Q1

令台北配給$X0$個，台中配給$X1$個，台南配給$X2$個，台東配給$X3$個

Objective: MAXIMIZE $1.6 * X0 + 1.4 * X1 + 1.9 * X2 + 1.2 * X3$(profit-maximization)

Constraint:

1. $X0 + X1 + X2 + X3 = 2000 * 0.6$(只能製造60%需求)
2. $620 * 0.5 \leq X0 \leq 620 * 0.7$(每個地區需滿足原需求的50%~70%)
3. $490 * 0.5 \leq X1 \leq 490 * 0.7$(每個地區需滿足原需求的50%~70%)
4. $510 * 0.5 \leq X2 \leq 510 * 0.7$(每個地區需滿足原需求的50%~70%)
5. $380 * 0.5 \leq X3 \leq 380 * 0.7$(每個地區需滿足原需求的50%~70%)

```
%pip install -i https://pypi.gurobi.com gurobipy
%pip install gurobipy>=10
import gurobipy as grb
import numpy as np
import math
import matplotlib.pyplot as plt
import scipy.stats

model_1 = grb.Model("Q1")

J = 4
x = model_1.addVars([j for j in range(J)], vtype = grb.GRB.CONTINUOUS, name = "x_1")

model_1.addConstr(grb.quicksum(x)==2000*0.6)

model_1.addConstr(x[0]>=620*0.5)
model_1.addConstr(x[0]<=620*0.7)

model_1.addConstr(x[1]>=490*0.5)
model_1.addConstr(x[1]<=490*0.7)

model_1.addConstr(x[2]>=510*0.5)
model_1.addConstr(x[2]<=510*0.7)

model_1.addConstr(x[3]>=380*0.5)
model_1.addConstr(x[3]<=380*0.7)


obj=1.6*x[0]+1.4*x[1]+1.9*x[2]+1.2*x[3]
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())
```

```
Looking in indexes: https://pypi.gurobi.com
Requirement already satisfied: gurobipy in /usr/local/lib/python3.10/dist-packages (11.0.1)
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, 4 columns and 12 nonzeros
```

```
Model fingerprint: 0xf54d41d5
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+02, 1e+03]
Presolve removed 8 rows and 0 columns
Presolve time: 0.01s
Presolved: 1 rows, 4 columns, 4 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.9315000e+03   1.225000e+01   0.000000e+00      0s
       1    1.9021000e+03   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.02 seconds (0.00 work units)
Optimal objective  1.902100000e+03
True
Maximize
  1.6 x_1[0] + 1.4 x_1[1] + 1.9 x_1[2] + 1.2 x_1[3]
Subject To
  R0: x_1[0] + x_1[1] + x_1[2] + x_1[3] = 1200
  R1: x_1[0] >= 310
  R2: x_1[0] <= 434
  R3: x_1[1] >= 245
  R4: x_1[1] <= 343
  R5: x_1[2] >= 255
  R6: x_1[2] <= 357
  R7: x_1[3] >= 190
  R8: x_1[3] <= 266
None
x_1[0] 408.0
x_1[1] 245.0
x_1[2] 357.0
x_1[3] 190.0
1902.1
<ipython-input-8-233c1dd009fd>:14: DeprecationWarning: Calling quicksum on a tupledict is deprecated, use .sum() instead.
  model_1.addConstr(grb.quicksum(x)==2000*0.6)
<ipython-input-8-233c1dd009fd>:35: DeprecationWarning: Model.display() is deprecated
  print(model_1.display())
```

Q2

令Oat放$X0$單位，Corn放$X1$單位，Alfalfa放$X2$單位，Peanut Hulls放$X3$單位

Objective: MINIMIZE $200 * X0 + 150 * X1 + 100 * X2 + 75 * X3$(cost-minimizing)

Constraint:

1. $[(0.6 * X0 + 0.8 * X1 + 0.55 * X2 + 0.4 * X3) + (0.9 * X0 + 0.3 * X1 + 0.6 * X2 + 0.8 * X3)]/(X0 + X1 + X2 + X3) \geq 0.6$
   (至少60%來自protein跟fiber)
2. $(0.5 * X0 + 0.7 * X1 + 0.4 * X2 + 1 * X3)/(X0 + X1 + X2 + X3) \leq 0.6$(fat低於60%)

