Instructions: (Please read carefully and follow them!)

Try to solve all problems on your own. If you have difficulties, ask the instructor or TAs.

The implementation of the optimization algorithms in this lab will involve extensive use of the `numpy` Python package. It would be useful for you to get to know some of the functionalities of `numpy` package. For details on `numpy` Python package, please consult `https://numpy.org/doc/stable/index.html`

For plotting purposes, please use `matplotlib.pyplot` package. You can find examples in the site `https://matplotlib.org/examples/`.

Please follow the instructions given below to prepare your solution notebooks:

- Please use different notebooks for solving different Exercise problems.
- The notebook name for Exercise 1 should be `YOURROLLNUMBER_IE684_Lab1_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab1_Ex2.ipynb`, etc.

There are only 2 exercises in this lab. Try to solve all problems on your own. If you have difficulties, ask the Instructors or TAs.

ALL QUESTIONS ARE COMPULSORY in this sheet. You can either print the answers using `print` command in your code or you can write the text in a separate text tab. To add text in your notebook, click `+Text`. Some questions require you to provide proper explanations; for such questions, write proper explanations in a text tab. Some questions require the answers to be written in LaTeX notation. Please see the demo video to know how to write LaTeX in Google notebooks. Some questions require plotting certain graphs. Please make sure that the plots are present in the submitted notebooks.

After completing this lab's exercises, click File → Download .ipynb and save your files to your local laptop/desktop. Create a folder with name `YOURROLLNUMBER_IE684_Lab1` and copy your `.ipynb` files to the folder. Then zip the folder to create `YOURROLLNUMBER_IE684_Lab1.zip`. Then upload only the `.zip` file to Moodle. **Please write question number clearly in your submission**. There will be PENALTY for students who do not follow the proper naming conventions in their submissions.

Please check the **submission deadline announced in moodle.**

1 Knapsack Problem

The Knapsack Problem is a classic optimization problem in computer science and combinatorial optimization. It's a problem of combinatorial optimization, where the goal is to select items to maximize the total value, subject to a constraint on the total weight. The problem gets its name from the analogy of a knapsack (or backpack) being filled with items, where the items have values and weights, and the goal is to maximize the value of the items in the knapsack without exceeding its capacity.

Here's the general formulation of the 0/1 Knapsack Problem:

Given a set of n items, each with a weight w_i and a value v_i, and a knapsack with a maximum weight capacity W, select a subset of the items to maximize the total value while ensuring that the total weight of the selected items does not exceed the capacity of the knapsack.

Mathematically, the problem can be formulated as follows:

$$\begin{aligned} &\text{Maximize} \quad \sum_{i=1}^{n} v_i \cdot x_i \\ &\text{subject to} \quad \sum_{i=1}^{n} w_i \cdot x_i \leq W \\ &\text{where} \quad x_i \in \{0,1\} \quad \text{for all} \quad i = 1, 2, \ldots, n \end{aligned}$$

The 0/1 Knapsack Problem is called "0/1" because it imposes the constraint that an item can either be taken entirely or not taken at all, meaning you cannot take a fraction of an item. This constraint makes the problem more challenging and is what distinguishes it from the linear relaxation of the problem called the fractional knapsack problem.

The feasible set of knapsack problem is called *Knapsack Set.* A subset $C \subset \{1, 2, ...n\}$ of indices is a *cover for knapsack set K* if $\Sigma_{i \in C} a_i > b$ and it is a minimal cover if $\Sigma_{i \in C \setminus \{j\}} a_i \leq b$ for every $j \in C$.

The Knapsack Problem and its variations have numerous real-world applications across various domains, including:

1. Resource Allocation.
2. Financial Portfolio Management.
3. Scheduling.
4. Data Compression.
5. Bin Packing.
6. Cryptocurrency mining.

There are various algorithms to solve the Knapsack Problem, including dynamic programming, greedy algorithms, and branch and bound techniques. The choice of algorithm often depends on the size of the problem and whether the items have integer or fractional weights.

1.1 Problem Statement

The manager of Luitporiya Company needs to schedule his 20 tasks in 40 hours week window. He made a table (given below) of each task with its expected duration and its value to the company. He realized that it is impossible to fit in all the task within this week itself, however he want to fit the tasks within the window in such a way that total value of the tasks in maximized.

Task No.	**Value**	**Time**
1	10	180
2	25	180
3	33	150
4	53	150
5	15	130
6	70	210
7	52	160
8	45	175
9	32	90
10	30	120
11	20	110
12	5	60
13	17	145
14	50	180
15	37	120
16	18	120
17	71	300
18	35	300
19	60	240
20	90	500

Exercise 1 *(30+10+10+10 marks)*

1. *Help the manager of Luitporiya Company in his scheduling problem by formulating and solving the problem using :*
   (a) *Integer Programming formulation and solver. Clearly describe the formulation with explicitly listing the variables, objective function and constraints.*
   (b) *Implementing Recursion based Programming. Clearly describe your logic. Your answer should give the optimal objective funstion as well the the variables.*
   (c) *Implementing Dynamic Programming. Clearly describe your logic. Your answer should give the optimal objective function as well the the variables.*
   Your answer should clearly express which tasks are chosen, according to the nomenclature of the problem. Note down the time taken by each of these methods and rank them.
2. *Suppose the manager decides it will do the most valuable tasks first. So he selects his most valuable tasks till the time constraint is fulfilled. Will the optimal solution remain same as 1.2? Show by implementation of this greedy algorithm.*
3. *Suppose the manager chooses first n tasks which have the better value by time ratio while maintaining feasibility of 40 hours. Will the optimal solution remain same as 1.2? Show by implementation of this greedy algorithm.*
4. *Consider the sets:* $K := \{x \in \{0,1\}^n : \Sigma_{i=1}^{n} a_i x_i \leq b\}$, $K^C := \{x \in \{0,1\}^n : \Sigma_{i \in C} x_i \leq |C| - 1, \textit{for every } C\}$ *(where C is a minimal cover for K) Show whether* $K \subseteq K^C, K^C \subseteq K$, *both or not necessarily either.*

2 Problem Statement:

Flipbakox company sells six types of boxes, ranging in volume from 0.15 to 0.75 cubic meter. The demand, size and selling price of each box is given below. The variable cost (in INR) of producing each box is equal to hundred times of box's volume. A fixed cost of INR 150 is incurred to produce any of a particular box. If the company desires, demand for a box may be satisfied by a box of larger size. Formulate and solve a discrete optimization problem whose solution will minimize the cost of meeting the demand for boxes.

Box	**Demand**	**Volume**	**Price**
1	300	0.15	300
2	450	0.25	300
3	80	0.35	350
4	500	0.5	500
5	150	0.6	600
6	800	0.75	800

Exercise 2 *(40 marks)*

1. *Formulate this as one of the discrete optimization problems you learnt in IE 684. Clearly describe the formulation with explicitly listing the variables, objective function and constraints.*
2. *Solve it using an algorithm you have used previously in IE 684 Lab. List a table of the optimal number of boxes of each size and state the optimal cost. Don't use solver.*
3. *Suppose only 1000 cubic meter of space is available for transport for Flipbakox company. So they may not be able to fulfill whole demand. Instead they would want to maximize their profit. Formulate this as an optimization problem. Clearly describe the formulation with explicitly listing the variables, objective function and constraints.*
4. *Solve the problem. You can use a solver. Write, explain and interpret the results. Your answer should clearly explain how many boxes of each type is required to be transported.*

References

1. 0-1 Knapsack Problem https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/