#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

#include<ctime>

#include<windows.h>

#include<stdlib.h>

#include<ctime>

using namespace std;

#define inf 0x3f3f3f3f

const int maxn=1010000;

int a[maxn],n;

void bubbleSort()//冒泡排序

{

//注释掉的语句为冒泡排序的优化程序，可使得当数组已有序时，提前终止循环

for(int i=0;i<n-1;i++){//从下标为0开始冒泡遍历

int tmp;

//bool flag=1;

for(int j=0;j<n-i-1;j++){

if(a[j]>a[j+1]){//如果相邻的元素下标大的元素权值更小，则对它们进行交换

tmp=a[j];

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

a[j+1]=tmp;

//flag=0;

}

}

//if(flag) break;

}

}

void sellectSort()//选择排序

{

//注释掉的语句为选择排序的优化程序，可使得当数组已有序时，提前终止循环

for(int i=0;i<n-1;i++){//从下标为0的数开始遍历

int pos=i,tmp;

//bool flag=1;

for(int j=i+1;j<n;j++){//如果存在下标比i大且权值更小的数，则记下它的位置

if(a[j]<a[pos])

pos=j;

//if(a[j]<a[j-1]) flag=0;

}

//if(flag) break;

if(pos!=i){//把i位置和下标为pos的位置进行交换，使得i位置的数是当前数组最小的

tmp=a[pos];

a[pos]=a[i];

a[i]=tmp;

}

}

}

void insertSort()//插入排序

{

for(int i=1;i<n;i++){//从下标为1的数开始遍历

int tmp=a[i],j=i-1;

while(tmp<a[j]&&j>=0){//如果下标比i的数，且值比a[i]大 则将其右移

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

j--;

}

a[j+1]=tmp;//把a[i]插入到j+1这个位置上

}

}

//归并排序

int L[maxn],R[maxn];

void merge(int left,int mid,int right){

int n1=mid-left;

int n2=right-mid;

for(int i=0;i<n1;i++) L[i]=a[left+i];//把左边子序列存储到L数组

for(int i=0;i<n2;i++) R[i]=a[mid+i];//把右边子序列存储到R数组

L[n1]=R[n2]=inf;//边界数为无穷大

int i=0,j=0;

//L 和 R数组都递增的序列

for(int k=left;k<right;k++){

if(L[i]<R[j])//如果L当前元素比较小 则取L当前元素

a[k]=L[i++];

else //否则取R当前元素

a[k]=R[j++];

}

}

void mergeSort(int left,int right){

if(left+1<right){

int mid=(left+right)/2;//把当前数组分为两部分

mergeSort(left,mid);//对左半部分排序

mergeSort(mid,right);//对右半部分排序

merge(left,mid,right);//对排好序的两个子序列 整合一起，得到整个有序的序列

}

}

void quickSort(int l,int r)//快速排序

{

int m,l1,r1;

if(l<r){

//以a[l]为标杆 把数组划分为大于a[l]的右半边 和小于a[l]的左半边

l1=l;r1=r;

int v=a[l1];

while(l1<r1){

while(l1<r1&&a[r1]>=v) r1--;//当右边元素值大于a[l]时，不用交换位置

a[l1]=a[r1];

while(l1<r1&&a[l1]<=v) l1++;//当左边元素值小于a[l]时，不用交换位置

a[r1]=a[l1];

}

a[l1]=v;

m=l1;

//对当前的划分好的两个子序列分别递归进行相同的操作

quickSort(l,m-1);

quickSort(m+1,r);

}

}

int Random()//该函数用于随机生成0到10^8的数

{

const int mod=100;

return rand()%mod+(rand()%mod)\*100+(rand()%mod)\*10000+(rand()%mod)\*1000000;

}

void process()//该函数用于随机生成数组规模为n的数，并储存在数组a中

{

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

int x=Random();//

//int x=random(100000);

if(!x) x++;//

a[i]=x;

}

}

// c数组用于储存各数组规模的大小 t为数据的测试次数 取20次

int c[10]={10000,20000,30000,40000,50000,60000},t=20;

double b[30];//b数组用于储存每次所用的时间 ms为单位

void solve1()

