* **1）给定*n*个物品，物品价值分别为*P*1，*P*2，…，*Pn*，物品重量分别*W*1，*W*2, …, *Wn*，背包容量为*M*。每种物品可部分装入到背包中。输出*X*1，*X*2，…，*Xn*，0**≤***Xi***≤**1, 使得最大，且** ≤**M。试设计一个算法求解该问题，分析算法的正确性。**

**问题分析：**

本问题为背包问题的变种，一开始考虑采用01背包问题动态规划的思想。但考虑到在该问题中，每件物品无需完全装入，可以每次都将单位价值最高的物体装入，直到装满为止。

**想法：**

先将整个物体列表按照单位价值大小排序后再往里面按顺序装到不能装为止。

**Code：**

from typing import List

'''

输入为: things列表其中每个列表元素都是个二元元列表第一个元素为物品价值第二个元素为物品的重量

w：背包容量

'''

def get\_maxValue(things:List[List],w)->float:

sorted\_things=sorted(things,key=lambda s: (s[0]/s[1]),reverse=True)

answer=0

remain=w

for i in range(len(things)):

if(remain<=sorted\_things[i][1]):

answer+=sorted\_things[i][0]\*remain/sorted\_things[i][1]

break

else:

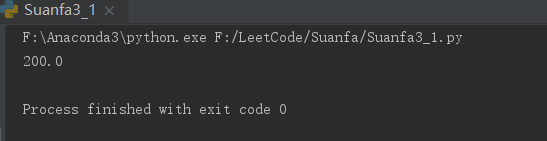
answer+=sorted\_things[i][0]

remain=remain-sorted\_things[i][1]

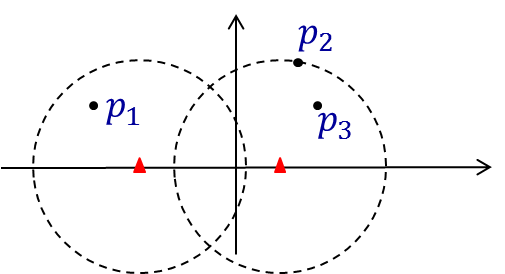
return answer

print(get\_maxValue([[50,10],[120,30],[60,20]],50))

**实验截图：**



* **2）海面上有一些船需要与陆地进行通信，需要在海岸线上布置一些基站。现将问题抽象为，在x轴上方，给出*N*条船的坐标，，，在x轴上安放的基站可以覆盖半径为d的区域内的所有点，问在x轴上至少要安放几个点才可以将x轴上方的点都覆盖起来。试设计一个算法求解该问题，并分析算法的正确性。**

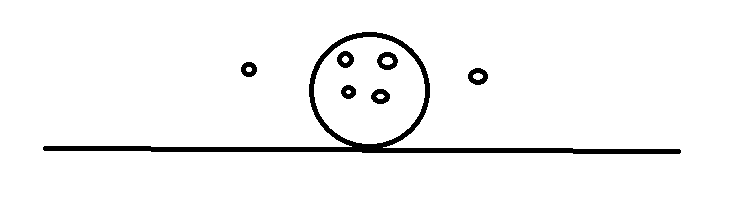


**问题分析：**

**想法一：考虑采用贪心算法，不断的获取可以为最多的船服务的基站**

**分析：**

该算法存在错误如：



这种情况下，会首先选择能包括进四条船的基站，按照这种算法至少有三个基站才能满足要求，但是显然这个图里两个基站就可以满足要求。

**想法二：**

将船点按照横坐标排序，不断获取能将最左边的点扩入的基站，并将被扩入的船移出点集。直到船点集为空。

**分析：**

假设最优解基站集合为现在去除最左边那个基站的基站集合为,去除最左边基站服务的船点集为。此时假设另有最优解所以此时。所以此时和假设条件矛盾，所以该算法有最优子结构。

**Code：**

from typing import List

'''

输入为船点集，第一个元素为横坐标第二个元素为纵坐标

'''

def get\_min\_J(P:List[List],d):

new\_P=sorted(P,key=lambda p : p[0])

print(new\_P)

Jcount=0

while new\_P!=[]:

x=pow(pow(d,2)-pow(new\_P[0][1],2),0.5)

