# << Only problem 3.1 will be graded. >>

## Linear optimization example

Ex. A small startup hardware company is selling smart gadgets. This company has two main products which are smartwatches, and smart TVs sold for 1,200 and 5,000 THB, respectively. The company decides that they will buy a 3D printer to manufacture their products instead of using Chinese factories. After some testing, a printer could produce a maximum of 50 watches or 10 TVs per day. Due to the issues with logistics a total 55 items could be produced per day. For the maintenance issue, at least 3 TVs have to be produced per day. Assuming that the demand for both items is unlimited, the company asks you how many watches and TVs should they produce per single printer to maximize revenue.

To solve this problem, first, we have to formulate this problem as a mathematical program.

To model the program we should :

1. Identifying the decision variable.
2. Identifying the objective.
3. Identifying the constraints.

A mathematical program for this example is :

### Decision variable

$x_1$ : The amount of smartwatches produced per day $x_2$ : The amount of TVs produced per day

$$Objective : max(1200x_1 + 5000x_2)$$

$$\begin{aligned} \frac{1}{50}x_1 + \frac{1}{10}x_2 &\leq 1 \\ s.t. \qquad x_1 + x_2 &\leq 55 \\ x_2 &\geq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

After the linear program is modeled, we then convert the program into a standard form.

$$Objective : min(-1200x_1 - 5000x_2))$$

$$\begin{aligned} \frac{1}{50}x_1 + \frac{1}{10}x_2 + x_3 &= 1 \\ s.t. \qquad x_1 + x_2 + x_4 &= 55 \\ x_2 - x_5 &= 3 \\ x_1, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

After the problem is converted into a standard form, we then convert it into a matrix form.

$$\begin{aligned} min \quad & \mathbf{c}^T\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

$$where \qquad \mathbf{c} = \begin{bmatrix} -1200 \\ -5000 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} \frac{1}{50} & \frac{1}{10} & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 1 \\ 55 \\ 3 \end{bmatrix}$$

After the matrix from is obtained, we then feed the matrixs into a linear optimization library to solve for an optimal solution and optimal value.

# Import library

In [1]:
```
from scipy.optimize import linprog
import numpy as np
```

# Creating a matrix form of the problem

In [2]:
```
c_T = np.array([-1200, -5000, 0, 0, 0])
A = np.array(
    [
     [1/50     , 1/10    , 1,  0,  0],
     [1          , 1         , 0, 1,  0],
     [0          , 1         , 0, 0, -1],
    ]
)
bound = [[0, None], [0, None], [0, None], [0, None], [0, None]] # bound for each
b = [1, 55, 3]
```

# Solving the optimization problem

It was found that x = [35, 3, 0, 17, 0], which means that we should produce 35 watches and 3 TVs per day.

In [3]:
```
result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
print("{}\n\n optimal value is {} \n optimal soultion is {}".format(result,resul
```

```
message: Optimization terminated successfully.
success: True
 status: 0
    fun: -57000.0
      x: [ 3.500e+01  3.000e+00  0.000e+00  1.700e+01  0.000e+00]
    nit: 4

optimal value is -57000.0
optimal soultion is [35.  3.  0. 17.  0.]
```

```
C:\Users\BBB\AppData\Local\Temp\ipykernel_9516\333211611.py:1: DeprecationWarnin
g: `method='simplex'` is deprecated and will be removed in SciPy 1.11.0. Please u
se one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
```

# Example of wrong matrix formation

Instead of producing at least 3 TVs, we mistype and produce at least 15 TVs instead.
Thus, a feasible solution could not be found.

In [4]:
```
new_b = [1, 55, 15]
result = linprog(c = c_T, A_eq = A, b_eq = new_b, bounds=bound, method='simplex'
print(result)
```

```
 message: Phase 1 of the simplex method failed to find a feasible solution. The p
seudo-objective function evaluates to 5.0e+00 which exceeds the required toleranc
e of 1e-09 for a solution to be considered 'close enough' to zero to be a basic s
olution. Consider increasing the tolerance to be greater than 5.0e+00. If this to
lerance is unacceptably  large the problem may be infeasible.
 success: False
  status: 2
     fun: -50000.0
       x: [ 0.000e+00  1.000e+01  0.000e+00  4.500e+01  0.000e+00]
     nit: 2
```

```
C:\Users\BBB\AppData\Local\Temp\ipykernel_9516\1043972956.py:2: DeprecationWarnin
g: `method='simplex'` is deprecated and will be removed in SciPy 1.11.0. Please u
se one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  result = linprog(c = c_T, A_eq = A, b_eq = new_b, bounds=bound, method='simple
x')
```

If we remove the production constraint. The solution becomes unbounded.

