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_Lab12_Ex1.ipynb`.
- Similarly, the notebook name for Exercise 2 should be `YOURROLLNUMBER_IE684_Lab12_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 Travelling Salesman Problem

Given a graph G (V,E), where V may represent a set of cities and E may represent distances between them, the Travelling Salesman Problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the original city?". It belongs to the most seductive problems in discrete optimization, thanks to a blend of complexity, applicability, and appeal to imagination.

The TSP can be formulated in many ways. Here is DFJ formulation of TSP:

Objective Function:

$$\text{Minimize:} \quad \sum_{e \in E} c_e x_e$$

Subject to the following constraints:

Every city is visited once and left once:

$$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i \in V$$

Elimination of sub-tours:

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall \phi \subset S \subset V$$

Binary variable constraints:

$$x_e \in \{0, 1\} \quad \forall e \in E$$

Where:

- $\delta(i) := e \in E$ such that edge e is incident on vertex i.
- $\delta(S) := e \in E$ such that edge e is incident on exactly one vertex of subset S of V.
- n is the total number of cities.

This problem shows up in practice in :

1. Routing of transportation.
2. Machine Scheduling.
3. Clustering.
4. Computer Wiring.
5. Curve reconstruction.

Table 1: Distance between old IITs

	BOMBAY	KANPUR	KHARAGPUR	MADRAS	DELHI	GUWAHATI	ROORKEE
BOMBAY	0	1289	1753	1364	1432	2586	1594
KANPUR		0	1167	1868	495	1432	630
KHARAGPUR			0	1562	1567	1057	1703
MADRAS				0	2207	2643	2343
DELHI					0	1891	212
GUWAHATI						0	2055
ROORKEE							0

1.1 Exercise 1:

1. Explain why the DFJ formulation will give exactly feasible tours.
2. What is the shortest way to visit all seven old IITs starting from IIT Bombay such that you visit each IIT exactly once? Code and solve an integer program using a solver. Use DFJ formulation. Data of distance as given in Table 1. Notice there are exponentially many constraints. Since there are exponentially many constraints, solve by relaxing sub-tour elimination constraints, if the optimal solution gives a sub-tour then add a constraint to remove it and repeat until you don't have any sub-tour.
3. The TSP polytope is defined as the convex hull of all feasible $x \in \{0,1\}^{|E|}$. How many extreme points the TSP polytope of the IIT problem will have?
4. Write a code that checks given a 0-1 vector whether it is an extreme point of the given TSP feasible region. Your function should take a $|E|$ dimensional 0-1 vector as input and print "extreme point" if it is an extreme point and "not an extreme point" if it is not. *(Hint: The problem is same as given a $|E|$ dimensional 0-1 vector, whether the vector is feasible or not).*

1.2 Exercise 2: Simulated Annealing

We will now use simulated annealing which is a probabilistic technique for approximating the global optimum of a given function. Our aim is to solve a Travelling Salesman Problem (TSP) using this technique.

Algorithm 1 Simulated Annealing for TSP

```
1: Generate a random initial solution and set $T_0$
2: Calculate the cost function using cost function.
3: for k = 0 through $max_{iter}$ do
4:     Generate a random neighboring state of the current solution using neighbor function and calculate its cost.
5:     $T_k$ = updated temperature given by cooling schedule.
6:     If $c_{new} \leq c_{old}$ move to new solution, else move to new solution with some probability given by $accept_prob()$
       function,
7:     In each iteration store the best solution you got till that time.
8: end for
9: Return the best solution.
```

1. Describe Travelling Salesman Problem.
2. We will consider a small instance of the problem with 11 cities. The distance between any two cities is given in TSP11.csv.

3. The pseudo-code for simulated annealing is given above. We start with an initial random state. A state is given by a particular sequence of cities which the salesman travels. This is known as a ”tour” of the salesman. Note that the salesman should travel to all the cities only once.

4. Given a tour of the salesman, compute the cost, which is the total distance travelled in that tour. Create *cost()* function which takes a sequence of cities as input and calculates the total distance travelled. For example, in a 4-city scenario, given the sequence of cities as [B,C,D,A], your function should calculate the total distance travelled in that particular order.

5. Now we will create the *neighbour()* function. In TSP, the neighbour of a state is defined as the new states obtained by swapping two consecutive cities. For example, in a 4-city case, every state has 4 neighbours and the neighbour of [A,B,C,D] are [B,A,C,D], [A,C,B,D], [A,B,D,C], [D,B,C,A]. Your function should choose one of the neighbours uniformly.

6. The acceptance probability is given by $e^{-(c_{\text{old}}-c_{\text{new}})/T_k}$ where c_{old} and c_{new} are the costs of the old state and the new state, respectively. T_k is the current temperature of the kth iteration. Explain the behaviour of acceptance probability when (1) new cost gets worse than the current one and (2) temperature decreases.

7. Usually, the temperature is started at a high value and slowly decreased to 0 in each iteration by a cooling schedule. Consider the cooling schedule given by $T_{k+1} = \alpha T_k$ for $\alpha \geq 0.80$ and $T_0 = 1$.

8. Run the simulated annealing algorithm and report the solution. Plot the cost function for each iteration and explain. Choose different T_0 values and observe the effect.

9. Come up with another cooling schedule (not of the form $T_{k+1} = \alpha T_k$) and comment on the change in behaviour of the algorithm.

10. Do the same for the slightly larger problem with 48 cities. The distance matrix is given in `TSP48.csv`.

References

1. Travelling Salesman Problem:
https://www.youtube.com/watch?v=X_V_3cX2O5E&list=PLJzsUPdFULDYBVVoyXsMIQIpFIIpLL0Qj

2. Simulated Annealing:
https://www.youtube.com/watch?v=FyyVbuLZav8&t=353s