# Perfect binary trees

# Outline

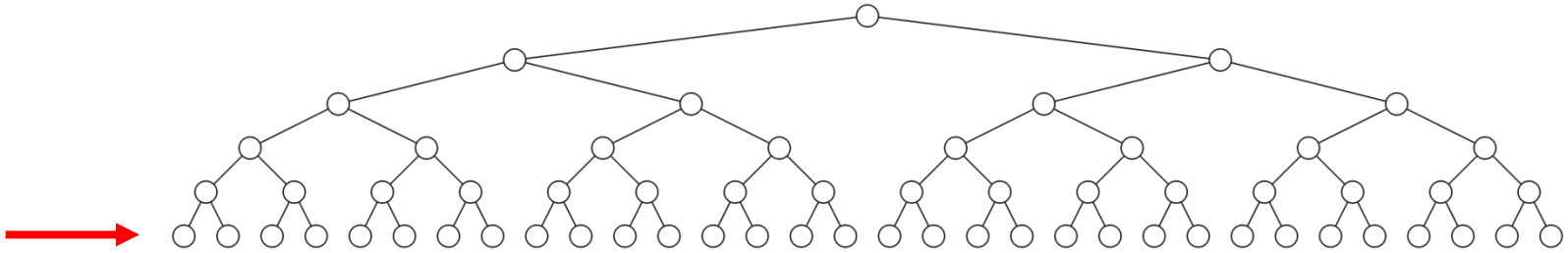
Introducing perfect binary trees

- Definitions and examples
- Number of nodes: $2^{h+1} - 1$
- Logarithmic height
- Number of leaf nodes: $2^h$

# Definition

Standard definition:

- A perfect binary tree of height $h$ is a binary tree where
  - All leaf nodes have the same depth $h$
  - All other nodes are full



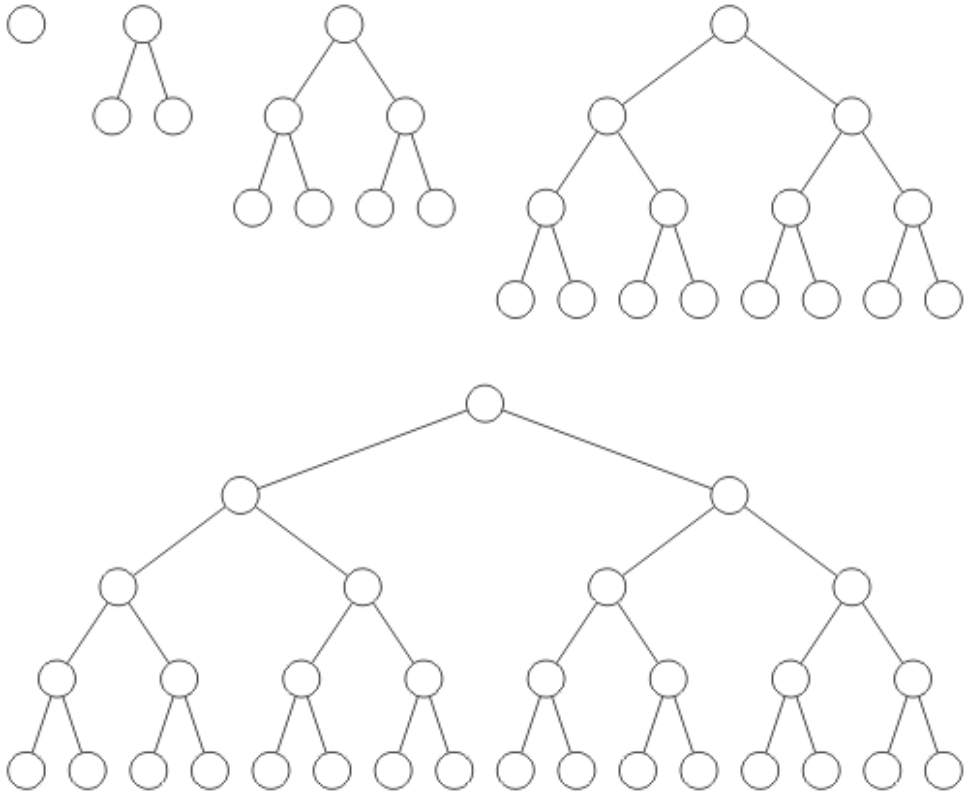
# Definition

Recursive definition:

