HW6 Solutions

For the simplicity, we only describe the recursions to be used in the dynamic programming. After the recursion is achieved, a standard dynamic programming with memoization is used to solve the problem.

1. Let $P(i)$ be the maximum profit from locations 1 to i. Charging stations should be set only k distance apart, so we get the following recurrence relation to find the maximum profit.

$$P(i) = \begin{cases} \max\Big(p_i + P(j), P(i+1)\Big) & i <= n \\ 0 & i > n \end{cases}$$

where j is the first location that is at least k distance from s_i.

2. Let $V(\mathcal{P}, M)$ be the maximum value one can get when the set of patents $\mathcal{P}$ is available for purchase and the budget is M.

$$V(\mathcal{P}, M) = \begin{cases} \max\Big(V(\mathcal{P} - \{p_{1^*}\}, M - m_{1^*}), V(\mathcal{P} - \{p_{1^*}\}, M)\Big) & |\mathcal{P}| > 0, M > 0 \\ 0 & |\mathcal{P}| = 0 \\ 0 & M \leq 0 \end{cases}$$

where p_{1^*} is the first patent in the set M.

3. Let $M(i,j)$ be the length of the longest palindrome from the string $x_i, \cdots, x_j$. Then, Recurrence for the maximum length palindrome substring can be defined as following:

$$M(i,j) = \begin{cases} M(i+1, j-1) + 2 & i < j, x_i = x_j \\ \max\Big(M(i+1,j), M(i,j-1)\Big) & i < j, x_i \neq x_j \\ 1 & i = j \end{cases}$$

4. Let's suppose the points are sorted based on their x coordinates in increasing order, say $v_1, v_2, \cdots, v_n$. Then, we need to find a right-going path from v_1 to v_n, then another disjoint left-coming **disjoint** (except v_1 and v_n) path from v_n to v_1, and we need to make sure all nodes are visited by the two disjoint paths while minimizing the total distance. (The reason we only consider disjoint paths is trivial – triangle inequality)

   Let $BTSP(i,j)$ denote the minimum total length of the two disjoint paths from v_1 to v_i and v_1 to v_k. Then, when $i = k = n$, $BTSP(n,n)$ will be the minimum total length of a path from v_1 to v_n and then a path back from v_n to v_1, which is the optimal bitonic tour. Note that $BTSP(i,j) = BTSP(j,i)$ for any i, j based on the definition.

$$BTSP(i,j) = \begin{cases} BTSP(i-1,j) + d(i-1,i) & i > j+1 \\ \min_{1 \leq k < j}\Big(BTSP(k,j) + d(k,j+1)\Big) & i = j+1 \\ \min_{1 \leq k < i}\Big(BTSP(i-1,k) + d(i-1,i) + d(k,i)\Big) & i = j \\ 0 & i = j = 0 \end{cases}$$

   Then we have the above recursion, and here are the explanations.

(a) $i > j+1$: The minimum distance path from v_1 to v_i must have been formed from the minimum distance path from v_1 to v_{i-1} plus the edge $e(i-1, i)$ because $i > j+1$, and v_j is at the left side of v_{i-1}.

(b) $i = j+1$: $v_i = v_{j+1}$. Then, since the paths need to be disjoint except the start and the end, it is not possible that the path from v_1 to v_{j+1} contain the node v_j since it is already in the other path from v_1 to v_j. Therefore, the path from v_1 to v_{j+1} could have been formed only by from a path v_1 to v_k where $k < j$ plus the edge $e(k, j+1)$.

(c) $i = j$: In this case, one of the two path must contain v_{i-1} (*i.e.,* contain $e(i-1, i)$) since all nodes must be visited, and the other path must have been formed by some path from v_1 to v_k plus the edge $e(k, i)$.

Note that $BTSP$ is symmetric, therefore we don't need to discuss $i = j-1$ (which is identical to the case where $j = i-1 \leftrightarrow i = j+1$ due to the symmetry) or $i < j-1$.

Time Complexity of finding Optimal Bitonic Path = Time complexity of sorting the points + Time complexity to fill matrix $BTSP(i, j)$ for all $1 \leq i, j \leq n$.

Sorting n points: $O(n \log n)$.

Filling the matrix: $O(n^2)$ because only the items in the diagonal line requires min function $O(n)$, and everything else can be calculated directly based on the adjacent cell ($O(1)$). That means, we have approximately $O(n)$ cells that need $O(n)$ operations and $O(n^2)$ cells that need $O(1)$ operations $\Rightarrow$ $O(n \cdot n + n^2 \cdot 1) = O(n^2)$.