Introduction

- Impossibility of algorithms: There are some problems that cannot be solved
  - We’ll visit this throughout the semester, especially with NP-completeness
  - Today’s example: there does not exist a general-purpose (comparison-based) algorithm to sort \( n \) elements in time \( o(n \log n) \)
  - Will show this by proving an \( \Omega(n \log n) \) lower bound on comparison-based sorting

Comparison-Based Sorting Algorithms

- What is a comparison-based sorting algorithm?
  - The sorted order it determines is based only on comparisons between the input elements
  - E.g., Insertion Sort, Selection Sort, Mergesort, Quicksort, Heapsort
- What is not a comparison-based sorting algorithm?
  - The sorted order it determines is based on additional information, e.g., bounds on the range of input values
  - E.g., Counting Sort, Radix Sort

Decision Trees

- A decision tree is a full binary tree that represents comparisons between elements performed by a particular sorting algorithm operating on a certain-sized input (\( n \) elements)
  - Key point: a tree represents algorithm’s behavior on all possible inputs of size \( n \)
  - Each internal node represents one comparison made by algorithm
    - Each node labeled as \( i : j \), which represents comparison \( A[i] \leq A[j] \)
    - If, in the particular input, it is the case that \( A[i] \leq A[j] \), then control flow moves to left child, otherwise to the right child
    - Each leaf represents a possible output of the algorithm, which is a permutation of the input
    - All permutations must be in the tree in order for algorithm to work properly

Example for Insertion Sort


Example for Insertion Sort (2)

- Example: \( A = [7, 8, 4] \)
  - First compare 7 to 8, then 8 to 4, then 7 to 4
  - Output permutation is \( (3, 1, 2) \), which implies sorted order is 4, 7, 8
Proof of Lower Bound

- Length of path from root to output leaf is number of comparisons made by algorithm on that input
- Worst-case number of comparisons is length of longest path (= height $h$)
- Number of leaves in tree is $n$!
- A binary tree of height $h$ has at most $2^h$ leaves
- Thus we have $2^h \geq n!$ ≥ $\sqrt{2\pi n} \left(\frac{n}{e}\right)^n$
- Take base-2 logs of both sides to get
  \[
  h \geq \lg \sqrt{2\pi} + (1/2) \lg n + n \lg n - n \lg e = \Omega(n \log n)
  \]
  \[\Rightarrow\text{ Every comparison-based sorting algorithm has an input that forces it to make } \Omega(n \log n) \text{ comparisons} \]
  \[\Rightarrow\text{ Mergesort and Heapsort are asymptotically optimal} \]

Another Lower Bound: Convex Hull

- Can use the lower bound on sorting to get a lower bound on the convex hull problem:
  - Given a set $Q \subseteq \{p_1, p_2, \ldots, p_n\}$ of $n$ points, each from $\mathbb{R}^2$, output $CH(Q)$, which is the smallest convex polygon $P$ such that each point from $Q$ is on $P$'s boundary or in its interior

Another Lower Bound: Convex Hull (cont’d)

- We will reduce the problem of sorting to that of finding a convex hull
- I.e., given any instance of the sorting problem $A = \{x_1, \ldots, x_n\}$, we will transform it to an instance of convex hull such that the time complexity of the new algorithm sorting will be no more than that of convex hull
  - If convex hull could be solved in time $o(n \log n)$ then so can sorting
  - Since that cannot happen, we know that convex hull is $\Omega(n \log n)$
- The reduction: transform $A$ to $Q = \{(x_1, x_2^1), (x_2, x_2^2), \ldots, (x_n, x_2^n)\}$
  - Takes $O(n)$ time
- Since the points on $Q$ are on a parabola, all points of $Q$ are on $CH(Q)$
  - Can read off the points of $CH(Q)$ in $O(n)$ time
  - Yields a sorted list of points from (any) $A$
- Time to sort $A$ is $O(n) + \text{ convex hull } + O(n)$
- If time for convex hull is $o(n \log n)$, then sorting is $o(n \log n)$
  - Convex hull time complexity is $\Omega(n \log n)$