Final Exam of Design and Analysis of Algorithms

June 15, 2022

1. (10%) In the prune and search algorithm we introduced in the class to solve the selection problem, the input elements are divided into groups of 5 elements. Please explain the reason why this algorithm can be considered as a prune and search algorithm? (5%) Will this algorithm still work in linear time if the input elements are divided into groups of 7 elements? (5%) Note that you need to explain the reason for your answer.

2. (10%) The simplified two-variable linear programming problem is defined as follows:

$$\begin{aligned} &\text{Minimize} && y \\ &\text{Subject to} && y \geq a_i x + b_i \ (i = 1,2, \dots, n) \end{aligned}$$

As introduced in class, this problem can be solved by using the prune and search approach. Please answer the following questions:

   (a) (2%) Let (x_0, y_0) be an optimum solution and $F(x) = \max_{1 \leq i \leq n} \{a_i x + b_i\}$. Is it true that x_0 must satisfy $F(x_0) = \min_{-\infty \leq x \leq \infty} F(x)$.

   (b) (2%) Given a value x_m, how do we know whether $x_0 < x_m, x_0 > x_m$ or $x_0 = x_m$?

   (c) (2%) If $x_0 < x_m$, then how can we prune those constraints that are useless?

   (d) (2%) How do we choose x_m so that we can prune $\left\lfloor \frac{n}{4} \right\rfloor$ constraints away for each iteration?

   (e) (2%) What is the time complexity of this prune and search approach?

3. (10%) Suppose that among n identical-looking coins, one is fake. With a balance scale (天平), we can compare any two sets of coins to determine whether these two sets weight the same, or which of the two sets is heavier than the other but not by how much. Use the prune and search strategy to design an efficient algorithm for detecting the fake coin (6%). For simplicity, we assume that the fake coin is heavier than the genuine one. Please also analyze the number of weighting needed by your algorithm in worst case (4%).

4. (10%) Given two strings $A = a_1 a_2 \dots a_m$ and $B = b_1 b_2 \dots b_n$, the longest common subsequence (LCS) problem is to find a longest common subsequence between A and B. Please design a dynamic program algorithm to solve the LCS problem (3%). What is the recursive formula for your dynamic programming algorithm (3%)? Note that you have to justify the correctness of your recursive formula. Also analyze the time complexity of your dynamic programming algorithm (4%).

5. (10%) Suppose that we are given a sequence of n matrices $A_1, A_2, \dots, A_n$, where matrix A_i has dimension $d_{i-1} \times d_i$ for each $1 \leq i \leq n$. In the standard matrix multiplication algorithm, the product of two matrices, say $A_1 \cdot A_2$, requires $d_0 \times d_1 \times d_2$ scalar multiplications. Moreover, the matrix multiplication is associative. For example, $A_1 \cdot (A_2 \cdot A_3) = (A_1 \cdot A_2) \cdot A_3$ (i.e., these two different parenthesizations yield the same product). However, different parenthesizations for a product of matrices may have different scalar multiplications. For example, $A_1 \cdot (A_2 \cdot A_3)$ has $d_1 \times d_2 \times d_3 + d_0 \times d_1 \times d_3$ scalar multiplications, but $(A_1 \cdot A_2) \cdot A_3$ has $d_0 \times d_1 \times d_2 + d_0 \times d_2 \times d_3$ scalar multiplications. A product of matrices is fully parenthesized if it is either a single matrix or the product of two fully parenthesized matrix products. Please design a dynamic programming algorithm to fully parenthesize the product $A_1 \cdot A_2 \cdot \dots \cdot A_n$ in a way that minimizes the total number of scalar multiplications (3%). What is the recursive formula for

your dynamic programming algorithm (3%)? Note that you have to justify the correctness of your recursive formula. Also analyze the time complexity of your dynamic programming algorithm (4%).

6. (10%) For the following profit matrix, please use the dynamic programming approach to find an optimal allocation of resources to maximize the total profit for those four projects and four resources. (6%) Please also analyze the time complexity of your dynamic programming approach in big-O notation, where we assume that there are m resources and n projects. (4%)

Project \ Resource	**1**	**2**	**3**	**4**
1	3	7	10	12
2	1	2	6	9
3	2	4	8	9
4	4	2	7	10

7. (5%) Use the resolution principle to determine whether the following Boolean formula is satisfiable or not.

$$\begin{aligned} & -x_1 \vee -x_2 \vee x_3 && (1) \\ \& \quad & -x_1 && (2) \\ \& \quad & x_2 \vee x_3 && (3) \\ \& \quad & -x_3 && (4) \end{aligned}$$

8. (5%) Prove or disprove that the halting problem is NP-hard.

9. (10%) Prove or disprove that 6-SAT (i.e., the satisfiability problem with each clause containing exactly 6 literals) is an NP-complete problem.

10. (10%) Given an undirected graph $G = (V, E)$, a starting vertex $x \in V$ and an ending vertex $y \in V$, the x-y path problem is to find whether there is a path from x to y that visits every vertex exactly once in G. Given a complete undirected graph $G = (V, E)$ and a real-valued cost $w(e)$ for each edge $e \in E$, the TSP problem is to find a cycle C that visits every vertex exactly once in G with the minimum sum $\sum_{e \in C} w(e)$ of edge costs. Prove or disprove that the x-y path problem reduces to the TSP problem.

11. (10%) Determine whether the following statements are correct or not and justify your answers.
    (a) (2%) The time complexity of the whole prune and search process is of the same order as the time complexity of each iteration.
    (b) (2%) If a problem is shown to be NP-complete, then it means that we are not able to find any polynomial time algorithm to solve this problem.
    (c) (2%) If P_1 and P_2 both are NP-complete problems, then we can conclude that $P_1 \propto P_2$ and $P_2 \propto P_1$.
    (d) (2%) Given a set of cities, the traveling salesperson problem is to find a shortest round-trip tour that visits each city exactly once and returns to the starting city. The traveling salesperson problem is known to be an NP-complete problem.
    (e) (2%) According to the NP-completeness theory, if any NP-complete problem can be solved in polynomial time, then NP = P.