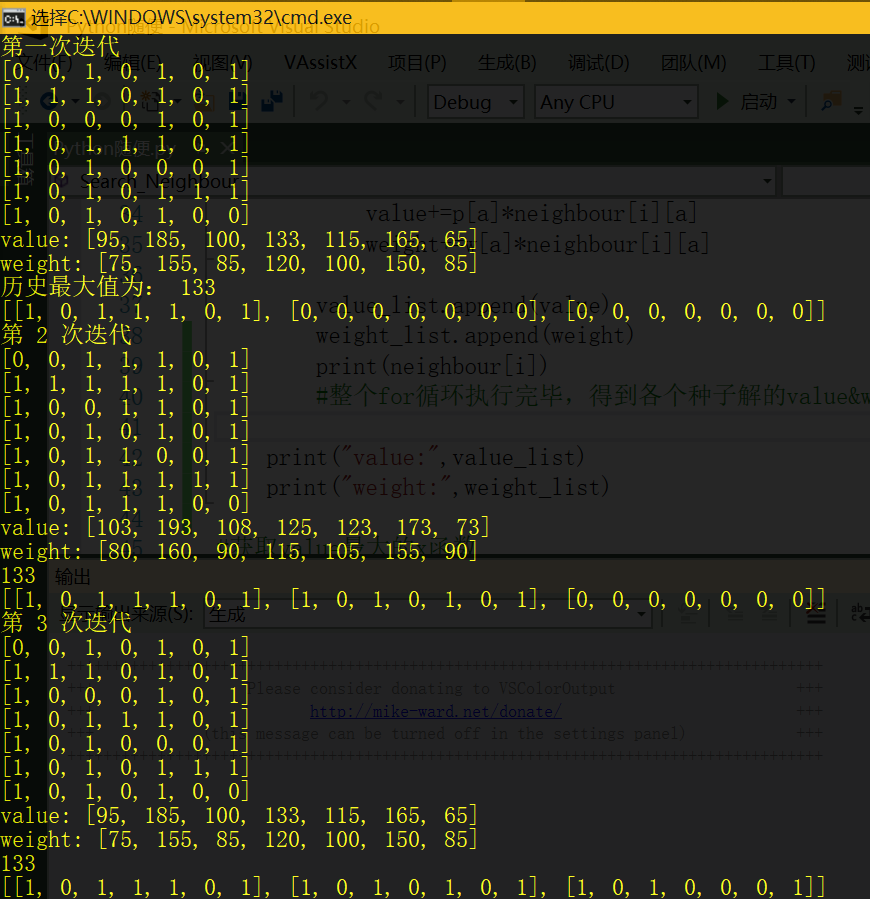
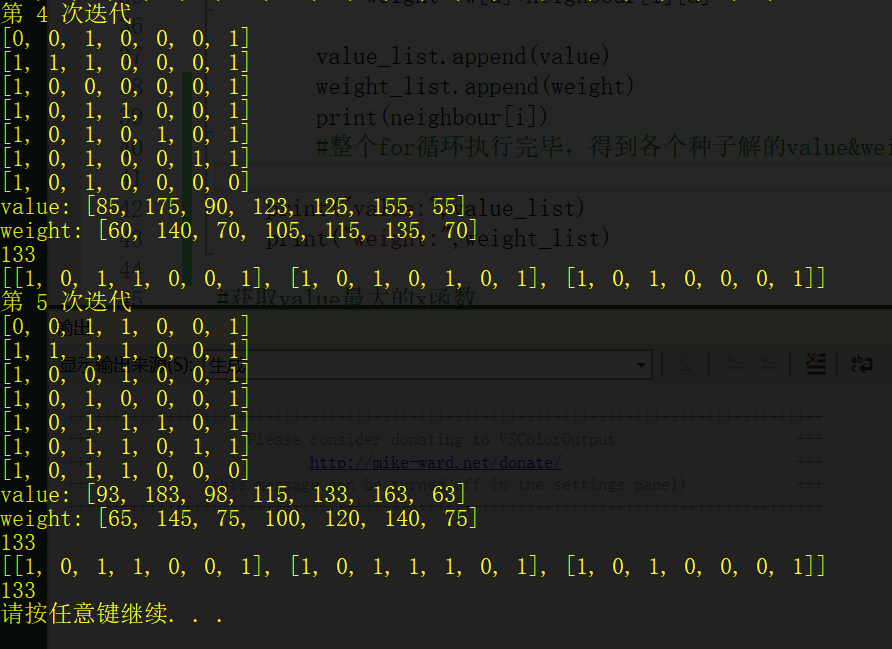
