Binary Trees and Lower Bound of Sorting

Instructor: Meng-Fen Chiang

COMPCSI220: WEEK 9

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

Slides adapted from Mark Wilson, Georgy Gimel'farb, Simone Linz and Tanya Gvozdeva

Definitions

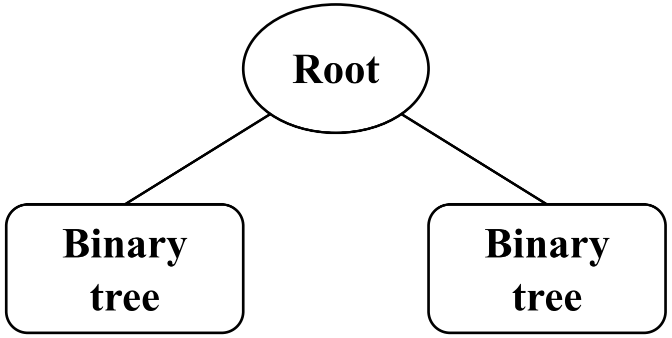
- **Definition (Very-very recursively)**

A binary tree is an object that is either empty or consists of a root node connected to an ordered pair of binary trees.

Possibility 1

Empty tree

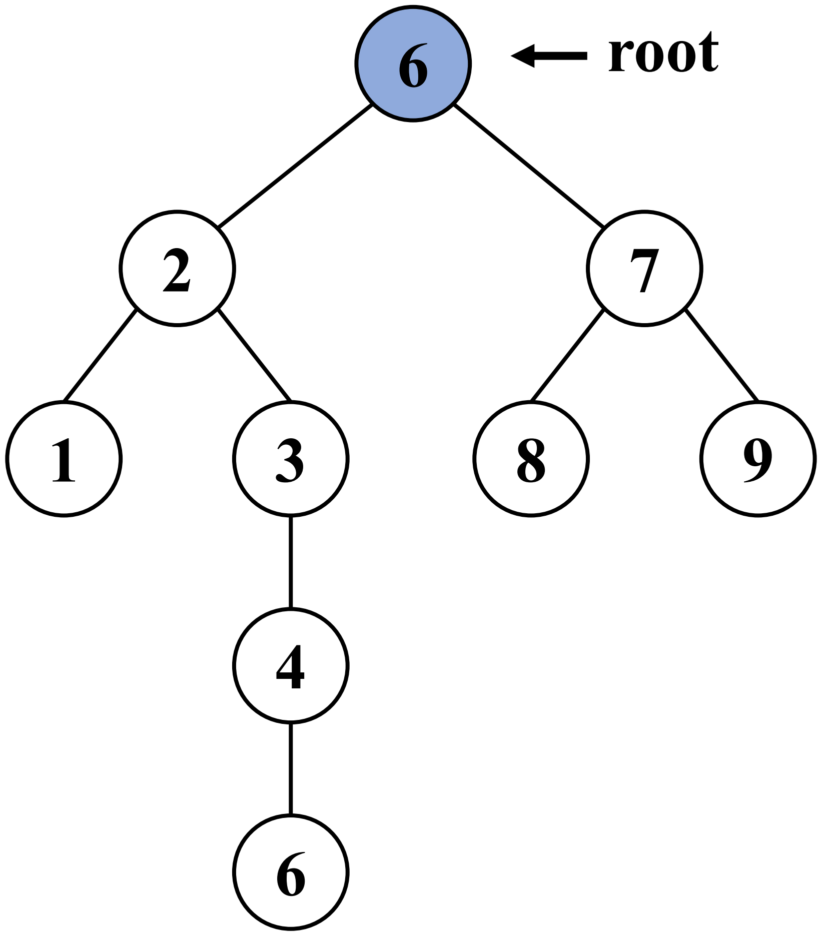
Possibility 2



Definitions (Contd.)

