**贪婪搜索——贪婪算法**

1. 实验目的：

了解贪婪搜索，用具体算法问题理解贪婪搜索（贪婪算法）的原理，学习用python来编写贪婪算法，了解什么是贪婪算法

1. 实验内容：

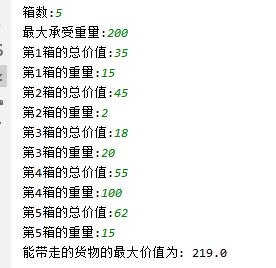
假设：

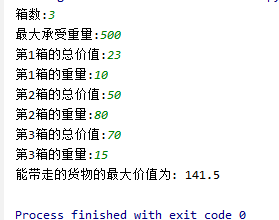
  现在有很多箱重量和价值都不相同的货物，每箱货物都可以分成任意散装组合运送。货车最多只能承受一定重量的货物。要求能运送的货物的最大总价值

可以对货物进行规定次序的排序，依次放入高价值货物后判断是否超出最大承重，超出则要依次减小重量知道找到最优的方案。

1. 实验代码
2. *#coding:utf-8***from** \_\_future\_\_ **import** division  
   input\_box = input(**'箱数:'**)  
   input\_h = input(**'最大承受重量:'**)  
   list\_a = []  
   list\_b = []*#备用的两个列表***for** i **in** range(1,int(input\_box)+1):  
    input\_Tvalue = input(**'第'**+str(i)+**'箱的总价值:'**)  
    input\_Theav = input(**'第'**+str(i)+**'箱的重量:'**)  
    avg = round(int(input\_Tvalue)/int(input\_Theav),1)*#每一箱单位重量的价值* list\_a.append(avg)*#添加到列表，用于比较* list\_b.append([int(input\_Theav),avg,0])*#中间的列表一个存放总重量，第二个存放单位价值，第三个存放是否该物品已被取走*list\_a.sort(reverse=**True**) *# 降序排序*sum =[0,0]*# 用于存放取走的总重量，第一个参数是取走的重量，第二个判断是否超出的备份信息*num =0  
   full = 0  
   **for** i **in** range(len(list\_a)):  
    **for** k **in** range(len(list\_b)):  
    **if** full == 0:*#是否超出最大承受量，未超出为0* **if** (list\_a[i] == list\_b[k][1]) **and** (list\_b[k][2]==0):  
    sum[1] = sum[0]*#备份* sum[0] = sum[0] + list\_b[k][0]*#取走的重量* v = list\_b[k][0]*#取走的重量* **if** sum[0] > int(input\_h):*#如果取走的重量超出最大承重量* full = 1t= list\_b[k][0]  
    **while True**:*#依次减去1单位的重量* z = sum[1] + t*#使用备份进行判断* **if** z <= int(input\_h):  
    **break** t = t-1  
    v=t*#等于最大承受量时，价值较大的一件物品应取走的数量* sum[0]=sum[1]*#从备份恢复* sum[0] = sum[0] + t*#取走数量* num = list\_a[i]\*v + num*#总价值* list\_b[k][2] = 1*#取走的标记*print (**'能带走的货物的最大价值为:'**,num)

1. 实验结果：





1. 实验总结

贪婪算法是指，在对问题求解时，总是做出在当前看来说最好的选择。也就是说，不从整体最优上加以考虑，而是在某种意义上的局部最优解。