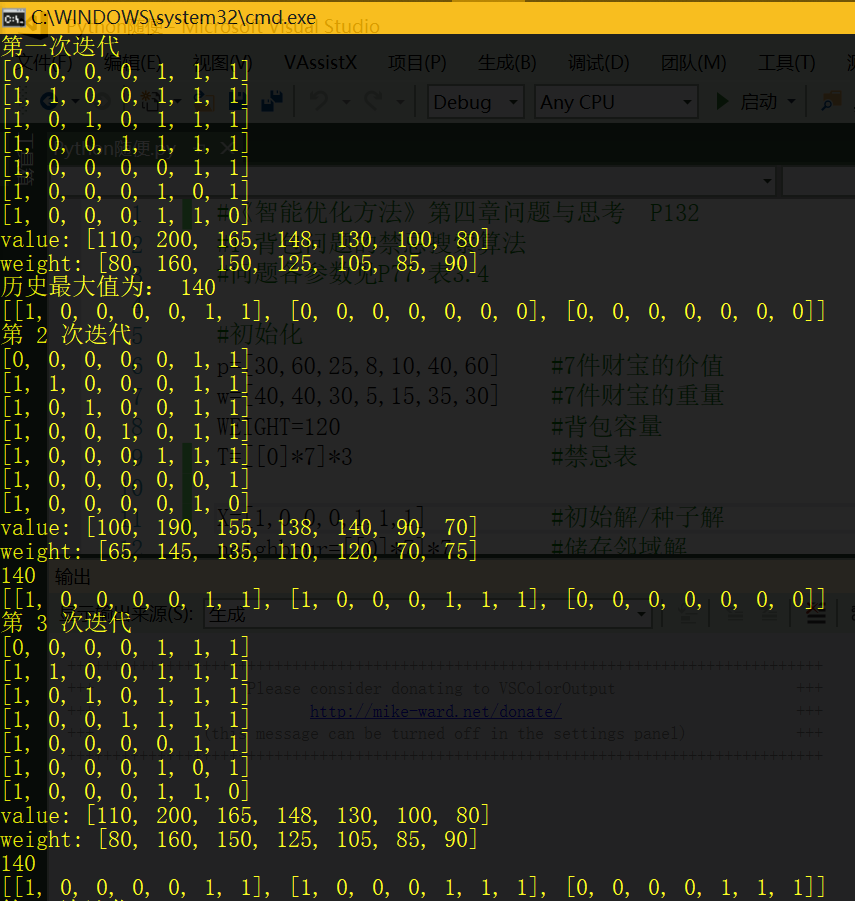
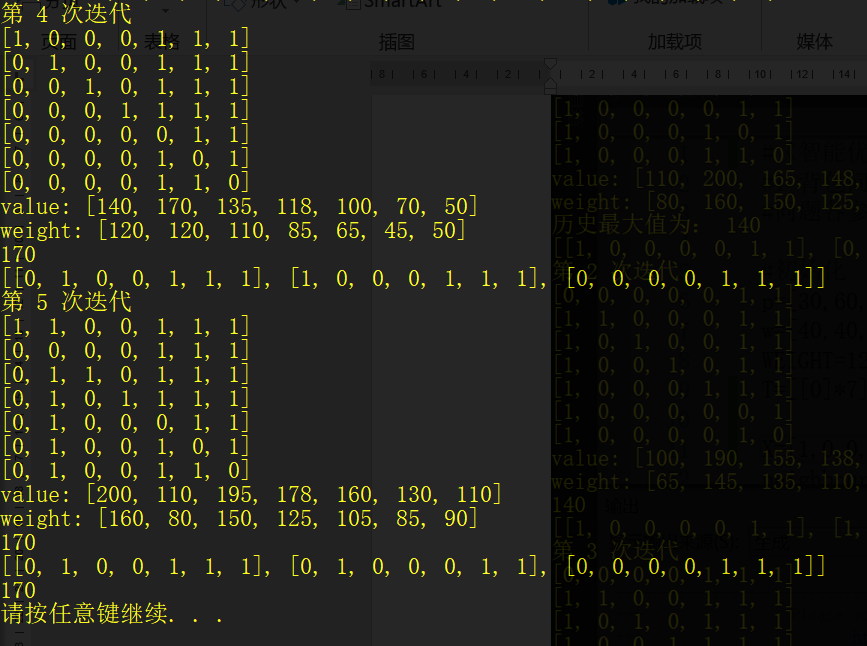
以X=1010101为初值 进行五次迭代 得到历史最优解为133





