MIDTERM I, WINTER 2020 — COMPUTER SCIENCE 251

Examiner: Luc Devroye
Instructions: Calculators and computers are not allowed.
Weight: All questions carry equal weight.

Question 1. Write upper bound recurrences (not solutions) for T_n, the worst-case the complexity for the following problems, where n is the input size. So, $T_n \leq$ (your answer). Example: for binary search using a binary comparison oracle, we have $T_n \leq T_{\lfloor (n+1)/2 \rfloor} + 1$.

(i) Mergesort on an array of size n (binary comparison oracle).
(ii) Binary search into a sorted list of size n using a ternary comparison oracle.
(iii) Strassen's method for multiplying two $n \times n$ matrices (RAM model).

Question 2. We are given four points in $\mathbb{R}^2$, $(x_1, y_1), \ldots, (x_4, y_4)$. The dot product of (x_i, y_i) and (x_j, y_j) is defined as $x_i x_j + y_i y_j$. We need to compute the six possible dot products between these four points (consider all $i \neq j, 1 \leq i, j \leq 4$). Explain how this can be done using at most eleven multiplications. (We are allowing any number of additions.)

Question 3. Give explicit decision tree lower bounds in the binary comparison oracle model for problems (i)-(iii), and answer (iv):

(i) Finding the third smallest of eight numbers.
(ii) Sorting seven numbers.
(iii) Merging a sorted list of three numbers with a sorted list of six numbers.
(iv) The problem is to try and guess an 11-bit password. An oracle accepts a candidate password and reveals how many bits are correct (correct value in the correct position). What is the decision tree lower bound for this problem?

Question 4. We want to count the number $N[n, k]$ of ways in which we can pay a bill of n dollars using bank notes of denominations $b_1 < b_2 < \cdots < b_k$. (Example: a bill of 12 dollars can be paid in three ways if we have bank notes of 2,5 and 10 dollars) The standard dynamic programming solution computes all sub-solutions $N[i, j]$ for $i \leq n$, $j \leq k$:

```
for i = 0 to n do
for j = 0 to m do
  if i = 0 then N[i,j] ← 1
  else if j = 0 then N[i,j] ← 0
  else XXX
```

Fill in the missing part indicated by XXX.

Question 5. This question is about the Landau symbols, and the notion of worst-case time T_n taken by an algorithm on a problem of size n.

(i) True or false: If T_n is the worst case complexity of the standard dynamic programming traveling salesman algorithm, then $T_n = \Omega(2^n)$.

(ii) True or false: If T_n is the worst case bit model complexity of Karatsuba multiplication, then $T_n = O(n \log^5 n)$.

(iii) True or false: If T_n is not polynomial in n, then $T_n = \Omega(c^n)$ for some $c > 1$.

Question 6.

(i) Explain why it is possible to split a $6n$ point cloud in $\mathbb{R}^3$ in six sets of n points each by three hyperplane cuts.

(ii) Consider an arbitrary binary tree with 2000 nodes. How small can the height h be?

(iii) What is the worst-case number of binary comparisons when we use mergesort to sort 16 numbers?

(iv) What is the number of binary comparisons when we use mergesort to sort 14 numbers that are already sorted?

Question 7. Give the worst-case time recurrence for finding the k-th smallest of n elements using a median-of-3 instead of the standard median-of-5 algorithm. Give its complexity, and explain in one line how you got that result.

Question 8. In dynamic programming, we have dealt with these examples: (A) the longest common subsequence problem, (B) the knapsack problem, (C) Fibonacci numbers, (D) binomial coefficients, (E) partitions of n into k non-empty sets, (F) job scheduling, (G) the assignment problem, (H) rod cutting, (I) optimal binary search trees, (J) the traveling salesman problem, (K) matrix chains. All of these compute all entries in an array or a 2-d matrix. Referring to the dynamic program solutions seen in class, circle all problems that only need an array, not a 2-d matrix.