Lecture 15: Lower Bounds for Comparison-Based Sorting

Agenda:

- Two useful trees in algorithm analysis (recall)
  - Recursion tree
  - Decision tree
- Decision tree sorting lower bound

Reading:

- Textbook pages 165 – 168

Two useful trees in algorithm analysis:

- Recursion tree
  - node $\longleftrightarrow$ recursive call
  - describes algorithm execution for one particular input by showing all calls made
  - one algorithm execution $\longleftrightarrow$ all nodes (a tree)
  - useful in analysis:
    sum the numbers of operations over all nodes

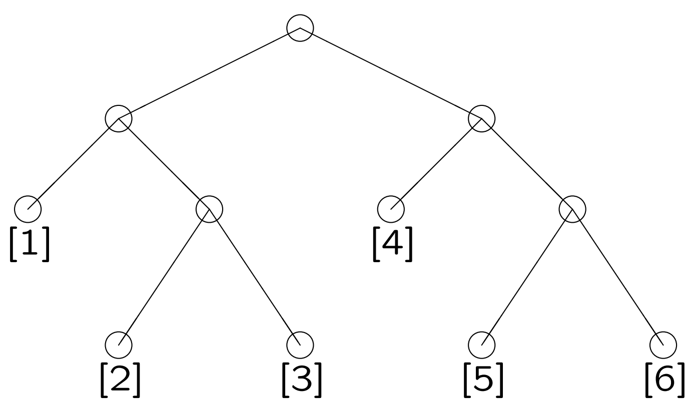
Recursion tree example:

- Mergesort pseudocode

```
Merge(A; lo, mid, hi)    **p 29
        **pre-condition:  lo ≤ mid ≤ hi
        **pre-condition:  A[lo, mid] and A[mid + 1, hi] sorted
        **post-condition:  A[lo, hi] sorted


MergeSort(A; lo, hi)    **p 32

    if lo < hi then
        mid ← ⌊(lo + hi)/2⌋
        MergeSort(A; lo, mid)
        MergeSort(A; mid + 1, hi)
        Merge(A; lo, mid, hi)
```



- For different input instance, the number of operations at each node could be different.

Two useful trees in algorithm analysis:

- Recursion tree
  - node $\longleftrightarrow$ recursion call
  - describes algorithm execution for one particular input by showing all calls made
  - one algorithm execution $\longleftrightarrow$ all nodes (a tree)
  - useful in analysis:
    sum the numbers of operations over all nodes

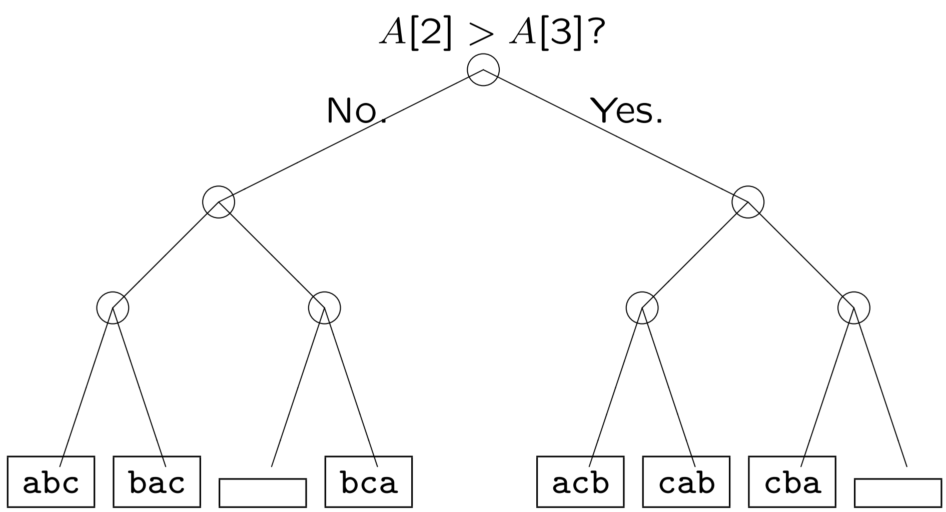
- Decision tree
  - node $\longleftrightarrow$ algorithm decision
  - describes algorithm execution for all possible inputs by showing all possible algorithm decisions
  - one algorithm execution $\longleftrightarrow$ one root-to-leaf path
  - useful in analysis:
    sum the numbers of operations over nodes on one path

Selectionsort decision tree:

- Assume input keys in array $A[1..3] = \{a, b, c\}$
- Tree node: if $A[k] > A[j]$ — 2-way key comparison
- Node label $A[j]$

```
SelectionSort(A; n)

if n ≥ 1 then
    for j ← n downto 2 do
        psn ← j
        for k ← j − 1 downto 1 do
            if A[k] > A[psn] then
                psn ← k
        exchange A[j] ↔ A[psn]
    return
```



- In every case — whatever input instance is, 3 KC !!!

Sorting lower bound:

- Comparison-based sort:

  keys can be (2-way) compared only !

- This lower bound argument considers only the comparison-based sorting algorithms. For example,
  - Insertionsort, Mergesort, Heapsort, Quicksort
  - *Selectionsort*, *Bubblesort*

- Binary tree facts:
  - Suppose there are t leaves and k levels. Then,
  - $t \leq 2^{k-1}$
  - So, $\lg t \leq (k-1)$
  - Equivalently, $k \geq 1 + \lg t$
    — binary tree with t leaves has *at least* $(1 + \lg t)$ levels

- Comparison-based sorting algorithm facts:
  - Look at its *Decision Tree*. We have,
  - It's a binary tree.
  - It should contain every possible permutation of the positions $\{1, 2, \ldots, n\}$.
  - So, it contains at least $n!$ leaves ...
  - Equivalently, it has at least $1 + \lg(n!)$ levels.
  - A longest root-to-leaf path of length at least $\lg(n!)$.
  - The worst case number of KC is at least $\lg(n!)$.
  - $\lg(n!) \in \Theta(n \log n)$

Sorting lower bound (cont'd):

- Key ideas in deriving the lower bound:
  - Decision tree
  - It's binary
  - Length of longest root-to-leaf path $\longleftrightarrow$ WC KC
  - The number of possible permutations $\longleftrightarrow$ number of leaves
- It doesn't hold for non-comparison-based sorting algorithm ...

  Check Chapter 8 for extra reading

Have you understood the lecture contents?

well	ok	not-at-all	topic
☐	☐	☐	recursion tree
☐	☐	☐	decision tree
☐	☐	☐	difference between them
☐	☐	☐	WC running time $\leftrightarrow$ longest path
☐	☐	☐	BC running time $\leftrightarrow$ shortest path
☐	☐	☐	Each leaf is a permutation
☐	☐	☐	Deriving the lower bound