HW 3

By ShineHale(yunsong Yang) U202115980

Before we solve the problems, we need to import the related library, like `numpy`, `cmath`, `pulp` (which is the most important)

Introduction to pulp

`pulp` lib is a open-source library which has a powerful capability for solving the linear programming

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pulp
import math
```

T1

Q1: how to allocate the amount of the water, the company can earn more profits

From the background of the question, we can find when the Water Diversion Management Fee is smaller, the profit is bigger. we can easily find the final target formula we wanna solve

$$max\ Z = 290x_{11} + 320x_{12} + 230x_{13} + 280x_{14} + 310x_{21} + 320x_{22} + 260x_{23} + 300x_{24} + 260x_{31} + 250x_{32} + 220x_{33}$$

and the constrains are as follows

$$x_{11} + x_{12} + x_{13} + x_{14} <= 50$$
$$x_{21} + x_{22} + x_{23} + x_{24} <= 60$$
$$x_{31} + x_{32} + x_{33} <= 50$$
$$30 <= x_{11} + x_{21} + x_{31} <= 80$$
$$70 <= x_{12} + x_{22} + x_{32} <= 140$$
$$10 <= x_{13} + x_{23} + x_{33} <= 30$$
$$10 <= x_{14} + x_{24} <= 50$$
$$x_{34} = 0$$

design the codes for solving as follow:

In [ ]:
```
prob1 = pulp.LpProblem('T1_Q1', pulp.LpMaximize)

x1 = [[pulp.LpVariable(f'x_{i + 1}_{j + 1}', upBound = 100, lowBound = 0, cat = 'Continuous')
incomes = np.array([[450 for j in range(4)] for i in range(3)])
price = np.array([[160, 130, 220, 170],
                  [140, 130, 190, 150],
                  [190, 200, 230, 0]])
profits = incomes - price

#define the function
tmp = x1[0][0] * profits[0][0]
for i in range(3):
    for j in range(4):
        if i == 0 and j == 0:
            continue
        tmp += x1[i][j] * profits[i][j]
prob1 += tmp
```

```python
#define the constrains
prob1 += x1[0][0] + x1[0][1] + x1[0][2] + x1[0][3] <= 50
prob1 += x1[1][0] + x1[1][1] + x1[1][2] + x1[1][3] <= 60
prob1 += x1[2][0] + x1[2][1] + x1[2][2] <= 50
prob1 += x1[0][0] + x1[1][0] + x1[2][0] >= 30
prob1 += x1[0][0] + x1[1][0] + x1[2][0] <= 80
prob1 += x1[0][1] + x1[1][1] + x1[2][1] >= 70
prob1 += x1[0][1] + x1[1][1] + x1[2][1] <= 140
prob1 += x1[0][2] + x1[1][2] + x1[2][2] >= 10
prob1 += x1[0][2] + x1[1][2] + x1[2][2] <= 30
prob1 += x1[0][3] + x1[1][3] >= 10
prob1 += x1[0][3] + x1[1][3] <= 50
prob1 += x1[2][3] == 0

prob1.solve()
print("Results:")
for variable in prob1.variables():
    print(variable.name, "=", variable.varValue)
print("Total Profits: ", pulp.value(prob1.objective))
```

```
Results:
x_1_1 = 0.0
x_1_2 = 50.0
x_1_3 = 0.0
x_1_4 = 0.0
x_2_1 = 0.0
x_2_2 = 50.0
x_2_3 = 0.0
x_2_4 = 10.0
x_3_1 = 40.0
x_3_2 = 0.0
x_3_3 = 10.0
x_3_4 = 0.0
Total Profits:  47600.0
```

From the outcome above, we can have the conclusion, only when all the waters are transported to the neighbours can the profits be the largest

Q2: when the supply of water has been double, how much the profit can be increased?

In [ ]:
```python
prob2 = pulp.LpProblem('T1_Q1', pulp.LpMaximize)

x2 = [[pulp.LpVariable(f'x2_{i + 1}_{j + 1}', upBound = 100, lowBound = 0, cat = 'Continuous'
incomes = np.array([[450 for j in range(4)] for i in range(3)])
price = np.array([[160, 130, 220, 170],
         [140, 130, 190, 150],
         [190, 200, 230, 0]])
profits = incomes - price

