姓名:王湛

学号:519432910019

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import combinations
from collections import deque

n=15
weight = np.array([70,73,77,80,82,87,90,94,98,106,110,113,115,118,120])#w is weight,c is capacit
capacity = 750
profit = np.array([135,139,149,150,156,163,173,184,192,201,210,214,221,229,240])
```

In [2]:
```
#所有可能的递归生成
def generate_binary_arrays(n, array=[]):
    if n == 0:
        print(array)
    else:
        array.append(0)
        generate_binary_arrays(n - 1, array)
        array.pop()
        array.append(1)
        generate_binary_arrays(n - 1, array)
        array.pop()

# 生成长度为15的所有由0和1组成的数组
length = 15
generate_binary_arrays(length)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0]
```

In [3]:
```
domain=[]
best_profit=0
best_comb=np.zeros(n)
#穷举法
for i in range(2**n):
    # 将整数转换为二进制字符串，并去掉开头的'0b'标识符
    binary_string = bin(i)[2:]
    comb = [int(bit) for bit in binary_string.zfill(n)]
    if np.dot(comb,weight)<= capacity:
        domain.append(comb)
        if best_profit<=np.dot(comb,profit):
            best_profit = np.dot(comb,profit)
            best_comb = comb

print(best_comb,np.dot(best_comb,weight),best_profit)#穷举法的最优解，重量，价值
```

```
[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1] 749 1458
```

In [4]:
```
def neighbour(a,dis):#以a为圆心，dis为半径的邻域中的所有元素
    neigh=[]
    for i in range(1,dis+1):#最少修改1位,最多修改dis位
        for j in list(combinations(range(len(a)),i)):#具体修改的位置
            acopy=a.copy()
            for k in j:#逐位置修改
                acopy[k]=(acopy[k]+1)%2
            if np.dot(acopy,weight)<capacity: #重量不超过capacity
                neigh.append(acopy)
    return neigh
```

In [5]:
```
def neigh_best(curr_comb,dis):
    neigh_best_comb=np.zeros(n)
    neigh_best_profit=0
    for i in neighbour(curr_comb,dis):#对于所有hamming距离小于等于2的点，通过改变几个个位置来遍
        flag=1
        for j in tabu_list:#确定i是否在禁忌表中
            if not(i-j).any():#只有所有元素均相等时.any()返回false，也就是i在tabu_list中时，fla
                flag=0
                break
        if np.dot(i,profit) > neigh_best_profit and flag:  #大于目前邻域内的最优值，且不在禁忌表
            neigh_best_comb=i.copy()
            neigh_best_profit=np.dot(i,profit)
    return neigh_best_comb
```

In [7]:
```
#tabu search
curr_comb=np.zeros(n)#这里可以随机生成初始点
tabu_comb=np.zeros(n)
tabu_profit=0
tabu_list=deque(maxlen=5)#禁忌队列长度为5
endstep=100 #最大运行步数，停止条件
dis=2
#在领域中选最好的一点，如果大于目前的最优值，直接替换，如果小于等于，则将目前的最优值和点列入禁忌
i=0
while i <=endstep:
    neigh_best_comb = neigh_best(curr_comb,dis)
    tabu_list.append(curr_comb)#重点：把上一步列为禁忌
#     print(tabu_list)
    if np.dot(neigh_best_comb,profit)>np.dot(tabu_comb,profit):
        tabu_comb=neigh_best_comb.copy()#只用保持对历史最优解的更新
    curr_comb=neigh_best_comb.copy()
    i=i+1
print(tabu_comb,np.dot(tabu_comb,weight),np.dot(tabu_comb,profit))#输出组合方式，重量，价值
```

```
[0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1.] 749.0 1449.0
```

In [8]:
```
def f(x):
    return -np.dot(x,profit)#按照极小化，这里应该是负的
```

In [9]:
```
def p(x_next,x):
    return e**-(f(x_next)-f(x))/T
```

In [10]:
```
#simulated annealing algorithm
e=2.71828
curr_comb=np.zeros(n)#这里可以随机生成初始点
sa_comb=np.zeros(n)
T=10#初始温度
k=0.5#降温速度
sa_profit=0
endstep=100 #最大运行步数，停止条件
dis=2
i=0
while i<=endstep:
    next_comb=random.choice(neighbour(curr_comb,dis))
    if f(next_comb)>f(curr_comb):
        if random.random()<p(next_comb,curr_comb):
            curr_comb = next_comb.copy()
    else:
        curr_comb = next_comb.copy()
    if f(next_comb)<f(sa_comb):
        sa_comb = next_comb.copy()
    T=k*T
    i=i+1

print(sa_comb,np.dot(sa_comb,weight),np.dot(sa_comb,profit))
```

```
[0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 1. 1. 1. 1. 1.] 748.0 1443.0
```