Lower bounds for Sorting (with comparisons)

Cormen Sect 8.1

Sorting algorithms we have seen so far are based on Comparisons $T(n) \simeq \#$ of comparisons

$a_1 \quad a_2 \quad a_3 \quad \dots \quad a_n$ to sort

Question is $a_i \overset{?}{\leq} a_j$

Comparison-based sorting algorithms

Pros : - change the effect by changin cmp function

- work for any kind of object where a cmp function can be defined

Cons - design a faster algorithm without limiting ourself to comparison

Prove that $\Omega(n \log n)$ time is the best we can hope for for comparison-based sorting algorithm

$A = a_1 a_2 a_3 \dots a_n$

Assume wlog that $\forall_{i,j} \; a_i \neq a_j$

Decision-tree model

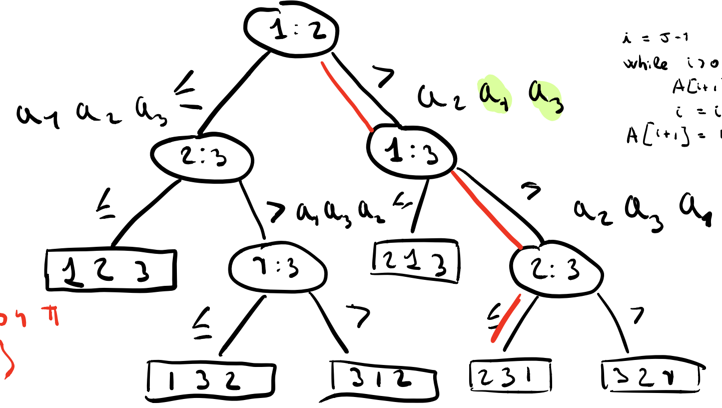
A decision tree is a fully binary tree that reppresent the comparisons performed by a fixed comparison-based sorting algorithm on any input of a fixed size n

Decision tree for insertion sort on $n = 3$

```
InsertionSort(A)      |A| = n
for j = 2 to A.length
        key = A[j]

        i = j - 1
        while i > 0 and A[i] > key
                A[i+1] = A[i]
                i = i - 1
        A[i+1] = key
```

$a_1 \; a_2 \; a_3$



a permutation π of $\{1, 2, 3\}$ that gives the input sorted

- The execution of the algorithm on a input of size n corresponds to a root to leaf path on the tree

e.g. $A =$

7	4	5
1	2	3

$\pi = 2 \quad 3 \quad 1$

A_s

4	5	7

- Running time of the algorithm is Ω (length of path)
  worst case time = Ω (height of tree)

- Any correct sorting algorithm must be able to produce any permutation
  why ? if not , it is wrong
  if $\pi = 1\ 3\ 2$ is not in a leaf
  there exists an input s.t.
  for which the algorithm is incorrect !

- There are at least $n!$ leaves

$$\boxed{\text{Any Sorting algorithm based on comparisons needs } \Omega(n \log n) \text{ comparisons}}$$

- Worst case time = Ω (height of the tree)
  $\Omega(h)$

- $\Omega(n!)$ leaves

- Binary tree of height h has no more than 2^h leaves

$$\boxed{2^h \geq n!} \Rightarrow h \geq \log(n!)$$

$$\Rightarrow h \geq \Omega(n \log n)$$

Exercise (***)

Binary search takes $\Theta(\log n)$ time (and comparisons) to find a key k in a sorted array A of size n.. Is this optimal in the comparison model?

Solution

Decision tree

- Internal nodes contain a index of the array say i, and the result is the comparison between $A[i]$ and k

- Leaves contain a index i from 1 to n (if k is in position i i.e. $A[i]==k$) or -1 (if k is not in A)
- An algorithm cannot be correct if there is a index not in the leaves $\Rightarrow$ # of leave is at least $n+1$
- Running time of the algorithm is Ω(height of the tree) in the worst case

- $2^h \geq n+1 \Rightarrow h = \Omega(\log n)$

if the is ternary

$$3^h \geq n+1 \Rightarrow h = \Omega(\log n)$$

* * *

- You cannot sort, in general, faster than $\Omega(n \log n)$ time with a sorting algorithm based on comparisons
- Assumptions allows you to sort faster
  - Few distinct elements e.g. binary array can be sorted in linear time
    $\Theta(n \log k)$ time to sort a array with k distinct elements