- A binary tree of height $h = 0$ is perfect
- A binary tree with height $h > 0$ is a perfect if both sub-trees are prefect binary trees of height $h - 1$

# Examples

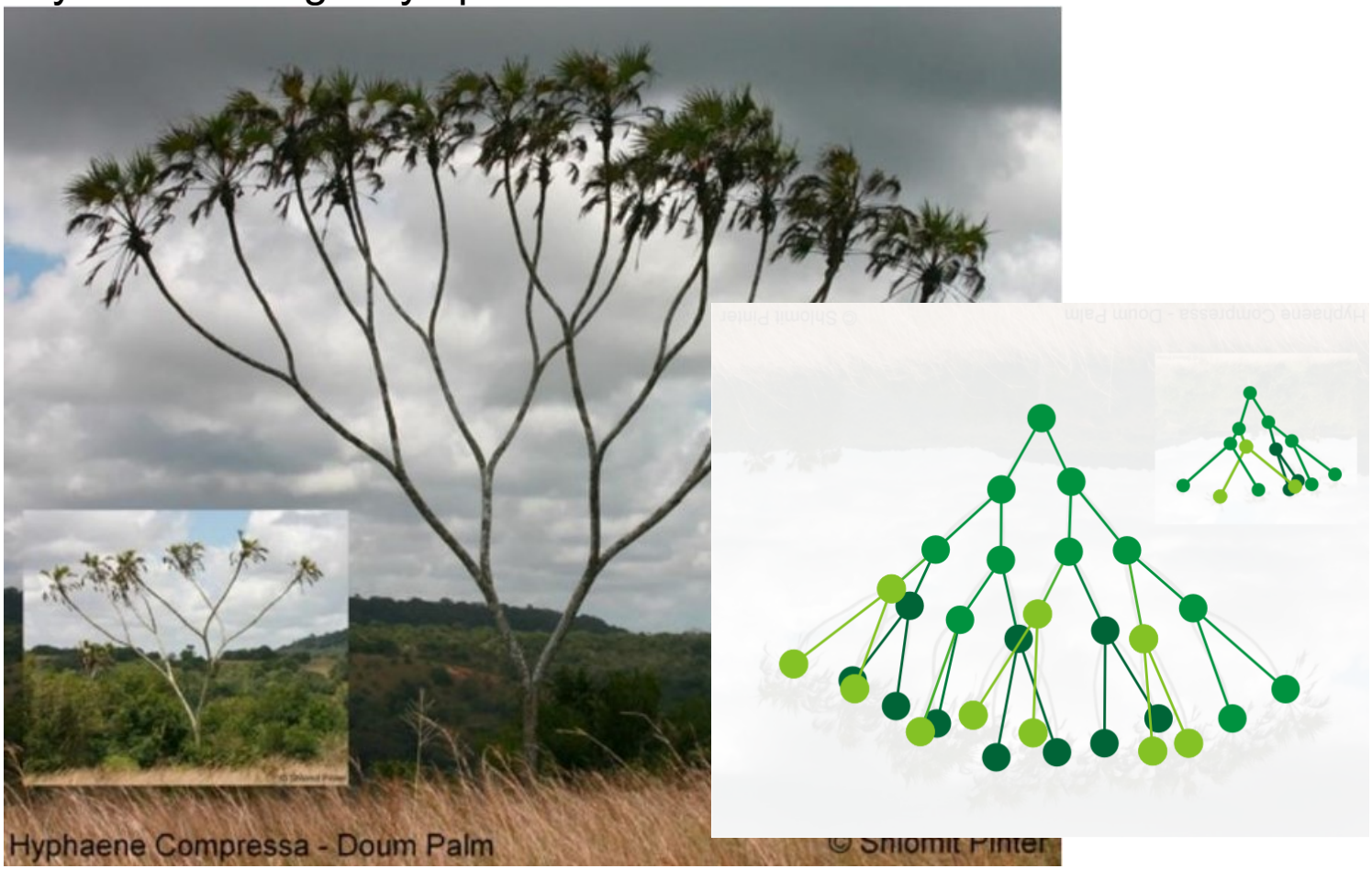
Perfect binary trees of height $h = 0, 1, 2, 3$ and $4$