- **Definition (Intuitive definition)**

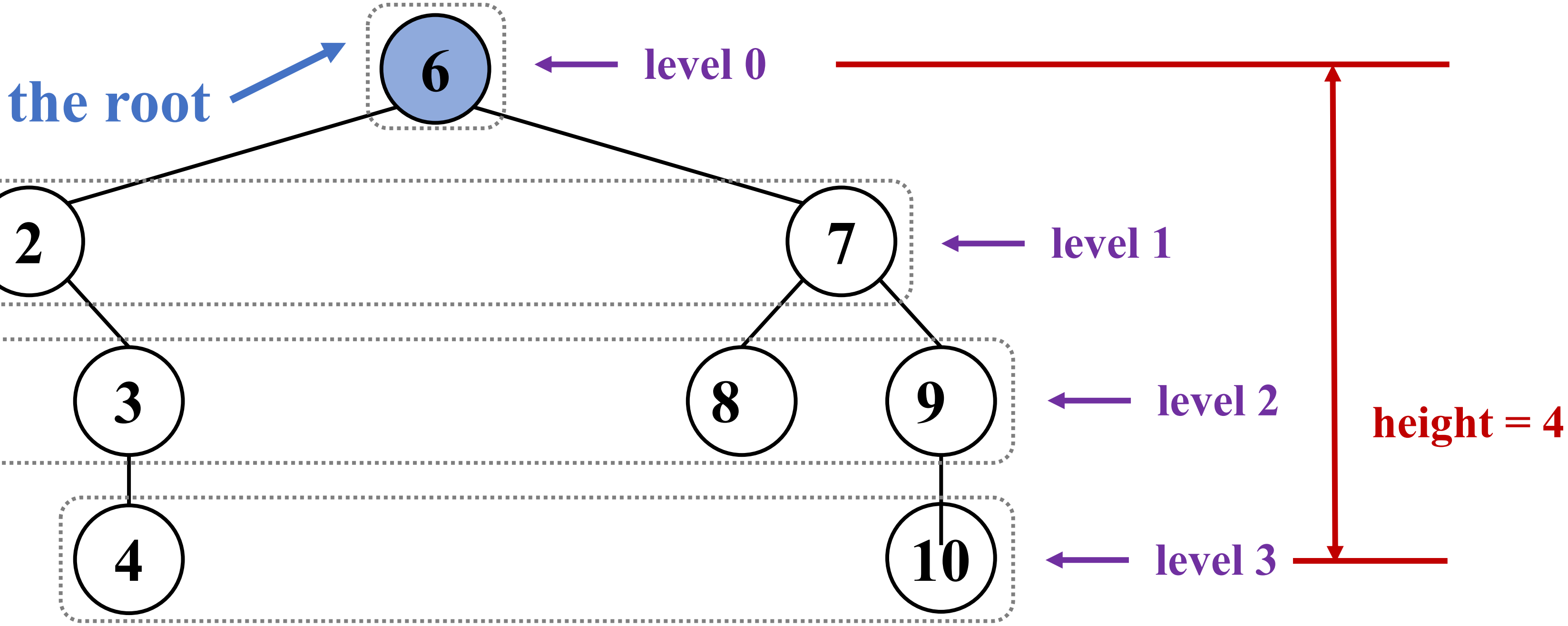
A binary tree is an object that is either empty or consists of nodes, which are connected to either 0,1, or 2 nodes under it. There is one special node called the root.



Definitions (Contd.)

- **Definition (Root and Height)**

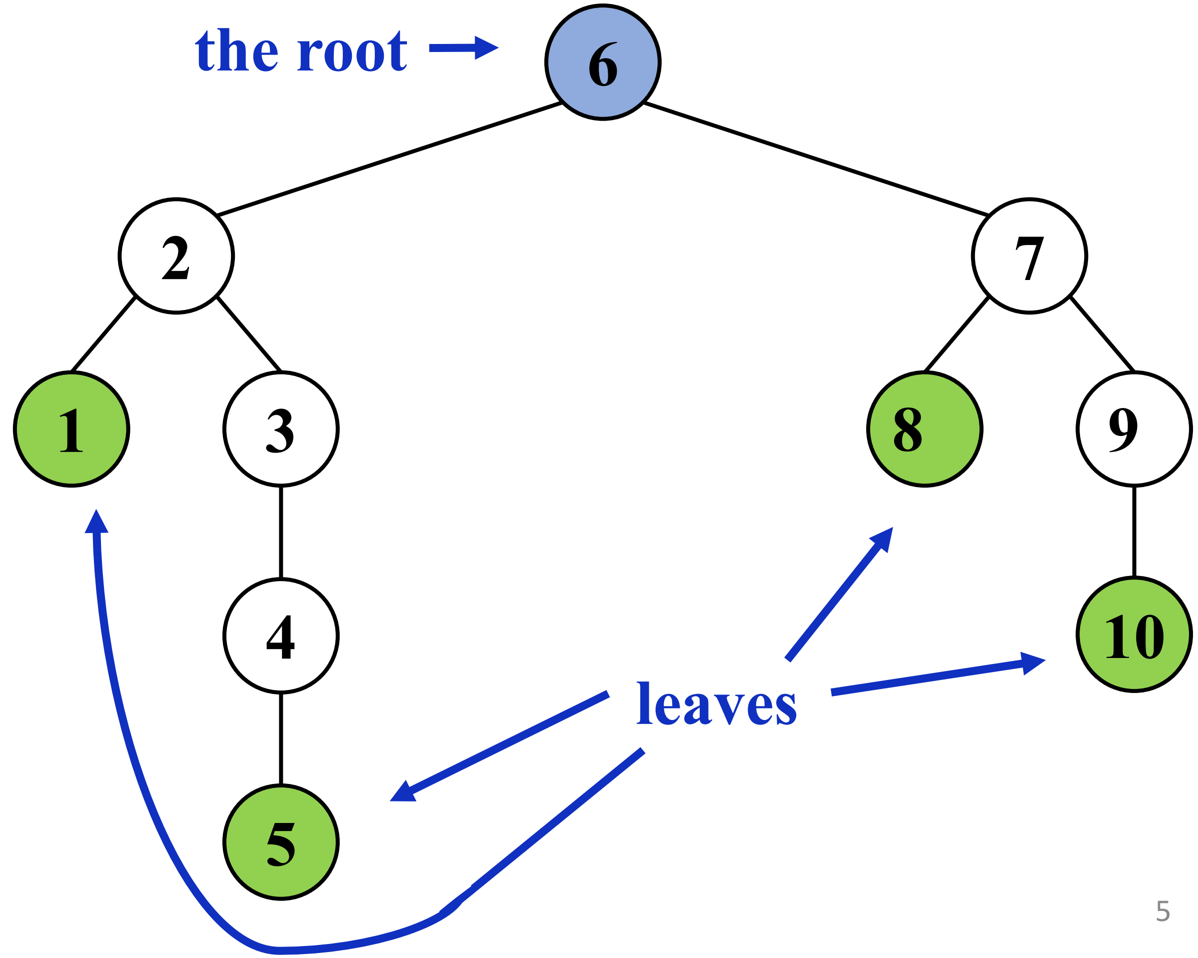
The level of a node is the length of the (unique) path from the root to that node. The height of a binary tree is the maximum level of its nodes.