{

//用QueryPerformanceFrequency函数来计时，需要用到头文件<windows.h>

LARGE\_INTEGER t1,t2,tc;// 用于记录排序所耗时间的频率

for(int i=0;i<5;i++){

n=c[i];

double sum=0;//统计总时间

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

sellectSort();//对数据进行选择排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//计算排序所耗时间

sum+=b[j];//统计总时间

}

//输出各测试的所耗时间

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

//输出平均所耗时间

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}//以下solve函数整体与solve1相同，不重复说明

void solve2()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

bubbleSort();//对数据进行冒泡排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

void solve3()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

mergeSort(0,n);//对数据进行合并排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

void solve4()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

insertSort();//对数据进行插入排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

void solve5()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;//

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

quickSort(0,n-1);//对数据进行快速排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

int main()

{

while(1){

//打印菜单

printf("请输入要测试的排序类型\n");

printf("0、退出 1、选择排序 2、冒泡排序 3、合并排序 4、插入排序 5、快速排序\n");

int k;

scanf("%d",&k);

if(!k) break;

//争对不同k进行不同排序

switch(k){

case 1:solve1();break;

case 2:solve2();break;

case 3:solve3();break;

case 4:solve4();break;

case 5:solve5();break;

}

printf("\n按任意键继续...\n");

}

}

//排序代码的测试

#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

using namespace std;

#define inf 0x3f3f3f3f

const int maxn=1010000;

int a[maxn],n;

void bubbleSort()

{

for(int i=0;i<n-1;i++){

int tmp;

for(int j=0;j<n-i-1;j++){

if(a[j]>a[j+1]){

tmp=a[j];

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

a[j+1]=tmp;

}

}

}

}

void sellectSort()

{

for(int i=0;i<n-1;i++){

int pos=i,tmp;

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

if(a[j]<a[pos])

pos=j;

}

if(pos!=i){

tmp=a[pos];

a[pos]=a[i];

a[i]=tmp;

}

}

}

void insertSort()

{

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

int tmp=a[i],j=i-1;

while(tmp<a[j]&&j>=0){

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

j--;

}

a[j+1]=tmp;

}

}

/\*

void merge(int left, int mid, int right) {

int n1 = mid - left;

int n2 = right - mid;

for (int i = 0; i < n1; i++) L[i] = A[left + i];

for (int i = 0; i < n2; i++) R[i] = A[mid + i];

L[n1] = R[n2] = maxnum;

int i = 0, j = 0;

for (int k = left; k < right; k++) {

cnt++;

if (L[i] <= R[j])

A[k] = L[i++];

else

A[k] = R[j++];

}

}\*/

int L[maxn],R[maxn];

void merge(int left,int mid,int right){

int n1=mid-left;

int n2=right-mid;

for(int i=0;i<n1;i++) L[i]=a[left+i];

for(int i=0;i<n2;i++) R[i]=a[mid+i];

L[n1]=R[n2]=inf;//

int i=0,j=0;

for(int k=left;k<right;k++){

if(L[i]<R[j])

a[k]=L[i++];

else

a[k]=R[j++];

}

}

void mergeSort(int left,int right){

if(left+1<right){

int mid=(left+right)/2;

mergeSort(left,mid);

mergeSort(mid,right);

merge(left,mid,right);

}

}

int partition(int l,int r)

{

int v=a[l];

while(l<r){

while(l<r&&a[r]>=v) r--;

a[l]=a[r];

while(l<r&&a[l]<=v) l++;

a[r]=a[l];

}

a[l]=v;

return l;

}

void quickSort(int l,int r)

{

int m;

if(l<r){

m=partition(l,r);

//print();

quickSort(l,m-1);

quickSort(m+1,r);

}

}

void print()

{

int c=0;

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

c++;

printf("%d ",a[i]);

if(c%10==0)

printf("\n");

}

printf("\n");

//printf("%d\n",a[n-1]);

}

void process()

{

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

int x=rand()%n+1;//

a[i]=x;

}

}

int main()

{

while(~scanf("%d",&n)){

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

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

process();

print();

//mergeSort(0,n);

quickSort(0,n-1);

//bubbleSort();

print();

}

}

//10亿个数筛最大的10个数 源代码

#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

#include<ctime>

#include<windows.h>

#include<stdlib.h>

#include<ctime>

using namespace std;

