Sheet 2

1-

2- Minimum # of nodes happens in a heap in which the last level contains only one node.

∴ Minimum # of nodes $= 2^0 + 2^1 + 2^2 + \dots 2^{h-1} + 1 = 2^h$

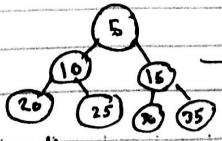
* Maximum # of nodes happens in a heap in which the last level is full of nodes.

∴ Maximum # of nodes $= 2^0 + 2^1 + 2^2 + \dots 2^h = 2^{h+1} - 1$

3- Smallest element will be at the last level of the max heap (will be one of the leaf nodes) for finding the smallest element in a max heap

↳ O(n)

4- Min heap is a complete binary tree where each parent is less than any of its children and a sorted array is a min heap if it is sorted in ascending order 5, 10, 15, 20, 25, 30, 35



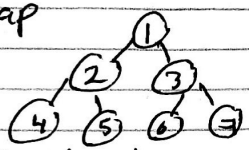
→ Min Heap

and sorted Array has the property that $A[i] \leq A[j]$ if $i < j$ and A is min heap if $A[i] <= A[\text{left}(i)]$

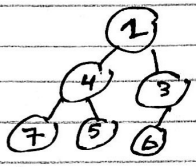
and A[i] <= A[Right(i)] where i < left(i) and i < right(i) This implies a sorted array

Date / / Object

if we have an array [1,2,3,4,5,6,7] we can say that a Min Heap

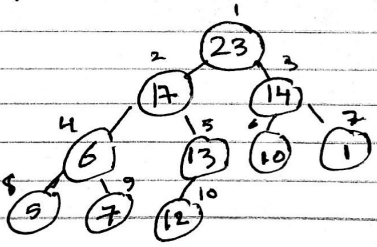


but if we pull [1] of the heap then reheapify it then it will be



then the array is no longer sorted [2,4,3,7,5,6]

5- [23, 17, 14, 6, 13, 10, 1, 5, 7, 12]



it's not a max heap because the node (4) has 2 children (2*i) (2*i+1)

(2*i) → 8 → has the value = 5 < 6

(2*i+1) → 9 → has the value = 7 > 6 → not valid in max heap

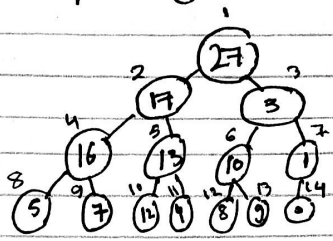
K.M.S.

Date / / Object

6. $A = [27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0]$

MAX HEAPify (A, 3) [i = 3]

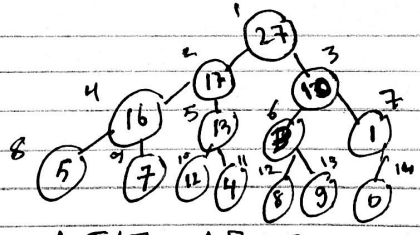
∴ L = Left(3) = 6 R = Right(3) = 7



∴ A[L] > A[3] ∴ largest = 6

∴ Exchange A[largest] && A[3]

then MaX HEAPify (A, 6)

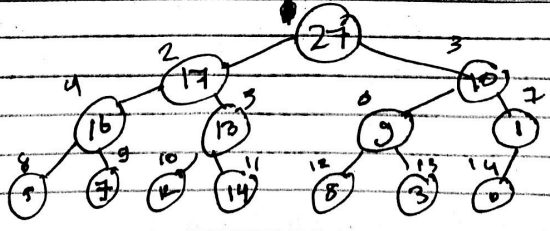


∴ L = 12
∴ R = 13

∴ A[L] > A[6] ∴ largest = 12

A[R] > A[largest] ∴ largest = 13

∴ Exchange A[largest] && A[6]



K.M.S.

Date / / Object

7. [5, 3, 17, 10, 84, 19, 6, 22, 9]