Definitions (Contd.)

- **Definition (Leaves)**

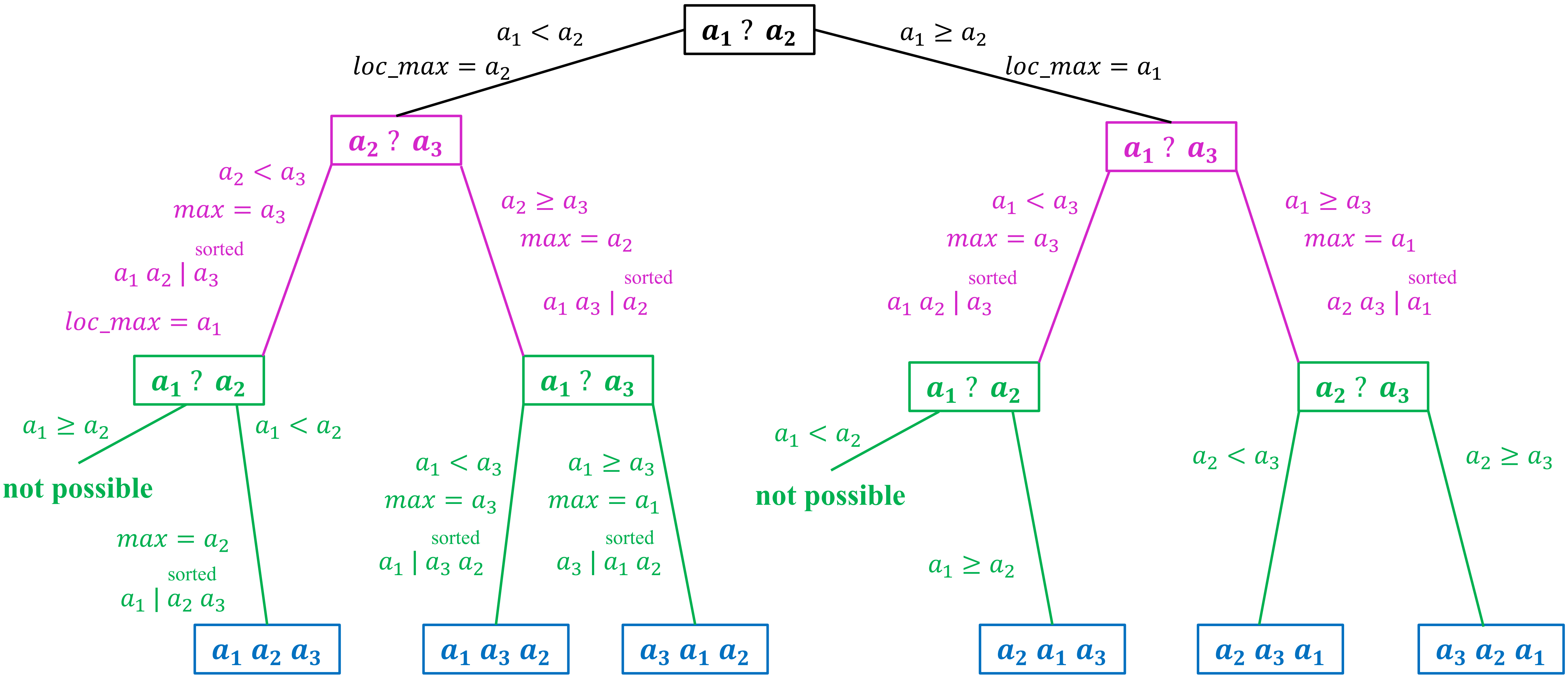
A leaf is a node, which is not connected to any other nodes on the next level.



