University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
G. Schmid

Algorithms and Data Structures – Conditional Course Winter Term 2026/27

Sample Solution Exercise Sheet 2

Due: Wednesday, 04 November 2026, 12:00

Exercise 1: Hacked? *(10 Points)*

A system administrator analyzes login times to detect suspicious behavior. If a user logs in twice in succession "inhumanly quickly", the user was probably a bot. You are given a list of n integers representing login times. Determine the smallest absolute difference between two times.

Example Consider the array

$$A = [-20, 10, -18, 15, -100, 30, 98].$$

We seek the smallest absolute difference between two distinct entries, namely

$$\min_{1 \le i < j \le n} |A[i] - A[j]|.$$

For this array, the minimum is $|-20-(-18)| = 2$.

(a) Implement a brute-force algorithm that checks every pair to find the minimum absolute difference. What is its asymptotic running time? *(2 Points)*

(b) Design a more efficient algorithm. First describe it in words, then implement it. Explain why it is correct and give its asymptotic running time.[1] *(6 Points)*

(c) The website provides `input.txt`, containing 1000000 distinct integers. Use your implementation to compute the answer for this input and report the result. *(2 Points)*

Sample Solution

The accompanying `analyze.py` implements both approaches.

(a) Check each pair of positions once and keep the smallest difference seen so far. There are

$$(n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}$$

pairs. Since each comparison takes constant time in the usual model, the running time is $\Theta(n^2)$.

(b) Sort the numbers first, then compare only neighbours in the sorted array. For the example, sorting gives

$$[-100, -20, -18, 10, 15, 30, 98].$$

The differences between neighbours are $80, 2, 28, 5, 15, 68$, so the answer is 2.

[1] The template includes a helper function `compare_runtimes`. Use it to see how your improved algorithm performs.

Why are neighbours enough? If two numbers have another number between them in sorted order, their distance cannot be smaller than the distance to that intermediate number. More precisely, for sorted values $b_1 \le \cdots \le b_n$ and $i < j$,

$$b_j - b_i \ge b_{i+1} - b_i.$$

Thus every pair has a neighbouring pair whose difference is no larger. Checking all neighbours therefore finds a smallest difference. This also works if values repeat: two equal neighbours give difference zero.

Sorting with Merge Sort takes $O(n \log n)$ time. The scan checks $n - 1$ differences and takes $O(n)$ time. Altogether, the running time is $O(n \log n)$. The supplied implementation uses Python's `sorted`, whose worst-case time has the same bound.

(c) For the supplied `input.txt`, the result is

$$\boxed{619145}.$$

Run `python analyze.py input.txt` to compute it. The brute-force method would check 499999500000 pairs for this input, so use the sorting approach.

Exercise 2: O-Notation *(10 Points)*

(a) Prove or disprove the following statements using the definition of Landau notation from the lecture.[2]

$$\sum_{i=0}^{n} i \in \Omega(n^2)$$

(2 Points)

$$n^{1/2} \cdot n^{1/3} \cdot n^{1/4} \in \Theta(n)$$

(2 Points)

$$3n \log n + 7n \in \Theta(n \log n)$$

(2 Points)

(b) Give two functions f, g such that

$$f(n) \in O(g(n)) \quad \text{but} \quad f(n) \notin \Theta(g(n)).$$

Prove that your functions have these properties. You may use limits for this part. *(4 Points)*

Sample Solution

Throughout this solution, log means logarithm to base 2.

(a) We consider the three statements in order.

(i) **True.** We have

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} \geq \frac{1}{2}n^2 \qquad (n \geq 1).$$

This is exactly the lower bound required by the definition of $\Omega(n^2)$, with $c = 1/2$ and $n_0 = 1$.

(ii) **False.** Adding the exponents gives

$$n^{1/2}n^{1/3}n^{1/4} = n^{13/12} = n \cdot n^{1/12}.$$

For this to be in $\Theta(n)$, it would in particular need an upper bound Cn for some constant $C > 0$ and all sufficiently large n. But for any proposed C and threshold n_0, choose an integer $n > \max\{n_0, C^{12}\}$. Then $n^{1/12} > C$, so $n^{13/12} > Cn$. No such upper bound exists; hence the function is not in $\Theta(n)$.

(iii) **True.** For $n \geq 2$, we have $\log n \geq 1$, so $7n \leq 7n \log n$. Therefore

$$3n \log n \leq 3n \log n + 7n \leq 10n \log n.$$

Both bounds hold from $n_0 = 2$ onwards. Taking $c_1 = 3$ and $c_2 = 10$ proves membership in $\Theta(n \log n)$ directly from the definition.

(b) Choose $f(n) = n$ and $g(n) = n^2$. Since $n \leq n^2$ for $n \geq 1$, we have $f \in O(g)$ with $c = 1$ and $n_0 = 1$.

However, a lower bound $cn^2 \leq n$ with a fixed $c > 0$ cannot hold for all large n: it fails whenever $n > 1/c$. Thus $f \notin \Omega(g)$ and therefore $f \notin \Theta(g)$.

[2] A proof using only limits receives no points.