```
%pip install gurobipy>=10
import gurobipy as grb
import numpy as np
import math
import matplotlib.pyplot as plt
import scipy.stats
```

Q1

令$X0$為組好的A數量，$X1$為組好的B數量，$X2$為短腳數量，$X3$為長腳數量，$X4$為桌板數量

Objective: MAXIMIZE $30 * X0 + 45 * X1$(profit-maximization)

Constraint:

1. $18 * X2 + 30 * X3 \leq 500 * 12$(桌腳長度總和不超過$500 * 12$inches)
2. $X0 + X1 = X4$(製作出來的桌子數量=桌板數量)
3. $X0 * 4 = X2$(一張桌子有四個桌腳)
4. $X1 * 4 = X3$(一張桌子有四個桌腳)
5. $0.1 * X2 + 0.1 * X3 + 0.5 * X4 + 0.3 * X0 + 0.3 * X1 \leq 80$(組裝時間不超過80小時)

```
model_1 = grb.Model("Q1")

I = 5
x = model_1.addVars([i for i in range(I)], vtype = grb.GRB.INTEGER, name = "x_1")

model_1.addConstr(18*x[2]+30*x[3]<=500*12)
model_1.addConstr(x[0]+x[1]-x[4]==0)
model_1.addConstr(x[0]-x[2]/4==0)
model_1.addConstr(x[1]-x[3]/4==0)
model_1.addConstr(0.1*x[2]+0.1*x[3]+0.5*x[4]+0.3*x[0]+0.3*x[1]<=80)

obj=30*x[0]+45*x[1]
model_1.setObjective(obj,grb.GRB.MAXIMIZE)
model_1.update()
model_1.optimize()

print(model_1.Status == grb.GRB.OPTIMAL)
print(model_1.display())

for v in model_1.getVars():
    print(v.VarName, v.X)

optobj=model_1.getObjective()
print(optobj.getValue())
```

```
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (linux64 - "Ubuntu 22.04.3 LTS")

CPU model: Intel(R) Xeon(R) CPU @ 2.20GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Optimize a model with 5 rows, 5 columns and 14 nonzeros
Model fingerprint: 0x393cf3e6
Variable types: 0 continuous, 5 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e-01, 3e+01]
  Objective range  [3e+01, 4e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [8e+01, 6e+03]
Found heuristic solution: objective -0.0000000
Presolve removed 3 rows and 3 columns
Presolve time: 0.00s
Presolved: 2 rows, 2 columns, 4 nonzeros
Variable types: 0 continuous, 2 integer (0 binary)
Found heuristic solution: objective 1995.0000000
```

```
Root relaxation: objective 2.370000e+03, 2 iterations, 0.00 seconds (0.00 work units)

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

*    0     0               0    2370.0000000 2370.00000  0.00%     -    0s

Explored 1 nodes (2 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 2 (of 2 available processors)

Solution count 3: 2370 1995 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 2.370000000000e+03, best bound 2.370000000000e+03, gap 0.0000%
True
Maximize
  30.0 x_1[0] + 45.0 x_1[1]
Subject To
  R0: 18.0 x_1[2] + 30.0 x_1[3] <= 6000
  R1: x_1[0] + x_1[1] + -1.0 x_1[4] = 0
  R2: x_1[0] + -0.25 x_1[2] = 0
  R3: x_1[1] + -0.25 x_1[3] = 0
  R4: 0.3 x_1[0] + 0.3 x_1[1] + 0.1 x_1[2] + 0.1 x_1[3] + 0.5 x_1[4] <= 80
General Integers
  ['x_1[0]', 'x_1[1]', 'x_1[2]', 'x_1[3]', 'x_1[4]']
None
x_1[0] 40.0
x_1[1] 26.0
x_1[2] 160.0
x_1[3] 104.0
x_1[4] 66.0
2370.0
<ipython-input-9-daff5d9d8b54>:18: DeprecationWarning: Model.display() is deprecated
  print(model_1.display())
```

Q2

令$X0$到$X6$為7位成員，positions為各自位置，ballHandlings為各自ballHandling分數，shootings為各自shooting分數，reboundings為各自rebounding分數，defenses為各自defense分數

Objective: MAXIMIZE defense分數

Constraint:

1. 只選五人(先發隊伍僅五人)
2. $(ballHandlings + shootings + reboundings)/5 \geq 2$(平均分大於2)
3. $X3 \geq X0$且$X4 \geq X0$(1上則4.5都要上)
4. $X1 + X2 \geq 1$(2.3至少上一個)
5. $X2 + X5 \leq 1$(3.6只能上一個)
6. G至少有4個人
7. F至少有2個人
8. C至少有1個人

```
model_2 = grb.Model("Q2")


positions=['G','C','G-F','F-C','G-F','F-C','G-F']
ballHandlings=[3,2,2,1,3,3,3]
shootings=[3,1,3,3,3,1,2]
reboundings=[1,3,2,3,3,2,2]
defenses=[3,2,2,1,3,3,1]

#positions=np.array((['G'],['C'],['G-F'],['F-C'],['G-F'],['F-C'],['G-F']))
#rates=np.array(([3,3,1,3],[2,1,3,2],[2,3,2,2],[1,3,3,1],[3,3,3,3],[3,1,2,3],[3,2,2,1]))


x = model_2.addVars(len(positions), vtype = grb.GRB.BINARY, name = "x_2")



model_2.addConstr(x.sum() == 5) #選五人
model_2.addConstr(sum(x[i] * (ballHandlings[i]+shootings[i]+reboundings[i]) for i in range(len(positions)
```

```
model_2.addConstr(x[3]-x[0]>=0)  #若1上則4.5皆上
model_2.addConstr(x[4]-x[0]>=0)  #若1上則4.5皆上
model_2.addConstr(x[1]+x[2]>=1)  #2.3必上至少一
model_2.addConstr(x[2]+x[5]<=1)  #3.6只上一個



g_positions = [1 if 'G' in pos else 0 for pos in positions]
model_2.addConstr(sum(x[i] * g_positions[i] for i in range(len(positions))) >= 4)

f_positions=[1 if 'F' in pos else 0 for pos in positions]
model_2.addConstr(sum(x[i] * f_positions[i] for i in range(len(positions))) >= 2)

c_positions=[1 if 'C' in pos else 0 for pos in positions]
model_2.addConstr(sum(x[i] * c_positions[i] for i in range(len(positions))) >= 1)


obj=sum(x[i] * defenses[i] for i in range(len(positions)))
model_2.setObjective(obj,grb.GRB.MAXIMIZE)

model_2.update()
model_2.optimize()

print(model_2.Status == grb.GRB.OPTIMAL)
print(model_2.display())

for v in model_2.getVars():
    print(v.VarName, v.X)

optobj=model_2.getObjective()
print(optobj.getValue())
```

```
Gurobi Optimizer version 11.0.1 build v11.0.1rc0 (linux64 - "Ubuntu 22.04.3 LTS")

CPU model: Intel(R) Xeon(R) CPU @ 2.20GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 1 physical cores, 2 logical processors, using up to 2 threads

Optimize a model with 9 rows, 7 columns and 34 nonzeros
Model fingerprint: 0xdff9d3be
Variable types: 0 continuous, 7 integer (7 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [1e+00, 3e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 5e+00]
Presolve removed 9 rows and 7 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.04 seconds (0.00 work units)
Thread count was 1 (of 2 available processors)

Solution count 1: 10

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+01, best bound 1.000000000000e+01, gap 0.0000%
True
Maximize
  3.0 x_2[0] + 2.0 x_2[1] + 2.0 x_2[2] + x_2[3] + 3.0 x_2[4] + 3.0 x_2[5] + x_2[6]
Subject To
  R0: x_2[0] + x_2[1] + x_2[2] + x_2[3] + x_2[4] + x_2[5] + x_2[6] = 5
R1: 1.4000000000000001 x_2[0] + 1.2000000000000002 x_2[1] + 1.4000000000000001 x_2[2] +
1.4000000000000001 x_2[3] + 1.8 x_2[4] + 1.2000000000000002 x_2[5] + 1.4000000000000001
 x_2[6] >= 2
  R2: -1.0 x_2[0] + x_2[3] >= 0
  R3: -1.0 x_2[0] + x_2[4] >= 0
  R4: x_2[1] + x_2[2] >= 1
  R5: x_2[2] + x_2[5] <= 1
  R6: x_2[0] + x_2[2] + x_2[4] + x_2[6] >= 4
  R7: x_2[2] + x_2[3] + x_2[4] + x_2[5] + x_2[6] >= 2
  R8: x_2[1] + x_2[3] + x_2[5] >= 1
Binaries
  ['x_2[0]', 'x_2[1]', 'x_2[2]', 'x_2[3]', 'x_2[4]', 'x_2[5]', 'x_2[6]']
None
x_2[0] 1.0
x_2[1] 0.0
x_2[2] 1.0
```

```
x_2[3] 1.0
x_2[4] 1.0
x_2[5] 0.0
x_2[6] 1.0
10.0
<ipython-input-12-809150bfbafb>:45: DeprecationWarning: Model.display() is deprecated
  print(model_2.display())
```