**《算法设计与分析》实验报告**

# 实验三 贪心算法

|  |  |  |  |
| --- | --- | --- | --- |
| **报告书** | | | |
| 姓名 | 吴宇敖 | 指导教师 |  |
| 学号 | 176001752 | 日 期 | 2020.4.18 |
| 班级 | 17计算机软件一班 |  |  |
| **实验内容** | | | |
| **[要求先在OJ上提交通过]**  1)小K没事干，他要搬砖头，为了达到较好的减肥效果，教练规定的方式很特别：每一次，小K可以把两堆砖头合并到一起，消耗的体力等于两堆砖头的重量之和。经过 n-1次合并后， 就只剩下一堆了。小K在搬砖头时总共消耗的体力等于每次合并所耗体力之和。小K为了偷懒，希望耗费的体力最小。  例如有 3堆砖头，数目依次为 1、2、9 。可以先将 1 、 2 堆合并，新堆数目为3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为12 。所以总共耗费体力 =3+12=15。可以证明 15为最小的体力耗费值。[OJ]  2)给定一个非负整数数组，假定你的初始位置为数组第一个位置。数组中的每个元素代表你在那个位置能够跳跃的最大长度。你的目标是到达最后一个下标位置，并且使用最少的跳跃次数。[OJ] | | | |
| **实验目的** | | | |
| 1. 通过实验来了解贪心算法 2. 完成两个题目 | | | |
| **实验过程和步骤** | | | |
| **以下仅供参考，请在报告中删除**  **【实验题目】**  **小K没事干，他要搬砖头，为了达到较好的减肥效果，教练规定的方式很特别：每一次，小K可以把两堆砖头合并到一起，消耗的体力等于两堆砖头的重量之和。经过 n-1次合并后， 就只剩下一堆了。小K在搬砖头时总共消耗的体力等于每次合并所耗体力之和。小K为了偷懒，希望耗费的体力最小。例如有 3堆砖头，数目依次为 1、2、9 。可以先将 1 、 2 堆合并，新堆数目为3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为12 。所以总共耗费体力 =3+12=15。可以证明 15为最小的体力耗费值。**  **输入要求：**  **共两行。**  **第一行是一个整数 n(1≤n≤1000) ，表示砖头堆数。**  **第二行n个整数，每个整数表示每堆砖头的砖头块数。**  **输出要求：**  **一个整数，也就是最小的体力耗费值。**  **【程序代码及注释】**  #include<iostream>  #include<algorithm>  using namespace std;  int main()  {  int n,res=0,cost,k;  cin>>n;  int num[1000]={0};//初始化  for(int i=0;i<n;i++)  cin>>num[i];  sort(num,num+n);//排序  for(int i=0;i<n-1;i++)  {  cost=num[i]+num[i+1];//将最小的两堆砖合并  res+=cost;//将花费的体力加到总体力上  num[i+1]=cost;  for(int j=i+2;j<n;j++)//将剩下的砖堆进行从小到大的整理  {  if(num[j-1]>num[j])  {  k=num[j-1];  num[j-1]=num[j];  num[j]=k;  }  else  break;  }  }  cout<<res<<endl;  //system("pause");  return 0;  }  **【实验结果及遇到的问题】（附输入输出）**    遇到的问题：一开始我不太清楚如何才是最小体力消耗值，之后才明白了每次将最小的两个砖堆进行合并，这样就是最小的体力消耗。  **【算法分析】**  这次实验我先对于输入的砖堆数进行从小到大的排序，将最小的两个砖堆进行合并后进行记录消耗的体力。之后对于合并后的总砖堆进行从小到大的排序，之后不断重复之前的操作，直到砖堆全部合并完毕。  **【实验题目】**  给定一个非负整数数组，假定你的初始位置为数组第一个位置。数组中的每个元素代表你在那个位置能够跳跃的最大长度。你的目标是到达最后一个下标位置，并且使用最少的跳跃次数。  **【程序代码及注释】**  #include<iostream>  using namespace std;  int main()  {  int num[500]={0},x,k=0;  while(cin>>x){//进行输入  num[k]=x;  k++;  char ch=cin.get();  if(ch=='\n')//判断什么时候输入完成  break;  }  int step,n=0,y,i=0;  while(1)  {  step=i+num[i];//记录当前可以到达的距离  if(k==1)//只有一个数时，直接退出循环  break;  if(step>=k-1)//当距离超过最后一个时结束循环  {  n++;  break;  }  y=step;  for(int j=i;j<=step;j++)//判断可以到达的距离中，选择哪一个才是最远的距离  {  if(num[j]+j>y)  {  y=num[j]+j;  i=j;  }  }  n++;//记录走了几次  step=y;  }  cout<<n<<endl;  //system("pause");  return 0;  }  **【实验结果及遇到的问题】（附输入输出）**    遇到的问题：一开始使用了scanf来进行输出，但是输入数据后无法进行之后的代码，之后换成了cin进行输入，再加上判断的语句就可以了。  **【算法分析】**  这次实验从输入的数据的第一位开始，定义最长的路径step为第一个数值的值再加上它的下标0，之后在step中可以到达的全部值中进行循环，判断step数据中可以到达的最长距离是哪一个，距离的判断为那个点的值加上那个点的下标，记录那个点的下标，这个点就为下一步到达的点。当step到达最后的那个点，或者超过那个点时，就可以在加一步，然后退出循环。 | | | |