Homework 6 solution

1. *Give a dynamic programming algorithm for the following problem. The input is an n sided convex polygon. Assume that the polygon is specified by the Cartesian coordinates of its vertices. The output should be the triangulation of the polygon in to $n-2$ triangles that minimizes the sums of the cuts required to create the triangles. Analyze the time complexity of your algorithm.*

   We name the polygon vertex from 1 to n.

   Let $d(i,j)$ denote the Euclidean distance between i and j.

   Let $S(i,j)$ denote the optimal sum of cuts of the polygon $\{i, i+1, \cdots, j\}$

   Then $S(i,j)$ can be computed as:

   $$\min\{S(i,k) + S(k,j) + d(i,k) + d(k,j)\} \text{ where } i+1 \le k \le j-1$$

   And $S(i,j) = 0$ if $j \le i+2$

   Build the DP matrix S and our goal is to compute $S(1,n)$

   Time complexity analysis:

   Computing the matrix needs $O(n^2)$ operations. To compute each cell of the matrix takes $O(n)$ operations. Thus the time complexity should be $O(n^3)$

2. *Give a dynamic programming algorithm to find the longest monotonically increasing subsequence of a sequence of n elements. Apply your algorithm to a sequence $A =< 1,3,2,4,6,13,14,15,5,6,8,12,13 >$. Note that in this example, sequence $< 1,3,4,6,13,14,15 >$ and $< 1,2,4,5,6,8,12,13 >$ both are monotonically increasing subsequence of A.*

   Longest Increasing Subsequence (LIS) problem

   Let $L(i)$ denote the longest increasing subsequence ending in i. Then $L(j)$ can be computed as follows:

   $$L(j+1) = \max\{L(i)\} + 1 \text{ iff } A(j+1) > A(i), \text{ where } 1 \le i \le j$$

   Apply Dynamic Programming according to the above formula.

3. *Consider a 2-D map with a horizontal river passing through its center. There are n cities on the southern bank with x-ccoordinates $a(1), \cdots, a(n)$ and n cites on the northern bank with x-coordinates $b(1), \cdots, b(n)$. You want to connect as many north-south pairs of cities as possible with bridges such that no two bridges cross. When connecting cites, you can only connect city i on the northern bank to city i on the southern bank. Give a dynamic programming algorithm to solve this problem and analyze the time complexity of your algorithm. Note that those x-coordinate values are not sorted, i.e. $a(i)$'s and $b(i)$'s is in an arbitrary order.*

   Sort according to southern bank i.e. move the position of $b(i)$ in the way how you move $a(i)$ during the sorting, and this can be convert into LIS problem. And use the solution of Q2 to find the LIS on the northern bank.

4. *Give a polynomial time algorithm for the following problem. The input consists of a sequence $R = R_1, \cdots, R_n$ of non-negative integers, and an integer k. The number R_i represents the number of users requesting some piece of information at time i (say from a www server). If the server broadcasts this information at some time t, the requests from all the users who requested the information strictly before time t have already been*

satisfied, and requests arrived at time t will receive service at the next broadcast time. The server can broadcast this information at most k times. The goal is to pick the k times to broadcast in order to minimize the total time (over all requests) that requests/users have to wait in order to have their requests satisfied. As an example, assume that the input wat $R = 3,4,0,5,2,7$ (so $n = 6$) and $k = 3$. Then one possible solution (there is no claim that this is the optimal solution) would be to broadcast at times $2,4,7$ (note that it is obvious that in every optimal schedule that there is a broadcast at time $n + 1$ if $R_n \neq 0$). The 2 requests at time 5 would then have to wait 2 time units. The 7 requests at time 6 would then have to wait 1 time units. Thus the total waiting time for this solution would be

$$3 * 1 + 4 * 2 + 5 * 3 + 2 * 2 + 7 * 1$$

Let $W(m, j)$ denote the optimal total waiting time of $R_1, \cdots, R_m$ and using j broadcasts where $R_m \neq 0$. Then $W(m, j)$ can be computed as the following formula:

$$W(m, j) = \min_{j \le i \le m} (W(i, j-1) + \Sigma_{a=i}^{m} (m + 1 - a) \cdot R_a)$$

Let $C(i) = \Sigma_{a=i}^{m} (m + 1 - a) \cdot R_a$

And we should compute DP matrix array W and DP array C. And our goal is to get $W(n, k)$.

Time complexity is $O(n^2)$