Computer Science C73
Scarborough Campus

September 30, 2019
University of Toronto

Homework Assignment #4
(worth 6% of the course grade)
Due: October 7, 2019, by 10 am

• ***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) A ***local minimum*** of a non-empty array A containing distinct numbers is an element of A that is smaller than all of its neighours. (Every element of A except the first and last has two neighbours; the first and last elements have one neighbour each, unless A has only one element. In that case, the single element of A has no neighours and is therefore, by definition, a local minimum.)

Give an algorithm that, given an array $A[1..n]$ containing $n > 1$ distinct numbers, finds a local minimum in $O(\log n)$ time. Justify the correctness and running time of your algorithm.

Note: You may not assume that the elements of A appear in any particular order. If A is sorted, it is trivial to find the minimum (which, of course, is a very special case of a local minimum) in $O(1)$ time!

Question 2. (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 $1 \leq i \leq j \leq n$, we define he ***value*** of subarray $A[i..j]$, denoted $\mathbf{val}(A[i..j])$, as the sum of all the entries of the subarray, i.e., $\sum_{t=i}^{j} A[t]$. We wish to find the maximum value of a subarray of A, i.e., $\max_{i,j \text{ s.t. } 1 \leq i \leq j \leq n} \mathbf{val}(A[i..j])$.

This can be done easily in $O(n^2)$ time as follows: By a "sweep" of $A[1..n]$ we first compute in $O(n)$ time an array $sum[0..n]$ such that $sum[0] = 0$, and for each i, $1 \leq i \leq n$, $sum[i] = \Sigma_{t=1}^{i} A[t]$. Then for each

pair i, j, $1 \leq i \leq j \leq n$, $\mathbf{val}(A[i..j])$ can be computed in $O(1)$ time as $sum[j] - sum[i-1]$. We can now find the maximum value of a subarray of A by considering every possible pair i, j, $1 \leq i \leq j \leq n$, finding the value of subarray $A[i..j]$ in $O(1)$ time by using the array sum as described above, and returning the maximum value over all such $O(n^2)$ pairs.

Give a divide-and-conquer algorithm to solve this problem that is asymptotically faster than $O(n^2)$. Explain why your algorithm is correct, and analyse its running time.

Question 3. (10 marks) In class we discussed Karatsuba's divide-and-conquer algorithm for integer multiplication, which multiplies n-bit numbers by recursively multiplying $n/2$ bit numbers (see Section 5.5 of the textbook). In this problem we show that it is possible to get an asymptotically faster algorithm by recursively multiplying $n/3$-bit numbers. To do this, we use ideas about fast multiplication of polynomials given their coefficients. In particular, we want to multiply two degree-2 polynomials in five (rather than the obvious nine) multiplications. Because here we'll be working with polynomials of fixed degree, we do not have to consider complex numbers as in the FFT.

a. (2 marks) We first consider ***polynomial evaluation***. Our input consists of the three coefficients of a degree-2 polynomial $p(x) = a_2x^2 + a_1x + a_0$, and we want to find the value of this polynomial at five integers $x = 0, 1, 2, 3, 4$.

Give a 5×3 matrix M_1 such that for every (row) vector $\vec{a} = (a_0, a_1, a_2)$,

$$M_1 \cdot \vec{a}^{\mathbf{T}} = \vec{v}^{\mathbf{T}}$$

where $\vec{v} = \big(p(0), p(1), p(2), p(3), p(4)\big)$ — i.e., the (row) vector consisting of the five values of $p(x)$ at $x = 0, 1, 2, 3, 4$.

b. (2 marks) We next consider ***polynomial interpolation***. Our input consists of the values of a degree-4 polynomial $p(x)$ at $x = 0, 1, 2, 3, 4$, and we want to find the coefficients of $p(x)$. That is, our input is a (row) vector $\vec{v} = (v_0, v_1, v_2, v_3, v_4)$, and we wish to find the coefficients $\vec{a} = (a_0, a_1, a_2, a_3, a_4)$ of a polynomial $p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ such that $(v_0, v_1, v_2, v_3, v_4) = \big(p(0), p(1), p(2), p(3), p(4)\big)$.

Give a 5×5 matrix M_2 of rational numbers such that for every $\vec{v} = (v_0, v_1, v_2, v_3, v_4)$, if we compute $\vec{a}^{\mathbf{T}} = M_2 \cdot \vec{v}^{\mathbf{T}}$ and let $(a_0, a_1, a_2, a_3, a_4) = \vec{a}$ and $p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$, then $(v_0, v_1, v_2, v_3, v_4) = \big(p(0), p(1), p(2), p(3), p(4)\big)$.

Hint. Consider part (a) and find the inverse of an appropriate matrix. You may use software (e.g., Mathematica) or a website (e.g., `http://www.math.odu.edu/~bogacki/lat/`) to help you with your calculations, but be sure to credit your source.

c. (6 marks) We now consider a divide-and-conquer algorithm for integer multiplication, where we multiply two n-bit numbers by recursively multiplying ***five*** $n/3$-bit numbers.

Suppose we are given two n-bit numbers A and B; for convenience, assume that n is a power of 3. Let a_2, a_1, a_0 be the integers comprising the leftmost third, middle third, and rightmost third of A; let b_2, b_1, b_0 be defined similarly for B. We then have

$$A = a_2(2^{n/3})^2 + a_1 2^{n/3} + a_0 \text{ and } B = b_2(2^{n/3})^2 + b_1 2^{n/3} + b_0.$$

Our goal is to compute

$$C = A \cdot B = c_4(2^{n/3})^4 + c_3(2^{n/3})^3 + c_2(2^{n/3})^2 + c_1 2^{n/3} + c_0$$

where $(c_0, c_1, c_2, c_3, c_4)$ are the coefficients of the degree-4 polynomial obtained by multiplying the two degree-2 polynomials $p(x) = a_2x^2 + a_1x + a_0$ and $q(x) = b_2x^2 + b_1x + b_0$.

Using the matrices M_1 and M_2 from parts (a) and (b), give the details of this algorithm. Give a recurrence that describes its time complexity, and explain why it does. Finally, explain why this algorithm is (asymptotically) faster than Karatsuba's algorithm discussed in class.