Decision Tree: Selection Sort

1. Split the input list into sorted and unsorted sublists.
2. Sorted sublist is initially empty, and the unsorted sublist is the whole list.
3. Find a maximal element of the unsorted part by sequential scan.
4. Move the maximal element to the head of the sorted part.
5. If the unsorted sublist is empty, then terminate else go to step 3

Selection sort: $a = [a_1, a_2, a_3]$ max of a: $loc\_max = a_1$



Selection sort: $a = [a_0, \dots, a_{n-1}]$

- The number of leaves is the number of all possible sorted orders on n elements. There are $n!$ ordered lists on n elements.

→ Thus, there are $\boldsymbol{n}!$ leaves

- The level of a leaf is the number of comparisons the algorithm needs to do to achieve this leaf.

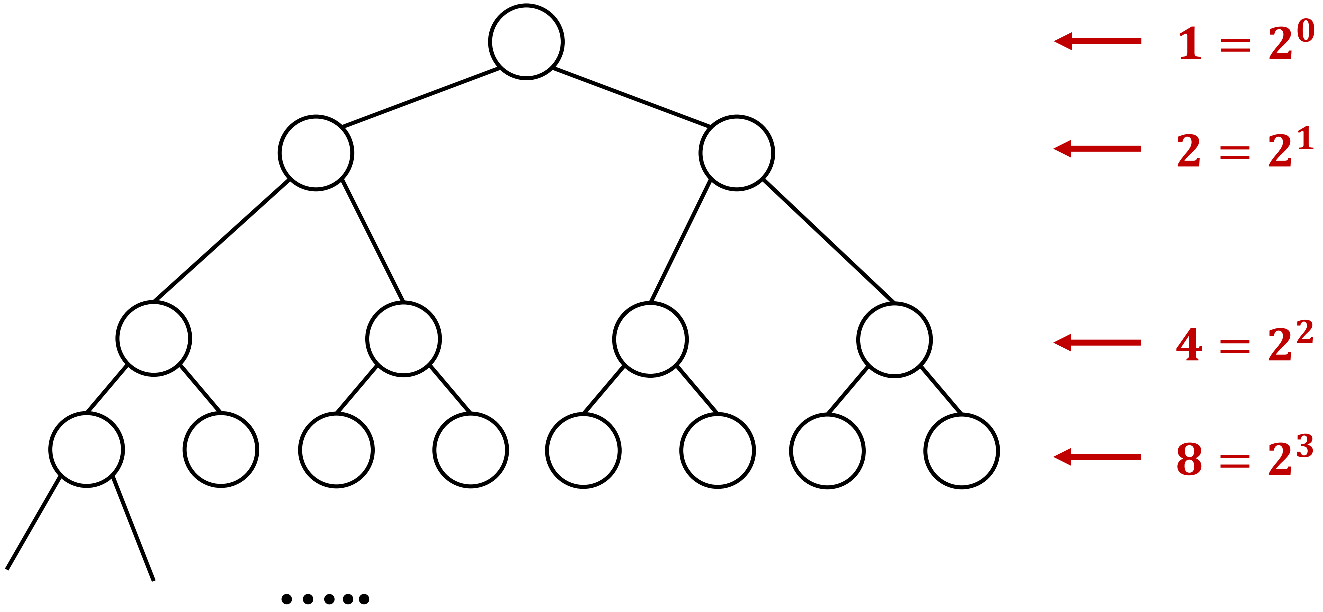
- The height of a decision tree is the max possible number of comparisons you may need to do using this algorithm.

→ The height is the number of comparisons in the worst case

- The runtime in the worst case is at least $\Omega(n \log_2 n)$

The Lower Bound of Sorting

- Any decision tree is a binary tree. So if you want to find the lower bound on the number of comparisons in the worst case, you need to find the smallest possible height of a binary tree with $n!$ leaves.



The Lower Bound of Sorting (Contd.)

- The height is the smallest if every level except for the last one is full.
- If the level i is full then there are 2^i elements.
- The last level contains all leaves
- h is the last level. All leaves could not fit into (h−1) level.

$$2^{h-1} < n!$$

- On the h−th level, there are 2^h nodes and all leaves can be on this level:

$$n! \leq 2^h$$
$$\log_2 n! \leq h$$

- The smallest value for h is $\log_2(n!)$ → $h \in \Omega(n \log_2 n)$

SUMMARY

- Decision Tree
- Lower Bound of Comparison-based Sorting