

# CS2351

# Data Structures

## Lecture 6:

## Lower Bound of Comparison Sorts

# About this lecture

- Lower bound of **any** comparison sorting algorithm
  - applies to insertion sort, selection sort, merge sort, heapsort, quicksort, ...
  - does **not** apply to counting sort, radix sort, bucket sort
- Based on **Decision Tree Model**

# Comparison Sort

- Comparison sort **only** uses **comparisons** between items to gain information about the **relative order** of items
- It's like the elements are stored in boxes, and we can only pick two boxes at a time to compare which one is larger, **without knowing their values**



# Worst-Case Running Time

Merge sort and heapsort are the "smartest" comparison sorting algorithms we have studied so far:

worst-case running time is  $\Theta(n \log n)$

Question: Do we have an even smarter algorithm? Say, runs in  $o(n \log n)$  time?

Answer: No! (main theorem in this lecture)

# Lower Bound

Theorem: Any comparison sorting algorithm requires  $\Omega(n \log n)$  comparisons to sort  $n$  distinct items in the worst case

Corollary: Any comparison sorting algorithm runs in  $\Omega(n \log n)$  time in the worst case

Corollary: Merge sort and Heapsort are (asymptotically) **optimal** comparison sorts

# Proof of Lower Bound

The main theorem **only** counts **comparison** operations, so we may assume all other operations (such as moving items) are **for free**

Consequently, any comparison sort can be viewed as performing in the following way:

1. Continuously gather relative ordering information between items
2. In the end, move items to correct positions

We use the above view in the proof

# Decision Tree of an Algorithm

Consider the following algorithm to sort 3 items **A**, **B**, and **C**:

Step 1: Compare **A** with **B**

Step 2: Compare **B** with **C**

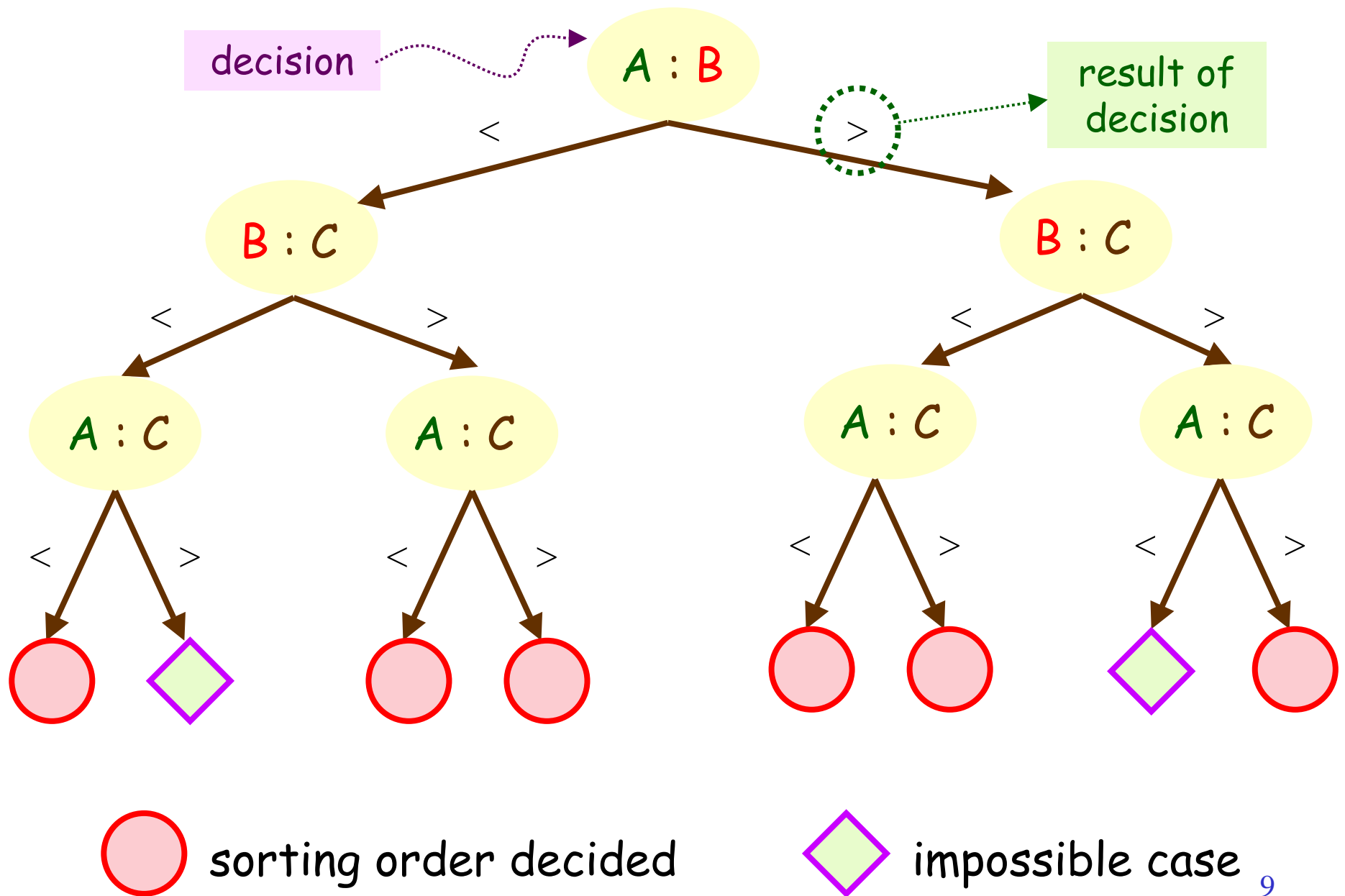
Step 3: Compare **A** with **C**

Afterwards, decide the sorting order of the 3 items

# Decision Tree of an Algorithm

- The previous algorithm always use 3 comparisons, and can sort the 3 items
- In particular, the comparisons used in different inputs (i.e., permutations) can be captured in a **decision tree**, as shown in the next slide:





# Decision Tree of an Algorithm

A cleverer algorithm may sort the 3 items, sometimes, using at most 2 comparisons:

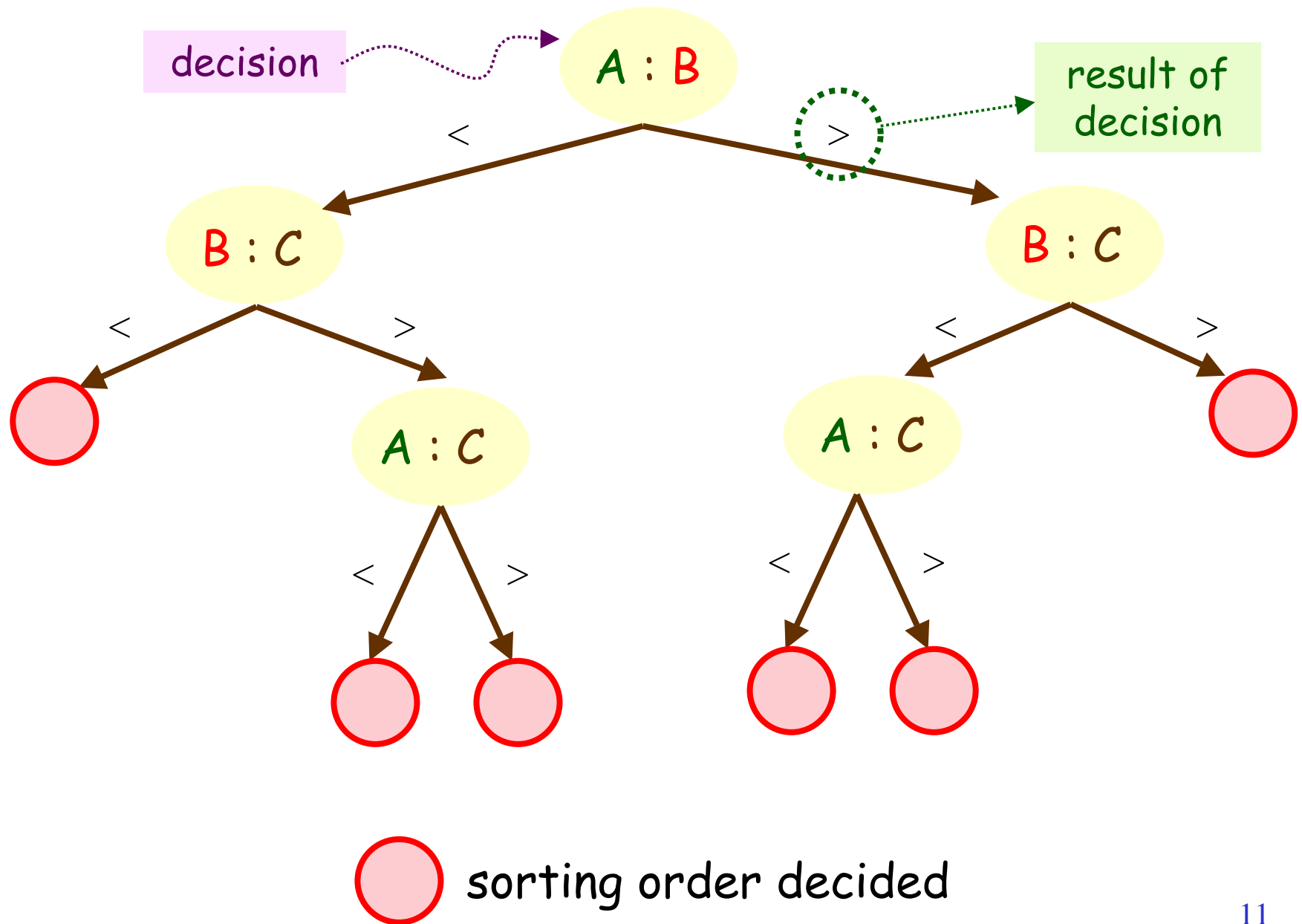
Step 1: Check if  $A > B$

Step 2: Check if  $B > C$

Step 3: Compare  $A$  with  $C$  if the result in Steps 1 and 2 are different

Afterwards, decide the sorting order

- Then, the decision tree becomes ...



# Properties of Decision Tree

In general, assume the input has  $n$  items

Then, for **ANY** comparison sort algorithm:

- Each of the  $n!$  permutations corresponds to a distinct leaf in the decision tree
- The **height** of the tree is the worst-case # of comparisons for any input

**Question:** What can be the **height** of the decision tree of the cleverest algorithm?

# Lower Bound on Height

- There are  $n!$  leaves [for any decision tree]
- Degree of each node is at most 2
- Let  $h$  = node-height of decision tree

So,  $n!$  = total # leaves  $\leq 2^h$

$$\begin{aligned}\rightarrow h &\geq \log(n!) = \log n + \log(n-1) + \dots \\ &\geq \log n + \dots + \log(n/2) \\ &\geq (n/2) \log(n/2) = \Omega(n \log n)\end{aligned}$$

We can also use Stirling's approximation:

$$n! = \sqrt{2\pi n} (n/e)^n (1 + \Theta(1/n))$$

# Proof of Lower Bound

Conclusion:

worst-case # of comparisons

= node-height of the decision tree

=  $\Omega(n \log n)$  [for any decision tree]

→ Any comparison sort, even the cleverest one, needs  $\Omega(n \log n)$  comparisons in the worst case