CS/ECE 374 Spring 2017
Homework 5 Problem 3

Renheng Ruan (rruan2)
Lanxiao Bai (lbai5)
Ho Yin Au (hoyinau2)

The McKing chain wants to open several restaurants along Red street in Shampoo-Banana. The possible locations are at $L_1, L_2, \ldots, L_n$ where L_i is at distance m_i meters from the start of Red street. Assume that the street is a straight line and the locations are in increasing order of distance from the starting point (thus $0 \leq m_1 < m_2 < \ldots < m_n$). McKing has collected some data indicating that opening a restaurant at location L_i will yield a profit of p_i independent of where the other restaurants are located. However, the city of Shampoo-Banana has a zoning law which requires that any two McKing locations should be D or more meters apart. Describe an algorithm that McKing can use to figure out the maximum profit it can obtain by opening restaurants while satisfying the city's zoning law.

Solution:

Let $MaxP(i, j)$ be the maximum profit for $0 \leq i \leq n$ and $0 \leq j \leq k$ by opening at most j restaurant in the location 1 to i such that the restaurants are at least D meters apart as required in the question.
Base Case: $MaxP(i, 0) = 0$ for all i and $MaxP(0, j) = 0$ for all j.
Let $MaxIndex(i)$ be the largest index $a < i$ such that $m_i - m_a \geq D$. We have the maximum profit $MaxP(i-1, j)$ if we do not open a restaurant at i. We also have the maximum profit $MaxP(MaxIndex(i), j-1) + p_i$ if we do open a restaurant at i and the nearest location to the left of i is $Maxindex(i)$, and in this case, we can open at most $j-1$ restaurant from 1 to $MaxIndex(i)$. p_i is the profit for the restaurant at i and $MaxP(MaxIndex(i), j-1)$ is the profit by opening at most $j-1$ restaurant in the location $MaxIndex(i)$ to $j-1$.

Thus, we have

$$MaxP(i, j) = \max\{MaxP(i-1, j), MaxP(MaxIndex(i), j-1) + p_i\}$$

We want to compute $MaxP(n, k)$. At first, $MaxIndex(i)$ takes $O(nlogn)$ time for sorting the m_i values. And then the time to compute $MaxP(i, j)$ from previously computed values is $O(1)$. However, the number of subproblems is $O(nk)$.

Thus, the algorithm takes $\boldsymbol{O(nk + nlogn)}$ time and $\boldsymbol{O(nk)}$ space.

The solution template is taken from https://www.coursehero.com/file/9896833/Sprin12-midterm2-solns/.

■