In [5]:
```
c_T = np.array([-1200, -5000,  0])
A = np.array(
    [
     [0, 1, -1]
    ]
)
bound = [[0, None], [0, None], [0, None],] # bound for each variables (0, inf)
b = [3]
result = linprog(c = c_T, A_eq = A, b_eq = b, bounds=bound, method='simplex')
print(result)
```

```
 message: If feasible, the problem is (trivially) unbounded due  to a zero column
in the constraint matrices. If you wish to check whether the problem is infeasibl
e, turn presolve off.
 success: False
  status: 3
     fun: -inf
       x: [       inf  0.000e+00  0.000e+00]
     nit: 0
```

```
C:\Users\BBB\AppData\Local\Temp\ipykernel_9516\2950711804.py:9: DeprecationWarnin
g: `method='simplex'` is deprecated and will be removed in SciPy 1.11.0. Please u
se one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  result = linprog(c = c_T, A_eq = A, b_eq = b, bounds=bound, method='simplex')
```

# Tips and tricks

You can create an identiy matrix by using `np.eye`.

In [6]:
```
x = np.eye(5)
print(x)
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

In [7]:
```
y = np.zeros((8, 8))
x = np.eye(5)
y[2: 7, 2: 7] =x
print(y)
```

```
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]
```

You can also index data diaganolly.

In [8]:
```
a = np.array([3, 2, 1])
print(np.diag(a))
```

```
[[3 0 0]
 [0 2 0]
 [0 0 1]]
```

In [9]:
```
y = np.zeros((6, 6))
y[2: 5, 2: 5] =np.diag(a)
print(y)
```

```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 3. 0. 0. 0.]
 [0. 0. 0. 2. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

In [10]:
```
y = np.zeros((6, 6))
di  = np.diag_indices(len(a))
y[di] = a
print(y)
y[di[0], di[1] + 1] = a* 2
print(y)
```

```
[[3. 0. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
[[3. 6. 0. 0. 0. 0.]
 [0. 2. 4. 0. 0. 0.]
 [0. 0. 1. 2. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

In [11]:
```python
# another useful python trick
bound = [[0, None]] * 5
print(bound)
```

```
[[0, None], [0, None], [0, None], [0, None], [0, None]]
```

# Problem 1 : Skill Check

Solve the following program :

$$Objective : max(3x + 4y)$$

$$s.t. \begin{array}{r} x + 2y \leq 7 \\ 3x - y \geq 0 \\ x - y \leq 2 \\ x, y \geq 0 \end{array}$$

In [12]:
```python
c_T = np.array([-3, -4, 0, 0, 0])
A = np.array(
    [
     [1  , 2   , 1,  0,  0],
     [3  , -1   , 0, -1,  0],
     [1  , -1   , 0, 0, 1],
    ]
)
bound = [[0, None], [0, None], [0, None], [0, None], [0, None]] # bound for each
b = [7, 0, 2]
```

In [13]:
```python
result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
print("{}\n\n optimal value is {} \n optimal soultion is {}".format(result,resul
```

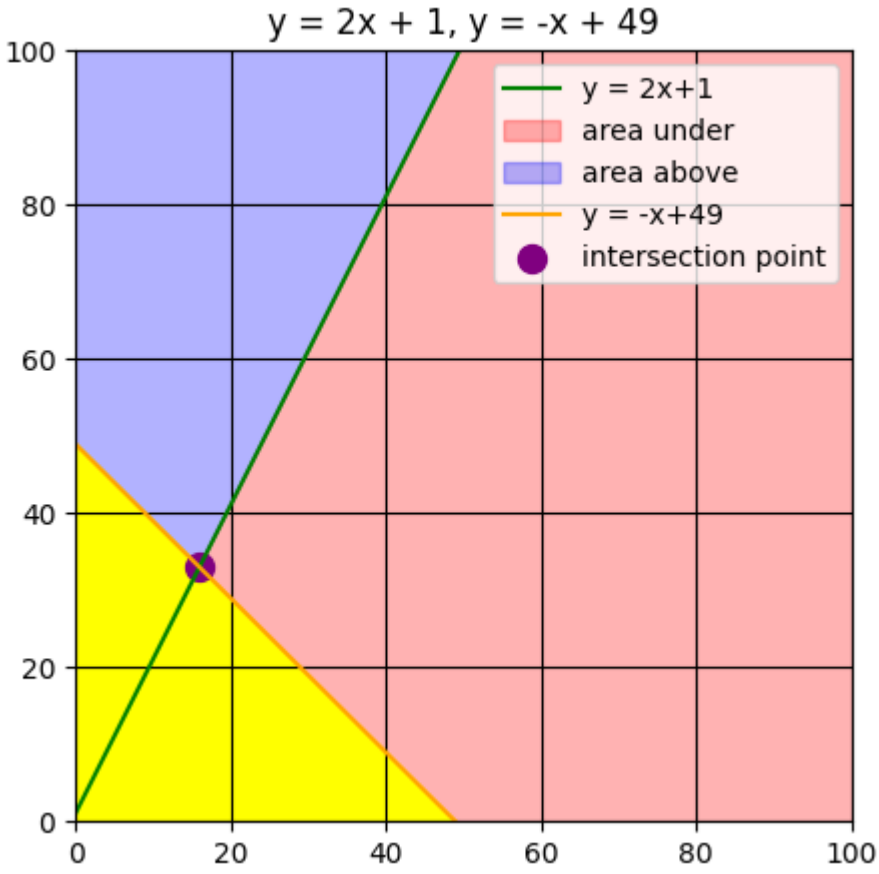
```
message: Optimization terminated successfully.
success: True
 status: 0
    fun: -17.666666666666668
      x: [ 3.667e+00  1.667e+00  0.000e+00  9.333e+00  0.000e+00]
    nit: 4

