**Implementation**

I have implemented following function for showing the optimum labels which optimally represents the graph.

**main.py:**

main function from where all the variables and function will be called and genetic algorithm will run here.

**crossover.py:**

Function which eventually returns a chromosome which is formed by crossover of two parent chromosomes.

**encode\_decode.py:**

contains:

decode(s):

returns a list of labels from a binary number(which is a chromosome)

encode(s):

takes a list as input and

returns a string of binary number

binary(n,x):

return binary a x-bit binary number of n

bin\_to\_dec(n):

takes a binary number as input and converts it to decimal form.

**fitness.py:**

this is my fitness function which basically returns size of label list, which is what I have to minimize.

**freq\_element.py:**

it has following functions :

freq\_element(graph):

it takes a graph as input and returns a list containing frequency of occurrence of all labels

dict\_create(graph):

takes graph as input and return a empty dictionary with edge vertices of graph as key.

**globals.py:**

contains all the global variables like number\_of\_iterations,number\_of\_labels,graph,etc.

**mutation.py:**

this function overall mutates the input chromosome.

It contains following function:

\_mutation(s):

It randomly adds a label which is not in chromosome s.

Mutation\_sorting(s,graph):

It returns a list(containing labels(s)) which is in decreasing order of frequency of occurrence in graph

**check\_spanning.py:**

this file contains following function

create\_subset\_graph(s,graph):

returns a sub-graph of graph such that it only contains edges of labels in s.

dfs(graph,visited,node):

this function depth first searches the graph to check the connectivity of graph i.e. whether the given sub-graph is a spanning tree of graph.

Checkz(visited):

After dfs we get a list (visited) which tells us whether a node is visited or not.

This function checks whether all nodes are visited or not.

is\_spanning\_tree(graph,labels):

it calls all above function viz. create\_subset\_graph,dfs,checkz

to check whether the graph formed by given labels form a spanning tree or not.