Searchs / Sorts 1

Exam Questions

/ 2 P   e) *Sorting algorithms:*

Below you see four sequences of snapshots, each obtained during the execution of one of the following five algorithms: `InsertionSort`, `SelectionSort`, ~~`QuickSort`~~, `MergeSort`, and `BubbleSort`. For each sequence, write down the corresponding algorithm.

8	6	4	2	5	1	3	7
6	4	2	5	1	3	7	8
4	2	5	1	3	6	7	8

Algorithm:

8	6	4	2	5	1	3	7
1	6	4	2	5	8	3	7
1	2	4	6	5	8	3	7

Algorithm:

8	6	4	2	5	1	3	7
6	4	2	5	1	3	7	8
4	2	5	1	3	6	7	8

not swapped

Algorithm: Bubble Sort

8	6	4	2	5	1	3	7
1	6	4	2	5	8	3	7
1	2	4	6	5	8	3	7

Algorithm: Selection Sort (with min val) Variant

8	6	4	2	5	1	3	7
6	8	2	4	1	5	3	7
2	4	6	8	1	3	5	7

Algorithm:

8	6	4	2	5	1	3	7
6	8	4	2	5	1	3	7
4	6	8	2	5	1	3	7

Algorithm:

8	6	4	2	5	1	3	7
6	8	2	4	1	5	3	7
2	4	6	8	1	3	5	7

Algorithm: Merge Sort

8	6	4	2	5	1	3	7
6	8	4	2	5	1	3	7
4	6	8	2	5	1	3	7

Algorithm: Insertion Sort

/ 4 P

g) *Sorting algorithms quiz:* For each of the following claims, state whether it is true or false. You get 1P for a correct answer, -1P for a wrong answer, 0P for a missing answer. You get at least 0 points in total.

Claim	true	false
There exist arrays of length n which can be sorted with `BubbleSort` after $\Theta(n)$ swaps.	☒	☐

[5, 1, 2, 3, 4] $n-1$ swaps $= \Theta(n)$

[1, 2, 3]   no swaps ✓

There exist arrays of length n for which the runtime of `InsertionSort` is $\Theta(n)$.	☐	☒

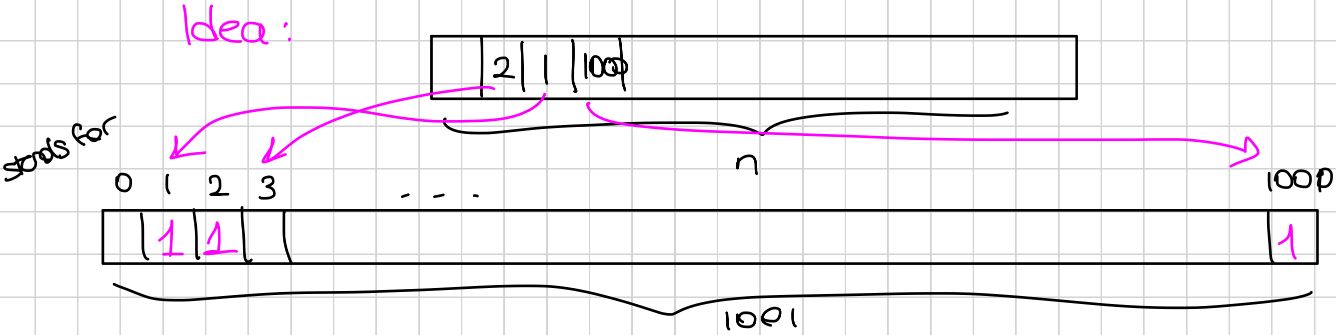
* Insertion Sort uses Binary Search to find the correct positions

Even if the array is already sorted, it's $\Theta(n \log n)$

Consider a sequence of n numbers $\{x_1, ..., x_n\}$, where $0 \leq x_i \leq 1000$, $\forall i = 1, ..., n$, is given as input. There exists an algorithm with runtime $O(n)$ which sorts any such sequence.	☒	☐

Key Realization: 1000 → a constant

Idea:



$n + 1001 \leq O(n)$

There exist a comparison-based sorting algorithm that can sort any array of length n in runtime $O(n)$.	☐	☒

Not possible!

