1 Assignment Problem

A construction company has four large bulldozers located at four different garages. The bulldozers are to be moved to four different construction sites. The distances in miles between the bulldozers and the construction sites are given below.

<table>
<thead>
<tr>
<th>Bulldozers/Sites</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>110</td>
<td>35</td>
<td>90</td>
<td>45</td>
</tr>
<tr>
<td>2</td>
<td>105</td>
<td>55</td>
<td>85</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>115</td>
<td>75</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>125</td>
<td>65</td>
<td>95</td>
<td>70</td>
</tr>
</tbody>
</table>

Each site must receive exactly one bulldozer. How should the bulldozers be moved to the construction sites in order to minimize the total distance traveled?

1.1 Complete Enumeration

1. The following graph shows a portion of the tree constructed by Complete Enumeration. Complete the (1 → B) branch using Complete Enumeration.
2. Write pseudo code that solves this problem by Complete Enumeration.

INPUT:
Distance[1,...,N][1,...N]: the distances in the table

OUTPUT:
X[1,...,N]: X[i]=j means bulldozer i goes to site j.

AssignmentCE(Distance[1,...,N][1,...,N])

1.2 Greedy Heuristic

Use Greedy Heuristic that was discussed in the class to find an upper bound.

1.3 Branch-And-Bound

1. What is the Row-Min lower bound, that is, if each bulldozer can move to whatever site that is closest to it?
2. What is the Column-Min lower bound, that is, if each site can receive whatever bulldozer that is closest to it?
3. Complete the (1 → B) branch of the following tree which solves the problem by branch and bound. Use Row-Min to determine the lower bound.

4. Complete all branches of the following tree which solves the problem by Branch-And-Bound. This time, use Column-Min to determine the lower bound. Use Breadth-First-Search.

5. Write pseudo code that solves this problem by Branch-And-Bound using Column-Min.
INPUT:
UpperBound: an upper bound computed by Greedy Heuristic
Distance[1,...,N][1,...N]: the distances in the table

OUTPUT:
X[1,...,N]: X[i]=j means bulldozer i goes to site j.

AssignmentBranchBound(Distance[1,...,N][1,...N], UpperBound)

2  Knapsack Problem

During a robbery, a burglar finds much more loot than he had expected and has to decide what to take. His bag (or “knapsack”) will hold a total weight of at most 11 pounds. There are 4 items to pick from, whose weights and values are shown below.

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

What’s the most valuable combination of items he can fit into his bag?

2.1  Complete Enumeration

1. Complete the tree below to show the Complete Enumeration.

2. Write pseudo code that solves this problem by Complete Enumeration.

KnapsackCE(CAPACITY, N, W[1,...,N], V[1,...,N])

2.2  Greedy Heuristic

Use Greedy Heuristic that was discussed in the class to find a lower bound. The greedy algorithm always picks the item with the highest value in the remaining items.
2.3 Branch-And-Bound

1. What are the relative value of these four items? Please complete the following table. Hint: relative value is value/weight.

<table>
<thead>
<tr>
<th>Item</th>
<th>Weight</th>
<th>Value</th>
<th>Relative Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

2. What is the upper bound of the total value that may be put in this bag? Use the technique learned in class to compute this value.

3. Complete the tree below which solves the problem by Branch-And-Bound. Selection rule: take the item with largest weight first. Use BFS (Breadth-First-Search). You must add the reason when crossing out a vertex.

4. Write pseudo code that solves this problem by Branch-And-Bound. Selection rule: take the item with largest weight first. Use BFS (Breadth-First-Search).

INPUT:
CAPACITY: the capacity of the knapsack
N: there are N items
W[1,...,N]: the weights of N items
V[1,...,N]: the values of N items
LowerBound: the lower bound computed by greedy heuristic.

OUTPUT:
X[1,...,N]: X[i]=1 means item i is in the knapsack, X[i]=0 means it is not in the knapsack.

KnapsackBranchBound(CAPACITY, N, W[1,...,N], V[1,...,N], LowerBound)
3 Proving NP-completeness

3.1 Bin Packing Problem

Problem: In the decision version of the Subset Sum problem we are given a sequence of non-negative integers \(a_1, ..., a_n\) and a target value \(t\). The question is whether there is a subset \(S \subseteq \{1, ..., n\}\) such that \(\sum_{i \in S} a_i = t\).

In the decision version of the Bin Packing problem we are given volumes \(v_1, ..., v_n\), a volume bound \(B\), and a target \(k\). The question is whether we can partition the integers \(v_1, ..., v_n\) into \(k\) subsets such that the integers in each subset sum to at most \(B\).

Given that the Subset Sum problem is NP-complete, prove that the Bin Packing problem is also NP-complete.

3.2 Clique Problem

Problem: Vertex Set of an undirected graph \(G = (V, E)\) covers all edges of the given graph \(G\). It is a subset \(V'\) of its vertices \(V\) such that for every edge \((u, v)\) in \(E\), either \('u'\) or \('v'\) is in the vertex cover. Given an undirected graph \(G\), the decision version of the Vertex Cover problem is to determine whether \(G\) has a vertex cover of a given size \(k\).

A Clique, in an undirected graph \(G = (V, E)\) is a subset \(C\) of its vertices \(V\) such that every two distinct vertices in \(C\) are connected by some edge in \(E\). Given an undirected graph \(G\), the decision version of the Clique problem is to determine whether a clique of size \(k\) exists in \(G\).

Given that the Vertex Cover problem is NP-complete, prove that the Clique problem is also NP-complete.

4 Approximation Algorithm for Bin Packing

Problem: Consider the following problem. Given \(n\) items of weights \(w_1, w_2, ..., w_n \leq 1\). You are tasked with the problem of putting these items into bins so that the total weight of all items in any one bin is at most \(1\). So for example, if you had items of weight 0.6, 0.5 and 0.3, you could put them all in different bins, or put the first and third in a bin together and the second in a different bin, but you could not put the first and second in the same bin because 0.6 + 0.5 > 1. Your objective is to do this while minimizing the total number of bins used. It turns out that this problem is NP-Hard (See Q3.1). Give a polynomial time algorithm that provides a 2-approximation for this problem, and prove correctness.

Hint: As long as your bins are full enough that you cannot combine any pair of them, the average weight in each bin is at least \(1/2\).