测试文档

递归快排实现代码:

void QuickSort\_Recursion(int \*a, int begin, int end)

{

if (begin < end)

{

int boundary = Partition(a, begin, end);

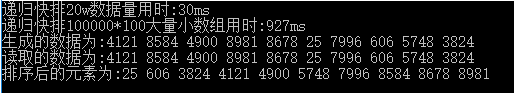
QuickSort\_Recursion(a, begin, boundary);

QuickSort\_Recursion(a, boundary + 1, end);

}

}

运行结果:



归并排序实现代码:

//归并数组

void MergeArray(int \*l,int l\_size,int \*r, int r\_size,int \*temp)

{

int i, j, k;

i = j = k = 0;

while (i<l\_size&&j<r\_size)

{

if (l[i] < r[j])

{

temp[k++] = l[i++];

}

else

{

temp[k++] = r[j++];

}

}

while (i < l\_size)

{

temp[k++] = l[i++];

}

while (j <r\_size)

{

temp[k++] = r[j++];

}

for (i = 0; i < (l\_size + r\_size); i++)

{

l[i] = temp[i];

}

}

//归并排序

void MergeSort(int \*a, int n, int \*temp)

{

if (n > 1)

{

int \*l = a;

int l\_size = n / 2;

int \*r = a + n / 2;

int r\_size = n - l\_size;

MergeSort(l, l\_size,temp);

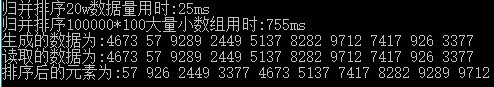
MergeSort(r, r\_size,temp);

MergeArray(l, l\_size, r, r\_size,temp);

}

}

测试结果:



插入排序实现代码:

void InsertSort(int \*a, int n)

{

int pre, cur;

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

{

pre = i - 1;

cur = a[i];

while (pre >= 0 && a[pre] >cur)

{

a[pre + 1] = a[pre];

pre--;

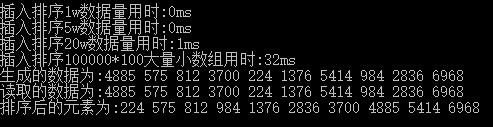
}

a[pre + 1] = cur;

}

}

测试结果:



非递归排序实现代码:

void QuickSort(int \*a, int begin, int end)

{

if (begin >= end)

return;

std::stack<int> s;

s.push(begin);

s.push(end);

while (!s.empty())

{

int en = s.top();

s.pop();

int be = s.top();

s.pop();

if (be < en)

{

int boundary = Partition(a, be, en);

s.push(be);

s.push(boundary);

s.push(boundary + 1);

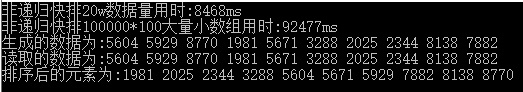
s.push(en);

}

}

}

测试结果：



计数排序实现代码：

void CountSort(int \*a, int n)

{

int i, j, k;

int cnt[MAX + 1] = { 0 };

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

cnt[a[i]]++;

k = 0;

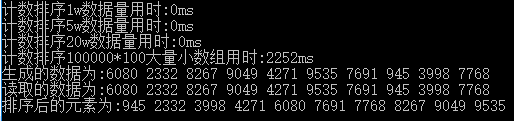
for (j = 0; j <= MAX; j++)

for (i = 1; i <= cnt[j]; i++)

a[k++] = j;

}

测试结果：



基数排序实现代码：

void RadixCountSort(int \*a, int n, int \*temp)

{

int exp;

int i, cnt[10] = { 0 };

for (exp = 1, i = 1; i <= 3; exp \*= 10, i++)

{

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

cnt[(a[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

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

for (i = n - 1; i >= 0; i--)

{

temp[cnt[(a[i] / exp) % 10] - 1] = a[i];

cnt[(a[i] / exp) % 10]--;

}

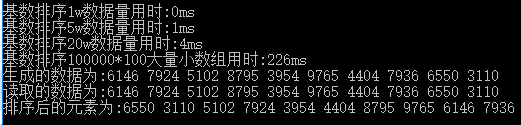
for (i = 0; i < n; i++)

a[i] = temp[i];

}

}

测试结果：



生成文本数据测试程序代码：

void Test(int \*f)

{

ofstream out("output.txt");

ifstream in("output.txt");

int i;

int b[10];

srand((unsigned int)time(NULL));

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

f[i] = rand() % 10000 + 1;

cout << "生成的数据为:";

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

cout << f[i] << " ";

cout << endl;

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

out << f[i] << " ";

out.close();

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

in >> b[i];

cout << "读取的数据为:";

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

cout << b[i] << " ";

cout << endl;

}

测试结果已显示在前面

//快速排序（枢轴存放）（辅助函数）

int Partition(int \*a, int begin, int end)

{

int i, j, x;

if (begin < end)

{

i = begin;

j = end - 1;

x = a[i];

while (i < j)

{

while (i <j && a[j] > x)

j--;

if (i < j)

a[i++] = a[j];

while (i < j && a[i] < x)

i++;

if (i < j)

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

}

a[i] = x;

}

return i;

}