#define the function
tmp = x2[0][0] * profits[0][0]
for i in range(3):
    for j in range(4):
        if i == 0 and j == 0:
            continue
        tmp += x2[i][j] * profits[i][j]
prob2 += tmp


#define the constrains
prob2 += x2[0][0] + x2[0][1] + x2[0][2] + x2[0][3] <= 100
prob2 += x2[1][0] + x2[1][1] + x2[1][2] + x2[1][3] <= 120
```

```
prob2 += x2[2][0] + x2[2][1] + x2[2][2] <= 100
prob2 += x2[0][0] + x2[1][0] + x2[2][0] >= 30
prob2 += x2[0][0] + x2[1][0] + x2[2][0] <= 80
prob2 += x2[0][1] + x2[1][1] + x2[2][1] >= 70
prob2 += x2[0][1] + x2[1][1] + x2[2][1] <= 140
prob2 += x2[0][2] + x2[1][2] + x2[2][2] >= 10
prob2 += x2[0][2] + x2[1][2] + x2[2][2] <= 30
prob2 += x2[0][3] + x2[1][3] >= 10
prob2 += x2[0][3] + x2[1][3] <= 50
prob2 += x2[2][3] == 0

prob2.solve()
print("Results:")
for variable in prob2.variables():
    print(variable.name, "=", variable.varValue)
print("Total Profits: ", pulp.value(prob2.objective))
print("Total increased profits is {0}".format(pulp.value(prob2.objective) - pulp.value(prob1.
```

```
Results:
x2_1_1 = 0.0
x2_1_2 = 100.0
x2_1_3 = 0.0
x2_1_4 = 0.0
x2_2_1 = 30.0
x2_2_2 = 40.0
x2_2_3 = 0.0
x2_2_4 = 50.0
x2_3_1 = 50.0
x2_3_2 = 0.0
x2_3_3 = 30.0
x2_3_4 = 0.0
Total Profits:  88700.0
Total increased profits is 41100.0
```

T2

Q:How should the procurement and processing of crude oil be arranged?

After reading the question, we have the assumption:

- Let $x1, x2, x3$ as the amount of buying the oil A in the price 10, 8, 6 repectively
- Let $x_{i,j}(i = 1, 2; j = 1, 2)$ as make $x_{i,j}$ oil from class i to class j

the target we can derive is:

$$max\ Z = (x_{1,1} + x_{2,1}) * 4.8 + (x_{1,2} + x_{2,2}) * 5.6 - 10x_1 - 8x_2 - 6x_3$$

and we can have the basic constrains:

$$x_{1,1} + x_{1,2} <= 500 + x_1 + x_2 + x_3$$
$$x_{2,1} + x_{2,2} <= 1000$$
$$\frac{x_{1,1}}{x_{1,1} + x_{2,1}} >= 0.5$$
$$\frac{x_{1,2}}{x_{1,2} + x_{2,2}} >= 0.6$$
$$(x_1 - 500)x_2 == 0$$
$$(x_2 - 500)x_3 == 0$$
$$x_1 + x_2 + x_3 <= 1500$$

since they are non-linear model, we need to introduce another variable for converting the problem to linear question, we can describe the function as follow:

$$f(x) = \begin{cases} 10x, & 0 \le x \le 500 \\ 8x + 1000, & 500 \le x \le 1000 \\ 6x + 3000, & 1000 \le x \le 1500 \end{cases}$$

so we can introduce the 0-1variables y_1, y_2, y_3 for indicating the conditions $0 \le x \le 500, 500 \le x \le 1000, 1000 \le x \le 1500$, so we have the new constrains for replacing the non-linear constrains.

$$500y_2 \le x1 \le 500y_1$$
$$500y_3 \le x2 \le 500y_2$$
$$x3 \le 500y_3$$

and the fraction constrains can be converted as follow:

$$x_{2,1} \le x_{1,1}$$
$$3x_{2,2} \le 2x_{1,2}$$

from all the above, we can have the codes as follow:

In [ ]:
```
prob3 = pulp.LpProblem('T2', pulp.LpMaximize)

x_m3_0 = [[pulp.LpVariable(f'X_{i + 1}_{j + 1}', lowBound = 0, upBound = 10000, cat = 'Contin
x_m3_1 = [pulp.LpVariable(f'x_{i + 1}', upBound = 500, lowBound = 0, cat = 'Continuous' ) for
y_m3 = [pulp.LpVariable(f'y_{i + 1}', lowBound = 0, upBound = 1, cat = 'Integer') for i in ra

q3 = (x_m3_0[0][0] + x_m3_0[1][0]) * 4.8 + (x_m3_0[0][1] + x_m3_0[1][1]) * 5.6 - 10 * x_m3_1[
prob3 += q3

