Algorithms and Data Structures

Exercise Session 3

https://n.ethz.ch/~ahmala/and

Moodle Quiz Week 4

- 5 minutes
- 5 problems
- Watch out for
  - Select one or more:
  - Select one:

Maximum Subarray Problem

- contiguous subset of an array
- Array: [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Password

Password

complexity3

Maximum Subarray Problem

- contiguous subset of an array
- Array: [-2, 1, -3, 4, -1, 2, 1, -5, 4]
- Output: 6 (subarray [4, -1, 2, 1])

Naive Method

Naive Method

- Idea: Check all possible subarrays
- Algorithm
  - Generate all subarrays
  - Calculate sum for each
  - Keep track of maximum sum

Divide and Conquer

Divide and Conquer

- Idea: Recursively divide the array and combine solutions
- Algorithm:
  - Divide array into two halves
  - Recursively find maximum subarray in left and right halves
  - Find maximum subarray crossing the midpoint
    - Find maximum suffix sum of left half
    - Find maximum prefix sum of right half
  - Return the maximum of the three

Master Theorem

Theorem

If $T(n) = aT\left(\left\lceil \frac{n}{b} \right\rceil\right) + O(n^d)$ (for constants $a > 0, b > 1, d \geq 0$), then:

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

Linear Solution

Linear Solution

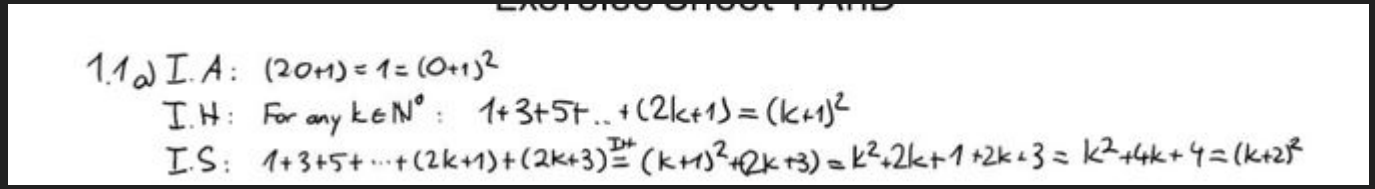
- Algorithm:
  - Initialize variables for current sum and maximum sum
  - Iterate through the array once
  - At each step, decide whether to start a new subarray or extend the existing one
  - Update maximum sum if current sum is larger

Can we do better than O(n)?

Source

https://cadmo.ethz.ch/education/lectures/HS16/DA/skript/skript.pdf#page=21

Common Mistake



1.1 a) I.A: $(2 \cdot 0+1) = 1 = (0+1)^2$

I.H: For any $k \in \mathbb{N}^0$: $1+3+5+\ldots+(2k+1) = (k+1)^2$

I.S: $1+3+5+\cdots+(2k+1)+(2k+3) \overset{IH}{=} (k+1)^2+(2k+3) = k^2+2k+1+2k+3 = k^2+4k+4 = (k+2)^2$

Last Week's Exercise Sheet

Exercise 2.3 *O-notation mistake.* **(1 point).**

Let $f : \mathbb{N} \to \mathbb{R}_+$ be a function, with $f(n) \leq O(n)$. A colleague tried to prove that $e^{f(n)} \leq O(e^n)$. You found their notes, in which they start with the statement they want to show, and derive a series of equivalent statements. The notes read:

$$
\begin{aligned}
e^{f(n)} &\leq O(e^n) && \text{— use Definition 1 on the first page} && (1)\\
e^{f(n)} &\leq C \cdot e^n, \text{ for some } C > 0 && \text{— take the log on both sides} && (2)\\
\log\left(e^{f(n)}\right) &\leq \log\left(C \cdot e^n\right) && \text{— } \log(C \cdot e^n) = \log C + n && (3)\\
f(n) &\leq \log C + n && \text{— } n + \log C \leq O(n) && (4)\\
f(n) &\leq O(n) && \text{— True by assumption, so we are done!} && (5)
\end{aligned}
$$

Forward & Backward Proofs

Omega Notation

Definition 1 (Ω-Notation)**.** For $f : N \to \mathbb{R}_+$,

$$\Omega(f) \coloneqq \{g : N \to \mathbb{R}_+ \mid f \leq O(g)\}.$$

We write $g \geq \Omega(f)$ instead of $g \in \Omega(f)$.

