Problem Set 12

Data Structures and Algorithms, Fall 2020

Due: Dec 17, in class.

Problem 1

After graduating from university, you decide to interview for a position at the Wall Street bank **Long Live Boole**. The managing director of the bank, Eloob Egroeg, poses a 'solve-or-die' problems to each new employee, which they must solve within 1 hour. Those who fail to solve the problem are fired!

Entering the bank for the first time, you notice that the employee offices are organized in a straight row, with a large T or F printed on the door of each office. Furthermore, between each adjacent pair of offices, there is a board marked by one of the symbols $\wedge$, $\vee$, or $\oplus$. When you ask about these arcane symbols, Eloob confirms that T and F represent the boolean values TRUE and FALSE, and the symbols on the boards represent the standard boolean operators AND, OR, and XOR. He also explains that these letters and symbols describe whether certain combinations of employees can work together successfully. At the start of any new project, Eloob hierarchically clusters his employees by adding parentheses to the sequence of symbols, to obtain an unambiguous boolean expression. The project is successful if this parenthesized boolean expression evaluates to T.

For example, if the bank has three employees, and the sequence of symbols on and between their doors is $T \wedge F \oplus T$, there is exactly one successful parenthesization scheme: $(T \wedge (F \oplus T))$. However, if the list of door symbols is $F \wedge T \oplus F$, there is no way to add parentheses to make the project successful.

Eloob finally poses his question: Describe an algorithm to decide whether a given sequence of symbols can be parenthesized so that the resulting boolean expression evaluates to T. Your input is an array $S[0 \cdots 2n]$, where $S[i] \in \{T, F\}$ when i is even, and $S[i] \in \{\wedge, \vee, \oplus\}$ when i is odd.

Problem 2

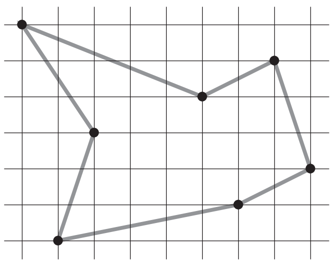
Suppose you are given an array $A[1 \cdots n]$ of numbers, which may be positive, negative, or zero, and which are not necessarily integers. Describe and analyze an algorithm that finds the largest product of elements in a contiguous subarray $A[i \cdots j]$.

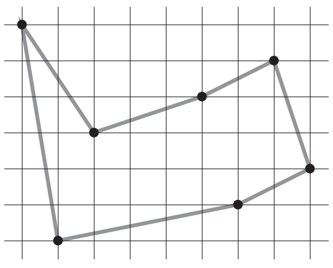
For example, given the array $[-6, 12, -7, 0, 14, -7, 5]$ as input, your algorithm should return 504; and given the one-element array $[-374]$ as input, your algorithm should return 1. (The empty interval is still an interval!) For the sake of analysis, assume that comparing, adding, or multiplying any pair of numbers takes $O(1)$ time. For full credit, your algorithm should have $O(n)$ time complexity and $O(1)$ space complexity.

Problem 3

Suppose we are given a set of n points in the plane, and we wish to find the shortest closed tour that connects all n points. Part (a) of the following figure shows the solution to a 7-point problem. The general problem is NP-hard, and its solution is therefore believed to require more than polynomial time.

We can simplify the problem by restricting our attention to *bitonic tours*, that is, tours that start at the leftmost point, go strictly rightward to the rightmost point, and then go strictly leftward back to the starting point. Part (b) of the following figure shows the shortest bitonic tour of the same 7 points. For finding optimal bitonic tours, a polynomial-time algorithm is possible.


(a)


(b)

Describe an $O(n^2)$-time algorithm for determining an optimal bitonic tour. You may assume that no two points have the same x-coordinate and that all operations on real numbers take unit time. *(Hint: Scan left to right, maintaining optimal possibilities for the two parts of the tour.)*

Problem 4

(a) Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

(b) Is the dynamic-programming algorithm you designed in part (a) a polynomial-time algorithm? Justify your answer.

Problem 5

Show that the class $\mathbf{P}$, viewed as a set of languages, is closed under union, intersection, concatenation, complement, and Kleene star. That is, if $L_1, L_2 \in \mathbf{P}$, then $L_1 \cup L_2 \in \mathbf{P}$, $L_1 \cap L_2 \in \mathbf{P}$, $L_1 L_2 \in \mathbf{P}$, $\overline{L_1} \in \mathbf{P}$, and $L_1^* \in \mathbf{P}$.

Problem 6

(a) Consider the language GRAPH-ISOMORPHISM $= \{\langle G_1, G_2\rangle : G_1$ and G_2 are isomorphic graphs$\}$. Prove that GRAPH-ISOMORPHISM $\in \mathbf{NP}$.

(b) Let ϕ be a boolean formula constructed from the boolean input variables $x_1, x_2, \cdots, x_k$, negations ($\neg$), ANDs ($\wedge$), ORs ($\vee$), and parentheses. The formula ϕ is a *tautology* if it evaluates to 1 for every assignment of 1 and 0 to the input variables. Define TAUTOLOGY as the language of boolean formulas that are tautologies. Show that TAUTOLOGY $\in \mathbf{coNP}$. (Read the textbook to understand the definition of $\mathbf{coNP}$.)

Problem 7

(a) Prove that $\mathbf{P} \subseteq \mathbf{coNP}$.

(b) Prove that if $\mathbf{NP} \neq \mathbf{coNP}$, then $\mathbf{P} \neq \mathbf{NP}$.