RAHUL ETHIRAJ
ethiraj@usc.edu

CSCI-570 - SPRING 2018 - HW3

1.) Graded Problems:

1.) n elements

i) Build a min-heap on n elements : $O(n)$

ii) Sort the min-heap : $O(n \log n)$

iii) If median is even,

min-heap $(n/2)$

" " is odd,

min-heap $(n/2 + 1)$

$\Bigg\} O(1)$

iv) Extract-Median ():

→ Get median using (iii)

→ decreasekey() - Replace min-heap (median) with $-\infty$

↳ $O(\log n)$

→ $-\infty$ is at the root now.

→ extractMin () - $O(\log n)$

→ heapify () - ~~$O(\log n)$~~ $O(2^h)$

v) Insert (): $O(\log n)$

→ Add a new key at the end of the tree

→ If new key is > parent,

End;

else,

heapify () : $O(2^h)$

v) Delete ():

Same as (iv)

2. <u>Online</u> <u>version</u>:

We have room to store <u>k</u> of the elements that are coming in.

First <u>k</u> elements : insert into Min_heap : $O(k)$

next $(n-k)$ elements :

if key value < root (min_heap), ignore

else,

deletemin () : $O(\log k)$

insert new element : $O(\log k)$

heapify (), if needed : $O(2^n)$

3.

i) Create an output array of size <u>$n * k$</u>.

ii) Assuming we have <u>k</u> arrays sorted in ascending order, having <u>n</u> elements.

- Create a min heap of size k : $O(k)$
- insert 1st element in all the arrays into the heap : ~~O(log)~~ $O(\log k)$

iii) Repeat <u>$n * k$</u> times

- Extractmin () and Store in output array : $O(\log k)$
- Replace heap root with next element from the array from which the element is extracted.

  If the array doesn't have any more elements, replace root with ∞.

  Heapify () : $O(2^k)$

4.)

Payoff $= a_1^{b_1} \times a_2^{b_2} \times a_3^{b_3} \ldots \times a_n^{b_n}$

Problem :

Need to maximise payoff.

Product = 1

- Put a_i in max_heap (A) : $O(n)$
- Put b_i in max_heap (B) : $O(n)$
- Extract root (maximum is always at the top) a_i and b_i

  Compute $a_i^{b_i}$ and store in product : $O(\log n)$
- Heapify (A, B) : $O(2^h)$
- Repeat for n items : $O(n \log n)$

2.) Practice Problems :

1.) G - strongly connected

n nodes : intersections

m edges : one-way streets : directed

a.) i) Run BFS from random s.

If all nodes are reached, Step ii) : $O(m+n)$

else,

Mayor is wrong

ii) Reverse the direction of all edges to get G^{inv} : $O(m+n)$

iii) Repeat the same for ~~...~~ G^{inv}, as (i) : $O(m+n)$

b.) Keep 's' from a) as 'Town Hall' and check whether it is strongly connected.

2.)

$G = (V, E, w)$

Shortest path $= \delta(s, u)$

$\delta(s, t) = ?$

$\delta(s, u) \cancel{+ \delta(u, t)}$

$\delta(s, v) + w(v, u) = \delta(s, u)$

Run BFS tree to find shortest path from s to v. : $O(V + E)$