EECS 3101 M: Design and Analysis of Algorithms

Suprakash Datta
Office: LAS 3043

Course page: `http://www.eecs.yorku.ca/course/3101M`
Also on Moodle

Lower Bounds - Intuition

- Can we beat the $\Omega(n \log n)$ lower bound for sorting?
- A: In general no, but in some special cases YES!
- Ch 7: Sorting in linear time
- We will prove the $\Omega(n \log n)$ lower bound.

Lower Bounds - Details

- What are we counting?
  – Running time? Memory? Number of times a specific operation is used?
- What (if any) are the assumptions?
- Is the model general enough?

Here we are interested in lower bounds for the WORST CASE.
So we will prove (directly or indirectly):
for any algorithm for a given problem, for each $n > 0$, there exists an input that make the algorithm take $\Omega(f(n))$ time.
Then $f(n)$ is a lower bound on the worst case running time.

Comparison-based Algorithms

- The algorithm only uses the results of comparisons, not values of elements (*)
- Very general – does not assume much about what type of data is being sorted
- However, other kinds of algorithms are possible!
- In this model, it is reasonable to count #comparisons. Note that the #comparisons is a lower bound on the running time of an algorithm.

(*) If values are used, lower bounds proved in this model are not lower bounds on the running time.

Lower Bound: Observations

- Lower bounds are rarely simple: there are virtually no known general techniques.
- So we must try ad hoc methods for each problem.
- We proved a lower bound on finding the maximum
- Sorting lower bounds:
    - Trivial: $\Omega(n)$ – every element must be in a comparison
    - Best possible result – $\Omega(n \log n)$ comparisons, since we already know several $O(n \log n)$ sorting algorithms
    - Difficulty: how do we reason about all possible comparison-based sorting algorithms?

The Decision Tree Model

Assumptions:

- All numbers are distinct
- All comparisons have form $a_i \leq a_j$ (since $a_i < a_j, a_i \leq a_j, a_i \geq a_j, a_i > a_j$ are equivalent)

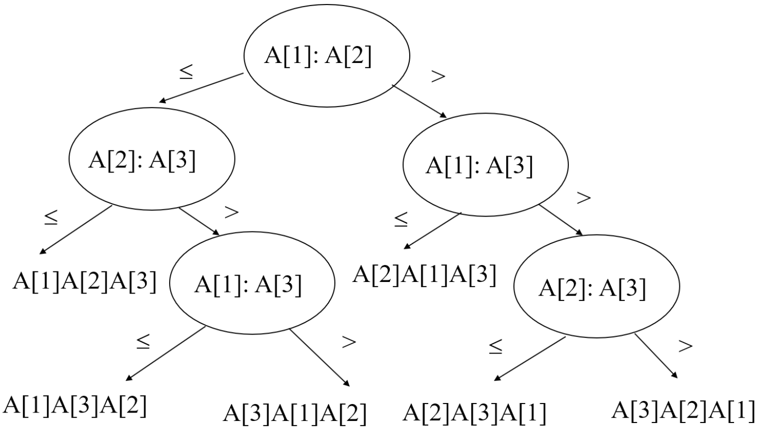
Decision tree structure

- Full binary tree
- Ignore control, movement, and all other operations, just use comparisons.
- suppose three elements $\langle a_1, a_2, a_3 \rangle$ with instance $\langle 6, 8, 5 \rangle$.

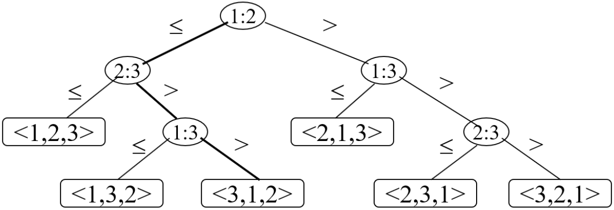
The Decision Tree Model - Example

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1..j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

Insertion Sort: Decision Tree



Insertion Sort: Another view



Internal node $i : j$ indicates comparison between a_i and a_j. Leaf node $\langle p_1, p_2, p_3 \rangle$ indicates ordering $a_{p_1} \leq a_{p_2} \leq a_{p_3}$ Path of bold lines indicates sorting path for $\langle 6, 8, 5 \rangle$. There are total $3! = 6$ possible permutations (paths).

The Decision Tree Model - Summary

- Only consider comparisons
- Each internal node $=$ 1 comparison
- Start at root, make the first comparison
    - if the outcome is $\leq$, take the LEFT branch
    - if the outcome is $>$, take the RIGHT branch
- Repeat at each internal node
- Each LEAF represents ONE correct ordering

Lower Bound on Sorting

- Claim: The decision tree must have at least $n!$ leaves. WHY?
- worst case number of comparisons $=$ the height of the decision tree
- Claim: Any comparison sort in the worst case needs $\Omega(n \log n)$ comparisons
- Suppose height of a decision tree is h, number of paths (i.e., permutations) is $n!$
- Since a binary tree of height h has at most 2^h leaves, $n \leq 2^h$
- So $h \geq \lg n! \in \Omega(n \lg n)$

Lower Bounds: Check your understanding

- Can you prove that any algorithm that searches for an element in a sorted array of size n must have running time $\Omega(\lg n)$?