optimal value is -17.666666666666668
optimal soultion is [3.66666667 1.66666667 0.         9.33333333 0.        ]
```

```
C:\Users\BBB\AppData\Local\Temp\ipykernel_9516\333211611.py:1: DeprecationWarnin
g: `method='simplex'` is deprecated and will be removed in SciPy 1.11.0. Please u
se one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
```

# Problem 1.2 :

Solve problem 1 by using a graphical method and draw an isoprofit line. Does the obtained solution the same as the one we get in problem 1? You can solve this problem on your tablet/paper or use a plotting library.

In [14]:
```
# basic matplotlib command
import matplotlib.pyplot as plt
x = np.linspace(0, 1000, 100)
y = 2 * x + 1
y2 = -x +  49

min_y, max_y = -10000, 10000

plt.figure(figsize = (5, 5))
plt.xlim(0, 100)
plt.ylim(0, 100)
plt.title('y = 2x + 1, y = -x + 49')
plt.plot(x , y, color = 'green' , label = 'y = 2x+1')
plt.fill_between(x, y, min_y, color = 'red', alpha = 0.3, label = 'area under')
plt.fill_between(x, y, max_y, color = 'blue', alpha = 0.3, label = 'area above')

plt.plot(x , y2, color = 'orange', label = 'y = -x+49')
plt.fill_between(x, y2, min_y, color = 'yellow', alpha = 1)

plt.scatter(16, 33, s = 100, color = 'purple',  label = 'intersection point')
plt.legend()
plt.grid(color = 'black')

plt.show()
```




## Problem 2 : Hamtaro factory

After Hamtaro finished planting the sunflower field, he then aims to open the Hamtaro factory, selling sunflower snacks to the fellow hamsters. However, the harvested sunflower seeds have different grades, which leads to different nutrition values. Nutrition values for each grade are shown in the table below. To sell the snack at the market, the HFA (Hamster Food Administration) states that the snack they produce must contain at least 60% protein and no more than 5% fat. To minimize the ingredient cost, how should Hamtaro mix the sunflower seeds to pass the HFA approval? Formulate the problem as a linear program and solve for an optimal solution.

| sunflower seed grade | % protein | % fat | cost per g (THB) |
|---|---|---|---|
| terrible | 25 | 10 | 0.01 |
| low | 40 | 7 | 0.3 |
| medium | 70 | 4 | 0.7 |
| high | 90 | 1 | 1.2 |

In [15]:
```
pass
```

## Problem 3.1 : Storage server management

A company is running a video analytic system. To store the analyzed videos, the company has 10 local storage servers to store the data, of which each of them could store 24 TB

per server. Recently, the company finds out that the servers they have will not be adequate in the recent future. Therefore, the company has to figure out a plan to scale up its resource. The CTO has come up with two solutions, which are buying new storage servers, and using a cloud storage service called 'SWA S3'. The cloud storage service charges 690 THB/TB per month. On the other hand, buying a new server costs 40,000 THB, but it could be used for a very long time. After several discussions, the company has projected the amount of storage required for each month. The projected data is shown in the table below. To minimize the cost, what should the company do to store the data? Formulate the problem as a linear program and solve for an optimal solution.

**Note 1 : The optimal solution does not have to be an integer. Note 2 : The company could buy new servers at any month.**

| Month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |:-:|:-:|:-:|:-:|:-:|:-:|:-:|:-:| | Estimated amount of storage required (TB) | 140 | 200 | 300 | 1000 | 1400| 500 | 600 | 900 | 700 |

3.1)

| month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Estimated storage required (TB) | 140 | 200 | 300 | 1000 | 1400 | 500 | 600 | 900 |

$x_i$ = จำนวน server ที่ซื้อตอนเดือนที่ i

$y_i$ = ——— cloud ———

obj : $\min\left(690(y_1+y_2+y_3+y_4+y_5+y_6+y_7+y_8)+40000(x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8)\right)$ #cost

st :

เพียงพอทุกเดือน

$240+y_1+24x_1 \geq 140 \longrightarrow y_1+24x_1-e_1=-100$

$240+y_2+24(x_1+x_2) \geq 200 \longrightarrow y_2+24(x_1+x_2)-e_2=-40$

$240+y_3+24(x_1+x_2+x_3) \geq 300 \longrightarrow y_3+24(x_1+x_2+x_3)-e_3=60$

$240+y_4+24(x_1+x_2+x_3+x_4) \geq 1000 \longrightarrow y_4+24(x_1+x_2+x_3+x_4)-e_4=760$

$240+y_5+24(x_1+x_2+x_3+x_4+x_5) \geq 1400 \longrightarrow y_5+24(x_1+x_2+x_3+x_4+x_5)-e_5=1160$

$240+y_6+24(x_1+x_2+x_3+x_4+x_5+x_6) \geq 500 \longrightarrow y_6+24(x_1+x_2+x_3+x_4+x_5+x_6)-e_6=260$

$240+y_7+24(x_1+x_2+x_3+x_4+x_5+x_6+x_7) \geq 600 \longrightarrow y_7+24(x_1+x_2+x_3+x_4+x_5+x_6+x_7)-e_7=360$

$240+y_8+24(x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8) \geq 900 \longrightarrow y_8+24(x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8)-e_8=660$

$x_i, y_i \geq 0$

C_T = [690] * 8 + [40000] * 8 + [0] * 8

X_T = [$y_1\ y_2\ y_3\ y_4\ y_5\ y_6\ y_7\ y_8\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7\ e_8$]

b = [-100 -40 60 760 1160 260 360 660]

$$A = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 24 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 24 & 24 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 24 & 24 & 24 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 24 & 24 & 24 & 24 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 24 & 24 & 24 & 24 & 24 & 24 & 24 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}$$

In [16]:

