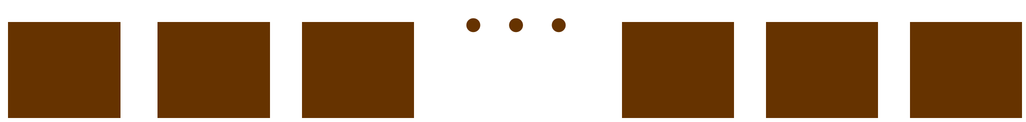
Chapter 8-1: Lower Bound of Comparison Sorts

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



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 Ω(n log n) comparisons to sort n distinct items in the worst case
- Corollary: Any comparison sorting algorithm runs in Ω(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:
  - Continuously gather relative ordering information between items
  - 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:

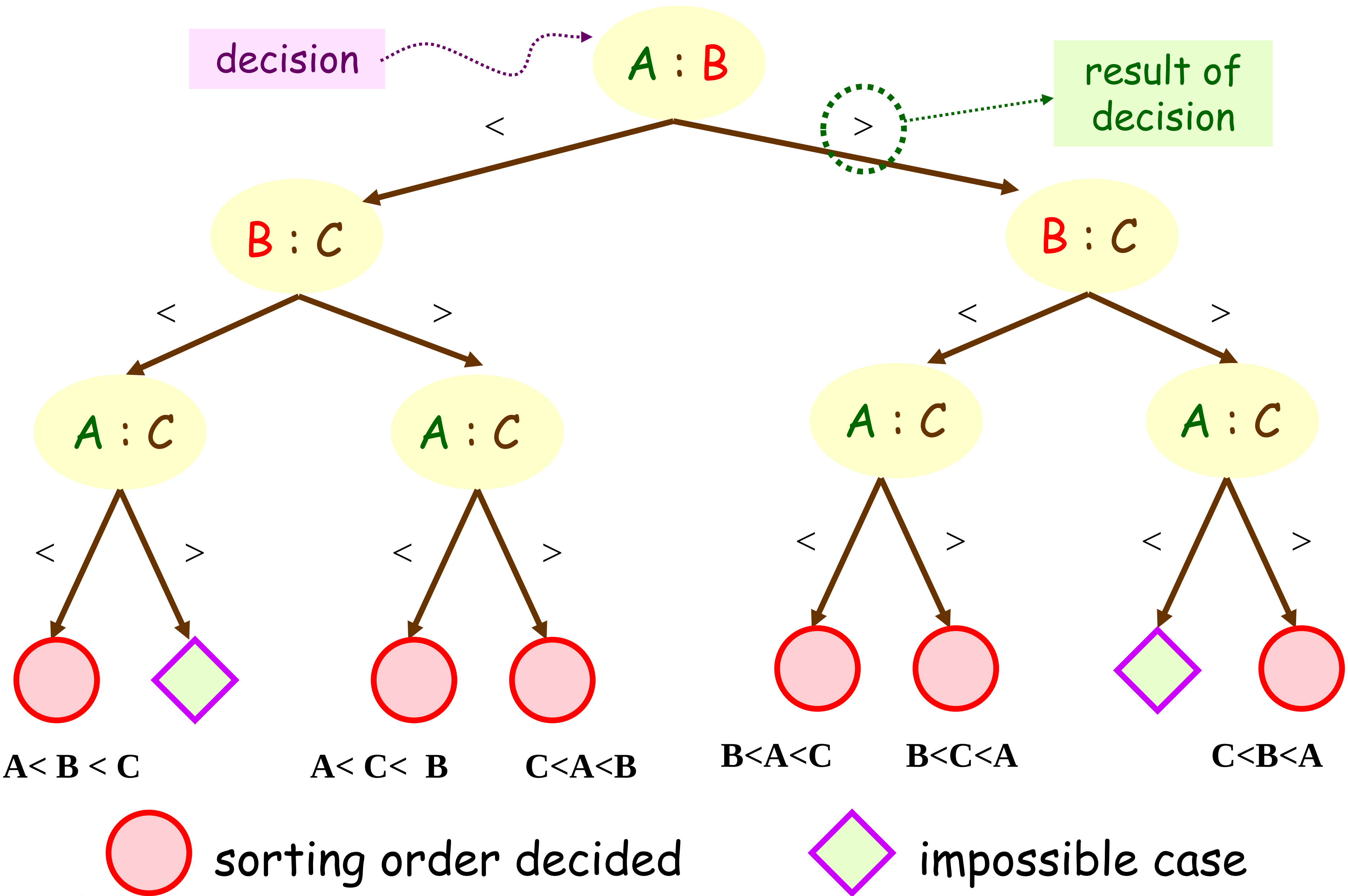
  Step 1: Compare A with B
  Step 2: Compare B with C
  Step 3: Compare A with C

