Recurrence Relations for Yuckdonalds Problem

exercise 6.3 from DPV

Yuckdonalds is considering opening a series of restaurants along Quaint Valley Highway (QVH). The \( n \) possible locations are along a straight line, and the distances of these locations from the start of QVH are, in miles and in increasing order, \( m_1, m_2, \ldots, m_n \). The constraints are as follows:

- At each location, Yuckdonalds may open at most one restaurant. The expected profit from opening a restaurant at location \( i \) is \( p_i \), where \( p_i > 0 \) and \( i = 1, 2, \ldots, n \).
- Any two restaurants should be at least \( k \) miles apart, where \( k \) is a positive integer.

Give an efficient algorithm to compute the maximum expected total profit subject to the given constraints.

**comment:** In both versions below we add a restaurant location at milepost \( m_0 = -\infty \) with profit \( p_0 = 0 \).

**version 1**

**subproblem:** \( RP(i) \) is the maximum profit available by placing restaurants at locations chosen from mileposts \( m_0, m_1, \ldots, m_i \) subject to the restrictions

- each restaurant is at least \( k \) miles from the nearest neighbor
- (important) a restaurant is placed at location \( i \)

**recurrence:**

\[
RP(i) = \begin{cases} 
0 & \text{if } i = 0 \\
p_i + \max \{ \text{ } RP(j) \mid 0 \leq j < i \text{ and } m_i - m_j \geq k \} & \text{otherwise.} 
\end{cases}
\]

**desired output:** \( \max_{1 \leq i \leq n} RP(i) \)

**implicit time:** \( O(n^2) \)

**version 2**

**subproblem:** \( RP(i) \) is the maximum profit available by placing restaurants at locations chosen from mileposts \( m_0, m_1, \ldots, m_i \) subject to the **one** restriction
- each restaurant is at least \( k \) miles from the nearest neighbor

\textit{define:} Let \( \pi(i) \) be the largest \( j < i \) such that \( m_i - m_j \leq k \) (the closest previous location before \( i \) that is at least \( k \) miles away). \textbf{Note:} all the \( \pi(i) \) values could be pre-computed in \( O(n) \) time. (I think!)

\textit{recurrence:}

\[
RP(i) = \begin{cases} 
0 & \text{if } i = 0 \\
\max[RP(i-1), p_i + RP(\pi(i))] & \text{otherwise.}
\end{cases}
\]

\textit{desired output:} \( RP(n) \)

\textit{implicit time:} \( O(n) \)