Informatics II, Spring 2024, Solution Exercise 4

Publication of exercise: March 11, 2024
Publication of solution: March 18, 2024
Exercise classes: March 18 - March 22, 2024

Learning Goal

- Learn how to solve problem with Divide and Conquer.
- Learn how to analyze Recurrences with Substitution, Recursion tree and Master method.

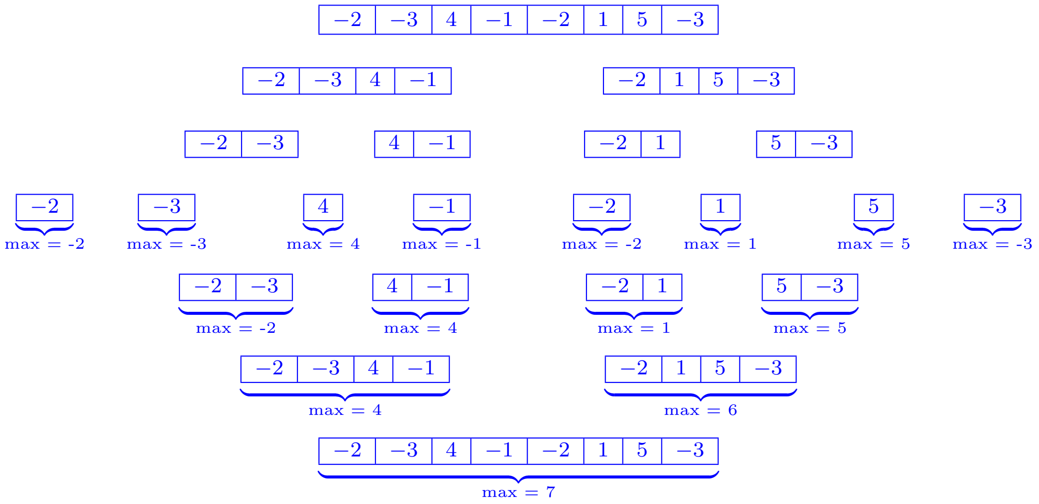
Task 1 [Easy]

The closest number problem involves finding the closest number in an array $A[...]$ with length n sorted in ascending order to a given number t. One integer a is closer to t than another integer b if $|a - t| < |b - t|$. Implement an algorithm with complexity $O(\log n)$ that finds the closest number to t in an array A sorted in ascending order. Use C code for your implementation. See task01.c

Task 2 [Medium]

The maximum subarray problem involves finding the contiguous subarray in an unordered array that has the largest sum. For example for array $A = [-1, 2, -4, 1, 9, -6, 7, -3, 5]$ the maximum subarray is $[1, 9, -6, 7, -3, 5]$ with a sum of 13. Use a divide and conquer approach to solve this problem by breaking it into subproblems and solving them recursively.

a) Draw a tree to illustrate the process of determining the maximum subarray in array $A = [-2, -3, 4, -1, -2, 1, 5, -3]$.



b) Implement a divide and conquer algorithm that finds the maximum subarray in an array A and returns its sum. Use C code for your implementation. See task02b.c

c) Determine the recurrence relation of your algorithm and its asymptotic tight bound.

Recurrence: $T(n) = 2T(\frac{n}{2}) + O(n)$
Asymptotic complexity: $\Theta(n \log n)$

Task 3 [Hard]

Given an array of n integers, find the majority element with a divide and conquer approach. The majority element is the element that has appeared more than $\lfloor \frac{n}{2} \rfloor$ times. You can assume that the majority element always exists. See task03.c

Task 4 [Medium]

Consider the recurrence $T(n) = 2T(n/2) + n\log(n) - n + O(\log(n))$ with $T(1) = 1$. Determine the Master method case that applies and the asymptotic complexity it yields.

□ Case 2 applies and yields complexity $\Theta(\log(n))$

□ Case 1 applies and yields complexity $\Theta(n)$

□ Case 3 applies and yields complexity $\Theta(n\log(n))$

□ Case 2 applies and yields complexity $\Theta(n\log(n))$

☑ None of the cases of the Master method can be applied.