CSE 331: Algorithm and Complexity	Summer 2020

Homework 4

Instructor: Xiangyu Guo **Deadline: Jul/27/2020**

Your Name: ______________________ Your Student ID: ________________

Problems	1	2	3	4	Total
Max. Score	10	20	20	20	70
Your Score					

Problem 1 (10 points). Consider the following optimum binary search tree instance. We have 5 elements e_1, e_2, e_3, e_4 and e_5 with $e_1 < e_2 < e_3 < e_4 < e_5$ and their frequencies are $f_1 = 5, f_2 = 25, f_3 = 15, f_4 = 10$ and $f_5 = 30$. Recall that the goal is to find a binary search tree for the 5 elements so as to minimize $\sum_{i=1}^{5} \text{depth}(e_i) f_i$, where $\text{depth}(e_i)$ is the depth of the element e_i in the tree. You need to output the best tree as well as its cost. You can try to complete the following tables and show the steps. In the two tables, $\text{opt}(i, j)$ is the cost of the best tree for the instance containing $e_i, e_{i+1}, ..., e_j$ and $\pi(i, j)$ is the root of the best tree.

$\text{opt}(i,j)$ \ j ; i	1	2	3	4	5
1	5				
2		25			
3			15		
4				10	
5					30

$\pi(i,j)$ \ j ; i	1	2	3	4	5
1	1				
2		2			
3			3		
4				4	
5					5

Table 1: *opt* and π tables for the optimum binary search tree instance. For cleanness of the table, we assume $opt(i, j) = 0$ if $j < i$ and there are not shown in the left table.

$$
\begin{aligned}
\text{opt}(1,2) &= \min\{0 + \text{opt}(2,2), \text{opt}(1,1) + 0\} + (f_1 + f_2) = \\
\text{opt}(2,3) &= \\
\text{opt}(3,4) &= \\
\text{opt}(4,5) &= \\
\text{opt}(1,3) &= \min\{0 + \text{opt}(2,3), \text{opt}(1,1) + \text{opt}(3,3), \text{opt}(1,2) + 0\} + (f_1 + f_2 + f_3) \\
&= \\
\text{opt}(2,4) &= \\
&= \\
\text{opt}(3,5) &= \\
&=
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{opt}(1,4) &= \min\{0+\mathrm{opt}(2,4), \mathrm{opt}(1,1)+\mathrm{opt}(3,4), \mathrm{opt}(1,2)+\mathrm{opt}(4,4), \mathrm{opt}(1,3)+0\} \\
&\qquad + (f_1+f_2+f_3+f_4) \\
&= \\
\mathrm{opt}(2,5) &= \\
\\
&= \\
\mathrm{opt}(1,5) &= \\
\\
&=
\end{aligned}
$$

Problem 2 (20 points) This problem asks for the maximum weighted independent set in a $2 \times n$ size grid. Formally, the set of vertices in the input graph G is $V = \{1,2\} \times \{1,2,3,\cdots,n\} = \big\{(r,c) : r \in \{1,2\}, c \in \{1,2,3,\cdots,n\}\big\}$. Two different vertices (r,c) and (r',c') in V are adjacent in G if and only if $|r-r'|+|c-c'|=1$. For every vertex $(r,c) \in V$, we are given the weight $w_{r,c} \geq 0$ of the vertex. The goal of the problem is to find an independent set of G with the maximum total weight. (Recall that $S \subseteq V$ is an independent set if there are no edges between any two vertices in S.) See Figure 1 for an example of an instance of the problem.

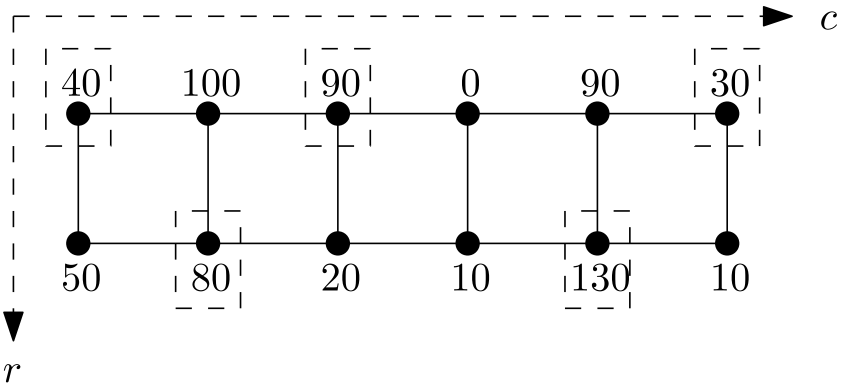

Figure 1: A maximum weighted independent set instance on a 2×6-grid. The weights of the vertices are given by the numbers. The vertices in rectangles form the maximum weighted independent set, with a total weight of 370.

Design an $O(n)$-time dynamic programming algorithm to solve the problem. For simplicity, you only need to output the *weight* of the maximum weighted independent set, not the actual set.

(**Hint:** If you could not solve the above problem, you can first try to solve the simpler problem when the grid size is $1 \times n$ instead of $2 \times n$ (that is, the input graph is a path on n verticies))

Problem 3 (20 points) You are managing the construction of billboards on the highway I-90, which runs west- east for M miles in New York State. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$, each in the interval $[0 \ldots M]$ (specifying their position along I-90, measured in miles from its west end.)

- If you place a billboard at location x_i (for each $1 \le i \le n$), you receive a revenue $r_i > 0$.
- The regulations imposed by NYS Highway Department require that two billboards must be placed at least > 5 miles apart.

We want to decide the locations to place the billboards to that the total revenue is maximized.

Formally, the input consists of two length-n arrays X and R, where $X[i]$ is the ith position that you can choose to put a billboard, and $R[i]$ is the revenue you get if you put the billboard at $X[i]$.

Example: $n = 4, M = 20, X = [x_1, x_2, x_3, x_4] = [6, 7, 12, 14]$ and $R = [r_1, r_2, r_3, r_4] = [5, 6, 5, 1]$. It can be checked the optimal solution would be to place the billboards at x_1 and x_3 with a total revenue of 10.

Describe a Dynamic Programming algorithm for solving this problem in $O(n)$ time. For simplicity, your algorithm only need to output the maximum *revenue*, not the actual locations to place the billboards.

Problem 4 (20 points). Given a sequence $A[1 .. n]$ of numbers, we say that A is an N-shaped sequence if there are two indices i, j such that $1 < i < j < n$ and

- $A[1] < A[2] < A[3] < \cdots < A[i]$,
- $A[i] > A[i+1] > A[i+2] > \cdots > A[j]$,
- $A[j] < A[j+1] < A[j+2] < \cdots < A[n]$.

For example $(3, 6, 9, 12, 11, 10, 12, 13, 17)$ is an N-shaped sequence.

Design an polynomial-time algorithm that, given an array A of n numbers, outputs the *length* of the longest N-shaped subsequence of A. (If no such subsequence exists, your algorithm can output $-\infty$). For example, if the input sequence is $(3, 1, 4, 6, 5, 7, 2)$, your algorithm should output 5 (because $(3, 4, 6, 5, 7)$ is the longest N-shaped subsequence).

You will get all the points if the running time of your algorithm is $O(n^2)$.

(**Hint:** Divide the task into three folds: (1)Finding the longest increasing subsequence; (2) Finding the longest Λ-shaped subsequence using the information of (1); (3) Finding the longest N-shaped subsequence using the information of (2))