Analysis of Build-Heap Algorithm

Manav Mantry

Problem Statement

(a) Show that in any heap containing n elements, the number of nodes at height h is at most:

$$\frac{n}{2^{h+1}}$$

(b) Using the above result, prove that the time complexity of the **Build-Heap** algorithm is $O(n)$.

Part (a): Number of Nodes at Height h

Definitions

- A heap is a **complete binary tree**.
- The **height** of a node is the number of edges on the longest downward path from that node to a leaf.

Observation

- Nodes with height h are located at level $H - h$, where H is the height of the heap.
- In a complete binary tree, the number of nodes at any level is at most the total number of nodes divided by the minimum number of nodes required below them.

Each node of height h must have at least:

$$2^{h+1} - 1$$

nodes in its subtree (including itself).

Upper Bound on Nodes at Height h

Since the total number of nodes in the heap is n, the number of nodes at height h, denoted by N_h, satisfies:

$$N_h \cdot (2^{h+1} - 1) \leq n$$

Ignoring constants for an upper bound:

$$N_h \leq \frac{n}{2^{h+1}}$$

Conclusion (Part a)

$$\boxed{\text{The number of nodes at height } h \text{ is at most } \frac{n}{2^{h+1}}}$$

Part (b): Time Complexity of Build-Heap

Understanding Build-Heap

The **Build-Heap** algorithm constructs a heap by calling `Heapify` on all non-leaf nodes, starting from the lowest internal node up to the root.

- The time complexity of `Heapify` on a node of height h is $O(h)$.
- Nodes with greater height take more time to heapify.

Total Time Complexity

Let the total time be $T(n)$. Then:

$$T(n) = \sum_{h=0}^{\lfloor \log n \rfloor} (\text{number of nodes at height } h) \times (\text{cost per node})$$

Using the result from Part (a):

$$T(n) = \sum_{h=0}^{\lfloor \log n \rfloor} \frac{n}{2^{h+1}} \cdot O(h)$$

Factoring out n:

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

Evaluating the Series

The series:

$$\sum_{h=0}^{\infty} \frac{h}{2^h}$$

is a convergent series and evaluates to a constant.

Hence:

$$\sum_{h=0}^{\lfloor \log n \rfloor} \frac{h}{2^{h+1}} = O(1)$$

Final Result

$$T(n) = O(n)$$

Conclusion

The time complexity of the Build-Heap algorithm is $O(n)$

This proves that Build-Heap runs in linear time despite individual heapify operations taking logarithmic time.