以初值X=1000111进行五次迭代 得到历史最优解为170 证明初值对禁忌搜索算法最终结果影响较大





源代码如下：

#《智能优化方法》第四章问题与思考 P132

#1 背包问题的禁忌搜索算法

#问题各参数见P77 表3.4

#初始化

p=[30,60,25,8,10,40,60] #7件财宝的价值

w=[40,40,30,5,15,35,30] #7件财宝的重量

WEIGHT=120 #背包容量

T=[[0]\*7]\*3 #禁忌表

X=[1,0,0,0,1,1,1] #初始解/种子解

neighbour=[[0]\*7]\*7 #储存邻域解

max\_value=X[0]\*p[0]+X[1]\*p[1]+X[2]\*p[2]+X[3]\*p[3]+X[4]\*p[4]+X[5]\*p[5]+X[6]\*p[6] #依据初始解给出

count=0 #禁忌表计数器

value\_list=[] #与neighbour一一对应记录value

weight\_list=[] #与neighbour一一对应记录weight

#邻域搜索函数 得到邻域解的value weight信息

def Search\_Neighbour(good\_x): #输出一个种子解

#每次循环寻找邻域一个解 所有循环结束得到邻域解的value--weight--list

for i in range(7):

neighbour[i]=good\_x[:] #初始邻域----传值不传指

#改变位值

if neighbour[i][i]==1:

neighbour[i][i]=0;

else:

neighbour[i][i]=1

#计算value && weight

value=weight=0

for a in range(7): #计算此邻域解value-weight

value+=p[a]\*neighbour[i][a]

weight+=w[a]\*neighbour[i][a]

value\_list.append(value)

weight\_list.append(weight)

print(neighbour[i])

#整个for循环执行完毕，得到各个种子解的value&weight -list与邻域解一一对应

print("value:",value\_list)

print("weight:",weight\_list)

#获取value最大的x函数

def Get\_Max\_Value\_x():

global max\_value,count

#数据清洗,剔除超重项，其余项均理论可用

for i in range(7):

if weight\_list[i]>120:

value\_list[i]=0

weight\_list[i]=0

#循环只是为了最后一个else的特殊处理

#正常情况下前两个if可解决，直接return退出循环

for i in range(7):

if (neighbour[value\_list.index(max(value\_list))] not in T): #如果最大值不在禁忌表，直接返回

max\_value = max( max(value\_list) , max\_value ) #更新历史最值

T[count%3]=neighbour[value\_list.index(max(value\_list))][:]

count+=1

return neighbour[value\_list.index(max(value\_list))][:]

else:

if max(value\_list)>max\_value: #虽然在禁忌表，但优于历史最优解，直接返回

max\_value=max(value\_list)

T[count%3]=neighbour[value\_list.index(max(value\_list))][:]

count+=1

return neighbour[value\_list.index(max(value\_list))][:]

else: #在禁忌表，不优于历史最优，不能破禁，寻找次优解----接受劣解

weight\_list[value\_list.index(max(value\_list))]=0

value\_list[value\_list.index(max(value\_list))]=0

#结束循环，重新寻找

for i in range(5):

#第一次迭代

if i==0:

print("第一次迭代")

Search\_Neighbour(X)

good\_x=Get\_Max\_Value\_x()

print("历史最大值为：",max\_value)

print(T)

else:

print("第",i+1,"次迭代")

value\_list=[]

weight\_list=[]

Search\_Neighbour(good\_x)

good\_x=Get\_Max\_Value\_x()

print(max\_value)

print(T)

print(max\_value)