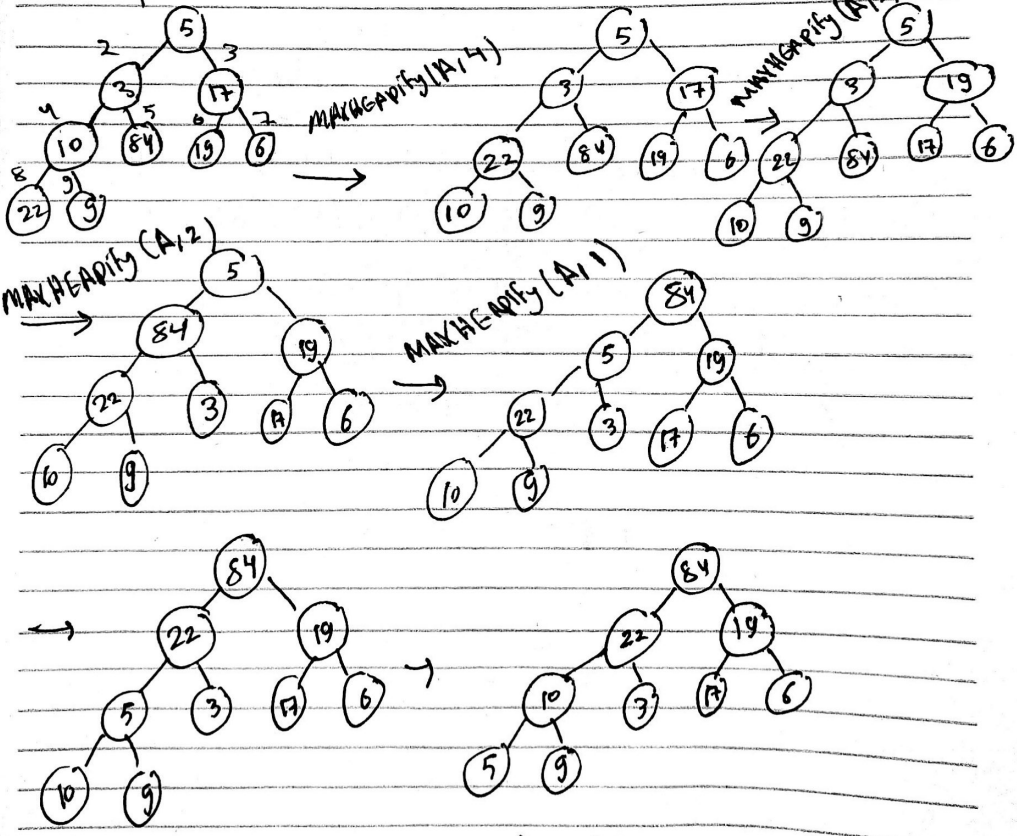
Build MAX HEAP

A.heap size = A.length = 9

for i = [A.length / 2] down to 1

MAXHEAPify (A, i) $O(n \log n)$

MAXHEAPify (A, 4) down to MAXHEAPify (A, 1)



The resulting array A = <84, 22, 19, 10, 3, 17, 6, 5, 9>

K.M.S.

Date / / Object

[8] The straightforward solution is to pick the smallest of the top elements in each list repeatedly this takes $k-1$ comparisons per element in total $O(k \cdot n)$ and keep the smallest element from each list in a heap, each element is augmented with the index of the lists where it comes from and we can perform a DeleteMin on the heap to find and delete the smallest element and insert the next element from the corresponding list. it takes $O(k)$ to build the heap for every element and it takes $O(\lg k)$ to delete Minimum and $O(\lg k)$ to insert the next one from the same list in total it takes $O(k + n \lg k)$

$= O(n \lg k)$

[9]

```
DeleteRoot (A, n)
lastelement ← A[n]
A[1] ⟷ lastelement
n ← n - 1
MinHeapify (A, 1)        O(n log n)
```

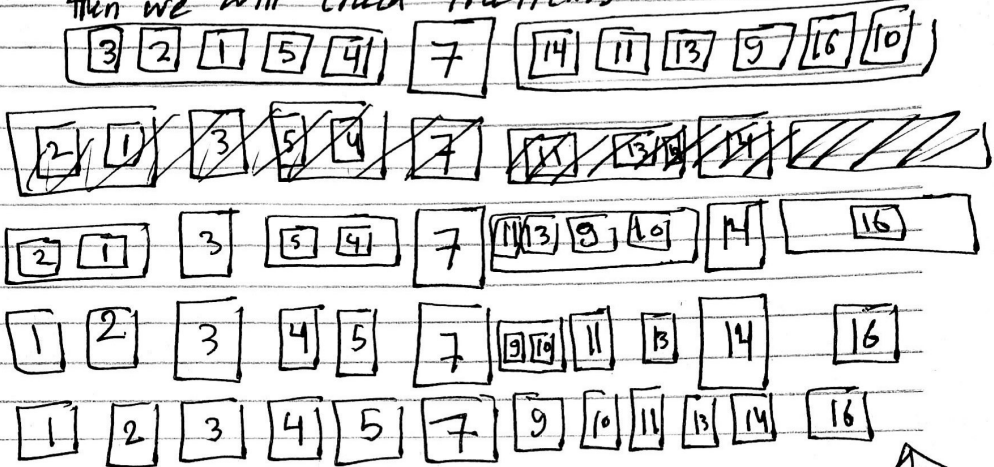
K.M.S

Date / / Object

[10] 7 is the pivot

Left < 7 < right

then we will check the items



= Sorted by Quick Sort

in average case $O(n \log n)$

in worst case $O(n^2)$

[11] 4-way mergesort divides the array into 4 subarrays and then calls a 4-way merge function which combines 4 sorted arrays into one sorted array and we will have index for each of the four sub arrays and it take 3 comparisons to determine the smallest of the four subarrays.

$T(n) = 4T(n/4) + n$ $O(n \log_4 n)$

if we divide the array into k equal subarrays

$O(n \log_k n)$

K.M.S.