Advanced Data Structures and Algorithms

Comprehensive Assignment Solutions

Arya Raag
Roll No: A125002

M.Tech (Computer Science and Engineering)

January 5, 2026

1 Heap Operations and Analysis

Question 3. (a) Show that in any heap containing n elements, the number of nodes at height h is at most $\lceil \frac{n}{2^{h+1}} \rceil$.
(b) Using the above result, prove that the time complexity of the Build-Heap algorithm is $O(n)$.

Detailed Solution:

Part (a): Nodes at Height h

Let h be the height of a node, defined as the number of edges on the longest simple path from the node down to a leaf. Leaves are at height 0.

By induction, we can see the structure of a complete binary tree:

- At height 0 (leaves), we have approximately $n/2$ nodes.
- At height 1, we have approximately $n/4$ nodes.
- Generally, at height h, we have roughly $n/2^{h+1}$ nodes.

Formally, the number of nodes of height h in any n-element heap is bounded by:

$$N_h \leq \left\lceil \frac{n}{2^{h+1}} \right\rceil$$

Part (b): Build-Heap Complexity Analysis

The `Build-Max-Heap` algorithm works by calling `Max-Heapify` on all non-leaf nodes, starting from the last non-leaf node down to the root.

The cost of `Max-Heapify` on a node of height h is $O(h)$. Therefore, the total cost $T(n)$ is the sum of the costs for all nodes:

$$T(n) = \sum_{h=0}^{\lfloor \lg n \rfloor} (\text{number of nodes at height } h) \cdot O(h)$$

Substituting the bound from Part (a):

$$T(n) \leq \sum_{h=0}^{\lfloor \lg n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil O(h)$$

Removing the ceiling function for the asymptotic bound (since $\lceil x \rceil < x+1$) and factoring out constants:

$$T(n) = O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right)$$

Summation Convergence

To evaluate the summation $\sum_{h=0}^{\infty} \frac{h}{2^h}$, consider the standard infinite series for $|x| < 1$:

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

Setting $x = 1/2$:

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2} = \frac{1/2}{1/4} = 2$$

Final Complexity

Since the infinite sum converges to a constant (2), the finite sum is also bounded by a constant.

$$T(n) = O(n \cdot 2) = O(n)$$

Conclusion

We have proven that building a heap from an unordered array takes **linear time**, $O(n)$. This is a significant improvement over the $O(n \log n)$ approach of inserting elements one by one, making `Build-Heap` the standard method for initializing priority queues and heapsort.