```
c_T = np.array([ 690, 690, 690, 690, 690, 690, 690, 690, 40000, 40000, 40000, 40
A = np.array(
    [
        [1]     + [0] * 7 +          [24]     + [0] * 7 + [-1] + [0] * 7,
        [0] * 1 + [1] + [0] * 6    + [24] * 2 + [0] * 6 + [0] * 1 + [-1] + [0] *
        [0] * 2 + [1] + [0] * 5    + [24] * 3 + [0] * 5+ [0] * 2 + [-1] + [0] *
        [0] * 3 + [1] + [0] * 4    + [24] * 4 + [0] * 4+ [0] * 3 + [-1] + [0] *
        [0] * 4 + [1] + [0] * 3    + [24] * 5 + [0] * 3+ [0] * 4 + [-1] + [0] *
        [0] * 5 + [1] + [0] * 2    + [24] * 6 + [0] * 2+ [0] * 5 + [-1] + [0] *
        [0] * 6 + [1] + [0] * 1    + [24] * 7 + [0] * 1+ [0] * 6 + [-1] + [0] *
        [0] * 7 + [1]              + [24] * 8 + [0] * 7 + [-1]
    ]
)
```

```
bound = [[0, None]]*24 # bound for each variables (0, inf)
b = [-100, -40, 60, 760, 1160, 260, 360, 660]
```

In [17]:
```
result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
print("{}\n\n optimal value is {} \n optimal soultion is {}".format(result,resul
print("---------------------")
print("cloud : ", result.x[0:8])
print("server : ", result.x[8:16])
print("e : ", result.x[16:24])
```

```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: 1514000.0
       x: [ 0.000e+00  0.000e+00 ...  3.000e+02  0.000e+00]
     nit: 12

 optimal value is 1514000.0
 optimal soultion is [  0.    0.    0.  100.  500.    0.    0.    0.    0.    0.
2.5  25.
   0.    0.    0.    0.  100.   40.    0.    0.    0.  400.  300.    0. ]
---------------------
cloud :  [  0.   0.   0. 100. 500.   0.   0.   0.]
server :  [ 0.   0.   2.5 25.   0.   0.   0.   0. ]
e :  [100.  40.   0.   0.   0. 400. 300.   0.]
```

