Priority queue: Abstract Data structure storing pending jobs with priorities with 2 operations

insert()

delete_max()

Naive: n = size of list ~~(Linear structure)~~

Unsorted list: insert $O(1)$

delete_max $O(n)$

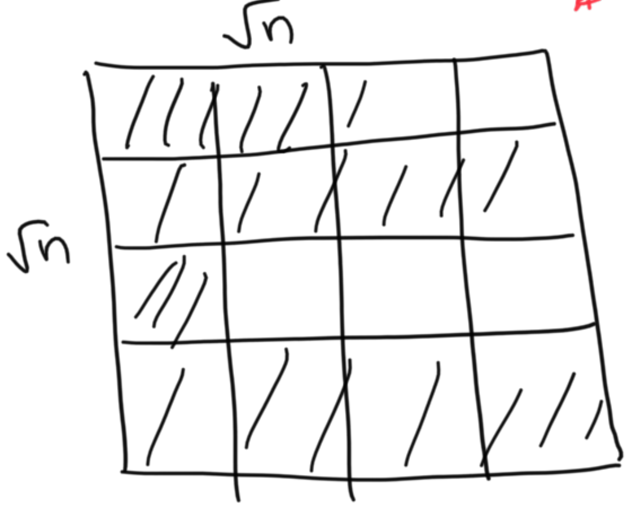
sorted list: insert $O(n)$

delete_max $O(1)$

Processing n jobs $\rightarrow$ n insertions
n deletions

$\rightarrow$ $O(n^2)$ for both naive approaches

Better (if you know beforehand # of jobs $\leq n$) (2-D structures)



$\sqrt{n} \times \sqrt{n}$ array

each row maintained in sorted array and

also maintain how many jobs in each row are there.

Insert $\rightarrow$ check end element of each row and insert first time there is space and inserting element $\geq$ end element of current row

$O(\sqrt{n})$

delete $\rightarrow$ pick max (end elements of rows) and delete it. $O(\sqrt{n})$

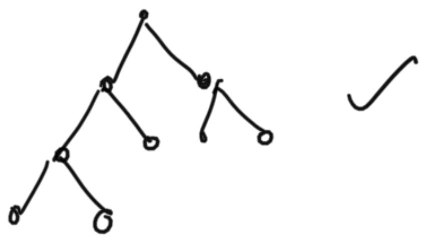
To process n jobs: $\left.\begin{array}{l} n \text{ inserts} \\ n \text{ delete-max} \end{array}\right\} O(n\sqrt{n})$

$= O(n^{3/2})$

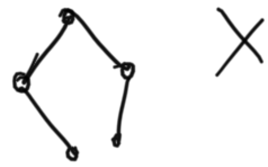
<u>much better</u>

<u>Heaps</u>

- binary tree with n nodes
- "balanced" ← height $\log n$



balanced



unbalanced

given n, unique balanced binary tree with n nodes

↖ # of jobs

Can implement priority queues so that

insert : $O(\log n)$

delete : $O(\log n)$

processing n jobs : $O(n \log n)$

(can grow heap as we go along.

no need to fix # of nodes in heap in advance)