HW7

Seyed Armin Vakil Ghahani
PSU ID: 914017982
CSE-565 Fall 2018
Collaboration with: Sara Mahdizadeh Shahri, Soheil Khadirsharbiyani, Muhammad Talha Imran

October 17, 2018

Problem 1. Longest path

Solution

- a) If we remove the edge (v_2, v_4) in the proposed example in the question, and add the edge (v_1, v_3), the longest path in the new graph is v_1, v_3, v_4, v_5. However, the proposed algorithm choose v_2 for its first iteration, and after that it chooses v_5. So, the output of the algorithm would be two that is incorrect.

- b) Suppose that $dp[i]$ equals to the longest path from v_i to v_n. Because the edges are all from left to right in this graph, we can calculate the value of $dp[i]$ from the following vertices with the following procedure:

$$dp[i] = max_{(v_j,v_i)\in E(G)} dp[j] + 1,$$

$$dp[n] = 0$$

  We should calculate the value of this array from n to 1. Hence, whenever we want a value of the outgoing vertex in this graph, it was calculated before, and we can use it because it is after the chosen vertex. Moreover the answer is located at $dp[1]$ that is defined as the longest path from v_1 to v_n.

- c) The time complexity of the proposed algorithm is $O(n+m)$ because for each vertex, we iterate on the outgoing vertices of it. So, the time complexity of the algorithm is the sum of the number of outgoing vertices for each vertex plus the $O(n)$ iteration on the vertices. As a result, the time complexity is $O(n + m)$.

Problem 2. Partition a string into words

Solution Suppose that $dp[i]$ equals to the maximum quality of the letter $y_i y_{i+1} ... y_n$. We can calculate the value of $dp[i]$ from the value of $dp[i+1], ..., dp[n]$ in $O(n)$. Suppose the word at the beginning of this letter is $y_i ... y_j$, so the value of $dp[i]$ can be $dp[j+1] + quality("y_i, ...y_j")$.

Consequently, if we calculate this value for each $j > i$ and get the maximum value from these, it will be the value of $dp[i]$. As a result, $dp[i]$ has the following recurrence formula:

$$dp[i] = max_{j=i}^{n} dp[j+1] + quality(''y_i, ...y_j''),$$

$$dp[n+1] = 0$$

The value of $dp[n+1]$ should be zero because it corresponds to the initial value of the algorithm which there is not any word in the letter, so the quality equals to zero. Moreover, the answer is located at $dp[1]$ which equals the maximum quality of $y_1...y_n$.

The time complexity of this algorithm is $O(n^2)$. However, if we know from the language that there is not any word longer than L, we can just check the value of $dp[i+1], ..., dp[i+L]$ to calculate the value of $dp[i]$, because we know that there is not any word longer than L. With this optimization, the time complexity of this algorithm would be $O(nL)$, which is better than $O(n^2)$ because L is constant.

Problem 3. Buy and sell

Solution Suppose that $dp[i]$ equals the minimum value of prices in days 1 to i. We can calculate this array by dynamic programming in $O(n)$ with the following procedure from the first entry ($dp[1]$) to the last entry ($dp[n]$):

$$dp[i] = min(dp[i-1], price(i)).$$

Now, when we have the minimum value of each segment from the beginning until each day, we can decide on each day that if we buy the product on previous days, how much can we profit if we sell the product at current day. It is because the profit at i^{th} day equals to the $price(i) - dp[i-1]$. We should calculate this value for all days from 2^{nd} until the n^{th} day, and consider the maximum value between them that is the maximum profit for the company.

The time complexity of this algorithm is $O(n)$ because at first, we should calculate the dp array that is $O(n)$, and after that, we should calculate the maximum value of the proposed formula for each day that is $O(n)$.