Homework 4

February 2020

1 Solving Recurrences

Solve the following Recurrence relations either by unrolling or by guess-and-prove (see course book chapter 5, Substituting a solution into the Mergesort recurrence). Compare your findings with the result from the master theorem presented in class (if applicable). The last two recurrences are more difficult and nothing we would expect you to be able to solve in an exam.

(a) $T(n) = T(n-1) + cn$

(b) $T(n) = 2T(n-1) + cn$

(c) $T(n) = 2T(n/2) + cn^2$

(d) $T(n) = 2T(n/2) + cn^3$

(e) $T(n) = 3T(n/2) + cn$

(f) $T(n) = 2T(n/2) + n \log n$

(g) $T(n) = 2T(n/2) + \sqrt{n}$

turn page

2 Multi-MergeSort

MergeSort is based on the merge procedure which takes as an input two sorted arrays of n elements each and return as an output a sorted array of $2n$ elements. For this problem, we present an extended version of this procedure "multi-merge". It takes as an input k sorted arrays of size n each and returns as an output an array of kn sorted elements.

1. First approach: use the merge procedure to merge the two first arrays, then use it again to merge the resulting array with the third array and so on until you merge all the k arrays.
   (a) Construct the runtime function $T(n, k)$ for this approach.
   (b) Calculate the time complexity of this algorithm.
2. Second approach: try to solve this problem based on a divide and conquer approach.
   (a) Explain your algorithm.
   (b) Provide a pseudo code for your algorithm and prove that it is correct.
   (c) Construct the runtime function $T(n, k)$ for this approach.