```
C:\Users\BBB\AppData\Local\Temp\ipykernel_9516\2474110378.py:1: DeprecationWarnin
g: `method='simplex'` is deprecated and will be removed in SciPy 1.11.0. Please u
se one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  result = linprog(c = c_T, A_eq = A, b_eq=b, bounds=bound, method='simplex')
```

## Problem 3.2

From problem 3. Would the optimal solution change if the company has to pay 2,000 THB maintenance fee per month for each local storage server they have?

In [20]:
```
pass
```

## Problem 4 : Task assignment

A data center company has a lot of servers to be maintained. Thus, the maintainers are required 24/7. To maintain the servers, the company has employed four technicians, of which one of them is a senior. Each technician has to work at least 40 hours per week, except for the senior one, which works exactly 36 hours per week. The wage is paid hourly, and every technician has different wages and availability. The maintenance is performed with exactly one person on duty. How should the company assign each person to be on duty to minimize the maintenance cost? Formulate the problem as a linear program and solve for an optimal solution.

| Technician | Wage per hour(weekday) | Wage per hour(weekend) | Maximum hours of avaiability | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Mon. | Tue. | Wed. | Thurs. | Fri. | Sat. | Sun. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Technician A | 70 | 80 | 12 | 10 | 12 | 0 | 8 | 8 | 4 |
| Technician B | 70 | 80 | 12 | 10 | 0 | 8 | 8 | 4 | 8 |
| Technician C | 80 | 75 | 0 | 10 | 8 | 8 | 0 | 10 | 10 |
| Senior technician D | 160 | 200 | 24 | 0 | 24 | 24 | 24 | 24 | 24 |

In [21]:
```
pass
```

# Problem 5 : Courier service

A company named "Curry" provides courier services to the customer. In town X, 10 customers are using this company's service. To satisfy the demand in this town, the company has placed the courier office as a hub for the couriers to collect customers' packages. However, only a limited amount of packages could be stockpiled in each office. The company has to pay the couriers 10 THB per kg per distance they have traveled in kilometer. Moreover, the package could be partially collected (divisible), i.e., a courier from office A and C could collect 200 kg, 300 kg package from customer 1. How should this company plan to send the couriers to collect the package such that the package collection cost is minimized? The amount of packages to be collected for each customer is shown in Table 2.1. The size of each package storage is shown in Table 2.2. The distance from the courier office to each customer is shown in Table 2.3. Formulate the problem as a linear program and solve for an optimal solution.

**Note : The optimal solution does not have to be an integer.**

Table 2.1. Amount of packages to be collected for each customer.

| Customer | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Amount of packages to be collected (kg) | 500 | 200 | 1000 | 2000 | 150 | 20 | 350 | 250 | 375 | 1 |

