1. **假设有 n 个值不同的元素存于数组 A( 1:n ) 中，另设一数组 C( 1:n )， 对每个元素 A[ i ]， C[ i ] 存放 A 中值小于A[ i ]的元素的个数，则 C[ i ]=0的元素即为最小元素； 然后根据 C 的值大小将 A 中的元素重新排列。 这种方法称为计数排序，试编程实现之。**

#include <iostream>

using namespace std;

void sort(int A[], int n)

{

int C[n]; // 辅助数组C，存放每个元素的小于自身的个数

for (auto &i : C)

i = 0;

for (int i = 0; i < n; i++) // 计算每个元素的小于自身的个数

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

if (A[j] < A[i])

C[i]++;

int B[n]; // 临时数组，存放排序后的元素

// C[i]表示原数组A中小于A[i]的元素个数，也即在排序后的数组B中，A[i]应该放置在B数组的第C[i]个位置

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

B[C[i]] = A[i];

for (int i = 0; i < n; i++) // 将B数组的元素复制回A数组

A[i] = B[i];

return;

}

int main()

{

// 测试数据

int A[] = {5, 2, 9, 1, 7, 6};

int n = sizeof(A) / sizeof(A[0]);

cout << "排序前的数组：";

for (auto i : A)

cout << i;

cout << endl;

sort(A, n);

cout << "排序后的数组：";

for (auto i : A)

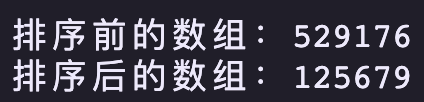
cout << i;

cout << endl;

return 0;

}

测试数据：



1. **假设有 n 个值不同的元素存于顺序结构中，要求不经排序选出自大至小的前k (k<<n) 个元素，问哪些方法可用，哪些方法比较次数最少？**

答案：冒泡排序比较次数n-1+n-2+...+n-j=nk-k(k+1)/2

简单选择排序同上

树形排序：附加2n-1个空间，比较次数(n-1)+(k-1)「log2n「

堆排序比较次数4n+(k-1)「log2n「

快速排序比较次数<2n

可以选择排序，堆排序。

对于堆排序而言，在建立一个含n个元素的堆时，进行关键字的比较的次数不超过4n，调整新建堆时的比较次数不超过2log2n次。

对于不经过排序而直接选择自大至小的前k个元素，可以采用以下几种方法：

可以遍历整个数组，找到最大的元素，放进一个新数组中。然后继续找数组中次大的，共k次，比较次数为O(nk)。

也可以用堆排序，需要进行n次元素的插入和调整操作，每次操作的复杂度为O(log k)，总的比较次数为O(n log k)，该方法比较次数较少。

1. **设有 n 个值不同的元素存于顺序结构中，问能否用比 2n-3 次少的比较次数找出该序列的最大值和最小值？若能，应如何实现，最坏情况如何？**

假设a1为最大值，a2为最小值。

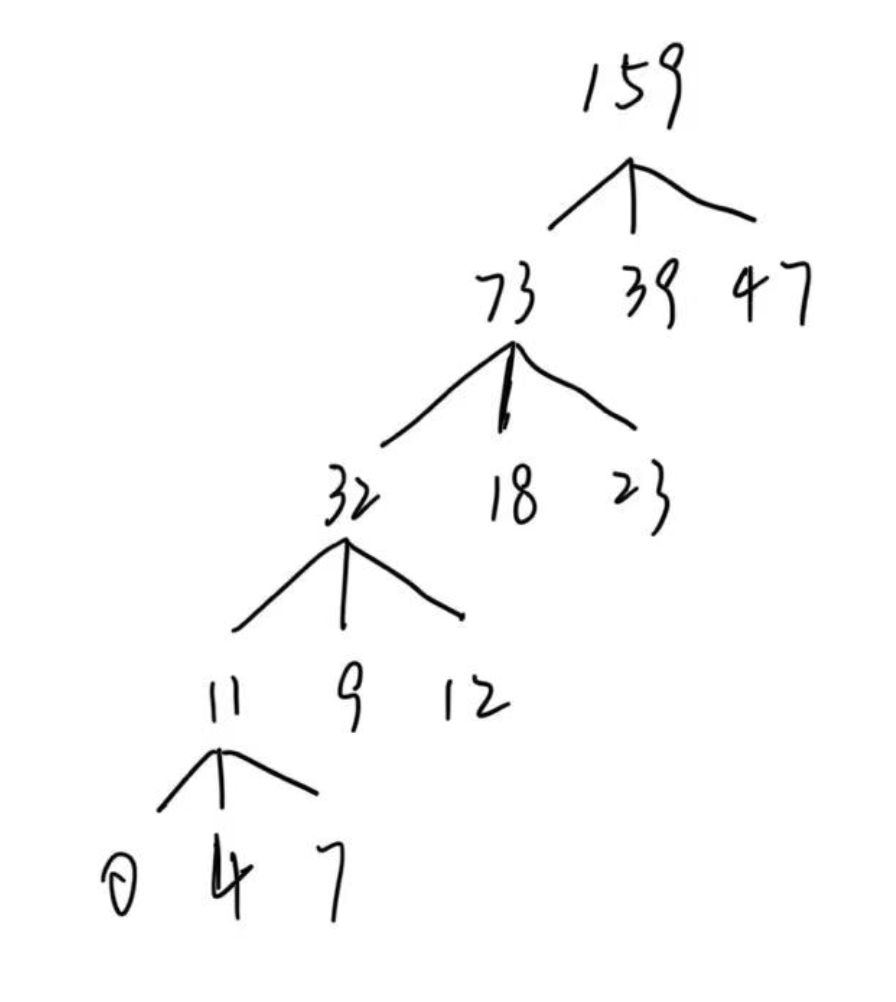
1. 比较a1和a2，若a1<a2，交换a1和a2；（比较一次）
2. 比较a3和a2，若a3<a2，交换a3和a2；反之再比较a3和a1，若a3>a1，交换a3和a1（比较一次或两次）
3. 比较a4和a2，若a4<a2，交换a4和a2；反之再比较a4和a1，若a4>a1，交换a4和a1（比较一次或两次）

…

n-1. 比较an和a2，若an<a2，交换an和a2；反之再比较an和a1，若an>a1，交换an和a1（比较一次或两次）

综上所述，最坏情况是(n-2)\*2+1=2n-3次。

1. **已知某文件经过置换-选择排序之后，得到长度分别为 47，9，39，18，4，12，23 和 7 的八个初始归并段。 试为 3-路平衡归并设计一个读写外存次数最少的归并方案，并求出总的读写外存次数。**



次数=[(4+7)\*4+(9+12)\*3+(18+23)\*2+(39+47)\*1]\*2=550。