Theta Notation

Definition 2 (Θ-Notation)**.** For $f : N \to \mathbb{R}_+$,

$$\Theta(f) \coloneqq \{g : N \to \mathbb{R}_+ \mid g \leq O(f) \text{ and } f \leq O(g)\}.$$

We write $g = \Theta(f)$ instead of $g \in \Theta(f)$.

In other words, for two functions $f, g : N \to \mathbb{R}_+$ we have

$$g \geq \Omega(f) \Leftrightarrow f \leq O(g)$$

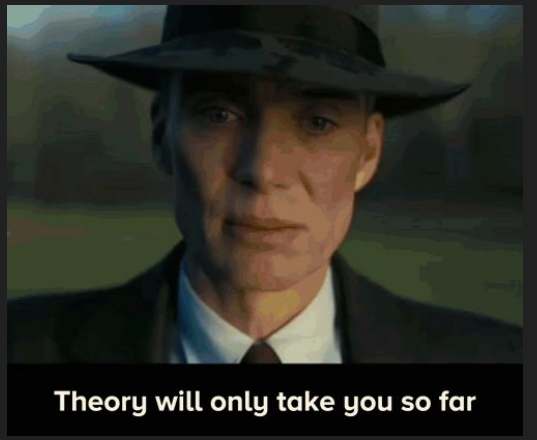
and

$$g = \Theta(f) \Leftrightarrow g \leq O(f) \text{ and } f \leq O(g).$$

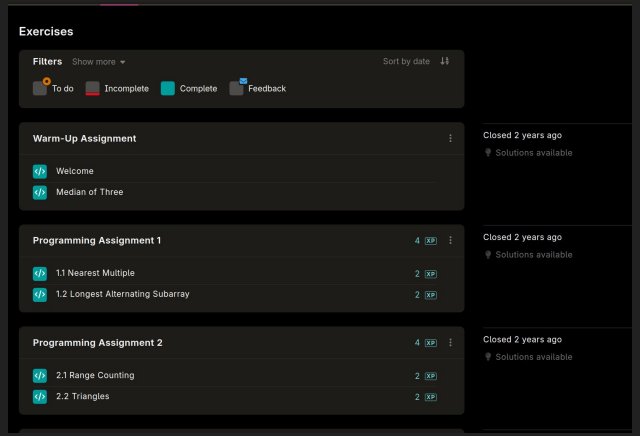
Code Expert

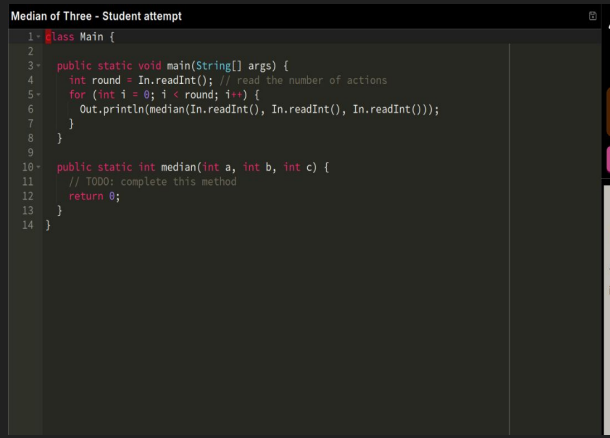
First programming task on Friday

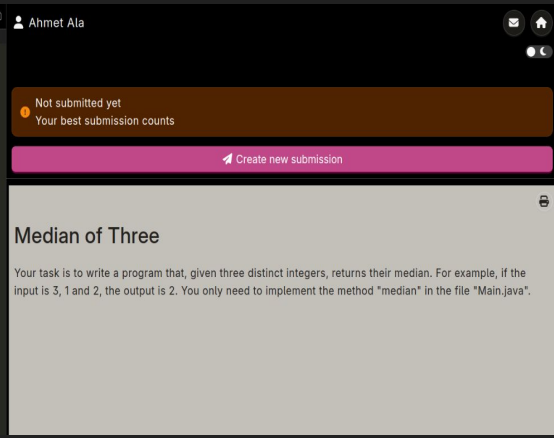
40% of the exam



You get one point for each passing test set. To pass both test sets correctly, your solution has to be in $O(\log n)$ time.







Peer Grading

Task 2.3