Table 2.2. The size of package storage for each office.

| Office | Storage size (kg) |
|---|---|
| A | 1500 |
| B | 1000 |
| C | 4000 |

Table 2.3. Distance from the office to each household (km)

| From / to | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| A | 4 | 3 | 8 | 1 | 2 | 1 | 3 | 2 | 4 | 4 |
| B | 3 | 1 | 1 | 8 | 3 | 4 | 5 | 7 | 9 | 8 |
| C | 8 | 6 | 5 | 3 | 1 | 6 | 7 | 2 | 7 | 5 |

In [22]:
```
pass
```

# Problem 6: VM placement

A startup company is provisioning virtual machines (VM) for cloud computing. Currently, the company offers three types of instances, which are a small, medium, and large instance. The detail for each type of instance is shown in the table below. To host the VMs, the company has a large number of servers to provide its service. All of the servers they possessed have 8 CPUs core. Recently, the company has found a new customer to use their service. The customer want to host 20 small, 9 medium, and 5 large instances for their company. Moreover, the customer also demands that only they could access the provided server, i.e., the company could not use the leftover CPUs to serve other customers despite having 2 CPUs left. How many servers should the company allocate to serve this customer? Formulate the problem as a linear program and solve for an optimal solution.

| Instance type | Number of CPUs provided |
|---|---|
| small | 2 CPUs |
| medium | 3 CPUs |
| large | 4 CPUs |

**Note : The optimal solution does not have to be an integer. Hint : How many way could we fit the instances into a single server?**

In [23]:
```
pass
```

# Problem 7: Linear binary classification model (Maximal Margin Classifier)

This problem will introduce you to a basic machine learning classification algortithm. Let's say that we have the dataset $D = \{(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\}$. Each element in $D$ is a datapoint $x_i$ in a form of tuple $(x_0, x_1)$ which belong to the class $y_i \in \{-1, 1\}$. Machine learning alogrithm aims to represent the dataset $D$ using sets of parameter $\theta$. For classification task, we use machine learning alogrithm to correctly classify the class $y$ given the datapoint $x$.

In this problem, we use a linear line $\theta_0 x_0 + \theta_1 x_1 + \theta_2 = 0$ as a classifier. The datapoints above and below the line will be classified as class 1 (positive) and -1 (negative)

respectively. The objective of this problem is to find the parameters of the linear classifier $\theta$ given $D$ by formulating the problem as a linear program. A figure below demonstrates that the line with the parameter $(\theta_0, \theta_1, \theta_2) = (1.5, 1, -5)$, i.e., $1.5x_0 + x_1 - 5 = 0$, could seperate the data into two classes.

In [24]:
```
import matplotlib.pyplot as plt
```

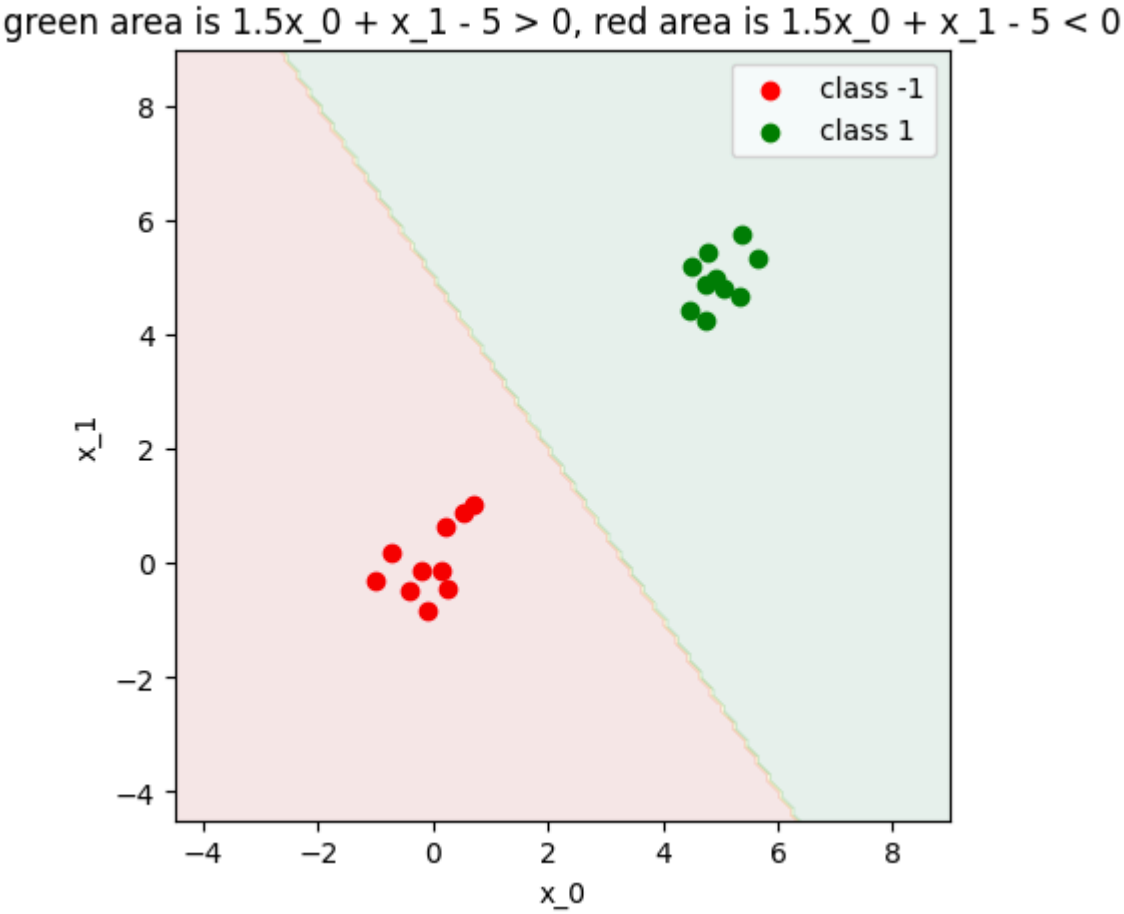
In [25]:
```
class_a = np.random.randn(10, 2)* 0.5 # generate class -1 data
class_b = np.random.randn(10, 2) * 0.5  + 5# generte class 1 data
```

In [26]:
```
print(class_a[0]) # example of datapoint
```

