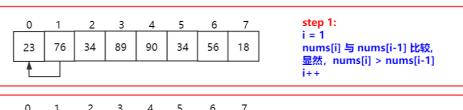
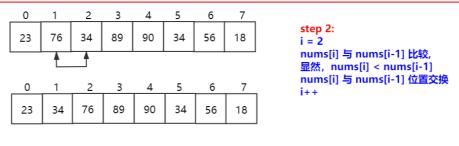
冒泡排序(交换排序)

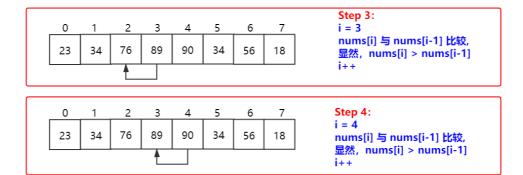
排序思想

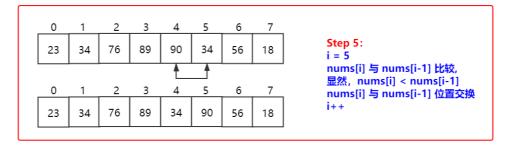
- 1. 比较相邻的元素。如果第一个比第二个大,就交换他们两个。
- 2. 对每一对相邻元素做同样的工作,从开始第一对到结尾的最后一对。在这一点,最后的元素应该会是最大的数。
- 3. 针对所有的元素重复以上的步骤,除了最后一个。
- 4. 持续每次对越来越少的元素重复上面的步骤,直到没有任何一对数字需要比较。

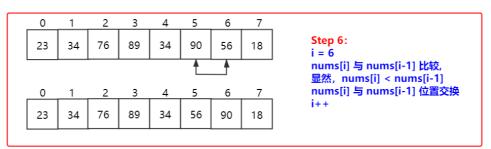
第1轮排序: [1,7]

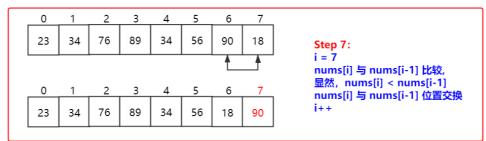












第2轮排序: [1,6] 第3轮排序: [1,5]

第4轮排序: [1,4]

第5轮排序: [1,3]

每一轮排序按上述排序进行,终点依次藏1, 每次排序都会有一个数的位置被定下来 第6轮排序: [1,2] 第7轮排序: [1,1]

冒泡排序运用时需要注意的

冒泡排序对于最坏的情况(严格递减/递增的数组),需要比较和移位的次数为 n(n-1)/2;

对于**最好的情况**(严格递增/递减的数组),需要比较的次数是 n-1 ,需要移位的次数是 0。

冒泡排序算法分析

冒泡排序的时间复杂度是 O(n^2) , 空间复杂度是 O(1) , 同时也是**稳定排序**, **全局有序** (每次排序将会有一个元素放在最终位置上) 。

代码实现

```
1 | class Solution{
    public:
2
        void bubbleSort(vector<int> &nums) {
3
             for(int i = 0; i < nums.size() - 1; i++) {</pre>
                 for(int j = 1; j < nums.size() - i; j++) {
 5
                     if(nums[j] < nums[j-1]) {
6
                         int tmp = nums[j-1];
 7
8
                         nums[j-1] = nums[j];
9
                         nums[j] = tmp;
10
                     }
11
                 }
12
            }
13
        }
14 };
```

加工后执行的结果

```
第0轮: 23, 76, 34, 89, 90, 34, 56, 18
第1轮: 23, 34, 76, 89, 34, 56, 18, 90
第2轮: 23, 34, 76, 34, 56, 18, 89, 90
第3轮: 23, 34, 34, 56, 18, 76, 89, 90
第4轮: 23, 34, 34, 18, 56, 76, 89, 90
第5轮: 23, 34, 18, 34, 56, 76, 89, 90
第6轮: 23, 18, 34, 34, 56, 76, 89, 90
第7轮: 18, 23, 34, 34, 56, 76, 89, 90
```

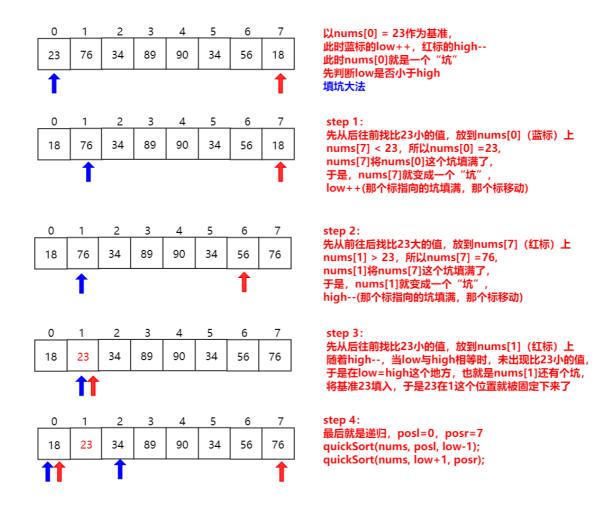
快速排序(交换排序)

排序思想

快速排序是一种划分交换排序。它采用了一种分治的策略,通常称其为分治法。

- 1. 先从数列中取出一个数作为基准数。
- 2. 分区过程,将比这个数大的数全放到它的右边,小于或等于它的数全放到它的左边。
- 3. 再对左右区间重复第二步,直到各区间只有一个数。

