Searching and Sorting

- Linear and Binary Search
- Order Statistics - MIN and MAX
- Comparison Based Sorting Algorithms
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
- Lower Bound on Comparison based sorting
- Non-Comparison Based Sorting - Integers Sorting
  - Counting Sort
  - Radix Sort

IMDAD ULLAH KHAN

Lower Bound on Sorting Algorithms

Trivial lower bound: Since any element must be part of at least one comparison, there must be at least $n/2 = \Omega(n)$ comparisons

Theorem: Any comparison-based algorithm requires $\Omega(n \log n)$ comparisons to sort n elements

- This means that $\Omega(n \log n)$ comparisons are necessary (lower bound)
- We know that $O(n \log n)$ comparisons are sufficient (upper bound) - INSERTION-SORT with BINARY-SEARCH
- Proving this theorem is non-trivial, as one has to argue about all known and unknown sorting algorithms

Lower Bound on Sorting Algorithms

Theorem: Any comparison-based algorithm requires $\Omega(n \log n)$ comparisons to sort n elements

Assumptions:

- All elements (numbers) are distinct ▷ only a technicality
- Comparison Based Algorithm
  - Only access to elements is via pairwise comparisons
  - No direct manipulations allowed
  - No decision based on values of elements or number of bits/digits
- A comparison is of the form $A[i] < A[j]$ (other forms are equivalent)

Lower Bound on Sorting Algorithms

Theorem: Any comparison-based algorithm requires $\Omega(n \log n)$ comparisons to sort n elements

The input is some permutation of the set $A = \{a_1, a_2, \ldots, a_n\}$

The output of sorting algorithm is a permutation $a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}$

▷ think of $\pi(i)$ as index of ith order statistics of A

Key observation is that for each of the possible $n!$ outputs, π, there exists a unique input, for which π is the only right output

Think about this one-to-one correspondence and keep thinking until it is clear

Lower Bound on Sorting Algorithms

Key observation is that for each of the possible $n!$ outputs, π, there exists a unique input, for which π is the only right output

Think about this one-to-one correspondence and keep thinking until it is clear

Assume

$a_1 < a_2 < a_3$

Input:	Output:
$[a_1, a_2, a_3]$	$[1, 2, 3]$
$[a_1, a_3, a_2]$	$[1, 3, 2]$
$[a_2, a_1, a_3]$	$[2, 1, 3]$
$[a_2, a_3, a_1]$	$[3, 1, 2]$
$[a_3, a_1, a_2]$	$[2, 3, 1]$
$[a_3, a_2, a_1]$	$[3, 2, 1]$

Lower Bound on Sorting Algorithms

Theorem: Any comparison-based algorithm requires $\Omega(n \log n)$ comparisons to sort n elements

The output permutation is determined solely based on the knowledge that the algorithm gains from the answers to comparisons

Answers to t comparisons make a t-bits string

$\triangleright$ (the algorithm's knowledge)

After performing t comparisons the number of different types of information the algorithm can obtain is 2^t

Lower Bound on Sorting Algorithms

Theorem: Any comparison-based algorithm requires $\Omega(n \log n)$ comparisons to sort n elements

After performing t comparisons the number of different types of information the algorithm can obtain is 2^t

Claim: $\mathbf{2^t \geq n!}$, for the algorithm to work correctly for all inputs

- Otherwise, by the pigeon-hole principle, there are at least two different inputs for which the algorithm gets the same knowledge
- The (deterministic) algorithm will output the same π for both inputs
- Hence at least one of the outputs will be wrong $\square$

$$2^t \geq n! \implies 2^t \geq \left(\frac{n}{2}\right)^{n/2} \implies t \geq \log \left(\frac{n}{2}\right)^{n/2} \implies t = \Omega(n \log n)$$

Lower Bound on Sorting Algorithms

Assume

$a_1 < a_2 < a_3$

Input:	Output:
$[a_1, a_2, a_3]$	$[1, 2, 3]$
$[a_1, a_3, a_2]$	$[1, 3, 2]$
$[a_2, a_1, a_3]$	$[2, 1, 3]$
$[a_2, a_3, a_1]$	$[3, 1, 2]$
$[a_3, a_1, a_2]$	$[2, 3, 1]$
$[a_3, a_2, a_1]$	$[3, 2, 1]$

- the output permutation solely depends on the series of comparison answers
- comparisons answers depend on the input
- inputs 'giving' same comparison answers lead to same output permutation
- If an algorith $\mathcal{A}$ always make $t < \log(n!)$ comparisons, the total number of different possible output permutations is $2^k < n!$
- In other words, there is some permutation $\mathcal{A}$ can never output (say perm. π)
- So $\mathcal{A}$ will fail on the input for which π is **the only correct** output