```
[-0.73859333  0.19525337]
```

In [27]:
```
def generate_decision_boundary(theta_0, theta_1, theta_2, xmin = -10,  ymin = -1
  '''
    Input : Parameter theta_0, theta_1, theta_2 of a linear classifier
    Output : Decision boundary ( 1 is positive, -1 is negative, 0 is seperating
  '''
  x_0, x_1 = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))
  prediction = theta_0 * x_0 + theta_1 * x_1 + theta_2
  prediction[prediction > 0] = 1
  prediction[prediction < 0] = -1
  return (x_0, x_1, prediction)

theta_0 = 1.5
theta_1 = 1
theta_2 = -5
decision_boundary = generate_decision_boundary(theta_0, theta_1, theta_2)
```

In [28]:
```
plt.figure(figsize = (5, 5))
plt.xlim(-4.5, 9)
plt.ylim (-4.5, 9)

plt.scatter(class_a[:, 0], class_a[:, 1], color = 'red', label = 'class -1')
plt.scatter(class_b[:, 0], class_b[:, 1], color = 'green', label = 'class 1')
plt.contourf(decision_boundary[0], decision_boundary[1], decision_boundary[2], c
plt.xlabel('x_0')
plt.ylabel('x_1')
plt.title('green area is 1.5x_0 + x_1 - 5 > 0, red area is 1.5x_0 + x_1 - 5 < 0'
plt.legend()
plt.show()
```




As you can see, the line $\theta_0 x_0 + \theta_1 x_1 + \theta_2 = 0$ could seperate the data into two classes by constraining $x_i$ which has the value of $\theta_0 x_{i,0} + \theta_1 x_{i,1} + \theta_2 > 0$ into a positive class $(y_i = 1)$ and $\theta_0 x_{i,0} + \theta_1 x_{i,1} + \beta_2 < 0$ into a negative class $(y_i = -1)$. Therefore, the value of $\theta$ could optimized by constraining the line to satisfy the condition of every datapoints w.r.t. its class, which result in the mathematical program shown below.

$$Objective:-$$

$$s.t. \quad \begin{aligned} \theta_0 x_{i,0} + \theta_1 x_{i,1} + \theta_2 &> 0 \\ \theta_0 x_{j,0} + \theta_1 x_{j,1} + \theta_2 &< 0 \\ \forall i, y_i &= 1 \\ \forall j, y_j &= -1 \end{aligned}$$

The program could also be written in a form of:

$$Objective:-$$

$$s.t.\ \forall i, y_i(\theta_0 x_{i,0} + \theta_1 x_{i,1} + \theta_2) > 0$$

The program above could return a feasible solution yet the wellness of the optimized value is not defined. For example, $H_2$ and $H_3$ in the image below are both acceptable solution using the program above. However, $H_3$ might be a preferred solution.

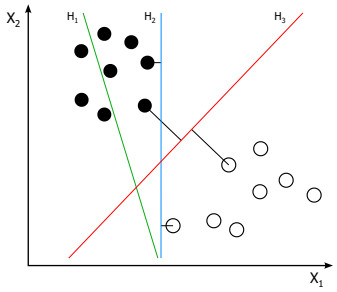



Image source:
https://upload.wikimedia.org/wikipedia/commons/b/b5/Svm_separating_hyperplanes_%28SV

Thus, we introduce the margin $\epsilon$ to define the wellness of the program. The margin is the shortest distance from the hyperplane to the closest datapoint. Maximizing $\epsilon$ would result in the line which has the largest separation between the two classes (maximum margin). As a result, the program becomes:

$$Objective: max(\epsilon)$$

$$\begin{aligned} & \theta_0 x_{i,0} + \theta_1 x_{i,1} + \theta_2 \geq \epsilon \\ & \theta_0 x_{j,0} + \theta_1 x_{j,1} + \theta_2 \leq -\epsilon \\ s.t. \quad & \forall i, y_i = 1 \\ & \forall j, y_j = -1 \\ & \theta_0, \theta_1 \in [-1, 1] \end{aligned}$$

The program could also be written in a form of:

$$Objective: max(\epsilon)$$

$$\begin{aligned} s.t. \quad & \forall i, y_i(\theta_0 x_{i,0} + \theta_1 x_{i,1} + \theta_2) \geq \epsilon \\ & \theta_0, \theta_1 \in [-1, 1] \end{aligned}$$

$\theta_0, \theta_1$ are both bounded to $[-1, 1]$ to prevent unbounded solution.

**Note: This is a simplified version of a linear support vector machine and many details are omitted. You can read more at https://en.wikipedia.org/wiki/Support-vector_machine#Linear_SVM if interested.**

## Problem 7.1

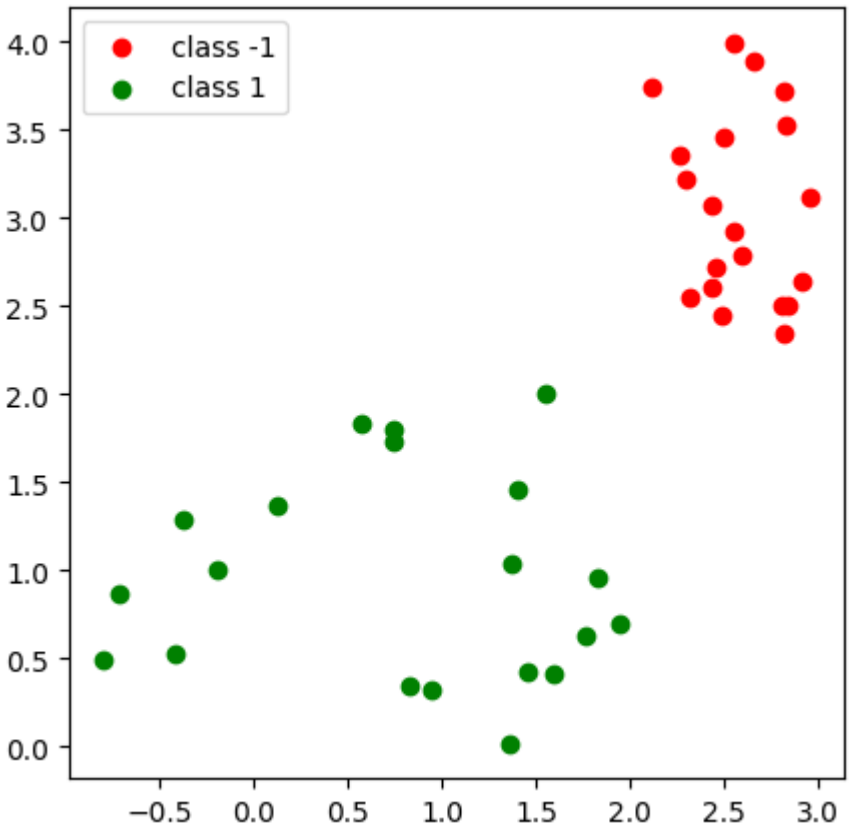
Convert the mathematical program above into a linear program. Then, use the converted program to solve for the line seperating the datapoints below. **Do not forget to visualize the solved line.**

In [29]:
```
class_a = np.array([[np.random.uniform(2.1, 3), np.random.uniform(2.3, 4)] for _
class_b = np.array([[np.random.uniform(-1, 2), np.random.uniform(0, 2.2)] for _
print(class_a.shape)
```