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

J=4
x = model_2.addVars([j for j in range(J)], vtype = grb.GRB.CONTINUOUS, name = "x_2")

model_2.addConstr(grb.quicksum(x)==1)
model_2.addConstr(((0.6*x[0]+0.8*x[1]+0.55*x[2]+0.4*x[3])+(0.9*x[0]+0.3*x[1]+0.6*x[2]+0.8*x[3]))>=0.6*(grb.qui
model_2.addConstr((0.5*x[0]+0.7*x[1]+0.4*x[2]+1*x[3])<=0.6*(grb.quicksum(x)))

obj=200*x[0]+150*x[1]+100*x[2]+75*x[3]
model_2.setObjective(obj,grb.GRB.MINIMIZE)
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 3 rows, 4 columns and 12 nonzeros
Model fingerprint: 0xb8a991ae
Coefficient statistics:
  Matrix range     [1e-01, 1e+00]
  Objective range  [8e+01, 2e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 1 rows and 1 columns
Presolve time: 0.02s
Presolved: 2 rows, 3 columns, 6 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.5000000e+01   1.048250e+00   0.000000e+00      0s
       1    9.1666667e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.07 seconds (0.00 work units)
Optimal objective  9.166666667e+01
True
Minimize
  200.0 x_2[0] + 150.0 x_2[1] + 100.0 x_2[2] + 75.0 x_2[3]
Subject To
  R0: x_2[0] + x_2[1] + x_2[2] + x_2[3] = 1
  R1: 0.9 x_2[0] + 0.5 x_2[1] + 0.5499999999999999 x_2[2] + 0.6000000000000002 x_2[3] >= 0
R2: -0.09999999999999998 x_2[0] + 0.09999999999999998 x_2[1] + -0.19999999999999996
 x_2[2] + 0.4 x_2[3] <= 0
None
x_2[0] 0.0
x_2[1] 0.0
x_2[2] 0.6666666666666667
x_2[3] 0.33333333333333326
91.66666666666666
<ipython-input-9-f49949fe6aaa>:6: DeprecationWarning: Calling quicksum on a tupledict is deprecated, use .sum() instead.
  model_2.addConstr(grb.quicksum(x)==1)
<ipython-input-9-f49949fe6aaa>:7: DeprecationWarning: Calling quicksum on a tupledict is deprecated, use .sum() instead.
  model_2.addConstr(((0.6*x[0]+0.8*x[1]+0.55*x[2]+0.4*x[3])+(0.9*x[0]+0.3*x[1]+0.6*x[2]+0.8*x[3]))>=0.6*(grb.quicksum(x)))
<ipython-input-9-f49949fe6aaa>:8: DeprecationWarning: Calling quicksum on a tupledict is deprecated, use .sum() instead.
  model_2.addConstr((0.5*x[0]+0.7*x[1]+0.4*x[2]+1*x[3])<=0.6*(grb.quicksum(x)))
<ipython-input-9-f49949fe6aaa>:16: DeprecationWarning: Model.display() is deprecated
  print(model_2.display())
```

Q3

令混合液第一種含量$X0$`,第二種含量$X1$，第三種含量$X2$，第四種含量$X3$

Objective:MINIMIZE $48 * X0 + 43 * X1 + 58 * X2 + 46 * X3$(cost minimizing)

Constraint:

1. $89 \leq (99 * X0 + 70 * X1 + 78 * X2 + 91 * X3) \leq 90$(first quality index在89~90間)
2. $270 \leq (210 * X0 + 335 * X1 + 280 * X2 + 265 * X3) \leq 280$(second quality index在270~280間)

```
model_3=grb.Model("Q3")

J=4
x = model_3.addVars([j for j in range(J)], vtype = grb.GRB.CONTINUOUS, name = "x_3")

model_3.addConstr((99*x[0]+70*x[1]+78*x[2]+91*x[3])>=89)
model_3.addConstr((99*x[0]+70*x[1]+78*x[2]+91*x[3])<=90)

model_3.addConstr((210*x[0]+335*x[1]+280*x[2]+265*x[3])>=270)
model_3.addConstr((210*x[0]+335*x[1]+280*x[2]+265*x[3])<=280)

obj=48*x[0]+43*x[1]+58*x[2]+46*x[3]
model_3.setObjective(obj,grb.GRB.MINIMIZE)
model_3.update()
model_3.optimize()

print(model_3.Status == grb.GRB.OPTIMAL)
print(model_3.display())

for v in model_3.getVars():
    print(v.VarName, v.X)

optobj=model_3.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 4 rows, 4 columns and 16 nonzeros
Model fingerprint: 0xc9340ca4
Coefficient statistics:
  Matrix range     [7e+01, 3e+02]
  Objective range  [4e+01, 6e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [9e+01, 3e+02]
Presolve removed 2 rows and 0 columns
Presolve time: 0.02s
Presolved: 2 rows, 6 columns, 10 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.800000e+01   0.000000e+00      0s
       2    4.5617512e+01   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.04 seconds (0.00 work units)
Optimal objective  4.561751152e+01
True
Minimize
  48.0 x_3[0] + 43.0 x_3[1] + 58.0 x_3[2] + 46.0 x_3[3]
Subject To
  R0: 99.0 x_3[0] + 70.0 x_3[1] + 78.0 x_3[2] + 91.0 x_3[3] >= 89
  R1: 99.0 x_3[0] + 70.0 x_3[1] + 78.0 x_3[2] + 91.0 x_3[3] <= 90
  R2: 210.0 x_3[0] + 335.0 x_3[1] + 280.0 x_3[2] + 265.0 x_3[3] >= 270
  R3: 210.0 x_3[0] + 335.0 x_3[1] + 280.0 x_3[2] + 265.0 x_3[3] <= 280
```