J\_x=new\_P[0][0]+x

tmpnew\_P=new\_P

for p in new\_P:

if pow(p[0]-J\_x,2)+pow(p[1],2)<=pow(d,2):

tmpnew\_P.remove(p)

new\_P=tmpnew\_P

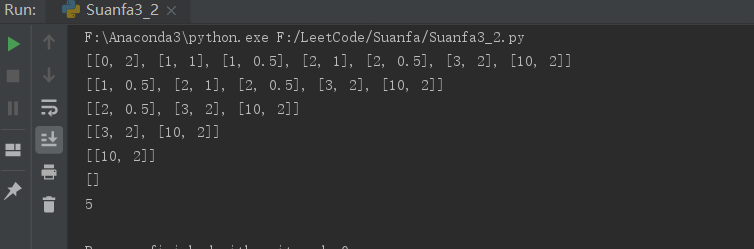
print(new\_P)

Jcount+=1

return Jcount

print(get\_min\_J([[0,2],[1,1],[1,0.5],[2,1],[2,0.5],[3,2],[10,2]],2))

**实验截图：**



* **某公司有个工厂和仓库。由于原材料等价格波动，工厂每个月的生产成本也会波动，令第个月产品的单位生产成本为（该月生产一个产品的成本为）。仓库储存产品的也有成本，假设每个月产品的单位储存成本为固定值1（存储一个产品一个月的成本为1）。令第个月需要供应给客户的产品数量为，仓库里的和生产的产品均可供应给客户。假设仓库的容量无限大，供应给客户剩余的产品可储存在仓库中。若已知个月中各月的单位生产成本、以及产品供应量，设计一算法决策每个月的产品生产数量，使得个月的总成本最低。例如：，，，则，即第1个月生产6个供应2个（代价22=4），储存4个供应给第2个月（代价(2+1)4=12），第3个月生产5个供应5个（代价35=15），使总成本4+12+15=31最小。**

**分析：**

考虑生产成本不一样，每个月都要上缴对应器件，所以从后往前倒着看。计算每个月那些器件从之前生产的最佳方案。

逆序遍历，遍历器件来自之前某个月使成本最低

**Code:**

from typing import List

'''

输入为两个List第一个存储了各个月的生产花费第二个存储了各个月需求

'''

def get\_min\_cost(c:List,y:List):

sumcost=0

for i in range(len(y)-1,-1,-1):

bestchoose=y[i]\*c[i]

for k in range(i):

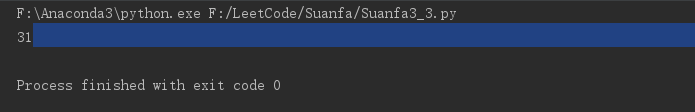
bestchoose=min(bestchoose,y[i]\*((i-k)+c[k]))

sumcost+=bestchoose

return sumcost

print(get\_min\_cost([2,5,3],[2,5,4]))

**实验截图：**



* **给定直线上 2n个点的序列P[1,2,… ,2n]，每个点 P[i]要么是白点要么是黑点，其中共有n个白点和 n个黑点，相邻两个点之间距离均为1，请设计一个算法将每个白点与一黑点相连，使得连线的总长度最小。例如，图中有4个白点和4个黑点，以图中方式相连，连线总长度为1+1+1+5=8。**

****

**题目分析：**

**想法：**

可以先将相距1的点配对再对剩下的相距2的点配对以此类推，直到点全部配对成功。

**思路分析：**

先将点集遍历，若点相邻1的有异色点则两个点置0。

再遍历点集若点相邻2的有异色点则两个点置0。

以此类推直到点集全0。

**Code:**

from typing import List

'''

传入为点集黑点为-1白点为1

'''

def minlen\_between\_black\_white(nodes:List):

unconnect=len(nodes)

trance=0

distance=1

while unconnect!=0:

for i in range(len(nodes)):

if nodes[i]!=0:

if i+distance<len(nodes)and nodes[i+distance]==-nodes[i]:

nodes[i]=0

nodes[i+distance]=0

unconnect-=2

trance+=distance

distance+=1

return trance

print(minlen\_between\_black\_white([-1,-1,1,-1,1,1,1,-1]))

**实验截图**：

