# Sabancı University
# Faculty of Engineering and Natural Sciences

## CS301 – Algorithms

## Homework 1

Due: April 2, 2023 @ 23.55
(Upload to SUCourse - **no late submission**)

**PLEASE NOTE**:

- Provide only the requested information and nothing more. Unreadable, unintelligible and irrelevant answers will not be considered.
- You can collaborate with your `TA/INSTRUCTOR ONLY` and discuss the solutions of the problems. However you have to write down the solutions on your own.
- Plagiarism will not be tolerated.

**Late Submission Policy**:

- Your homework grade will be decided by multiplying what you normally get from your answers by a "submission time factor (STF)".
- If you submit on time (i.e. before the deadline), your STF is 1. So, you don't lose anything.
- If you submit late, you will lose 0.01 of your STF for every 5 mins of delay.
- We will not accept any homework later than 500 mins after the deadline.
- SUCourse+'s timestamp will be used for STF computation.
- If you submit multiple times, the last submission time will be used.

| Question | Points | Score |
|---|---|---|
| 1 | 20 | |
| 2 | 20 | |
| 3 | 50 | |
| 4 | 10 | |
| Total: | 100 | |

**Question 1** [20 points]

(a) [5 points] What is the form of the input array that triggers the worst case of the insertion sort?

**When the input array is in descending order (reverse ordered), it triggers the worst-case scenario for insertion sort. Each element will need to be compared with every other element to find the correct position.**

(b) [5 points] What is the complexity of this worst–case behavior in Θ notation?

**Worst-case time complexity is Θ(n^2).**

(c) [10 points] Explain how this particular form of the array results in this complexity.

**There are two parts to an insertion sort. Namely: sorted and unsorted. The first element is the only one in the sorted part, while the remaining elements are in the unsorted part. The algorithm iteratively chooses the first element that is not sorted and compares it to the items in the array's sorted portion. The chosen element is then inserted into the appropriate location. As each element in the unsorted part of the input array is smaller than all the elements in the sorted part, it should be compared and moved to the beginning of the sorted part when the input array is arranged in descending order. As a result, the elements' number of comparisons and shifts grows linearly. The total needed comparisons and shifts can be expressed as follows: 1 + 2 ……. + (n-1). The equation adds up to (n-1) n/2. The dominating part is n^2/2. Constant factors and lower order terms are dropped. Therefore it produces a time complexity of Θ(n^2).**

**Question 2** [20 points]

(a) [5 points] What is the form of the input array that triggers the best case of the insertion sort?

**When the input array is already sorted (in ascending order), it triggers the best-case scenario of insertion sort. We don't need any additional operations.**

(b) [5 points] What is the complexity of this best–case behavior in Θ notation?

**Best-case time complexity is Θ(n).**

(c) [10 points] Explain how this particular form of the array results in this complexity.

**I've already mentioned the form of the insertion sort on question 1c. So in the best-case, each element in the unsorted part of the input array is already at the proper position in relation to the elements in the sorted part when the input array is arranged in ascending order. Each element simply needs to be compared once to ensure that it is in the right place by the algorithm, that's the only operation we need. Hence, as the array contains n elements, there will be n comparisons since only one comparison is required for each element. The number of comparisons determines the time complexity in this scenario because shifts or swaps are not necessary. In Θ notation, the complexity is represented as Θ(n), as the algorithm's running time is proportional to the number of elements in the array.**

**Question 3** [50 points]

Suppose that you are trying to prove $(5n + 4)^2 = O(n^2)$ by using the formal definition of $O$–notation, where $n \geq 0$.

In order to show that $(5n + 4)^2 = O(n^2)$ by using the formal definition of $O$–notation, we need to pick constants $c$ and $n_0$ such that for any $n \geq n_0$ we have

$$(5n + 4)^2 \leq cn^2 \tag{1}$$

(a) [25 points] If you use $n_0 = 2$, what is the smallest $c$ value that makes the proof go through?

**For n0 = 2;**
**(5 * 2 + 4)^2 ≤ c * (2^2)**
**(10 + 4)^2 ≤ c * 4;**
**14^2 ≤ 4c**
**196 ≤ 4c**

**When we divide both sides by 4;**
**49 ≤ c**

**Thus, the smallest c value that makes the proof go through is 49.**
**The inequality (5n + 4)^2 ≤ 49n^2 holds for all n ≥ 2.**

(b) [25 points] If you use $c = 36$, what is the smallest $n_0$ value that makes the proof go through?

**For c = 36, and finding the smallest n0 value such that (5n + 4)^2 ≤ 36n^2 holds for all n ≥ n0;**
**Try different n values starting from n=1.**

**n = 1:**
**(5 * 1 + 4)^2 = 9^2 = 81**
**36 * (1^2) = 36**

**81 > 36, so the inequality doesn't hold for n = 1.**

**n = 2:**
**(5 * 2 + 4)^2 = (14)^2 = 196**
**36 * (2^2) = 144**

**196 > 144, so the inequality doesn't hold for n = 2.**

**n = 3:**
**(5 * 3 + 4)^2 = (19)^2 = 361**
**36 * (3^2) = 324**

**361 > 324, so the inequality doesn't hold for n = 3.**

**n = 4:**
**(5 * 4 + 4)^2 = (24)^2 = 576**
**36 * (4^2) = 576**

**576 = 576, so the inequality holds for n = 4.**

**Therefore, the smallest n0 value that makes the proof go through when c = 36 is n0 = 4.**

**Question 4** [10 points]

Rank the following functions in descending order with respect to their growth rates.

$\lg(n!)$ $n!$ $n2^n$ $2^{2^n}$ $\lg^2 n$ $(\lg n)!$

**2^2^n > n*2^n > n! > (lg n)! > lg(n!) > lg^2 n**

**Handwritten version:** $2^{2^n} > n2^n > n! > (\lg n)! > \lg(n!) > \lg^2 n$