Assignment 4
COMP 3804, Fall 2018
Upload in cuLearn by 1PM on December 5, 2018

1 Guidelines

General guidelines are as follows:

1. Since we are only accepting assignments via CU-Learn, 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 clearly readable. If we can’t read, we can’t mark!

3. Please cite all the references (including web-sites, names of friends, etc.) which you used/consulted as the source of information for each of the questions.

4. All questions/problems 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 in pseudo code.
   (b) Prove that your algorithm is correct
   (c) Analyze the running time.

6. You can assume that a graph $G = (V, E)$ uses adjacency list representation.

2 Problems

1. The dynamic programming method for the matrix chain multiplication, which we did in the class, outputs the minimum number of operations (multiplications), but actually doesn’t tell us in which order one needs to multiply the matrices. What modifications will you make in the pseudo-code, so that we can also determine in which order one needs to multiply them? Apply your pseudo-code to multiply the following chain of matrices in an optimal way: $A_1A_2A_3A_4A_5A_6$, where the sequence of dimensions of these matrices are $3, 5, 3, 7, 2, 6, 4$ (i.e., the dimension of $A_1$ is $3 \times 5$, $A_2$ is $5 \times 3$, $A_3$ is $3 \times 7$, $A_4$ is $7 \times 2$, $A_5$ is $2 \times 6$, and $A_6$ is $6 \times 4$). Please show your work.
2. Given an unlimited supply of coins of denominations \( x_1, \ldots, x_n \), we wish to make change for a certain value \( X \). (We are not worried about finding the minimum set of coins that adds to \( X \); we just want to make change; and also note that sometimes we may not be able to make an exact change). Design a dynamic programming algorithm running in \( O(nX) \) time, that can decide whether the change for the amount \( X \) can be made or not? (Note that all the quantities \( (X, x_1, \ldots, x_n) \) are integers.)

3. Consider the following variation of the above problem. Given coins of denominations \( x_1, \ldots, x_n \), we wish to make change for a certain value \( X \). But for any denomination, we can use at most one coin (absolutely no repetitions). (We are not worried about finding the minimum set of coins which adds up to \( X \); we just want to make change; and also note that sometimes we may not be able to make an exact change). Design a dynamic programming algorithm running in \( O(nX) \) time, that can decide whether the change for the amount \( X \) can be made or not?

4. Assume that you have a chocolate bar of length \( n \) inches. You have a satisfaction chart, that indicates that if you eat a piece of length \( i \) inches, for \( 1 \leq i \leq n \), then you get a satisfaction of \( s_i \). We can assume that all the quantities involved \( (n, i, s_1, s_2, \ldots, s_n) \) are positive integers. You want to decide, using dynamic programming, what is the best way to make pieces of your chocolate bar so that you get a maximum satisfaction when you consume the whole bar. What is the time complexity of your dynamic programming algorithm?

For an example, suppose that the bar is 5-inches long, and if your satisfaction chart says:

<table>
<thead>
<tr>
<th>Length of Piece in inches (i)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfaction(s_i)</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

Then various ways to partition the 5-inch bar with its satisfaction values are as follows:

- \( 5 = 2 + 3 \), with satisfaction \( 7 + 9 = 16 \).
- \( 5 = 1 + 1 + 3 \), with satisfaction \( 2 + 2 + 9 = 13 \).
- \( 5 = 2 + 2 + 1 \), with satisfaction \( 7 + 7 + 2 = 16 \).
- No split, \( 5 = 5 \), with satisfaction \( 12 = 12 \).

(Hint: Get an actual chocolate bar, solve the problem correctly, and pass me the bill!)

5. Let \( T = (V, E) \) be a binary tree on \( n \) nodes. Note that \( T \) may not be a balanced tree. You want to find a subset of vertices \( S \subset V \), such that for each edge \( e = (uv) \in E \), atleast one of \( u \) or \( v \) is in \( S \). Design an algorithm, running in polynomial time in \( n \), that finds smallest such set \( S \) (i.e. \( |S| \) is minimized among all such subsets \( S \subset V \)).

6. State in your own words what are the complexity classes \( \mathcal{P} \), \( \mathcal{NP} \), \( \mathcal{NP} \)-Hard, and \( \mathcal{NP} \)-Complete. Give an example of a problem in each of these classes. Are all problems in \( \mathcal{P} \) in \( \mathcal{NP} \)?
7. Define what is a Vertex Cover and what is an Independent Set in a simple undirected graph \( G = (V, E) \). Define the decision versions of the problems of computing (a) a minimum vertex cover and (b) a largest Independent Set in \( G \). Provide a polynomial time reduction of the Vertex Cover problem to the Independent Set problem. Show that the algorithm that transforms one problem to the other problem runs in polynomial time, and it is a valid reduction (i.e., solution of one problem can be obtained from the solution of other problem).

8. Recall the SATISFIABILITY problem. Given \( n \)-Boolean variables, \( x_1, \ldots, x_n \), a Boolean formula \( \phi \) in the Conjunctive-Normal Form (CNF) is made of AND of \( k > 0 \) clauses \( \phi = C_1 \land C_2 \land C_3 \cdots \land C_k \), where each clause \( C_i \) is made of OR’s of one or more literals. Each literal is either a Boolean variable or its complement. For example, \( C_3 = (x_1 \lor \overline{x_4} \lor x_7) \) is a clause made of three literals. The SATISFIABILITY problem is whether we can find a satisfying assignment for \( \phi \), i.e., finding an assignment of Boolean values to \( x_1, \ldots, x_n \) that makes \( \phi \) true.

Given an undirected graph \( G = (V, E) \), we say that \( G \) can be 3-colored if we can assign one of the colors from \{Red, Blue, Green\} to each vertex so that for any edge \( e = (u, v) \in E \), \( u \) and \( v \) should get different colors. The decision version of the 3-coloring problem is to know whether there is an assignment of colors to vertices so that \( G \) is 3-colored.

Recall the definition of polynomial time reducibility and show that the 3-coloring problem is polynomial time reducible to the SATISFIABILITY problem.

Hint: Note that for each vertex \( v_i \), you need to assign it one of the colors from \{Red, Blue, Green\}. If \( v_i \) is assigned Red (Blue or Green) color, then we can say it’s corresponding Boolean variable is \( R_i \) (\( B_i \) or \( G_i \), respectively). Express the coloring constraints in terms of clauses. For example \( v_i \) needs to get at least one of the colors, therefore we can say that the corresponding clause is \( (R_i \cup B_i \cup G_i) \). Come up with expressions for no vertex receiving more than one color, no two neighboring vertices receiving the same color, etc.