Problem (Problem 1a)**.** *Consider a graph $G = (V, E)$.*
The node set V represents station sites, and each edge in E represents a pair of sites between which the distance is within the given bound.
The goal is to find the minimum number of service units to cover all station sites.

Model (Minimum Dominating Set (MDS))**.** *Modelling*
Variables:

- $x_i = 1$ *if station i has a service unit, 0 otherwise*

Objective: *Minimize the number of service units*

$$\textit{minimize} \sum_{i \in V} x_i \qquad \textit{(minimize total number of service units)}$$

subject to

$$x_i + \sum_{j \in N_i} x_j \geq 1 \quad \forall i \in V \qquad \textit{(each station must be covered by at least one service unit)}$$

$$x_i \in \{0, 1\} \quad \forall i \in V \qquad \textit{(binary decision variables)}$$

Solution. Solution to Mimimum Dominating Set

Table 1: Number of service units required for each pmed instance.

Instance	**Number of Service Units**
pmed1	24
pmed2	24
pmed3	24
pmed4	25
pmed5	23

Problem (Problem 1b)**.** *The vector $c \in \mathbb{Z}^E$ represents the distance in question. A cluster is a set consisting of all stations serviced from the same service unit. The diameter of a cluster is the largest distance between any two stations belonging to the cluster.*
The number of service units can not be larger than bound b, where $b = z + 2$ and z is the number of service units in the previous optimal solution.
The goal is to minimize the maximum diameter across all clusters.

Model (Minimum largest cluster distance)**.** *Modelling*
Variables:

- $x_i = 1$ *if station i has a service unit, 0 otherwise*
- $y_{ij} = 1$ *if station j is assigned to service unit at station i, 0 otherwise*
- $D =$ *maximum diameter across all clusters (objective value)*

Objective: *Minimize the maximum cluster diameter*

$$
\begin{aligned}
&\textit{minimize} \quad D \\
&\textit{subject to} \\
&\sum_{i=1}^{n} y_{ij} = 1 \quad \forall j \in V && \textit{(each node is assigned to exactly one service unit)} \\
&y_{ij} \leq x_i \quad \forall i, j \in V && \textit{(can only assign to locations with service units)} \\
&\sum_{i=1}^{n} x_i \leq b && \textit{(limit on number of service units)} \\
c_{jk} \cdot (y_{ij} + y_{ik} - 1) \leq D \quad &\forall i, j, k \in V, j < k && \textit{(maximum diameter constraint)} \\
x_i \in \{0, 1\} \quad &\forall i \in V && \textit{(binary decision variables)} \\
y_{ij} \in \{0, 1\} \quad &\forall i, j \in V && \textit{(binary assignment variables)}
\end{aligned}
$$

Solution. Solution to Mimimum Largest Cluster Distance

Instance	P Value	Max Diameter	Time (s)	Status
pmed1.txt	26	146.00	3605	Optimal
pmed2.txt	26	128.00	3610	Optimal
pmed3.txt	26	137.00	3659	Optimal
pmed4.txt	27	144.00	3606	Optimal
pmed5.txt	25	119.00	3606	Optimal
Total execution time: 18086 seconds				

Table 2: The objective value represents the maximum distance between any two nodes in the same cluster.Time limit was set to 3600 seconds.

Problem (Problem 2)**.** *Dynamic Programming, Integer Programming, and Greedy approaches to the Knapsack Problem.*

Model (Integer Knapsack)**.**

$$\textit{Let } DP[i,w] = \textit{maximum value using first } i \textit{ items with weight at most } w \tag{1}$$

$$DP[0,w] = 0 \quad \forall w \in \{0,1,\ldots,C\} \tag{2}$$

$$DP[i,w] = \begin{cases} \max(DP[i-1,w], DP[i-1,w-w_i] + v_i) & \textit{if } w_i \leq w \\ DP[i-1,w] & \textit{otherwise} \end{cases} \tag{3}$$

Model (Integer Programming)**.**

$$\textit{maximize} \quad \sum_{i=1}^{n} v_i x_i \tag{4}$$

$$\textit{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{5}$$

$$x_i \in \{0,1\} \quad \forall i \in \{1,2,\ldots,n\} \tag{6}$$

Model (Greedy Heurstic)**.**

$$\textit{1. Compute ratio } r_i = \frac{v_i}{w_i} \textit{ for each item } i \tag{7}$$

$$\textit{2. Sort items by } r_i \textit{ in descending order} \tag{8}$$

$$\textit{3. For each item } i \textit{ in the sorted list:} \tag{9}$$

$$\textit{If } w_i \leq \textit{remaining capacity, add item } i \textit{ to knapsack} \tag{10}$$

Knapsack Results

Instance	Items	Capacity	DP Value	DP Time	IP Value	IP Time	Greedy Value	Greedy Time	Greedy Gap
s002.kp	198	49000	63100	7.47 s	63100.0	0.23 s	62653	0.0001 s	0.71%
s003.kp	498	125125	159925	49.37 s	159925.0	0.16 s	159587	0.0002 s	0.21%
s004.kp	498	128232	163332	50.66 s	163332.0	0.13 s	163124	0.0002 s	0.13%
s000.kp	48	14239	17439	0.52 s	17439.0	0.11 s	16837	0.0000 s	3.45%
s001.kp	98	25391	32191	1.92 s	32191.0	0.14 s	31551	0.0001 s	1.99%

Table 1: Comparison of Dynamic Programming, Integer Programming, and Greedy approaches to the Knapsack Problem.

Definition (IP value vs Greedy value)**.** *We observe that the greedy heuristic is very close to the IP value on all instances*
The greedy heuristic is much faster than the IP algorithm.
The greedy heuristic is a good approximation algorithm for the knapsack problem.

Definition (IP running time vs DP running time)**.** *The IP algorithm is faster than the DP algorithm on all instances and much faster when the capacity is high.*

Definition (IP vs DP)**.** *To make the DP algorithm run faster than the IP algorithm we needed to decrease the C from 500 to 40 on the s004.kp instance. This is because the DP algorithm has a time complexity of $O(n \cdot C)$, where n is the number of items and C is the capacity of the knapsack.*
The IP algorithm has a time complexity of $O(n \cdot 2^n)$, where n is the number of items. The DP algorithm is faster than the IP algorithm when C is small compared to n.