#define the constrains
prob3 += x_m3_0[0][0] + x_m3_0[0][1] == 500 + x_m3_1[0] + x_m3_1[1] + x_m3_1[2]
prob3 += x_m3_0[1][0] + x_m3_0[1][1] <= 1000

prob3 += 500 * y_m3[1] <= x_m3_1[0]
prob3 += x_m3_1[0] <= 500 * y_m3[0]

prob3 += 500 * y_m3[2] <= x_m3_1[1]
prob3 += x_m3_1[1] <= 500 * y_m3[1]

prob3 += x_m3_1[2] <= 500 * y_m3[2]

prob3 += x_m3_0[1][0] <= x_m3_0[0][0]
prob3 += 3 * x_m3_0[1][1] <= 2 * x_m3_0[0][1]

prob3.solve()
print('Result:')
for variable in prob3.variables():
    print(variable.name, "=", variable.varValue)
print("Total Profits: ", pulp.value(prob3.objective))
```

```
Result:
X_1_1 = 0.0
X_1_2 = 0.0
X_2_1 = 1500.0
X_2_2 = 1000.0
x_1 = 500.0
x_2 = 500.0
x_3 = 0.0
y_1 = 1.0
y_2 = 1.0
y_3 = 1.0
Total Profits:  5000.0
```

From the value, we can easily find when we buy 1000 kg oil A and then put all the 1500kg oil A and 1000kg oil B to make the second class oil can have the largest profits, which can generate about 2500kg the second oil.

The profits are 5000k dollars.

T3

Q:How to arrange weekly production?

First of all, we should make analysis of the question, we can have the assumptions:

- Let $x_i, (i = 1, 2, 3, 4)$ as the amount of using the pattern i as generating matierals
- Let x as the total amout of the circle materials, y as the total amount of the rectangle materials, so we need $x >= 2y$
- the wasting materials are all the areas that are not used times the per price.

we can easily have the codes for solving the problem as follow:

In [ ]:
```
prob4 = pulp.LpProblem('T3', pulp.LpMaximize)

x_m4 = [pulp.LpVariable(f'x_{i + 1}', upBound = 50000, lowBound = 0, cat = 'Integer') for i i
tot_x = x_m4[0] * 10 + x_m4[1] * 4 + x_m4[2] * 16 + x_m4[3] * 5
tot_y = x_m4[0] + x_m4[1] * 2 + x_m4[3] * 4
tot_n1 = x_m4[0] + x_m4[1] + x_m4[2]

#define the problem
prob4 += tot_y * 0.1  - (tot_n1 * 24 * 24 + x_m4[3] * 32 *28 - tot_y * (2 * 2.5 * 2.5 * math.

#define the constrains
prob4 += tot_n1 <= 50000
prob4 += x_m4[3] <= 20000
prob4 += tot_x >= 2 * tot_y
prob4 += 1.5 * x_m4[0] + 2 * x_m4[1] + x_m4[2] + 3 * x_m4[3] <= 40 * 3600

prob4.solve()
print('T3 Result:')
for variable in prob4.variables():
    print(variable.name, "=", variable.varValue)
print("Total Profits: ", pulp.value(prob4.objective))
print("Total cans: ", pulp.value(tot_y))
print("Total heads: ", pulp.value(tot_x))
```

```
T3 Result:
x_1 = 0.0
x_2 = 40125.0
x_3 = 3750.0
x_4 = 20000.0
Total Profits:  4298.013921110274
Total cans:  160250.0
Total heads:  320500.0
```


```
x_1 = 0.0
x_2 = 40125.0
x_3 = 3750.0
x_4 = 20000.0
Total Profits:  4298.013921110274
Total cans:  160250.0
Total heads:  320500.0
```

From the calculation above, we can easily find the maximum profits can be earned is 4298.014, and we can generate 160250 cans. the conditions of the generating are as follows

- `pattern 2` : 40125
- `pattern 3` : 3750
- `pattern 4` : 20000