/ 3 P

g) *Sorting algorithms:*

i) Consider the sequence 6, 5, 4, 1, 2, 3. How many swaps does `Bubble Sort` perform to sort this sequence? *Give the exact number of swaps required.*

6 5 4 1 2 3   5 swaps for 6

5 4 1 2 3 6   4 swaps for 5

4 1 2 3 5 6   3 swaps for 4

+

12 swaps

ii) Consider the sequence 6, 5, 4, 1, 2, 3. How many swaps does `Selection Sort` perform to sort this sequence? *Give the exact number of swaps required.*

6 5 4 1 2 3

3 5 4 1 2 | 6

3 2 4 1 | 5 6

3 2 1 | 4 5 6

1 2 3 4 5 6

4 swaps

iii) Let $n \in \mathbb{N}$ be an even number and consider the sequence with the following structure:

$$2, 1, 4, 3, 6, 5, \ldots, n, n-1.$$

How many swaps does `Insertion Sort` perform to sort this sequence? *Give the exact number, not just the asymptotics.*

2 1 4 3 6 5 ... n n−1

We need 1 swap per pair : $\frac{n}{2}$ swaps

Formal Correctness Proof Example

Bubble Sort

Algorithm 1 Bubble Sort (input: array $A[1 \dots n]$).

---

for $j = 1, \dots, n$ **do**
  for $i = 1, \dots, n-1$ **do**
    if $A[i] > A[i+1]$ **then**
      Swap $A[i]$ and $A[i+1]$

Prove correctness of this algorithm by mathematical induction.

Hint: *Use the invariant $I(j)$ that was introduced in the lecture: "After j iterations the j largest elements are at the correct place."* → to prove

Solution:

We prove the invariant in the hint by mathematical induction on j.

- **Base Case.**
  We prove the statement for $j = 1$. Assume that the largest element of A is at position l in the beginning. After the first $l - 1$ iterations of the second for-loop, it is still at position l. For all further steps with $i \geq l$, $A[i]$ contains the largest element and thus the largest element is swapped to position $i + 1$. Hence, in the end the largest element is at position n, which shows $I(1)$.

- **Induction Hypothesis.**
  We assume that the invariant is true for $j = k$ for some $k \in \mathbb{N}, k < n$, i.e. after k iterations the k largest elements are at the correct position.

- **Inductive Step.**
  We must show that the invariant also holds for $j = k + 1$. By the induction hypothesis the k largest elements are at the correct position after k steps, i.e. at the positions $A[n-k+1 \dots n]$. We now consider step $k + 1$. Note that in this iteration the positions of the k largest elements are not changed since for $i \geq n - k$, we will never have $A[i] > A[i+1]$. Thus, in order to show $I(k+1)$ it is enough to show that after step $k + 1$ also the $(k+1)$st largest element is at the correct position. The $(k+1)$st largest element is the largest element of $A[1 \dots n-k]$ (all elements that are larger than it come later by $I(k)$). Thus, by the argumentation in the base case, after $i = n - k - 1$ iterations in the second for-loop, it is at position $A[n-k]$. But for the other k iterations of the second for-loop, nothing changes as was already argued before (the largest elements do not change their position). Thus, after step $k + 1$, the $k + 1$ largest elements are at the correct position, which shows $I(k+1)$.

By the principle of mathematical induction, $I(j)$ is true for all $j \in \mathbb{N}, j \leq n$. In particular, $I(n)$ holds, which means that after the first n iterations the n largest elements are at the correct position. This shows that after n steps the array is sorted, which shows correctness of the Bubble Sort algorithm.

☆ Make sure you have all the key ideas !
↳ try to do it in detail