**题目描述 Description**

  在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

    每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过n-1次合并之后，就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

    因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为1，并且已知果子的种类数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

    例如有3种果子，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。所以多多总共耗费体力=3+12=15。可以证明15为最小的体力耗费值。

**输入描述 Input Description**

 输入包括两行，第一行是一个整数n(1<＝n<=10000)，表示果子的种类数。第二行包含n个整数，用空格分隔，第i个整数ai(1<＝ai<=20000)是第i种果子的数目。

**输出描述 Output Description**

输出包括一行，这一行只包含一个整数，也就是最小的体力耗费值。输入数据保证这个值小于231。

**样例输入 Sample Input**

3   
1 2 9

**样例输出 Sample Output**

15

**数据范围及提示 Data Size & Hint**

对于30％的数据，保证有n<=1000：   
对于50％的数据，保证有n<=5000；   
对于全部的数据，保证有n<=10000。

就是贪心，每次合并两堆最小的，注意，当最小的两堆合并完了之后的新堆可不一定是最小的了，所以，要用一种类似冒泡的方法把他插到它应在的位置

（如果每次合并完之后都排序的话会超时）

#include<stdio.h>

#include<iostream>

#include<algorithm>

using namespace std;

int a[10010];

int n;

int main()

{

cin>>n;

for(int i=1;i<=n;i++)

cin>>a[i];

sort(a+1,a+n+1);

int sum(0);

for(int i=2;i<=n;i++)

{

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

sum+=a[i];

int k=i;

while(a[k]>a[k+1]&&k+1<=n)

{

int t=a[k];

a[k]=a[k+1];

a[k+1]=t;

k++;

}

}

cout<<sum<<endl;

return 0;

}

用优先队列

#include<iostream>

#include<stdio.h>

#include<functional>

#include<queue>

using namespace std;

int main()

{

// freopen("input.txt","r",stdin);

int n,x;

while(cin>>n&&n)

{

priority\_queue<int,vector<int>,greater<int> >q;

for(int i=1;i<=n;i++)

{

cin>>x;

q.push(x);

}

int ans=0,tmp(0);

while(q.size()>1)

{

tmp=0;

tmp+=q.top();

q.pop();

tmp+=q.top();

q.pop();

ans+=tmp;

q.push(tmp);//注意这个tmp还要入队

}

cout<<ans<<"\n";

}

return 0;

}

priority\_queue<**int**,vector<**int**>,greater<**int**> >q;中的最后两个>>中间要有一个空格；

C++优先队列类似队列，但是在这个数据结构中的元素按照一定的断言排列有序。  
1.empty() 如果优先队列为空，则返回真  
2.pop() 删除第一个元素  
3.push() 加入一个元素  
4.size() 返回优先队列中拥有的元素的个数  
5.top() 返回优先队列中有最高优先级的元素  
  
优先级队列可以用向量(vector)或双向队列(deque)来实现(注意list container 不能用来实现queue，因为list 的迭代器不是任意存取iterator，而pop 中用到堆排序时是要求randomaccess iterator 的!)：  
priority\_queue<vector<int>, less<int>> pq1; // 使用递增less<int>函数对象排序  
priority\_queue<deque<int>, greater<int>> pq2; // 使用递减greater<int>函数对象排序  
其成员函数有“判空(empty)” 、“尺寸(Size)” 、“栈顶元素(top)” 、“压栈(push)” 、“弹栈(pop)”等。

Greater和less是很好用的，以后用在快排等里面也好

加个#include<functional>