# Examples

Perfect binary trees of height $h = 3$ and $h = 4$

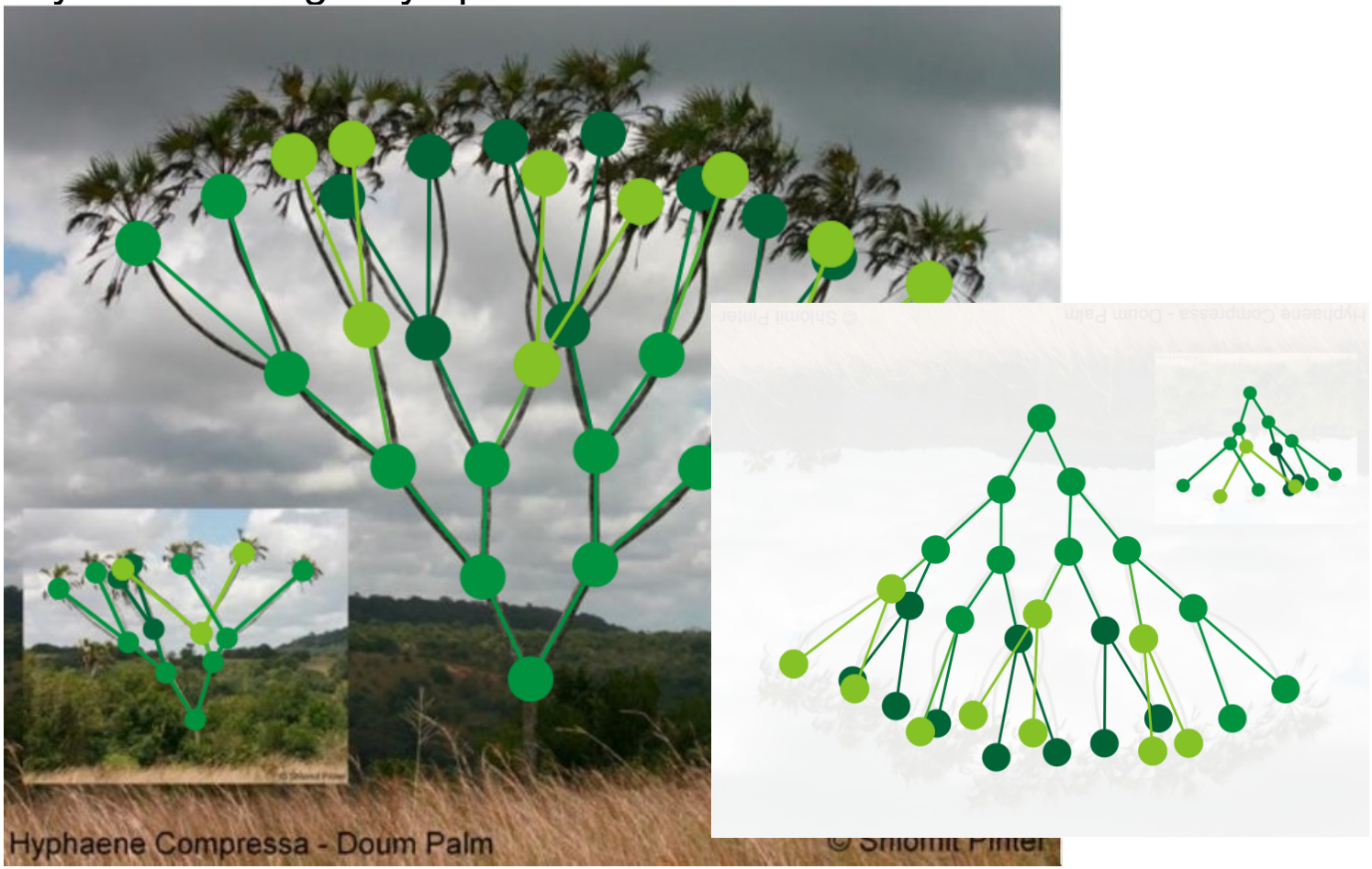
– Note they're the wrong-way up…



# Examples

Perfect binary trees of height $h = 3$ and $h = 4$

– Note they're the wrong-way up…



# Theorems

We will now look at four theorems that describe the properties of perfect binary trees:

- A perfect tree has $2^{h+1} - 1$ nodes
- The height is $\Theta(\log(n))$
- There are $2^h$ leaf nodes
- The average depth of a node is $\Theta(\log(n))$

The results of these theorems will allow us to determine the optimal run-time properties of operations on binary trees

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

Theorem

A perfect binary tree of height $h$ has $2^{h+1} - 1$ nodes

Proof:

We will use mathematical induction:

1. Show that it is true for $h = 0$
2. Assume it is true for an arbitrary $h$
3. Show that the truth for $h$ implies the truth for $h + 1$

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

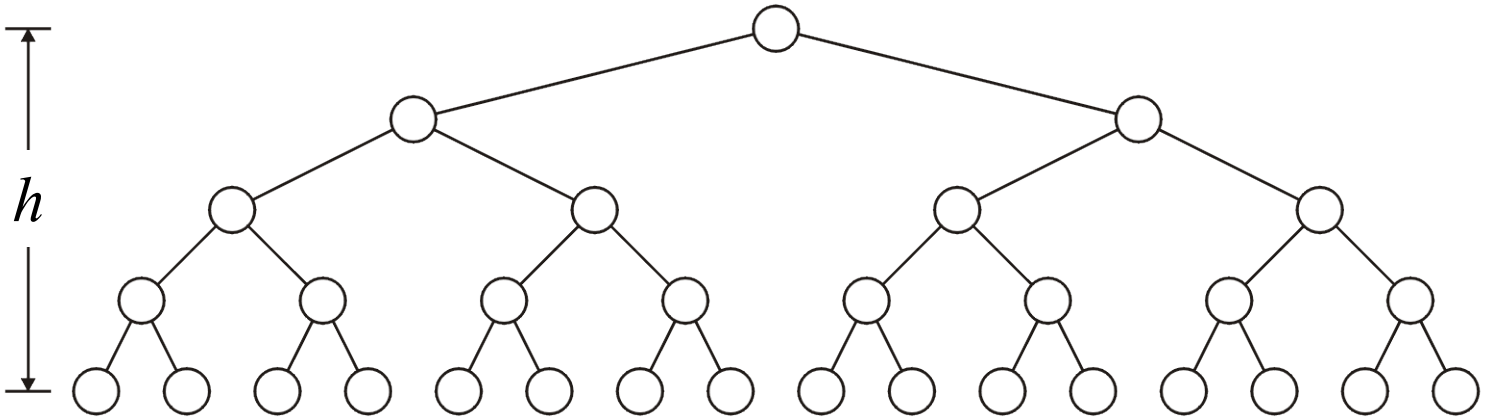
The base case:

- When $h = 0$ we have a single node $n = 1$
- The formula is correct: $2^{0+1} - 1 = 1$