- Afterwards, decide the sorting order of the 3 items

Decision Tree of an Algorithm

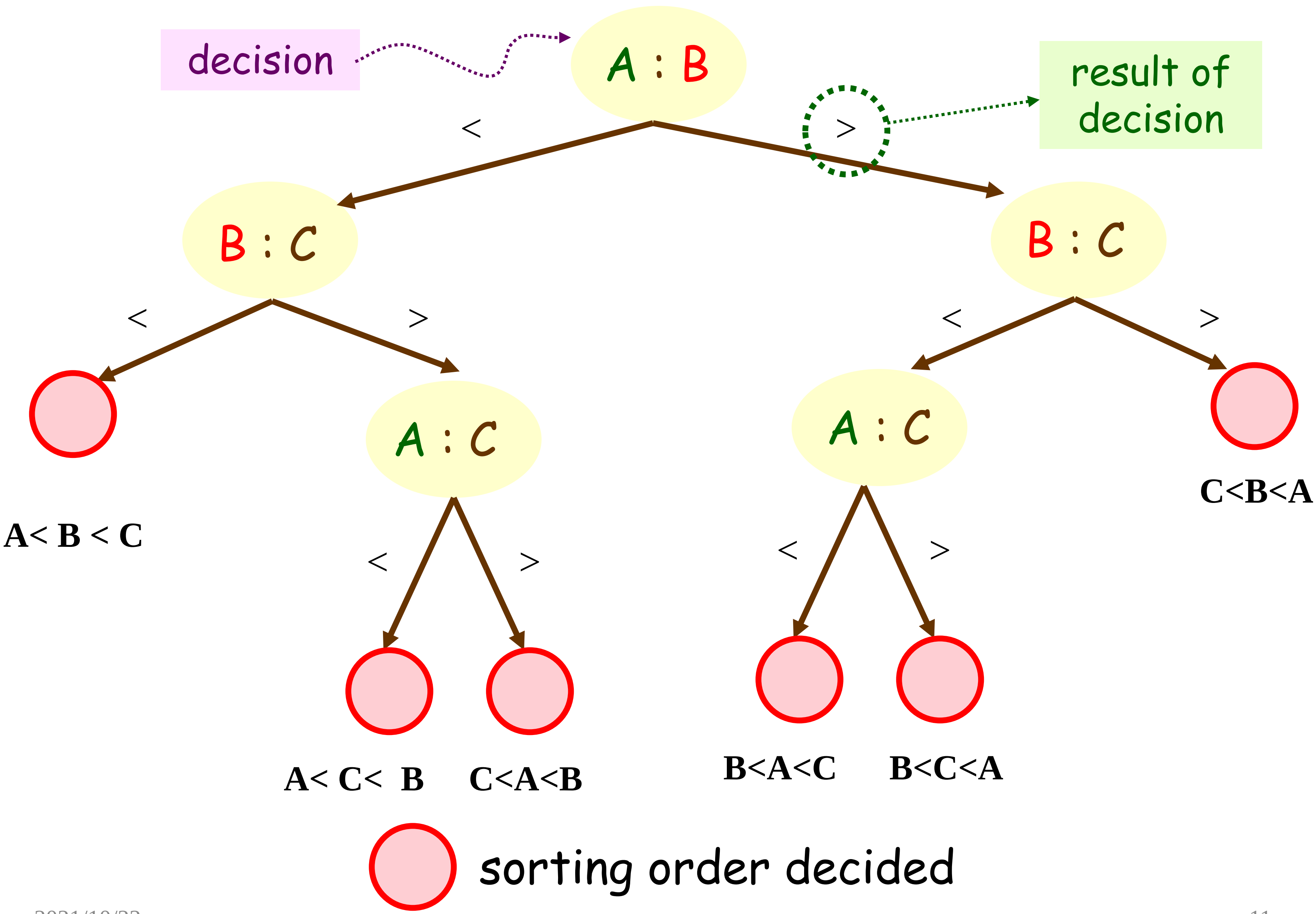
- The previous algorithm always use 3 comparisons, and can sort the 3 items

- In particular, the comparisons used in different inputs (i.e., permutations) can be captured in a decision tree, as shown in the next slide:

