CSCI 570 - Fall 2019 - HW 7

Junzhe Liu, 2270250947

October 24, 2019

1 Graded Problems

1.1 Suppose you have a DAG with costs $c_e > 0$ on each edge and a distinguished vertex s. Give a dynamic programming algorithm to find the most expensive path in the graph that begins at s. Prove your algorithms runtime and correctness. For full credit, your algorithms runtime should be linear.

Algorithm 1 DAG most expensive path

1: From s do BFS, find the topological sorting: $s, a_1, a_2, \ldots, a_n$.
2: According to the topological order, complete the dist[] array:
3: **for** all the nodes a_i: **do**
4: $dist[a_i] = \max_j\{dist[a_j] + edge[a_j][a_i]\}$, here $j < i$, and a_j is the direct ancestor of a_i (there is a directed edge from a_j pointing to a_i).
5: **end for**
6: find the largest number in the dist array

Inspired by the fact that this graph is DAG, then a topological order can be acquired by the topological sorting. Considering that a node can be have multiple ancestors, we use a array $dist[v]$ to store the longest path length from s to node v and fill in this array with the topological order. According to the definition, when we're filling in $dist[v]$, all the ancestors of node v have been reached and thus by comparing them we can acquire the longest path from s to v.

1.2 Given a sequence $a_1, a_2, ..., a_n$ of n numbers, describe an $O(n^2)$ algorithm to find the longest monotonically increasing sub-sequence.

Let array $A[i]$ be the input array and $LIS[i]$ stores the length of the LIS such that $A[i]$ is the last element.
The recurrence function is:

$$LIS[i] = \begin{cases} \max_j \{LIS[j] + 1\} & i > j \text{ and } A[i] > A[j] \\ 1 & \text{if no such j exists} \end{cases} \tag{1}$$

Use bottom-up to fill in the LIS array, from 1 to n, the answer will be $\max_i LIS[i]$. Clearly this algorithm has a complexity of $O(n^2)$.

1.3 Suppose you are in Casino with your friend, and you are interested in playing a game against your friend by alternating turns. The game contains a row of n coins of values $v(i)$, where n is even. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money you can definitely win if you move first.

Let $v[i]$ be the input coins and $d[i][j]$ be the maximum amount of values you can receive for a sub-array from index i to j. Let us suppose that the opponent is as clever as the user, which means for every choice, one will pick away the coin that minimizes the value of the remaining sub-array for the opponent. The recurrence function is:

$$d[i][j] = \begin{cases} A[i] & \text{if } i = j \\ \max\{A[i], A[j]\} & \text{if } i = j-1 \\ \max\{v[i] + \min\{d[i+2][j], d[i+1][j-1]\}, v[j] + \min\{d[i][j-2], d[i+1][j-1]\}\} & \text{otherwise} \end{cases} \tag{2}$$

The solution for both the user and the opponent all compile to this function. For every pick in sub-array i, j, one can only pick $v[i]$ or $v[j]$, and let the remaining array for the opponent to pick. Since both of them tries to minimize the opponent's value, they will pick away the coin that the remaining array has the minimum value, therefore by comparing the value of the pick-away coin plus the minimum value of the remaining sub-array, one will choose to pick away the coin which maximizes the sum.

1.4 Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces.

Let $v[i]$ be the array that contains the price of a rod that is of the length i inches and let $p[i]$ be the maximum value that a rod with length i can sell. The recurrence function will be:

$$p[i] = \begin{cases} v[1] & i = 1 \\ \max\{v[i], \{\max_{j<i/2}\{p[j] + p[i-j]\}\} & otherwise \end{cases} \tag{3}$$

2 Practice Problems

2.1 Kleinberg and Tardos, Chapter 6, Exercise 5.

2.2 Kleinberg and Tardos, Chapter 6, Exercise 12.

2.3 Kleinberg and Tardos, Chapter 6, Exercise 20