1. (20) Let $G = (V, E, w)$ be a weighted undirected graph. Let U be a subset of V, such that $G[V - U]$ is connected. Design an algorithm to find a minimum spanning tree of G in which all vertexes in U are leaves. Note that the spanning tree may have other leaves, but the weight of the tree should be minimized.

2. (20) Let $G = (V, E, w)$ be a connected weighted undirected graph. Given a vertex $s \in V$ and a shortest path tree T_s with respect to the source s, design a linear time algorithm for checking whether the shortest path tree T_s is correct or not.

3. (20) Let $X = x_1, x_2, \ldots, x_n$ be a sequence of n integers. A sub-sequence of X is a sequence obtained from X by deleting some elements. Give an $O(n^2)$ algorithm to find the longest monotonically increasing sub-sequences of X.

4. (20) Give a dynamic programming solution to the $\{0, 1\}$-knapsack problem that runs in $O(nW)$ time, where n is the number of items and W is the maximum weight of items that can put in the knapsack.

5. (40) Given a string of characters $s_1 s_2 \ldots s_n$. It is believed that the string is a document in which all spaces and punctuations have been removed. The following methods can be used to insert spaces back into the document.

   (a) Constructed a graph from the string $s_1 s_2 \ldots s_n$ and then find a path in the graph.

   (b) Solve the problem by dynamic programming.

   Use the following string

   `wewillmeetatmidnight`

   to explain the above two methods. You may assume that the only words are

   `a at me meet mid midnight night we will`

6. (20) Let $M_1, M_2, \ldots, M_n$ be a sequence of matrices. Each matrix M_i has dimension $r_{i-1} \times r_i$. The minimum cost of multiplying matrices $M_1 \times M_2 \times \ldots \times M_n$ can be computed as follows. Let $c_{i,j}$, $i \le j$, be the minimum cost of multiplying matrices $M_i \times M_{i+1} \times \ldots \times M_j$. Then we can show that

$$c_{i,j} = \begin{cases} 0 & \text{if } j = i \\ \min\limits_{i \le k < j} \{c_{i,k} + c_{k+1,j} + r_{i-1} r_k r_j\} & \text{if } j > i. \end{cases}$$

   (a) (10) Design an efficient algorithm to compute these $c_{i,j}$'s in a way that when $c_{i,j}$ is to be computed, all the $c_{i,k}$'s and all the $c_{k+1,j}$'s which are needed have already been computed.

   (b) (5) Use $n = 8$ as an example to show the order of the $c_{i,j}$'s are computed by your algorithm.

   (c) (5) Which $c_{i,j}$ is the minimum cost to multiply these matrices?