Assignment 1 : Question 3 (smart health)

Student_ID : 1115290

In [14]:

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

from string import*
from pandas import *
```

In [15]:

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

In [27]:

```


P = [
        [50,56,62,71,123,59],
        [113,147,149,61,112,113],
        [111,121,132,134,123,56],#-----------------------creating matrix
        [130,95,149,53,107,109],
        [118,83,108,101,60,62],
        [131,53,88,129,127,139]
]


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

```
[[ 50  56  62  71 123  59]
 [113 147 149  61 112 113]
 [111 121 132 134 123  56]
 [130  95 149  53 107 109]
 [118  83 108 101  60  62]
 [131  53  88 129 127 139]]
```

In [17]:

```
P[1]
```

Out[17]:

```
[113, 147, 149, 61, 112, 113]
```

In [ ]:

```

```

In [18]:

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

In [ ]:

```
```

In [19]:

```
M = m.addMVar((6,6), lb=0,ub=1, vtype=GRB.BINARY)#----------------creating 7by7 binary matr
```

In [ ]:

```
```

In [20]:

```
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)
```

In [21]:

```
new = np.zeros([6,6]).astype(int)#-------------
```

In [22]:

```
#-- 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]

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

In [23]:

```
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 12 rows, 36 columns and 72 nonzeros
Model fingerprint: 0xb47767c4
Variable types: 0 continuous, 36 integer (36 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [5e+01, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 636.0000000
Presolve time: 0.00s
Presolved: 12 rows, 36 columns, 72 nonzeros
Variable types: 0 continuous, 36 integer (36 binary)

Root relaxation: objective 3.970000e+02, 10 iterations, 0.00 seconds

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

*    0     0               0     397.0000000  397.00000  0.00%     -    0s

Explored 0 nodes (10 simplex iterations) in 0.01 seconds
Thread count was 8 (of 8 available processors)

Solution count 2: 397 636

Optimal solution found (tolerance 1.00e-04)
Best objective 3.970000000000e+02, best bound 3.970000000000e+02, gap 0.000
0%
```

In [24]:

```
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 [25]:

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

Out[25]:

```
array([[  0,   0,  62,   0,   0,   0],
       [113,   0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,  56],
       [  0,   0,   0,  53,   0,   0],
       [  0,   0,   0,   0,  60,   0],
       [  0,  53,   0,   0,   0,   0]])
```

In [26]:

```
minutess = new.sum()
print("minutes", + minutess)#----------------------converting to hours
hours = minutess/60
print("hours", + hours)
```

```
minutes 397
hours 6.616666666666666
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: