Lower Bound for Comparison Sort

ESO207

Indian Institute of Technology, Kanpur

Overview

- We have seen several sorting algorithms so far, Heapsort, Mergesort and Quicksort.
- Heapsort and Mergesort can sort n numbers in time $O(n \log n)$ time. Quicksort achieves it on average.
- For each of these inputs, we can produce an input sequence of n numbers for which the algorithm requires $\Omega(n \log n)$ time.
- But could we sort faster?

Comparison Sorts

- These algorithms are **comparison sorts**, that is, the sorted order they determine is based only on comparisons between the input elements.
- We will show that any comparison sort must make $\Omega(n \log n)$ comparisons in the worst case to sort n numbers (elements).
- Thus Heapsort and Mergesort are optimal comparison sort algorithms to within a constant factor.

Comparison Sort

- In a comparison sort, we use only comparisons between elements to gain order information about and input sequence $a_1, a_2, \ldots, a_n$.
- That is, given two elements a_i and a_j, we perform a comparison
  $a_i < a_j, a_i \leq a_j, a_i \geq a_j$, etc. to determine their relative order.
- Assume all elements are distinct.
- So any of the comparisons,
  $a_i \leq a_j, a_i < a_j, a_i \geq a_j, a_i > a_j$ are equivalent and give the same information.

Decision tree model for Comparison Sort Algorithms

- View a comparison sort as a decision tree.
- A **decision tree** is a full binary tree whose nodes represent comparisons between elements that are performed when that particular sorting algorithm runs on input of the given size.
- Control, data movement and other aspects of the algorithm are ignored (a conceptual model).

Decision tree model

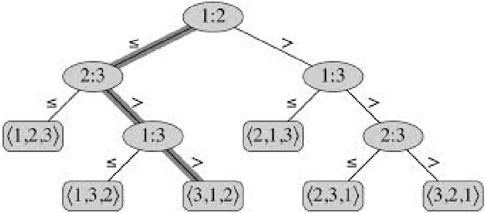

Figure: Depiction of a decision-tree model for insertion sort. $i : j$ represents comparison between $A[i]$ and $A[j]$.

- In a decision tree, each internal node is annotated by $i : j$ for some $1 \leq i, j \leq n$.
- Each leaf is annotated by the sorted permutation $(\pi(1), \pi(2), \ldots, \pi(n))$.
- The execution of the sorting algorithm corresponds to tracing a path from the root to a leaf.

Decision tree model

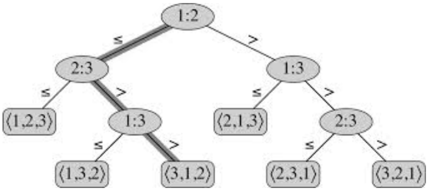

Figure: A decision-tree model for insertion sort. $i : j$ represents comparison between $A[i]$ and $A[j]$.

- Each internal node depicts a comparison $A[i] \leq A[j]$.
- If $A[i] \leq A[j]$ then we go to the left sub-tree, now knowing that $A[i] < A[j]$.
- Otherwise, we go to the right sub-tree, now knowing that $A[i] > A[j]$.

Decision tree model

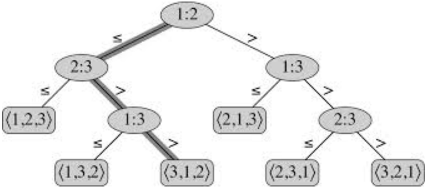

Figure: A decision-tree model for insertion sort. $i : j$ represents comparison between $A[i]$ and $A[j]$.

- The left sub-tree dictates subsequent comparisons once we know that $A[i] < A[j]$.
- The right sub-tree dictates subsequent comparisons once we know that $A[i] > A[j]$.
- When we come to the leaf, the sorting algorithm has established the ordering

$$A[\pi(1)] \leq A[\pi(2)] \leq \ldots \leq A[\pi(n)]$$

- Each of the $n!$ permutations must occur as a leaf (at least once) in the decision-tree model of any correct sorting algorithm, since each permutation may be the sorted order.

Decision tree model

- Each of the leaves in the decision tree must be reachable in a downward path corresponding to an actual execution of the comparison sort.
- The length of the longest simple path from the root of a decision tree to any of its leaves represents the worst-case number of comparisons that the corresponding sorting algorithm performs.
- So, the worst-case number of comparisons for a given comparison-sort algorithm equals the height of its decision tree.
- Therefore, a lower bound on the height of any decision tree is a lower bound on the running time of a comparison sort algorithm.

Lower Bound

- Consider any decision tree corresponding to a correct comparison sort algorithm.
- Suppose its height is h and it has l leaves.
- Then, $l \geq n!$, since each of the permutations must appear as a label for some leaf.
- A binary tree of height h has at most 2^h leaves.
- Therefore,

$$2^h \geq l \geq n!$$

Lower Bound

- $$2^h \geq n!$$
- Taking logarithms (base 2)
$$h \geq \Omega(n \log n)$$
- This follows from Stirling's approximation $\ln(n!) \approx n \ln n - (n - 1/2) + \Theta(1)$.
- Thus, any comparison-sort algorithm requires $\Omega(n \log n)$ time.
- **Corollary**: Heapsort and Mergesort are *asymptotically* optimal comparison sorts.