1. (40 points) Consider the following problem. The input consists of a graph $G = (V, E)$, where each edge $e \in E$ has an associated positive weight $w_e$. The output should be a collection $S$ of edges, such that the subgraph $G = (V, S)$ is connected, and the sum of the weights of the edges in $S$ is minimized. For each of the following greedy algorithms, either prove that the algorithm correctly solves the problem, or prove that the algorithm does not correctly solve the problem. Your proof of correctness must use an exchange argument.

(a) **Greedy Algorithm UP**: Consider the edges from smallest weight to largest weight. Then add an edge $e$ to $S$ if and only if $S \cup \{e\}$ does not contain a cycle.

(b) **Greedy Algorithm DOWN**: Consider the edges from largest weight to smallest weight. Then add an edge $e$ to $S$ if and only if $S \cup \{e\}$ does not contain a cycle.

2. (20 points) Consider the problem where the input is $n$ keys $K_1, \ldots, K_n$, where $K_1 < K_2 < \ldots < K_n$, and $n$ associated probabilities $p_1, \ldots, p_n$, where $\sum_{i=1}^{n} p_i = 1$. Here $p_i$ is the probability that key $K_i$ is accessed. The output should be the minimum possible expected access time for binary search tree $T$ on these keys. The expected access time for a tree $T$ is the sum over the keys of the depth of the key in $T$ times the probability that key is accessed, $\sum_{i=1}^{n} (\text{depth of } K_i \text{ in } T) \times (p_i)$.

(a) Let $A(i, j)$ be the minimum possible expected access time. Give pseudo-code for a recursive algorithm that computes $A(i, j)$.

(b) Show that the running time of your algorithm is exponential.

(c) Give pseudo-code for a polynomial-time iterative algorithm that computes $A(i, j)$.

(d) Assume that you have 4 keys with probabilities $p_1 = 0.4$, $p_2 = 0.3$, $p_3 = 0.1$ and $p_4 = 0.2$. Fill in the last entry in following table using your algorithm. Show your work.

<table>
<thead>
<tr>
<th></th>
<th>j=1</th>
<th>j=2</th>
<th>j=3</th>
<th>j=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td>.4</td>
<td>.3</td>
<td>.1</td>
<td>.2</td>
</tr>
<tr>
<td>i=2</td>
<td></td>
<td>.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>i=3</td>
<td></td>
<td></td>
<td>.4</td>
<td></td>
</tr>
<tr>
<td>i=4</td>
<td></td>
<td></td>
<td></td>
<td>.2</td>
</tr>
</tbody>
</table>

Table 1: Table for $A(i, j)$
3. (20 points) Consider the following problem:

Input: set of positive integers \( V = \{v_1, \ldots, v_n\} \)

Output: A subset \( S \) of \( V \) such that \( \sum_{i \in S} (v_i)^7 = \prod_{i \in S} v_i \).

Give dynamic programming algorithm whose running time is polynomial in \( n + L \), where \( L = \max(\sum_{i=1}^{n} (v_i)^7, \prod_{i=1}^{n} v_i) \). You can get 15 points for giving an algorithm to determine whether such a set \( S \) exists. For the full 20 points, you need to explain how to actually construct \( S \).

4. (20 points) The input to this problem is \( n \) points \( x_1, \ldots, x_n \) on a line, and \( n \) associated deadline times \( d_1, \ldots, d_n \). A feasible solution is a path/walk \( P \) that starts at the origin, and visits each point \( x_i \) after traveling at most \( d_i \) units on \( P \). Alternatively, if a unit speed vehicle followed \( P \), then each point \( x_i \) must be reached by time \( d_i \). Note that \( P \) need not be simple, that is, it can backtrack over territory that it has already covered. The objective of the problem is to determine the length of the shortest path \( P \) satisfying these constraints, or to determine that there is no path that satisfies these constraints. Give a polynomial time dynamic programming algorithm for this problem.

So for example:

- If the input points were \( x_1 = 1, x_2 = 2 \) and \( x_3 = -1 \), and the deadlines where \( d_1 = 3 \), \( d_2 = 2 \), and \( d_3 = 6 \), then the shortest feasible path is of length 5. This path would move from the origin to \( x_2 \) and then to \( x_3 \).

- If the input points were \( x_1 = 1, x_2 = 2 \) and \( x_3 = -1 \), and the deadlines where \( d_1 = 3 \), \( d_2 = 5 \), and \( d_3 = 6 \), then the shortest feasible path is of length 4. This path would move from the origin to \( x_3 \) and then to \( x_2 \).

- If the input points were \( x_1 = 1, x_2 = 2 \) and \( x_3 = -1 \), and the deadlines where \( d_1 = 3 \), \( d_2 = 3 \), and \( d_3 = 3 \), then there is no feasible path.