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

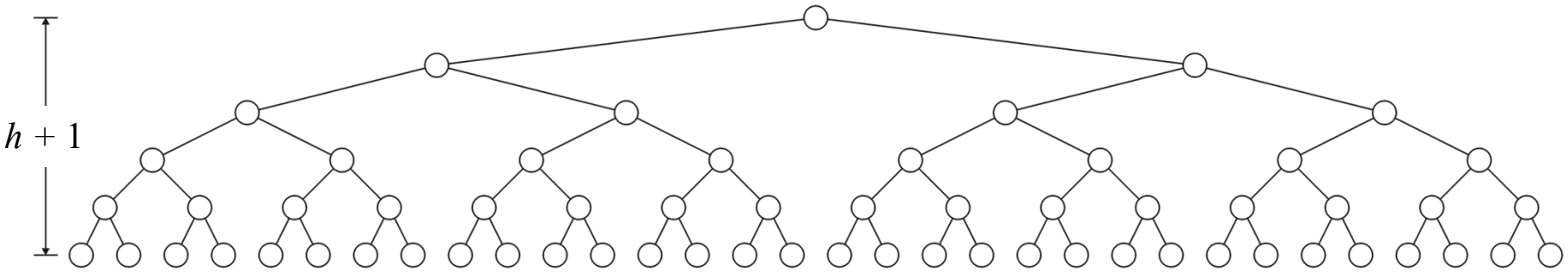
The inductive step:

- If the height of the tree is $h$, then assume that the number of nodes is $n = 2^{h+1} - 1$



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

We must show that a tree of height $h + 1$ has
$n = 2^{(h+1)+1} - 1 = 2^{h+2} - 1$ nodes

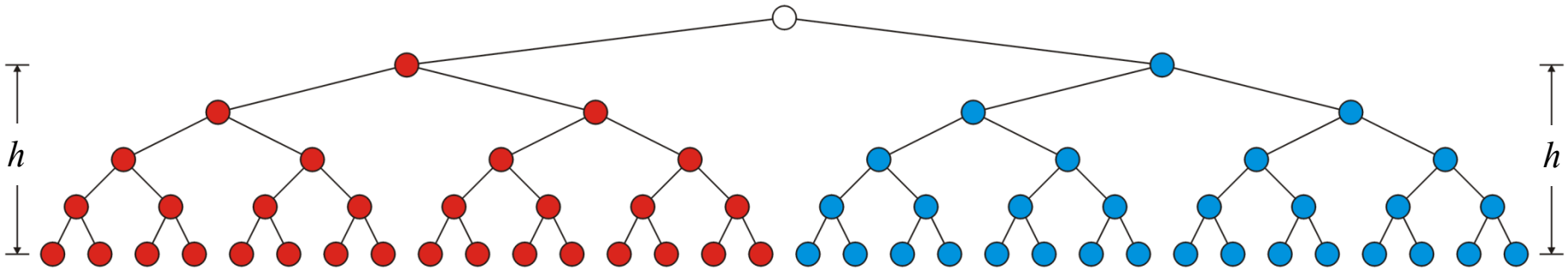


# $2^{h+1} - 1$ Nodes

Using the recursive definition, both sub-trees are perfect trees of height $h$

- By assumption, each sub-tree has $2^{h+1} - 1$ nodes
- Therefore the total number of nodes is

$$(2^{h+1} - 1) + 1 + (2^{h+1} - 1) = 2^{h+2} - 1$$



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

Consequently

The statement is true for $h = 0$ and the truth of the statement for an arbitrary $h$ implies the truth of the statement for $h + 1$.

Therefore, by the process of mathematical induction, the statement is true for all $h \geq 0$

# Logarithmic Height

Theorem

A perfect binary tree with $n$ nodes has height $\log(n + 1) - 1$

Proof

Solving $n = 2^{h+1} - 1$ for $h$:

$$n + 1 = 2^{h+1}$$

$$\log(n + 1) = h + 1$$

$$h = \log(n + 1) - 1$$

# $2^h$ Leaf Nodes

**Theorem**

A perfect binary tree with height $h$ has $2^h$ leaf nodes

Again, we will use induction on the height. When $h = 0$, there is $2^0 = 1$ node and that node is a leaf node.

