MTech CSE – 1st Semester
Student ID: **A125023**
Student Name: **SRIJITA VERMA**

Question

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

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

Answer

Assumptions and Model

We assume a binary heap, as defined in standard algorithm texts (CLRS-style), with the following properties:

- The heap is stored as a complete binary tree.
- The heap-order property (min-heap or max-heap) is irrelevant for this problem; only the tree structure matters.

Note that the heap-order property (min-heap or max-heap) plays no role in this argument; the proof depends solely on the structural completeness of the heap.

Definitions

Definition 1: Complete Binary Tree

A binary tree is complete if:

- Every level except possibly the last is completely filled.
- Nodes in the last level are filled from left to right.

A heap always satisfies this property.

Definition 2: Height of a Node

The height of a node is defined as the number of edges on the longest downward path from that node to a leaf.

Thus:

- Leaves have height 0.
- A node whose children are leaves has height 1.
- The root has the maximum height in the heap.

Goal Restated Precisely

Let:

- n be the total number of nodes in the heap,
- $h \geq 0$ be a fixed integer,
- N_h be the number of nodes whose height is exactly h.

We must prove:

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

Core Insight of the Proof

The proof relies on counting arguments based on subtree sizes.

A node of height h must have a sufficiently large subtree below it. Because the heap is complete, such subtrees cannot be arbitrarily small.

Step 1: Minimum Size of a Subtree of Height h

Consider any node v of height h.

By definition, the longest downward path from v to a leaf has length h. Therefore, the subtree rooted at v has at least $h + 1$ levels.

Lemma 1

The minimum number of nodes in a binary tree of height h is:

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

For the purpose of deriving an upper bound, we may use the weaker inequality that each such subtree contains at least 2^{h+1} nodes. Replacing $2^{h+1} - 1$ by 2^{h+1} simplifies the counting argument without affecting the correctness of the bound, since the resulting inequality remains valid.

Justification

The smallest tree of height h is a perfect binary tree. Such a tree has:

- 1 node at level 0,
- 2 nodes at level 1,
- 2^h nodes at level h.

Thus, the total number of nodes is:

$$1 + 2 + 4 + \cdots + 2^h = 2^{h+1} - 1.$$

Simplification for Counting

Since:

$$2^{h+1} - 1 \geq 2^{h+1}/2,$$

we use the weaker but sufficient bound:

$$\text{Subtree size} \geq 2^{h+1}.$$

This simplifies the algebra while preserving correctness.

Step 2: Disjointness of Subtrees

Consider all nodes of height exactly h.

Key Observation

No node of height h can be an ancestor of another node of height h.

Therefore, the subtrees rooted at nodes of height h are pairwise disjoint. No node in the heap belongs to more than one such subtree.

Step 3: Global Counting Argument

Let N_h be the number of nodes at height h.

Each such node roots a subtree containing at least 2^{h+1} nodes. Hence, the total number of nodes covered by these subtrees is at least:

$$N_h \cdot 2^{h+1}.$$

Since the heap contains only n nodes in total:

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

Step 4: Solving the Inequality

Dividing both sides by 2^{h+1}:

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

Since N_h must be an integer, we take the ceiling:

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

Illustrative Example

For example, in a heap with $n = 15$ nodes, there can be at most $\left\lceil \frac{15}{2^{h+1}} \right\rceil$ nodes of height h. For $h = 2$, this bound gives at most 2 nodes, which is consistent with the structure of a complete binary heap.

Final Result

In a heap with n elements, the number of nodes of height h is at most:

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

As a consistency check, when $h = 0$, the bound yields at most $\left\lceil \frac{n}{2} \right\rceil$ nodes, which matches the fact that in a complete binary tree, at most half of the nodes can be leaves.

Intuition (Conceptual Explanation)

- Nodes closer to the root have larger height.
- Larger height implies larger required subtrees.
- Since the total number of nodes is fixed, only a few nodes can have large height.

Therefore:

- Most nodes in a heap are close to the leaves.
- Very few nodes are near the root.

Importance of This Result

This bound is fundamental in algorithm analysis, especially for:

- Build-Heap,

- Heapify,
- proving that Build-Heap runs in $O(n)$ time.

It allows us to weight the cost of Heapify by the number of nodes at each height.

Observation

This bound captures the intuition that heaps contain many nodes near the leaves and very few near the root. This structural property plays a crucial role in the linear-time analysis of the Build-Heap algorithm.