Weekly Assignment 9

November 13, 2019

Exercise 1. *Weight: 10%*

You have to sort 1 GB of data with only 100 MB of available main memory. You may use any combination of merge sort, heap sort, or insertion sort. Describe an efficient approach for sorting the data.

Exercise 2. *Weight: 5%*

A list of k strings, each consisting of n symbols, is sorted into lexicographic order using the merge sort algorithm. What is the worst case time complexity of this computation, expressed in terms of k and n?

Exercise 3. *Weight: 5%*

Assume that a merge sort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 mints?

1. 256
2. 1024
3. 2048
4. 512

Exercise 4. *Weight: 20%*

Given a sorted array of distinct integers $A[1 \ldots n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $\mathcal{O}(\lg n)$ and prove its correctness.

Exercise 5. *Weight: 20%*

The net income (positive or negative) of a company during the last n years is represented as an array $\mathcal{I} = (i_1, i_2, \ldots, i_n)$. For a contiguous period $x \ldots y$, we define the accrued income as $AI = (i_x + i_{x+1} + \ldots + i_y)$. We are now interested in the best period of the history of the company, where the accrued income is maximal. For instance, the maximal accrued income (MAI) for $\mathcal{I} = (-5, 1, 3, -1, 10, -2, 0)$ is $MAI = 1 + 3 - 1 + 10 = 13$.

Give a divide-and-conquer algorithm to compute the MAI and analyse its complexity.

Exercise 6. *Weight: 20%*

Mark, John and David had to solve an algorithmic problem of size n. Each of them came up with a different divide-and-conquer solution. Mark solved the problem by recursively solving two subproblems of size n–1 and then combining the solutions in constant time. John decided to divide the problems into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $\mathcal{O}(n^2)$ time. Finally, David came up with a division of the problem into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

1. Give the asymptotic complexity of Mark, John and David's algorithms using the substitution method.
2. Which is the best?

Exercise 7. *Weight: 20%*

Design a $\theta(n \lg n)$ time algorithm for the following problem: "Given set U of n integers and another integer m, find out whether or not there exist two elements in U whose sum is exactly m".