```
(20, 2)
```

In [30]:
```
plt.figure(figsize = (5, 5))
plt.scatter(class_a[:, 0], class_a[:, 1], color = 'red', label = 'class -1')
plt.scatter(class_b[:, 0], class_b[:, 1], color = 'green', label = 'class 1')
```

```
plt.legend()
plt.show()
```



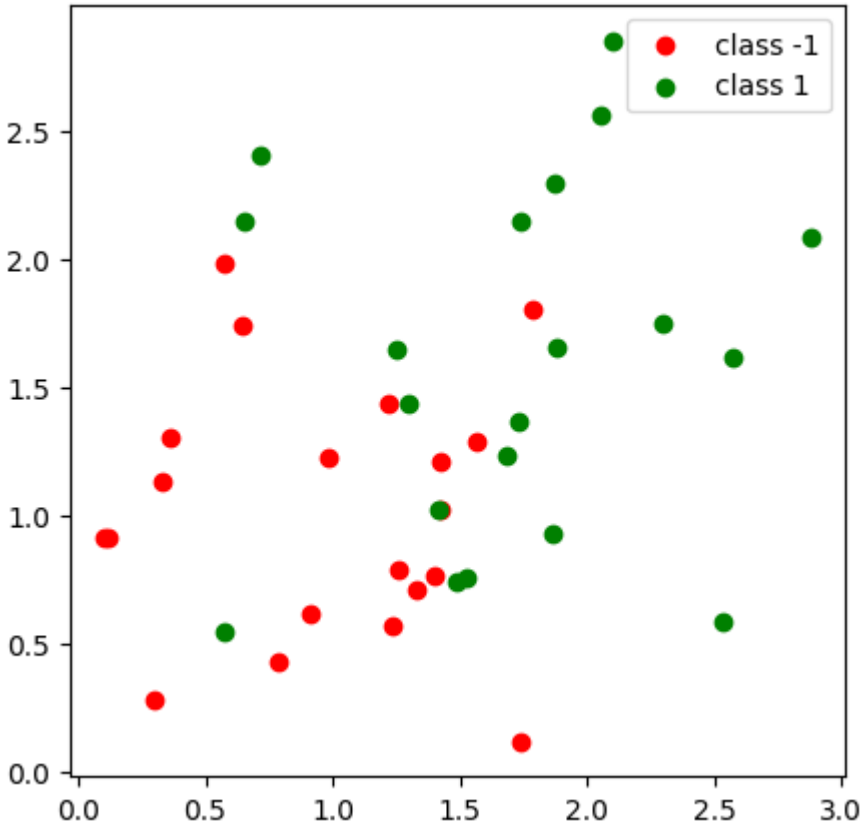

In [31]:
```
pass
```

## Problem 7.2

Given the datapoints below, repeat 7.1 and observe the result. What is happening to the solution and why does this happen?

In [32]:
```
class_a = np.random.uniform(0, 2, (20, 2) ) # class -1 data
class_b = np.random.uniform(0.5, 3, (20, 2)) # class 1 data
```

In [33]:
```
plt.figure(figsize = (5, 5))
plt.scatter(class_a[:, 0], class_a[:, 1], color = 'red', label = 'class -1')
plt.scatter(class_b[:, 0], class_b[:, 1], color = 'green', label = 'class 1')
plt.legend()
plt.show()
```




In [34]:
```
pass
```