CSE 347 Analysis of Algorithms February 11, 2020

Sample Midterm Exam 1

Instructor: Brendan Juba

This is a sample midterm exam. It is intended to be representative of the content, length, and difficulty of the questions you will be asked on your midterm exam.

Instructions. Answer questions as clearly and concisely as you can. Try to fit your solutions in the space provided, continuing on the provided extra sheets of paper if necessary. We will award partial credit based on the work that you provide; if you get stuck, *be sure to explain what you are trying to do.*

This is a *closed book* exam. The use of laptops, phones, smartwatches, etc. is *not allowed.* You are allowed to reference a crib sheet that you prepared yourself, contained entirely on a single 8.5×11" (US letter size) sheet of paper. Any other written material is not permitted.

You may use (without proof) any theorem that has been stated in lecture. Except on the Reading Solutions problem, you may also reference the result of the homework problems you were assigned without proof. You *may not* use the result of other problems from the book without providing a proof.

Name:

Student ID:

Problem Number	Points Possible	Points Received
1	25	
2	25	
3	30	
TOTAL	80	

This page intentionally left blank.

1. *(25 points)* Suppose that we wish to multiply a sequence of matrices, $A_1 A_2 \cdots, A_{m-1}$, in which the ith matrix has dimensions $n_i \times n_{i+1}$ for varying integers n_i. The total number of operations performed may depend significantly on the order in which we multiply the matrices: for example, if we have matrices that are $n \times 1$, $1 \times n$, and $n \times 1$, it is advantageous to first multiply the second two matrices (essentially an inner product, taking only n steps), followed by multiplying the first $n \times 1$ matrix by the 1×1 result, in another n steps. If we had performed the multiplication in the other order, we would first produce the $n \times n$ product (taking n^2 steps) followed by the product of an $n \times n$ matrix and a $n \times 1$ matrix, essentially a matrix-vector product, using another $O(n^2)$ steps. Consider the following algorithm:

input : Sequence of matrix dimensions $n_1, n_2, \ldots, n_m$
begin
  Initialize a $(m-1) \times (m-1)$ table M such that $M[i,i] = 0$ for all i and a $(m-2) \times (m-1)$ table C
  for $\ell = 2, \ldots, m-1$ **do**
    for $i = 1 \ldots, m - \ell$ **do**
      Put $j \leftarrow i + \ell - 1$, $M[i,j] \leftarrow \infty$
      for $k = i+1, \ldots, j-1$ **do**
        if $M[i,k] + M[k+1,j] + n_i n_{k+1} n_{j+1} < M[i,j]$ **then**
          Put $M[i,j] \leftarrow M[i,k] + M[k+1,j] + n_i n_{k+1} n_{j+1}$,
          Put $C[i,j] \leftarrow k$.
        end
      end
    end
  end
end
return M, C

(a) *(15 points)* Prove that in the table M, $M[i,j]$ contains the optimal cost of multplying the matrices in the range $i, i+1, \ldots, j$, assuming we use the naïve $n_i n_{i+1} n_{i+2}$-time algorithm for multiplying matrices of dimension $n_i \times n_{i+1}$ and $n_{i+1} \times n_{i+2}$.

This page intentionally left blank.

(b) *(10 points)* Give an algorithm that uses the returned M and C to produce a binary tree in which the leaves are labeled by the index of one of the matrices A_i, such that a recursive algorithm that multiplies (using the naïve algorithm) the matrix obtained from recursing on the left subtree with the matrix obtained from recursing on the right subtree will return the matrix product $A_1 A_2 \cdots A_{m-1}$ using this optimal number of operations. (You must prove that your algorithm is correct and runs in polynomial time.)

This page intentionally left blank.

2. *(25 points)* **Reading solutions.** The following is a problem that appeared on your homework. *For this problem only,* you may not simply reference the fact that it appeared on your homework. You must present a solution to this problem.

   Suppose you are given a sequence S of n *events*. (The same event may occur more than once in S.) You want to determine if a short subsequence S' of events occurs in S, not necessarily consecutively. That is, you want to know if there is a way to delete events from S so that the remaining events, in order, are equal to S'. (The same event may also occur more than once in S'.) Supposing that S' is of length m, give an $O(m+n)$ time algorithm to determine if S' is a subsequnce of S.

This page intentionally left blank.

3. *(30 points)* Suppose you are given an array of integers. Give an algorithm that returns an array in which only the *first* occurrence of an element remains, and these first occurrences appear in the same order as the original list. Your algorithm must run in time $O(n \log n)$. Prove its correctness and running time bound.