CS4311
Design and Analysis of Algorithms

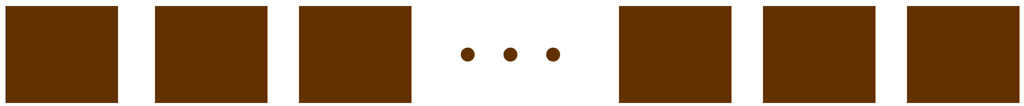
Lecture 7:
Lower Bound of Comparison Sorts

About this lecture

- Prove lower bound of any comparison sorting algorithm
  - applies to insertion sort, selection sort, mergesort, 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
- However, we don't know 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

Proof of Lower Bound

Now, consider a particular comparison sort algorithm *C*, running on some input *A*[1..n]

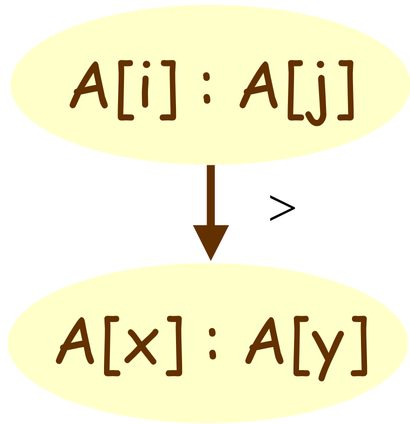
- At the beginning, *C* will make a decision to compare some items, say *A*[i] with *A*[j]

C's first decision $A[i] : A[j]$

Proof of Lower Bound

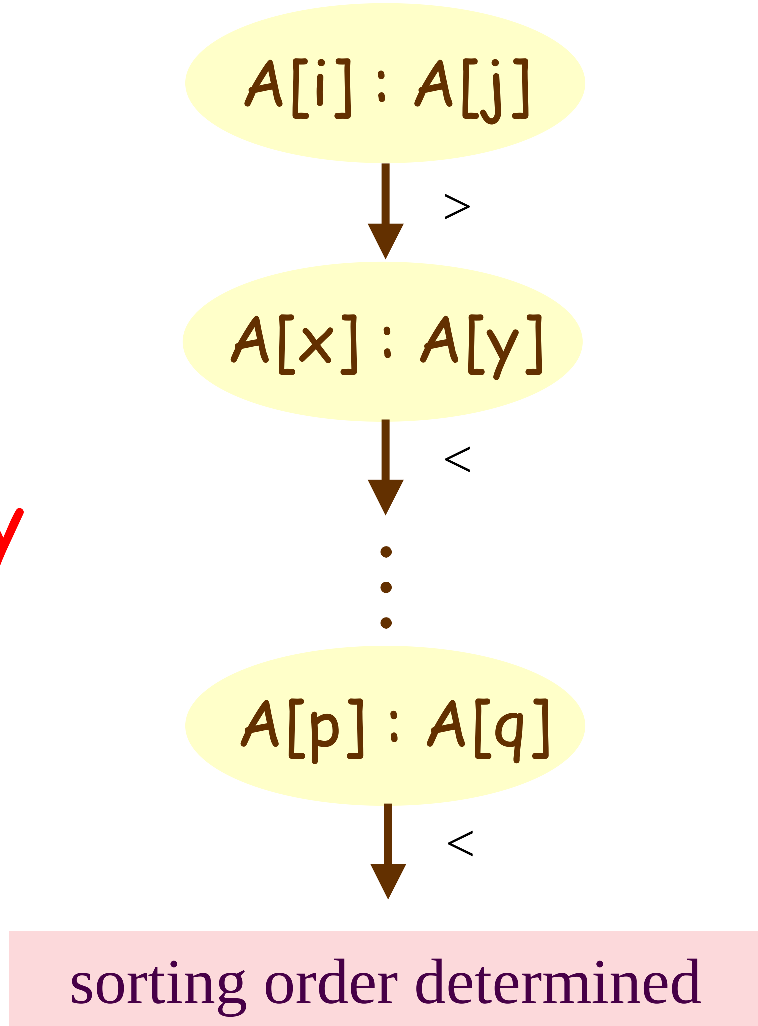
- Suppose $A[i] > A[j]$. C will then make another decision, say, to compare $A[x]$ with $A[y]$

C's second decision



Proof of Lower Bound

- The process continues until there is enough information to determine exactly the sorting order



Proof of Lower Bound

Suppose that the content of A[1..n] is changed and C is run again on this A[1..n]

Question: Which two items are compared at the beginning? Why?

