# Dynamic Programming Practice

**Problem 1.** Manhattan Tourist Problem

Imagine walking through Manhattan in New York City. You may see the Empire State Building, the World Trade Center memorial, Wall Street, FAO Schwarz, Macy's, Carnegie Hall, Broadway, Times Square, Central Park, etc. The streets of Manhattan are mostly arranged in a rectangular grid, similar to what you see below. You are starting at the northwest corner (source) and will walk south and east to the southeast corner (destination) without ever going west or north. Along the way, you can walk by various sites of interest; the number of such sites on each block is indicated next to the arrow. The question is: What route through the streets will take you by the largest number of sites of interest? Follow the arrows, but make good choices about which route to take.

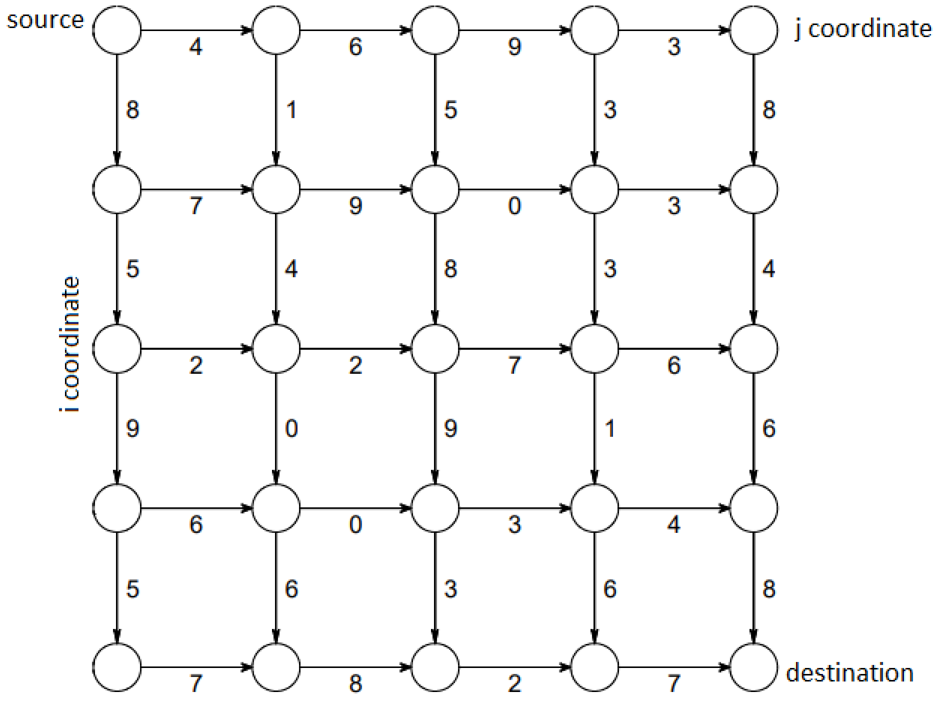

Figure 1: Problem 1.

(a) Work through the example in the picture, and determine the maximum number of attractions you could see.

(b) Present a recursive DP formulation to solve this problem. (Just the rule; you don't need to write any pseudo-code).
Be sure to include (1) the basis case(s), and (2) how to obtain the final answer given your formulation.
Let $h[i][j]$ = attractions between $h[i][j-1]$ and $h[i][j]$ ($h$ = horizontal weights)
Let $v[i][j]$ = attractions between $v[i-1][j]$ and $v[i][j]$ ($v$ = vertical weights)
Let $M(i,j)$ = the max number of attractions you can pass on the path from $(0,0)$ to $(i,j)$.

**Problem 2.** Interview Questions

The following prompts, or some variation of them, show up in technical interviews quite often. The key to solving each of these efficiently is to use dynamic programming.

For each of the following, derive the recursive DP formulation to solve the problem. Just as above, you only need to derive the rule, not the code.

(a) **Longest Increasing Subsequence:** Given a sequence $X = \langle x_1, ..., x_n \rangle$, an increasing subsequence is any subsequence of elements of $X$ that are in strictly increasing order. The longest increasing subsequence (LIS) is the increasing subsequence of maximum length. For example, the LIS of $X = \langle 10, 22, 9, 33, 21, 50, 41, 60, 80 \rangle$ is $\langle 10, 22, 33, 50, 60, 80 \rangle$, and its length is 6. Present an efficient algorithm, which given an $n$-element sequence computes the length of its LIS.

(b) **Longest Palindromic Substring:** Given a string, find the length of the longest substring which is a palindrome. For example, if the given string is "forgeeksskeegfor", the output should be 11 since "geeksskeeg" is the longest palindromic substring.

(c) **Double Knapsack:** Given an array $A$ containing the weight of $n$ distinct items, and two knapsacks that can withstand $W_1$ and $W_2$ weights, the task is to find the sum of the largest subset of the array $A$, that can be fit in the two knapsacks. You may not break any items in two, i.e an item should be put in one of the bags as a whole. For example, if $A = \langle 8, 3, 2 \rangle$, $W_1 = 10$, $W_2 = 3$, your output should be 13. Another example: if $A = \langle 8, 5, 3 \rangle, W_1 = 10, W_2 = 3$, your output should be 11.