**Assignment 1: Optimisation, Constraint-Satisfaction, and Least Squares**
**Due Date: March 1st 2020 @ 11:59 PM**

Instructions:

1. This assignment can be done in groups of up to three students
2. You should submit your assignment to the email addresses:
   - inzamam.rahaman@sta.uwi.edu
   - shivramx@gmail.com

   You should use the subject: "COMP3608 Assignment 1", and you should submit a single zip folder named using the student ids of all members separated by commas. This **zip** folder should contain a README.md or readme.txt file with the names and id numbers of all group members.
3. You should separate your solutions to each component into separate sub-directories named P1, P2, and P3

# Part 1 - Backtracking and CSPs

In Hall's Marriage Problem, we have two sets: a set of women, $W$, and a set of men $M$. We wish to pair men to women for marriage. Each woman has a set of men that they would like to marry. Each man is fine with being paired with any woman. Every woman can be paired with one and only one man and vice versa. We want to find a pairing of men to women such these constraints are upheld. If no such assignment is possible, then we want to report failure.

You are provided code to solve Hall's Marriage Problem. For the a sake of convenience, every woman is labelled with an integer, and likewise every man is labelled with an integer In addition, you are provided a .json file describing in instance of the problem. Draw the search tree (similar to the tree from the end of lecture #2) showing the backtracking procedure used to solve this instance of the problem. You are free to put print statements in the code provided to help you. You are to submit a clear picture of the backtracking tree you derived.

# Part 2 - Optimisation #1

A cellular provider, AZ&Z, wants to expand their operations into another country. To do this, they must construct new cell towers. They have decided on seven potential sites for cell tower construction. On the assumption that all cell towers cost the same, they want to minimise the number of cell towers they need to construct.

There are 15 neighborhoods they would like to cover, and each cell tower covers multiple neighborhoods. The neighborhoods covered by each cell tower are described in the following table:

| Location | Covered Neighbors |
|---|---|
| 1 | 1,2 |
| 2 | 2,3,5 |
| 3 | 1,7,9,10 |
| 4 | 4,6,8,9 |
| 5 | 6,7,9,11 |
| 6 | 5,7,10,12,14 |
| 7 | 12,13,14,15 |

Write MiniZinc or Julia code that determines the cell towers they need to construct. (Hint: Research the Set Cover Problem)

# Part 3 - Optimisation #2

A manufacturer has three plants and four distribution centres. Each plant can manufacture a fixed amount of product, and each distribution centre requires a minimum quantity of each product. Transportation costs per truckload between each plant-distribution centre pair differ. They want to satisfy the demands of each distribution centre at minimum cost. The cost details are shown below:

| | Shipping Cost ($) per Truckload | | | | |
|---|---|---|---|---|---|
| | Warehouse | | | | |
| | 1 | 2 | 3 | 4 | Output |
| Cannery 1 | 464 | 513 | 654 | 867 | 75 |
| Cannery 2 | 352 | 416 | 690 | 791 | 125 |
| Cannery 3 | 995 | 682 | 388 | 685 | 100 |
| Allocation | 80 | 65 | 70 | 85 | |

Write MiniZinc or Julia code that determines the number of truckloads each plant should send to each distribution centre. (Hint: Read up on Transportation problems)

## Part 3 - Least Squares

Suppose we have a network comprising of $n$ links. Each link has an unknown delay. We perform a series of experiments involving sending packets through specific paths and record the time between sending and receiving (i.e. the overall delay). Note that processing delays and transmission delays would "corrupt" these results. That being said, we can use least squares to approximate the delay across each link.

Suppose we ran $m$ experiments. All experiments can be encoded into an $m \times n$ matrix $E$ where

$$E_{ij} \begin{cases} 1 \text{ if experiment } i \text{ uses link } j \\ 0 \text{ otherwise} \end{cases}$$

We store the results of the experiment in a vector $d \in \mathbb{R}^m$.

You are given both $E$ and $d$ in two appropriately named .npy files. This (https://docs.scipy.org/doc/numpy/reference/generated/numpy.load.html#numpy.load) documentation should help you read in the files. You are free to use the least squares code developed in class. Note, that when you read in the arrays, you need to convert to torch tensors