CSE 431/531: Algorithm Analysis and Design Spring 2020

Homework 4

Instructor: Shi Li **Deadline: 4/22/2020**

Your Name: _____ Your Student ID: _____

Problems	1	2	3	Total
Max. Score	14	18	18	50
Your Score				

Problem 1 (14 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, $opt(i, j)$ 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.

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

$\pi(i,j)$ i \ j	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.

$opt(1,2) = \min\{0 + opt(2,2), opt(1,1) + 0\} + (f_1 + f_2) =$

$opt(2,3) =$

$opt(3,4) =$

$opt(4,5) =$

$opt(1,3) = \min\{0 + opt(2,3), opt(1,1) + opt(3,3), opt(1,2) + 0\} + (f_1 + f_2 + f_3)$

$=$

$opt(2,4) =$

$=$

$opt(3,5) =$

$=$

$$opt(1,4) = \min\{0 + opt(2,4), opt(1,1) + opt(3,4), opt(1,2) + opt(4,4), opt(1,3) + 0\}$$
$$+ (f_1 + f_2 + f_3 + f_4)$$
$$=$$
$$opt(2,5) =$$

$$=$$
$$opt(1,5) =$$

$$=$$

Problem 2 (18 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.

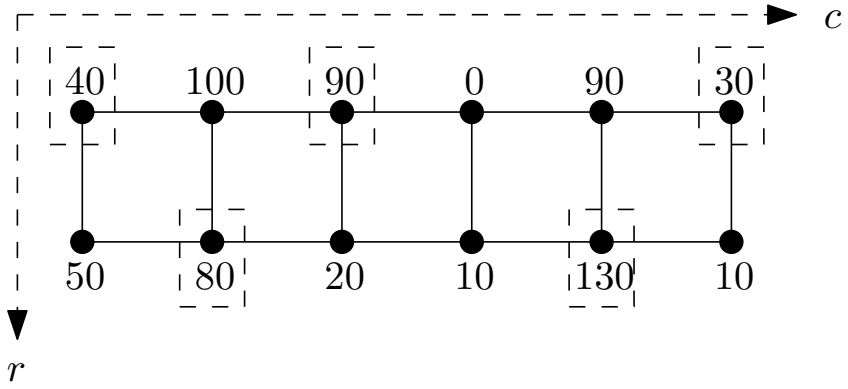

Figure 1: A maximum weighted independent set instance on a 2×6-grid. The weights of the vertices are given by the numbers. The vertices in rectangles form the maximum weighted independent set, with a total weight of 370.

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.

If you could not solve the above problem, you can 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 (18 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 ($(3, 4, 6, 5, 7)$ is the longest N-shaped subsequence).

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