Homework 8 Solution

1. *You are climbing a stair case. It takes* n *steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top? Give a dynamic programming algorithm to solve this problem. Note:* n *is a positive integer.*

   To climb to stair i, the previous status can only be on stair $i-1$ and stair $(i-2)$. Let $S(i)$ denote the distinct ways to climb to stair i. Thus, we can use the formula:

   $$S(i) = S(i-1) + S(i-2)$$

   Note that $S(0) = 1, S(1) = 1$.
   Build the DP array, and our goal is to compute $S(n)$.
   This is a linear time algorithm.

2. *You are given an array* $A = (a_1, a_2, \cdots, a_n)$ *of* n *positive integers. The objective is to find a pair* (a_i, a_j) *such that* $i < j$ *and* $d_j - d_i$ *is maximum and positive. For example, for* $A = (9,2,4,3,8)$*, a solution is* $(2,8)$*. For* $A = (9,6,5,3,1)$*, there is no such a pair. Find a dynamic programming algorithm to solve this problem.*

   Let $M(i)$ denote the minimum value of $(a_1, a_2, \cdots, a_i)$.
   Let $S(i)$ denote the optimal solution of $(a_1, a_2, \cdots, a_i)$. Noted that $S(i)$ is difference between the pair.
   Then we can have the following two formulas:

   $$M(i) = \begin{cases} a_i \text{ if } a_i < M(i-1) \\ M(i-1) \text{ otherwise} \end{cases}$$

   $$S(i) = \begin{cases} a_i \text{ if } a_i - M(i) > S(i-1) \\ S(i-1) \text{ otherwise} \end{cases}$$

   Build two DP array M and S. And set $M(1) = a_i, S(1) = -\infty$. The goal is to compute $M(n)$ and $S(n)$.
   This is a linear time algorithm.

3. *Given two strings* a *and* b*, check if* a *is a subsequence (not necessarily consequent subsequence) of* b*.*
   Example:
   a *=aegis76*
   b *=evatookplanetohamburgonsomedayin1976*
   Then a *is a subsequence of* b*.*
   Give a dynamic programming algorithm to solve this problem.

   Let $M(i)$ denote how many string does A match with $(b_1, b_2, \cdots, b_i)$.
   Then we have the formula:

   $$M(i) = \begin{cases} S(i-1), \text{if } a_{M(i-1)+1} \neq b_i \\ S(i-1) + 1 \text{ otherwise} \end{cases}$$

   Initially $M(1) = 1$ if $a_1 = b_1$, $M(1) = 0$ otherwise. Our goal is to compute $M(n)$. If $M(n) = m$, then it is true, or it is false. This is a linear time algorithm.

4. *Given a string s, find the longest palindromic subsequence (LPS) length in s. Example: s =axbddb, the LPS length is 4. Give a dynamic programming algorithm to solve this problem.*

Let $L(i,j)$ denote the LPS length between i and j.

Then we have:

$$L(i,j) = \begin{cases} L(i+1,j-1)+2 \text{ if } s(i)=s(j) \\ \max\{L(i,j-1), L(i+1,j)\} \text{ otherwise} \end{cases}$$

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

Time complexity analysis: each block takes $O(1)$ to compute. Thus the algorithm takes $O(n^2)$