#define inf 0x3f3f3f3f

const int maxn=1010000;

int a[maxn],n;

// c数组用于储存各数组规模的大小 t为数据的测试次数 取20次

//int c[10]={10000,20000,30000,40000,50000,60000};

int c[10]={100000,200000,300000,400000,1000000,6000000};

int t=20;

double b[30];//b数组用于储存每次所用的时间 ms为单位

void print()

{

int c=0;

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

c++;

printf("%d ",a[i]);

if(c%10==0){

printf("\n");

//break;

}

}

printf("\n");

//printf("%d\n",a[n-1]);

}

void print2()//输出后10个数

{

int c=0;

for(int i=n;i>=1;i--){

c++;

printf("%d ",a[i]);

if(c%10==0){

printf("\n");

break;

}

}

printf("\n");

//printf("%d\n",a[n-1]);

}

void heapadjust(int i,int n)

{

int pos=i;

int l=2\*i,r=2\*i+1;

if(l<=n&&a[l]>a[pos])//

pos=l;

if(r<=n&&a[r]>a[pos])//

pos=r;

if(pos!=i){

swap(a[pos],a[i]);

heapadjust(pos,n);

}

}

void heapSort()

{

for(int i=n/2;i>=1;i--)

heapadjust(i,n);

int h=0;

for(int i=n;i>=1;i--){

//printf("cur:%d : ",h++);

//print();

if(h==10) break;//

swap(a[1],a[i]);

heapadjust(1,i-1);

}

}

void quickSort(int l,int r)//快速排序

{

int m,l1,r1;

if(l<r){

//以a[l]为标杆 把数组划分为大于a[l]的右半边 和小于a[l]的左半边

l1=l;r1=r;

int v=a[l1];

while(l1<r1){

while(l1<r1&&a[r1]>=v) r1--;//当右边元素值大于a[l]时，不用交换位置

a[l1]=a[r1];

while(l1<r1&&a[l1]<=v) l1++;//当左边元素值小于a[l]时，不用交换位置

a[r1]=a[l1];

}

a[l1]=v;

m=l1;

//对当前的划分好的两个子序列分别递归进行相同的操作

if((n-(m-1)<10))

quickSort(l,m-1);

quickSort(m+1,r);

}

}

int Random()//该函数用于随机生成0到10^8的数

{

const int mod=100;

return rand()%mod+(rand()%mod)\*100+(rand()%mod)\*10000+(rand()%mod)\*1000000;

}

void process()//该函数用于随机生成数组规模为n的数，并储存在数组a中

{

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

int x=Random();//

//int x=random(100000);

if(!x) x++;//

a[i]=x;

}

}

void solve5()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;//

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

quickSort(0,n-1);//对数据进行快速排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

void solve6()

{

LARGE\_INTEGER t1,t2,tc;

for(int i=0;i<5;i++){

n=c[i];

double sum=0;//

for(int j=0;j<t;j++){

process();

QueryPerformanceFrequency(&tc);

QueryPerformanceCounter(&t1);

heapSort();//对数据进行快速排序

QueryPerformanceCounter(&t2);

b[j]=(t2.QuadPart-t1.QuadPart)\*1000.0/tc.QuadPart;//

sum+=b[j];

}

printf("数据规模为%d时：\n",n);

printf("20组样本时间：\n");

for(int i=0;i<10;i++)

printf("%.2f ",b[i]);

printf("\n");

for(int i=10;i<t;i++)

printf("%.2f ",b[i]);

printf("\n");

printf("平均所花时间为: %.2f\n",sum/t);

printf("\n\n");

}

}

void test()

{

for(int i=0;i<5;i++){

n=c[i];

printf("用快排 n为%d时，最大的10个数：\n",n);//

//printf("用堆排 n为%d时，最大的10个数：\n",n);//

process();

//print();

quickSort(1,n);//

//heapSort();

//print();

print2();

}

}

int main()

{

//test();

while(1){

//打印菜单

printf("请输入要测试的排序类型\n");

printf("0、退出 1、堆排序 2、快速排序\n");

int k;

scanf("%d",&k);

if(!k) break;

//争对不同k进行不同排序

switch(k){

case 1:solve6();break;

case 2:solve5();break;

}

printf("\n按任意键继续...\n");

}

}