虽然快速排序称为分治法,但分治法这三个字显然无法很好的概括快速排序的全部步骤。因此我的对快速排序作了进一步的说明:挖坑填数+分治法



快速排序运用时需要注意的

快速排序对于**最坏的情况**(严格递减/递增的数组),需要比较和移位的次数为n(n-1)/2;对于**最好的情况**(严格递增/递减的数组),需要比较的次数是n-1,需要移位的次数是0。

快速排序算法分析

快速排序最坏的时间复杂度是 $O(n^2)$,平均时间复杂度为O(nlogn),空间复杂度最好是O(logn),最坏是O(n),

但是快速排序是一种**不稳定排序**,全局有序。

代码实现

```
class Solution{
 1
 2
    public:
 3
        int counter=0;
 4
        void quickSort(vector<int> &nums, int low, int high) {
 5
             if(low<high) {</pre>
                 int num = nums[low];
 6
 7
                 int posl = low;
                 int posr = high;
 8
 9
                 while(low < high) {</pre>
10
                     //从后往前找比num小的值
```

```
while(nums[high] >= num && low < high) {</pre>
11
12
                            high--;
13
                       }
                       if(low < high){</pre>
14
                            nums[low++] = nums[high];
15
16
                       while(nums[low] <= num && low < high) {</pre>
17
                            low++;
18
19
                       }
                       if(low < high){</pre>
20
                            nums[high--] = nums[low];
21
                       }
22
                  }
23
                  nums[low] = num;
24
                  quickSort(nums, posl, low-1);
25
                  quickSort(nums, low+1, posr);
26
27
              }
         }
28
29 \ \ \ ;
```

加工后执行的结果

```
第0轮: 23, 76, 34, 89, 90, 34, 56, 18
第1轮: 18, 23, 34, 89, 90, 34, 56, 76
第2轮: 18, 23, 34, 89, 90, 34, 56, 76
第3轮: 18, 23, 34, 76, 56, 34, 89, 90
第4轮: 18, 23, 34, 34, 56, 76, 89, 90
```

测试代码

冒泡排序测试代码:

```
#include <stdio.h>
2
    #include <vector>
 3
    using namespace std;
4
5
    class Solution{
6
    public:
7
        void bubbleSort(vector<int> &nums) {
8
                    // 查看
            printf("第0轮: ");
9
10
            for(int j = 0; j < nums.size(); j++) {
                printf("%d",nums[j]);
11
12
                if(j!=nums.size()-1) printf(",");
13
            }
            printf("\n");
14
15
            for(int i = 0; i < nums.size() - 1; i++) {
16
                for(int j = 1; j < nums.size() - i; j++) {
17
                    if(nums[j] < nums[j-1]) {
18
                         int tmp = nums[j-1];
19
                         nums[j-1] = nums[j];
20
                         nums[j] = tmp;
```

```
21
22
                }
                // 查看
23
                 printf("第%d轮: ",i+1);
24
25
                for(int j = 0; j < nums.size(); j++) {
                     printf("%d",nums[j]);
26
                     if(j!=nums.size()-1) printf(",");
27
28
                 }
                printf("\n");
29
30
            }
        }
31
    };
32
33
    int main() {
34
35
        vector<int> v;
36
        v.push_back(23);
37
        v.push_back(76);
38
        v.push_back(34);
39
        v.push_back(89);
40
        v.push_back(90);
        v.push_back(34);
41
42
        v.push_back(56);
43
        v.push_back(18);
        Solution solution;
44
45
        solution.bubbleSort(v);
46
        return 0;
47 }
```

快速排序测试代码

```
1 #include <stdio.h>
2
    #include <vector>
    using namespace std;
4
5
    class Solution{
6
    public:
        int counter=0;
7
        void quickSort(vector<int> &nums, int low, int high) {
8
            if(low<high) {</pre>
9
10
                 // 查看
                 printf("第%d轮: ",counter);
11
                 for(int j = 0; j < nums.size(); j++) {
12
                     printf("%d",nums[j]);
13
                     if(j!=nums.size()-1) printf(",");
14
15
                 }
                 printf("\n");
16
17
                 counter++;
                 int num = nums[low];
18
19
                 int posl = low;
                 int posr = high;
20
21
                 while(low < high) {</pre>
```

```
//从后往前找比num小的值
22
23
                     while(nums[high] >= num && low < high) {</pre>
                         high--;
24
25
                     }
                     if(low < high){</pre>
26
27
                         nums[low++] = nums[high];
28
                     }
29
                     while(nums[low] <= num && low < high) {</pre>
30
                         low++;
31
                     }
32
                     if(low < high){</pre>
                         nums[high--] = nums[low];
33
                     }
34
35
                 }
                 nums[low] = num;
36
                 quickSort(nums, posl, low-1);
37
                 quickSort(nums, low+1, posr);
38
             }
39
        }
40
41
    };
42
    int main() {
43
44
        vector<int> v;
        v.push_back(23);
45
46
        v.push_back(76);
        v.push_back(34);
47
        v.push_back(89);
48
49
        v.push_back(90);
50
        v.push_back(34);
        v.push_back(56);
51
52
        v.push_back(18);
53
        Solution solution;
        solution.quickSort(v,0,7);
54
55
        return 0;
56 }
```