Computer Science C73 September 29, 2021
Scarborough Campus University of Toronto

Homework Assignment #4

(worth 6% of the course grade)
Due: October 6, 2020, by 11:59 pm

• ***You must submit your assignment through the Crowdmark system.*** *You will receive by email an invitation through which you can submit your work in the form of separate PDF docucments with your answers to each question of the assignment. To work with a partner, you and your partner must form a group on Crowdmark. Crowdmark does not enforce a limit on the size of groups.* ***The course policy that limits the size of each group to at most two remains in effect:*** *submissions by groups of more than two persons will not be graded.*
• *It is your responsibility to ensure that the PDF files you submit are legible. To this end, I encourage you to learn and use the LaTex typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You are not required to produce the PDF files you submit using LaTex; you may produce it any way you wish, as long as the resulting document is legible.*
• *By virtue of submitting this assignment you (and your partner, if you have one) acknowledge that you are aware of the policy on homework collaboration for this course.*[a]
• *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, or is in the assigned sections of the official course textbooks, by referring to it.*
• *Unless we explicitly state otherwise, you may describe algorithms in high-level pseudocode or point-form English, whichever leads to a clearer and simpler description.*
• *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.*

[a] "In each homework assignment you may collaborate with at most one other student who is currently taking CSCC73. If you collaborate with another student on an assignment, you and your partner must submit only one copy of your solution, with both of your names. The solution will be graded in the usual way and both partners will receive the same mark. Collaboration involving more than two students is not allowed. **For help with your homework you may consult only the instructor, TAs, your homework partner (if you have one), your textbook, and your class notes. You may not consult any other source.**"

Question 1. (10 marks) Let $A[1..n]$ be an array containing (possibly negative) integers, where n is a power of 2. For any i, j such that $1 \le i \le j \le n$, we define the ***value*** of subarray $A[i..j]$, denoted $\mathbf{val}(A[i..j])$, to the sum of all the integers of the subarray, i.e., $\mathbf{val}(A[i..j]) = \sum_{t=i}^{j} A[t]$. We wish to find the maximum value of any subarray of A, i.e., $\max_{i,j \text{ s.t. } 1 \le i \le j \le n} \mathbf{val}(A[i..j])$.

This can be easily done in $O(n^2)$ time. Give a divide-and-conquer algorithm that solves this problem and is asymptotically faster than $O(n^2)$. For full marks your algorithm should run in $O(n)$ time, but an $O(n \log n)$ algorithm will get you a significant amount of part credit. Explain why your algorithm is correct, and analyze its running time.

Question 2. (10 marks) In class we discussed Karatsuba's divide-and-conquer algorithm $\textsc{FastMult}(x, y)$, which takes as inputs the binary representations x and y of two non-negative integers, and computes the binary representation of their product.

a. Following is an algorithm (with line 3 left incomplete), which takes as input a positive integer n that is a power of two, and computes the binary representation of the number ten to nth power (i.e., the number that is written in decimal as a 1 followed by n 0s).

```
   POWEROFTENTOBINARY($n$)
1  if $n = 1$ then return "1010" (the binary string that represents the number ten)
2  else
3      $x :=$ ???
4      return FASTMULT($x, x$)
```

Fill in the missing expression in line 3, explain why this algorithm works, and analyze its running time.

b. Following is an algorithm (with line 6 left incomplete), which takes as input the decimal representation x of a non-negative integer whose number of digits is a power of 2, and computes the binary representation of that number. Assume that $x[1..\textbf{length}(x)]$ is a string of digits $0, 1, \ldots, 9$, where $x[1]$ is the most significant and $x[\textbf{length}(x)]$ the least significant digit of the input, and that you have defined an array **binary**$[d]$, which, for any digit d, gives the binary representation of d; i.e., **binary**$[0] = 0$, **binary**$[1] = 1$, **binary**$[2] = 10$, up to **binary**$[9] = 1001$.

```
   DECIMALTOBINARY($x$)
1  if length($x$) = 1 then return binary[$x$]
2  else
3      $n :=$ length($x$)
4      $x_L := x[1..n/2]$
5      $x_R := x[n/2+1..n]$
6      return ???
```

Fill in the missing expression in line 6, explain why this algorithm works, and analyze its running time. In constructing the missing expression, you may assume that, in addition to the algorithms POWEROFTENTOBINARY and FASTMULT, you are given a function SUM(a, b), which takes as input the binary representations a and b of two non-negative integers and returns the binary representation of their sum in $O\Big(\max\big(\textbf{length}(a), \textbf{length}(b)\big)\Big)$ time.

Question 3. (10 marks) Describe an algorithm that takes as input an array $A[1..n]$ of (not necessarily distinct) integers. Your algorithm returns an integer x that appears in more than $n/4$ positions of A (i.e., $|\{i : 1 \le i \le n \text{ and } A[i] = x\}| > n/4$), if such an x exists; and returns the special value NIL if no such x exists. Your algorithm should run in $O(n)$ time in the worst case. (It is trivial to solve the problem in $O(n \log n)$ time by sorting A.)

Explain why your algorithm is correct, and analyze its running time. Note that the integers in A may be enormous, and so you cannot use a direct-address table.