Answer: A[i] and A[j]. C has no way to tell the differences between the current input and the previous one

Proof of Lower Bound

Now, suppose $A[i] > A[j]$ in this new $A[1..n]$

Question: Which two items are compared next? Why?

Answer: $A[x]$ and $A[y]$. For similar reason, C cannot tell the differences between the current input and the previous one

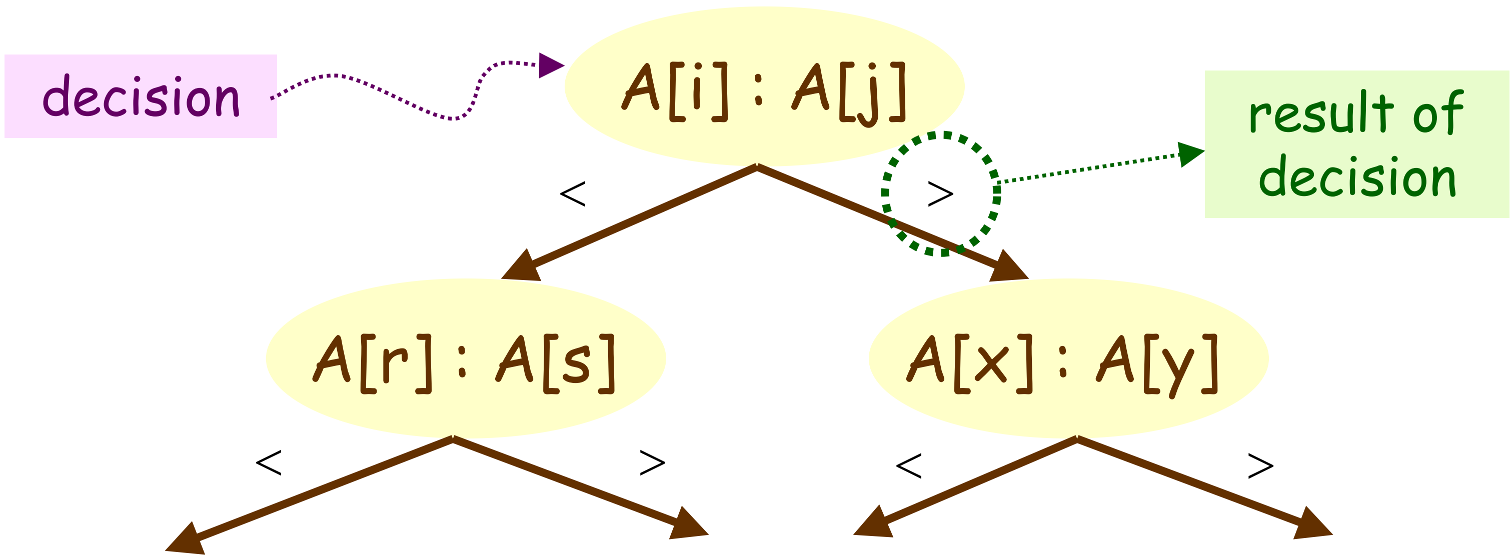
Proof of Lower Bound

Extending this idea further, we can obtain an important observation:

> If the sequence of previous decisions and the corresponding results are the same, *C* will always make the same decision next

Proof of Lower Bound

If we consider running *C* on all different kinds of inputs, the possible sequences of decisions can be captured by a tree:



This is called the decision tree of the algorithm *C*

Properties of Decision Tree

1. Each leaf of a decision tree corresponds to at most one kind of input (why?)
2. The height of the tree is the maximum # of comparisons for any kind of input using the algorithm → worst-case comparisons

Question: Any lower bound on the height?

Lower Bound on Height

- We have $n!$ different kinds of inputs (why?)
- Degree of each node is at most 2
- Let h = height of decision tree of C

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

$$\Rightarrow \quad h \geq \log(n!) = \log n + \log(n-1) + \ldots$$
$$\geq \log n + \ldots + \log(n/2)$$
$$\geq (n/2)\log(n/2) = \Omega(n \log n)$$

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 in *C*
= height of decision tree of *C*
= Ω(n log n)

We have made no special assumptions on *C* except it is a comparison sort

→ Lower bound is true for any comparison sort (so, the proof completes)