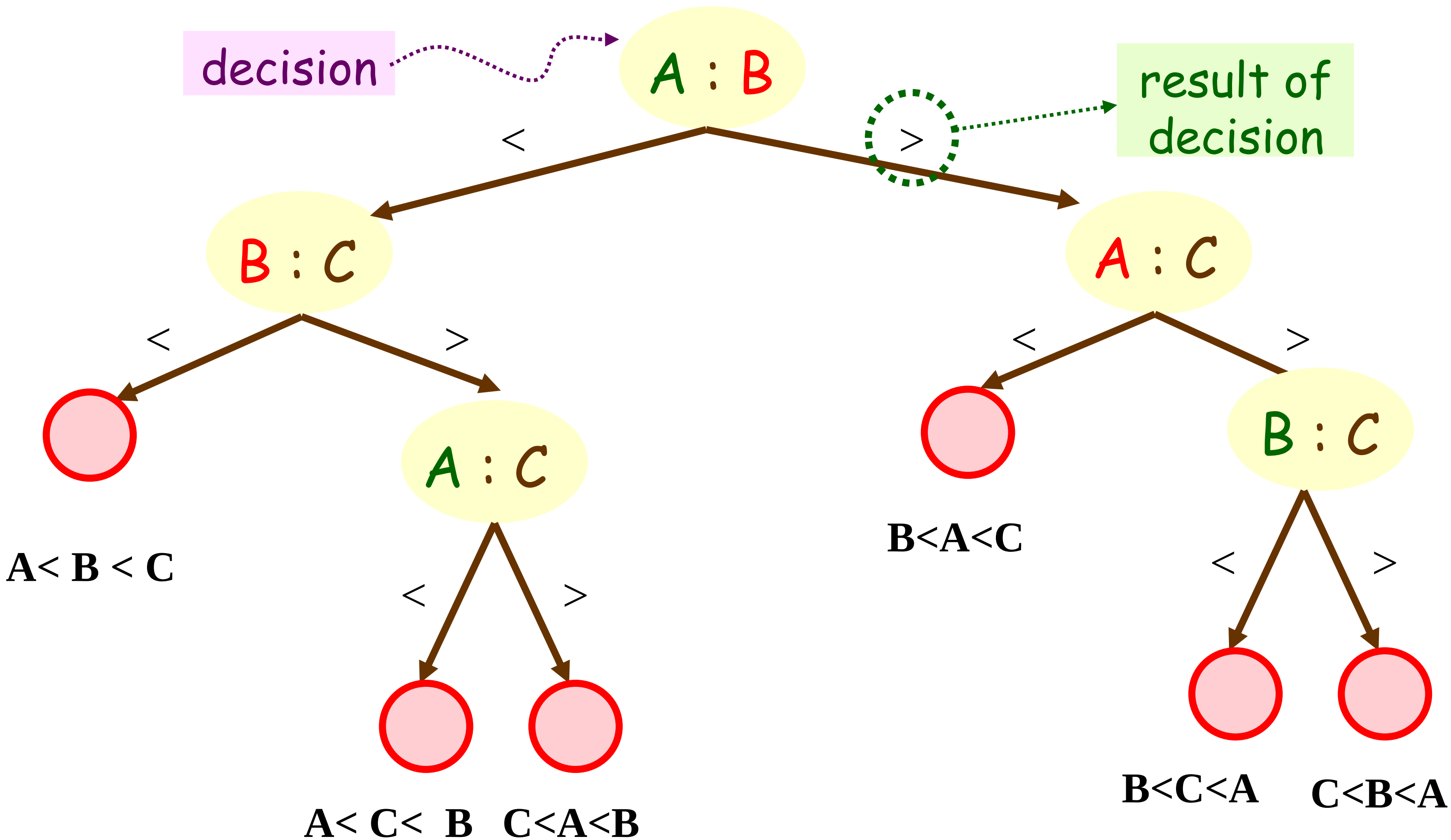

decision
A : B
result of decision
<
>
B : C
B : C
<
>
<
>
A : C
A : C
A : C
A : C
<
>
<
>
<
>
<
>
A< B < C
A< C< B
C<A<B
B<A<C
B<C<A
C<B<A
sorting order decided
impossible case

