Assignment 1 : Question 3 (Opitimization method)

In [ ]:

```
from gurobipy import *#--------------importing required libraries
import numpy as np

from string import*
from pandas import *
```

In [2]:

```
m = Model("Question 3 problem")#---creating the model
```

```
Using license file C:\Users\Aloukik Aditya\gurobi.lic
Academic license - for non-commercial use only
```

In [3]:

```


P = [
        [5,4,6,7,1,5,6],
        [9,8,5,1,1,2,3],
        [1,7,4,6,2,3,5],
        [1,1,2,4,2,6,2],#-----------------------creating matrix
        [15,12,1,3,10,8,2],
        [16,17,1,1,6,6,2],
        [3,5,8,1,2,1,1]
]
r1 = [5,4,6,7,1,5,6]

print(np.matrix(P))
print(len(P))
```

```
[[ 5  4  6  7  1  5  6]
 [ 9  8  5  1  1  2  3]
 [ 1  7  4  6  2  3  5]
 [ 1  1  2  4  2  6  2]
 [15 12  1  3 10  8  2]
 [16 17  1  1  6  6  2]
 [ 3  5  8  1  2  1  1]]
7
```

In [4]:

```
P[1]
```

Out[4]:

```
[9, 8, 5, 1, 1, 2, 3]
```

In [ ]:

```

```

In [5]:

```
#y = m.addMVar((7,7), vtype=GRB.BINARY)
```

In [ ]:

```

```

In [6]:

```
M = m.addMVar((7,7), lb=0,ub=1, vtype=GRB.BINARY)#----------------creating 7by7 binary matr
```

In [ ]:

```

```

In [8]:

```
for i in range(len(P)):#-------------------this loop will do algorithm part, whihc means se
    const = 0
    const_c = 0

    for j in range(len(P)):#------------------for values
        const += M[j][i]#--------------------------it means that only values with 1 will be
        const_c += M[i][j] #---------------------it means value with 1 will be selected in
    m.addConstr(const == 1)
    m.addConstr(const_c == 1)




# for i in range(len(P)):
#     const_c = 0
#     for j in range(len(P)):#------------------for values
#         const_c += M[i][j]
#     m.addConstr(const_c == 1)
```

In [9]:

```
new = np.zeros([7,7]).astype(int)#-------------
```

In [10]:

```
#-- objective function
# obj = 0
# for v in M:
#     obj += v
#m.setObjective(obj, GRB.MINIMIZE)

obj = 0
for i in range(len(P)):

    for j in range(len(P)):
        obj += P[i][j]*M[i][j]







# obj2 = 0
# #------------------------- multiplying each box with, C variable(which tells wether tool
# obj1 = 0
# for i in range(len(P)):
#     obj2 = 0
#     for j in range(len(P)):
#         obj2 += P[j][i]
#     obj1+= obj2 + P[i][j]
#obj = new.sum()


m.setObjective(obj, GRB.MAXIMIZE)#------------- this will help minimizing cost and number v
```

In [ ]:

```

```

In [11]:

```
m.optimize()#-------------using optimize function and it will show the result
```

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (win64)
Optimize a model with 14 rows, 49 columns and 98 nonzeros
Model fingerprint: 0x38cb43b9
Variable types: 0 continuous, 49 integer (49 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 29.0000000
Presolve time: 0.01s
Presolved: 14 rows, 49 columns, 98 nonzeros
Variable types: 0 continuous, 49 integer (49 binary)

Root relaxation: objective 6.200000e+01, 16 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      62.0000000   62.00000  0.00%     -    0s

Explored 0 nodes (16 simplex iterations) in 0.04 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 62 29

Optimal solution found (tolerance 1.00e-04)
Best objective 6.200000000000e+01, best bound 6.200000000000e+01, gap 0.000
0%
```

In [12]:

```
for i in range(len(P)):

    for j in range(len(P)):
        new[i][j]= P[i][j]*M[i][j].X#--------------------for loop to see the selected grids
```

In [ ]:

```

```

In [13]:

```
new#----------------------grids which are selected are showed below
```

Out[13]:

```
array([[ 0,  0,  0,  7,  0,  0,  0],
       [ 9,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  5],
       [ 0,  0,  0,  0,  0,  6,  0],
       [ 0,  0,  0,  0, 10,  0,  0],
       [ 0, 17,  0,  0,  0,  0,  0],
       [ 0,  0,  8,  0,  0,  0,  0]])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```