Assignment 3

COMP 3804, Fall 2022

Upload in Brighspace by 11:59 PM on December 6, 2022

1 Guidelines

General guidelines are as follows:

1. Since we are only accepting assignments via the Brighspace system, and we will discuss the solutions immediately after the due date, no late submissions will be entertained after the cut-off time & date.
2. Please write clearly and answer questions precisely. It is your responsibility to ensure that what is uploaded is readable. If we can't read, we can't mark!
3. Please cite all the references (including websites, names of friends, etc.) that you used/consulted as the source of information for each of the questions.
4. All questions carry equal marks.
5. When a question asks you to design an algorithm - it **requires** you to
    (a) Clearly spell out the **steps** of your algorithm as a pseudo code.
    (b) **Prove** that your algorithm is correct
    (c) **Analyze** the running time.

2 Problems

1. Find the longest common sequence of the following two DNA strings, X and Y- construct the table and show how you have obtained the longest sequence using the table. X= AGTCGGATA and Y=ACCGGCTA.

2. Assume that $O(pqr)$ number of operations (multiplications and additions) are required to compute the product PQ of two matrices P of dimension $p \times r$ and Q of dimension $r \times q$. Note that the resulting product matrix has dimension $p \times q$ (i.e., p rows and q columns). As input, we are given six matrices A, B, C, D, E, F and their dimensions are as follows:
   - A is 5×12,
   - B is 12×4,
   - C is 4×10,
   - D is 10×5,
   - E is 5×40, and
   - F is 40×6.

   What is the least number of operations required to compute the product $ABCDEF$? Justify your answer.

3. We are given a sequence of n integers, $a_1, \ldots, a_n$, some of which may be negative. For a contiguous subsequence $a_i, \ldots, a_j$, where $1 \leq i \leq j \leq n$, define $\Delta[i,j] = a_i + \cdots + a_j$. In $O(n)$ time, determine a pair of indices (i,j), where $1 \leq i \leq j \leq n$, such that $\Delta[i,j] \geq \Delta[\alpha, \beta]$ for any choice of α, β, where $1 \leq \alpha \leq \beta \leq n$. (Hint: Think of dynamic programming and consider the subsequence ending at j that maximizes $\Delta[i,j]$ for each choice of $j \in \{1, \ldots, n\}$.)

4. Let A be a $m \times n$ matrix where each element is 0 or 1. We are interested in finding the largest square sub-matrix of A such that each of its elements is 1. Design a dynamic-programming algorithm, running in $O(mn)$ time, that finds such a largest square sub-matrix in A. For example, let $A = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$.
   In this case, the 3×3 square submatrix formed by columns $2, 3, 4$ and rows $4, 5, 6$ of all 1s should be returned by the algorithm.

5. You are given a set of n positive numbers $A = \{a_1, \ldots, a_n\}$ and a positive integer t. Design a dynamic programming algorithm running in $O(nt)$ time that decides whether there exists a subset $A' \subseteq A$ such that $\sum_{x \in A'} x = t$. Note that each element of A can be used at most once. Is the run-time of your algorithm polynomial with respect to the size of the input?

6. Let $G = (V, E)$ be a undirected graph. Each vertex $v \in V$ has a positive weight $w(v) > 0$. A subset $S \subseteq V$ is a *cover* of G if for any edge $e = (uv) \in E$, $u \in S$ or $v \in S$. The weight of the cover S is the sum total of the weights of the vertices in S, i.e. $w(S) = \sum_{v \in S} w(v)$. The input to the cover decision problem consists of the graph G and a positive real number Δ, and we need to decide whether there exists a cover $S \subseteq V$ of G, such that $w(S) \leq \Delta$? It is known that the decision problem for the general graphs is **NP**-Complete. Show that the cover decision problem on trees can be solved in polynomial time. (Hint: Think of dynamic programming.)

7. Consider the following decision problem on satisfying inequalities. Let A be an integer $m \times n$ matrix. Let b be a vector of length m where each coordinate is an integer. You need to decide whether there exists a vector x of length n such that each of its coordinates is 0 or 1 and $Ax \leq b$. For example, let $A = \begin{bmatrix} 1 & 0 & -3 \\ 1 & -1 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} -2 \\ 3 \end{bmatrix}$. Then $x = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, satisfies $Ax \leq b$. Show that the problem of satisfying (integer) inequalities is NP-Complete. (Hint: Provide a polynomial-time reduction from 3CNF-SAT problem.)

8. Let $G = (V, E)$ be a simple undirected graph on n-vertices and let k be a positive integer. A set of k vertices $V' \subseteq V$ forms a clique, if for every pair of vertices $u, v \in V'$ we have $uv \in E$. We have seen that the problem Clique(G, k) of deciding whether G contains a clique of size k is **NP**-Complete. Now consider a variant of the clique problem where we need to decide if G has a clique of size at least $n/4$. Let us call this problem the *Quarter-Clique Problem (QCP)*. Show the following:

   (a) Show that $QCP \in$ **NP**.

   (b) Show that the QCP problem is **NP**-Complete. You may provide a polynomial-time reduction from the Clique(G, k) problem. Justify why your reduction works following the definition of polynomial-time reducibility. (Hint: Consider creating a supergraph G' of G with additional vertices and edges.)