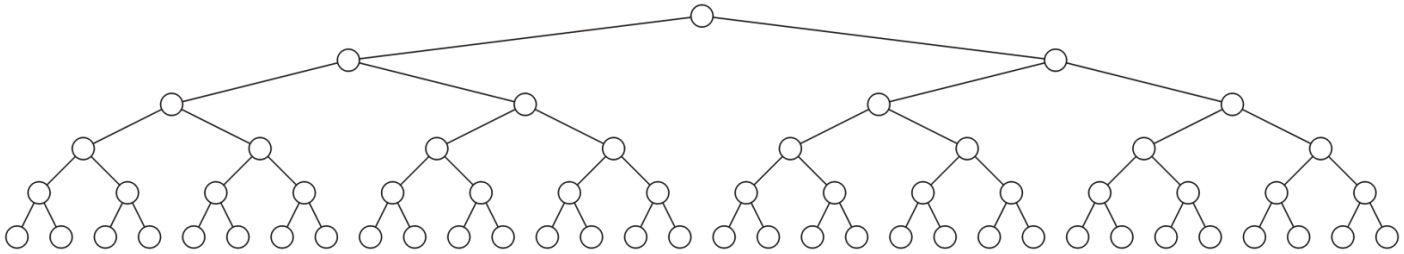
If we assume a perfect tree of height $h$ has $2^h$ leaf nodes, then a perfect tree of height $h + 1$ has two sub-trees, both of which are of height $h$ and both of which have $2^h$ leaf nodes. Therefore, the perfect tree of height $h + 1$ must have $2 \cdot 2^h = 2^{h+1}$ leaf nodes. ∎

# The Average Depth of a Node

Consequence:

- 50-50 chance that a randomly selected node is a leaf node

The average depth of a node in a perfect binary tree is



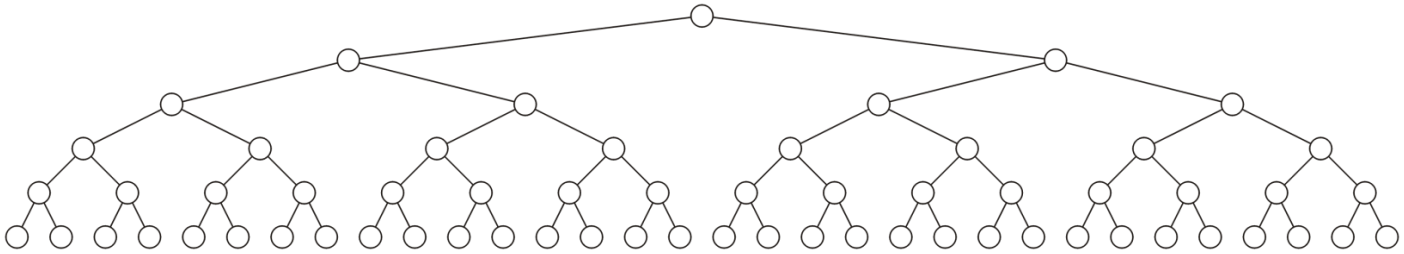
| Depth | Count |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |

# Optional material for proof: The Average Depth of a Node

Consequence:

- 50-50 chance that a randomly selected node is a leaf node

The average depth of a node in a perfect binary tree is



| Depth | Count |
|---|---|
| 0 | 1 |
| 1 | 2 |
| 2 | 4 |
| 3 | 8 |
| 4 | 16 |
| 5 | 32 |

Sum of the depths

Number of nodes

$$\frac{\sum_{k=0}^{h} k2^k}{2^{h+1}-1} = \frac{h2^{h+1}-2^{h+1}+2}{2^{h+1}-1} = \frac{h(2^{h+1}-1)-(2^{h+1}-1)+1+h}{2^{h+1}-1}$$

$$= h-1+\frac{h+1}{2^{h+1}-1} \approx \Theta(\log(n))$$

# Applications

Perfect binary trees are considered to be the *ideal* case

- The height and average depth are both $\Theta(\log(n))$

We will attempt to find trees which are as close as possible to perfect binary trees

# Summary

We have defined perfect binary trees and discussed:

- The number of nodes: $n = 2^{h+1} - 1$
- The height: $\log(n+1) - 1$
- The number of leaves: $2^h$
- Half the nodes are leaves
  - Average depth is $\Theta(\log(n))$
- It is an ideal case

# References

[1] Donald E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd Ed., Addison Wesley, 1998, §7.2.3, p.144.

[2] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill, 1990

[3] Weiss, *Data Structures and Algorithm Analysis.*

[4] Douglas Wilhelm Harder, MMath