CS435DE - Lab 9

Saurab Ghimire
417555

Problem 1: Solution
Binary heap properties:
For index `i`, left child is at `2i`, right child at `2i + 1`.
So for node at index 3 (value = 3):
Left child index = 6, value = 10
Right child index = 7, value = 1
Among A[3], A[6], A[7]:
A[6] (10) is the largest.

Step 2: Swap and recurse
Since A[6] > A[3], swap them:
After swapping A[3] and A[6]:
A = {27, 17, 10, 16, 13, 3, 1, 5, 7, 12, 4, 8, 9, 0}
Now recurse into index 6 (value = 3):
Swap A[6] and A[13]:
After swap A[6] and A[13]:
A = {27, 17, 10, 16, 13, 9, 1, 5, 7, 12, 4, 8, 3, 0}
Now recurse into index 13 (value = 3):
A = {27, 17, 10, 16, 13, 9, 1, 5, 7, 12, 4, 8, 3, 0}

Problem 2: Solution

We have 9 elements:

Array: { 5 3 17 10 84 19 6 22 9}

Start from floor(n/2) = floor(9/2) = 4

Call function on indexes 4, 3, 2, 1

Max_Heapify(4): A[4] = 10, left = 22 (A[8]), right = 9 (A[9]), max = 22
Swap A[4] and A[8] {5, 3, 17, 22, 84, 19, 6, 10, 9}

Max_Heapify(3): A[3] = 17, left = 19 (A[6]), right = 6 (A[7]), max = 19
Swap A[3] and A[6] {5, 3, 19, 22, 84, 17, 6, 10, 9}

Max_Heapify(2): A[2] = 3, left = 22 (A[4]), right = 84 (A[5]), max = 84
Swap A[2] and A[5] {5, 84, 19, 22, 3, 17, 6, 10, 9}
Recurse at 5: no children done

Max_Heapify(1): A[1] = 5, left = 84 (A[2]), right = 19 (A[3]), max = 84
Swap A[1] and A[2] {84, 5, 19, 22, 3, 17, 6, 10, 9}
Recurse at 2: A[2] = 5, left = 22 (A[4]), right = 3 (A[5]), max = 22
Swap A[2] and A[4] {84, 22, 19, 5, 3, 17, 6, 10, 9}
Recurse at 4: A[4] = 5, left = 10 (A[8]), right = 9 (A[9]), max = 10
Swap A[4] and A[8] {84, 22, 19, 10, 3, 17, 6, 5, 9}

Final heap array: {84, 22, 19, 10, 3, 17, 6, 5, 9}

Problem 3: Solution

Initial array:
5 13 2 25 7 17 20 8 4

Build Max Heap:
Heapify(3): 25 is fine
Heapify(2): swap 2 and 20 : 5 13 20 25 7 17 2 8 4
Heapify(1): swap 13 and 25 :   5 25 20 13 7 17 2 8 4
Heapify(0): swap 5 and 25 :   25 5 20 13 7 17 2 8 4
Heapify(1): swap 5 and 13 :   25 13 20 5 7 17 2 8 4
Heapify(3): swap 5 and 8 :   25 13 20 8 7 17 2 5 4
(Max Heap built)
25 13 20 8 7 17 2 5 4

Heapsort:
Swap 25 and 4, heapify :   20 13 4 8 7 17 2 5 25
Swap 20 and 5, heapify :   17 13 5 8 7 4 2 20 25
Swap 17 and 2, heapify :   13 8 5 2 7 4 17 20 25
Swap 13 and 4, heapify :   8 7 5 2 4 13 17 20 25
Swap 8 and 4, heapify :   7 4 5 2 8 13 17 20 25
Swap 7 and 2, heapify :   5 4 2 7 8 13 17 20 25
Swap 5 and 2 :   4 2 5 7 8 13 17 20 25
Swap 4 and 2 :   2 4 5 7 8 13 17 20 25

Sorted array:
2 4 5 7 8 13 17 20 25

Problem 4: Solution

A heap is a complete binary tree, meaning all levels are filled except possibly the last, which is filled from left to right. The height of a node is defined as the number of edges on the longest path from that node to a leaf. So, nodes at height 0 are leaves, height 1 are their parents, and so on.

Now, in a complete binary tree:

- At most $\lceil n/2 \rceil$ nodes can be at height 0 (leaves)
- At most $\lceil n/4 \rceil$ nodes can be at height 1
- At most $\lceil n/8 \rceil$ nodes can be at height 2
- And so on...

This pattern continues because each time we move up one level in the tree, the number of nodes is at most half of the level below.

So, in general, the number of nodes at height h is at most $\lceil n / 2^{(h+1)} \rceil$.

Therefore, we have shown that in an n-element heap, there are at most $\lceil n / 2^{(h+1)} \rceil$ nodes of height h.