CSE 551 Assignment 3
Solutions

March 17, 2021

Submission Instructions: Deadline is **11:59pm on 03/16**. Late sub-missions will be penalized, therefore please ensure that you submit (file upload is completed) before the deadline. Additionally, you can download the submitted file to verify if the file was uploaded correctly. Submit your answers electronically, in a single PDF, via *Canvas*. **Please type up the answers and keep in mind that we'll be checking for plagiarism**.
Furthermore, please note that the graders will grade 2 out of the 4 questions randomly. Therefore, if the grader decides to check questions 1 and 4, and you haven't answered question 4, you'll lose points for question 4. Hence, please answer all the questions.

1. Find the optimal order of multiplying four matrices whose sequence of dimensions is (13, 5, 89, 3, 34), i.e., the First matrix is 13 x 5, the second matrix is 5 x 89 and so on. Furthermore, compute the fewest number of multiplications needed to find the resulting matrix. Show all the steps of your computation.

   Solution: The optimal number of multiplications required is 2856. Let A be 13×5, B be 5×89, C be 89×3 and D be 3×34. The optimal order is $(A \times (B \times C)) \times D$.

2. Let $A_1, A_2, ..., A_n$ be matrices where the dimensions of A_i are $d_{i-1} * d_i$ for $i = 1, ..., n$. Here is a strategy to compute the best order (i.e., the order that requires the fewest number of multiplications) in which to perform the matrix multiplications to compute $A_1, A_2, ..., A_n$. At each step, choose the largest remaining dimension (from among $d_1, ..., d_{n-1}$) and multiply two adjacent matrices that share the same dimension. State with reasoning whether the above strategy will always minimize the number of multiplications necessary to multiply the matrix chain $A_1, ..., A_n$ (i.e., if you think that this will result in the optimal number of multiplications, then please provide arguments as to why you think so, else, provide a counter example).

   Solution: The above strategy will not result in the optimal number of multiplications. Here is the counter example. Consider the following matrices and their dimensions: M_1, M_2, M_3 with dimensions $(1 \times 1, 1 \times 2, 2 \times 3)$. The above method will multiply $(M_1 \times (M_2 \times M_3))$ resulting in 9 multiplications $(1 \times 2 \times 3 + 1 \times 1 \times 3)$. However, if we multiply $((M_1 \times M_2) \times M_3)$, we will end up with 8 multiplications $(1 \times 1 \times 2 + 1 \times 2 \times 3)$.

3. Using Dynamic Programming technique, find the optimal solution to the Traveling Salesman Problem for the following data set (Distance Matrix).

$$M = \begin{bmatrix} 0 & 2 & 5 & 9 \\ 10 & 0 & 2 & 8 \\ 3 & 8 & 0 & 6 \\ 9 & 2 & 6 & 0 \end{bmatrix}$$

   Show all your work. Also show the node ordering in the optimal tour.

   Solution: $g(2, \Phi) = 10$, $g(3, \Phi) = 3$, $g(4, \Phi) = 9$.
   $g(2, 3) = L_{23} + g(3, \Phi) = 5$
   $g(2, 4) = L_{24} + g(4, \Phi) = 17$
   $g(3, 2) = L_{32} + g(2, \Phi) = 18$
   $g(3, 4) = L_{34} + g(4, \Phi) = 15$
   $g(4, 2) = L_{42} + g(2, \Phi) = 12$
   $g(4, 3) = L_{43} + g(3, \Phi) = 9$

   $g(2, \{3, 4\}) = min(L_{23} + g(3, 4), L_{24} + g(4, 3)) = min(2 + 15, 8 + 9) = 17$
   $g(3, \{2, 4\}) = min(L_{32} + g(2, 4), L_{34} + g(4, 2)) = min(8 + 17, 6 + 12) = 18$
   $g(4, \{2, 3\}) = min(L_{42} + g(2, 3), L_{43} + g(3, 2)) = min(2 + 5, 6 + 18) = 7$

$g(1, \{2,3,4\}) = min(L_{12} + g(2,(3,4)), L_{13} + g(3,(2,4)), L_{14} + g(4,(2,3))) = min(2+17, 5+18, 9+7) = 16$

Optimal tour: $1 \to 4 \to 2 \to 3 \to 1$.

4. A subset of nodes in $S \subset V$ is a "Special Node Set (SNS)" of a graph $G = (V, E)$, if there are no edges between the nodes in S. The SNS of the largest cardinality is referred to as "Largest Special Node Set (LSNS)". For instance, in Fig. 1, the set $\{1, 5\}$ form an SNS but the set $\{1, 4, 5\}$ do not because there is an edge between nodes 4 and 5. The largest SNS set here is $\{2, 3, 6\}$.

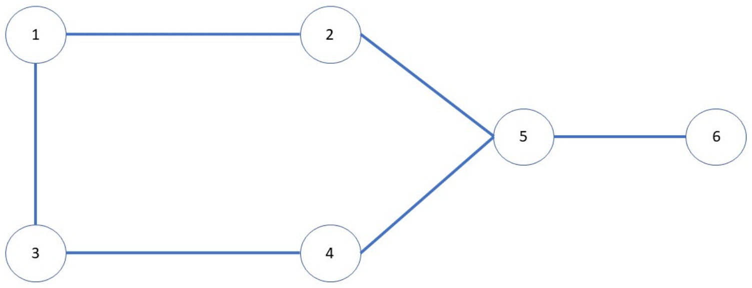

Figure 1: Figure for Q4

Design a Dynamic Programming based algorithm to compute the LSNS of a tree. Write down the recurrence relation on which your algorithm is based and show how the recurrence relation is being used by your algorithm to compute the LSNS of the tree. Analyze your algorithm to find it's complexity. **Show all our work**. Can your algorithm be used to find the LSNS of a graph which isn't a tree? If your answer is "yes", explain how it can be used, otherwise, explain why it can't be used to compute the LSNS of a graph which isn't a tree.

Solution: If the given graph is a tree, by using Dynamic Programming, we can show that the LSNS problem can be solved in linear time. Consider the following:

Root the tree at an arbitrary vertex. Now each vertex defines a subtree. Suppose that we proceed in a bottom up manner and we know the size of the LSNS of all subtrees below a node j. There can be two cases regarding the LSNS in the subtree rooted at j: either j is in the LSNS or it is not. If j is not in the solution, then the LSNS is simply the union of the LSNS of the subtrees of the children of j. However, if j is in the solution, then the LSNS includes j and the union of the LSNS of the grandchildren of j. The recursive equation can be written as follows:

$$I(j) := max\{ \sum_{k\ child\ of\ j} I(k),\ 1 + \sum_{k\ grandchild\ of\ j} I(k)\} \tag{1}$$

For each vertex, the algorithm only looks at its children and its grandchildren; hence, each vertex j is looked at only three times: when the algorithm is processing vertex j, when it is processing $j's$ parent, and when it is processing $j's$ grandparent. Since each vertex is looked at only a constant number of times, the total number of steps is $O(n)$.

In the case of trees, we know for sure that there are no cycles present. This is not the case for graphs. As a result, it will be difficult to isolate the children/grandchildren in the above recurrence relation, due to the presence of cycles. Therefore, the above algorithm cannot be used to compute the LSNS of a graph which isn't a tree.