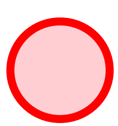
Decision Tree of an Algorithm

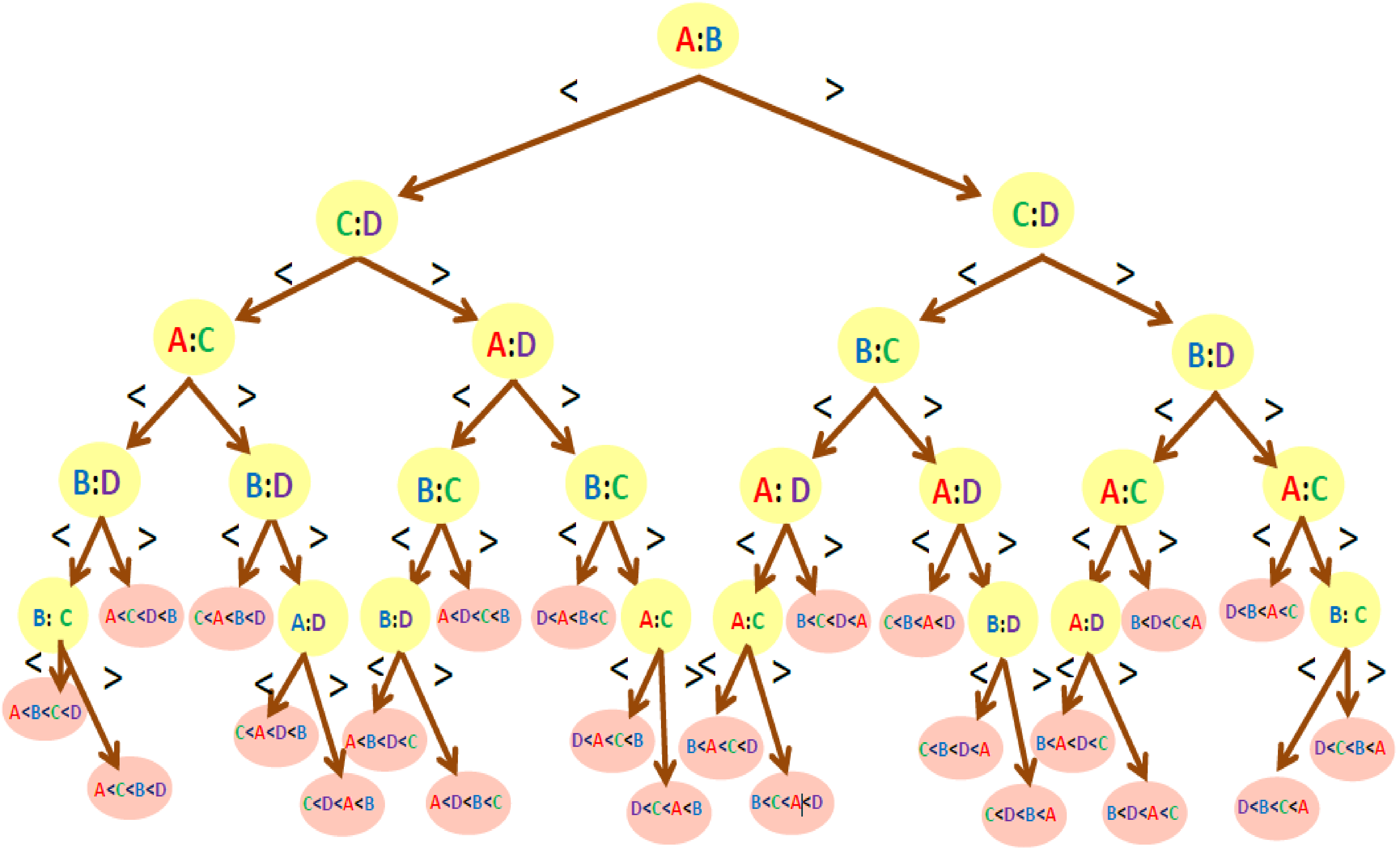
- A cleverer algorithm may sort the 3 items, sometimes, using at most 2 comparisons:
  - ✓ Step 1: Check if A > B
  - ✓ Step 2: Check if B > C
  - ✓ Step 3: Compare A with C if the result in Steps 1 and 2 are different
- Afterwards, decide the sorting order
- Then, the decision tree becomes …


decision
A : B
result of decision
<
>
B : C
B : C
<
>
<
>
A : C
A : C
C<B<A
A< B < C
<
>
<
>
A< C< B
C<A<B
B<A<C
B<C<A
sorting order decided

The decision tree for Insertion sort




sorting order decided



Look like merge sort? Can you modify this decision tree to be a decision tree of merge sort?

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 $\leq 2^h$

➔ $h \geq \log(n!) = \log n + \log(n-1) + \ldots$

$\geq \log n + \ldots + \log(n/2)$

$\geq (n/2) \log(n/2) = \Omega(n \log n)$

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

  = Ω(n log n) [for any decision tree]

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

→ Heapsort and merge sort are asymptotically optimal comparison sorts

Practice at Home

- Exercises: 8.1-1, 8.1-3, 8.1-4
- Please give a merge sort decision tree with four elements *a*, *b*, *c*, and *d*.
- Please give a decision tree for insertion sort operating on four elements *a*, *b*, *c*, and *d*.