Computer Science CSC373H
St. George Campus

January 13, 2021
University of Toronto

Homework Assignment #1

Due: January 20, 2021, by 11:00 am

- **You must submit your assignment through the Crowdmark system.** You will receive by email an invitation through which you can submit your work. If you haven't used Crowdmark before, give yourself plenty of time to figure it out!
- You must submit a **separate** PDF document with for **each** question of the assignment.
- To work with one or two partners, you and your partner(s) must form a **group** on Crowdmark (one submission only per group). We allow groups of up to three students, submissions by groups of more than three students will not be graded.
- The PDF file that you submit for each question must be typeset (**not** handwritten) and clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.
- If this assignment is submitted by a group of two or three students, for each assignment question the PDF file that you submit should contain:
  1. The name(s) of the student(s) who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.
- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy for this course, as stated in: `http://www.cs.toronto.edu/~sam/teaching/373/#HomeworkCollaboration`.
- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class (in lectures or tutorials) by referring to it.
- Unless we explicitly state otherwise, you should justify your answers. Your paper will be graded based on the correctness and efficiency of your answers, and the clarity, precision, and conciseness of your presentation.
- The total length of your pdf submission should be no more than 4 pages long in a 10pt font.

Question 1. (15 marks) You are given n piles of graded papers, each pile containing m papers sorted by increasing grade. Your task is to merge these n sorted piles into a single pile of nm papers, also sorted by increasing grade.

a. (5 marks) Giangiovanni Merluscone proposes the following algorithm to do it. Merge the first two sorted piles into a sorted pile; then merge the resulting pile with the third sorted pile; then merge the resulting pile with the fourth sorted pile, etc. What is the running time of this algorithm in terms of n and m? Recall that merging two sorted piles of papers can be done in time proportional to ***the size of the resulting pile***.

b. (10 marks) Give a more efficient ***divide-and-conquer*** algorithm to solve this problem. Describe your algorithm ***clearly and concisely*** in English (you don't need to give the pseudo-code). What is the (worst-case) running time of your algorithm? You don't need to prove the correctness of your algorithm, but you must justify its running time by giving (and solving) the running time recurrence relation. For simplicity you may assume that n or m is an exact power of 2.

Question 2. (20 marks) Given an array A of n integers, some of which may be negative, we want to find S_{ij} such that:

$$S_{ij} = \sum_{r=i}^{j} A[r]$$

is the maximum sum over all possible i and j, $1 \leq i \leq j \leq n$.

For both parts below, first describe your algorithm ***clearly and concisely*** in English and then give its pseudo-code. You should also give the running time recurrence relation, and use the master theorem to justify the asymptotic running time. For simplicity you may assume that n is an exact power of 2.

a. (10 marks) Give a simple divide-and-conquer algorithm that runs in $O(n \log n)$ worst-case time.

b. (10 marks) By improving the efficiency of the "combine" part of your algorithm, derive a more efficient divide-and-conquer algorithm that runs in worst-case $O(n)$ time.

HINT: To facilitate the "combine" part of divide-and-conquer, each subproblem recursive call now returns more than just the optimal solution of the subproblem.