问题描述

　　Huffman树在编码中有着广泛的应用。在这里，我们只关心Huffman树的构造过程。  
　　给出一列数{*pi*}={*p*0, *p*1, …, *pn*-1}，用这列数构造Huffman树的过程如下：  
　　1. 找到{*pi*}中最小的两个数，设为*pa*和*pb*，将*pa*和*pb*从{*pi*}中删除掉，然后将它们的和加入到{*pi*}中。这个过程的费用记为*pa* +*pb*。  
　　2. 重复步骤1，直到{*pi*}中只剩下一个数。  
　　在上面的操作过程中，把所有的费用相加，就得到了构造Huffman树的总费用。  
　　本题任务：对于给定的一个数列，现在请你求出用该数列构造Huffman树的总费用。  
  
　　例如，对于数列{*pi*}={5, 3, 8, 2, 9}，Huffman树的构造过程如下：  
　　1. 找到{5, 3, 8, 2, 9}中最小的两个数，分别是2和3，从{*pi*}中删除它们并将和5加入，得到{5, 8, 9, 5}，费用为5。  
　　2. 找到{5, 8, 9, 5}中最小的两个数，分别是5和5，从{*pi*}中删除它们并将和10加入，得到{8, 9, 10}，费用为10。  
　　3. 找到{8, 9, 10}中最小的两个数，分别是8和9，从{*pi*}中删除它们并将和17加入，得到{10, 17}，费用为17。  
　　4. 找到{10, 17}中最小的两个数，分别是10和17，从{*pi*}中删除它们并将和27加入，得到{27}，费用为27。  
　　5. 现在，数列中只剩下一个数27，构造过程结束，总费用为5+10+17+27=59。

输入格式

　　输入的第一行包含一个正整数*n*（*n*<=100）。  
　　接下来是*n*个正整数，表示*p*0, *p*1, …, *pn*-1，每个数不超过1000。

输出格式

　　输出用这些数构造Huffman树的总费用。

样例输入

5  
5 3 8 2 9

样例输出

59

锦囊1

贪心算法。

锦囊2

按题目要求处理即可。

import java.io.\*;

import java.math.BigInteger;

public class Main {

public static void main(String[] args)throws IOException {

BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));

int n=Integer.parseInt(buf.readLine());

String str=buf.readLine();

String s[]=str.split(" ");

int sum=0;

int a[] =new int[n];

int num=0;

for(int i=0;i<n;i++)

a[i]=Integer.parseInt(s[i]);

java.util.Arrays.sort(a);

if(n>1){

sum=a[0]+a[1];

num=sum;

a[1]=num;

for(int j=2;j<n;j++){

if(j<n){

java.util.Arrays.sort(a);

num=a[j-1]+a[j];

a[j]=num;

java.util.Arrays.sort(a);

sum+=num;

}

}

}

else {

sum=a[0];

}

System.out.println(sum);

}

}

#include <stdio.h>

typedef struct

{

int a[100];

int len;

}huf;

int sum=0;

int del(huf\* in,int t)

{

int i,j;

for(i=0;i<in->len && in->a[i]!=t;i++);

for(;i<in->len-1;i++)

in->a[i]=in->a[i+1];

in->len--;

return 1;

}

int add(huf\* in,int t)

{

in->a[in->len]=t;

in->len++;

}

int find\_two\_mins(huf\* in)

{

int i,j,t;

int mina,minb;

for(i=0;i<in->len-1;i++)

for(j=i+1;j<in->len;j++)

if(in->a[i]>in->a[j])

{

t=in->a[i];

in->a[i]=in->a[j];

in->a[j]=t;

}

mina=in->a[0];

minb=in->a[1];

del(in,mina);

del(in,minb);

add(in,mina+minb);

return mina+minb;

}

int main()

{

huf in;

int i,j,n;

scanf("%d",&n);

in.len=n;

for(i=0;i<n;i++)

scanf("%d",&in.a[i]);

while(1)

{

if(in.len==2)

{

sum=sum+in.a[0]+in.a[1];

break;

}

sum+=find\_two\_mins(&in);

}

printf("%d",sum);

return 0;

}

#include<iostream>

#include<queue>

using namespace std;

priority\_queue<int, vector<int>, greater<int> > pq; //构造从小到大的优先队列

int main() {

int n;

cin >> n;

while (!pq.empty())

pq.pop();

int x, s;

for (int i = 0; i < n; i++) {

cin >> x;

pq.push(x);

}

int sum = 0;

while (pq.size() > 1) {

s = pq.top();

pq.pop();

s += pq.top();

pq.pop();

sum += s;

pq.push(s);

}

cout << sum << endl;

}