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:**

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

**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

**define:** Let $\pi(i)$ be the largest $j < i$ such that $m_i - m_j \geq k$ (the closest previous location before $i$ that is at least $k$ miles away). **Note:** all the $\pi(i)$ values could be precomputed in $O(n)$ time. (I think!)

**recurrence:**

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

**desired output:** $RP(n)$

**implicit time:** $O(n)$

**iterative code for version 2**

The following pseudo-code will compute $RP$ in a bottom-up (iterative) manner, storing the values in an array. It will not pre-compute $\pi$, but will do so on the fly. The time bound will be $O(n^2)$ this way.

**input is provided by**
- int $k$ (spacing)
- int arrays $p[0..n]$ (profits), and
- $m[0..n]$ (mileposts)

**we use arrays**
- $RP[0..n]$ (max-profit so far) and
- $PI[1..n]$ (closest possible previous loc)

$RP[0]=0$

for $i=1$ to $n$

\[ j = i-1 \]

while $m[i]-m[j]<k$ do $j=j-1$

$PI[i]=j$

if ( $RP[i-1] \geq RP[PI[i]]+p[i]$ )

then $RP[i] = RP[i-1]$

else $RP[i] = RP[PI[i]]+p[i]$

return $RP[n]$

**modified code for version 2**

To determine the locations chosen, we include a boolean array $USE$ to save the choices.
RP[0]=0

for i=1 to n
    j=i-1
    while m[i]-m[j]<k do j=j-1
    PI[i]=j
    if ( RP[i-1] >= RP[PI[i]]+p[i] )
        then
            RP[i] = RP[i-1]
            USE[i] = false
        else
            RP[i] = RP[PI[i]]+p[i]
            USE[i] = true

i=n
while i>0
    if USE[i]
        then
            report 'place a restaurant at location i at milepost m[i]'
            i = PI[i]
    else
        i = i-1