

1、选择排序介绍

- 选择排序(Selection sort)是一种简单直观的排序算法。
- 它的基本思想是：首先在未排序的数列中找到最小(or最大)元素，然后将其存放到数列的起始位置；接着，再从剩余未排序的元素中继续寻找最小(or最大)元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

2、实现原理



选择排序原理示意图



JackieZheng

排序流程

- 第1趟: $i=0$ 。找出 $a[1...5]$ 中的最小值 $a[3]=10$, 然后将 $a[0]$ 和 $a[3]$ 互换。数列变化: 20,40,30,10,60,50 \rightarrow 10,40,30,20,60,50
- 第2趟: $i=1$ 。找出 $a[2...5]$ 中的最小值 $a[3]=20$, 然后将 $a[1]$ 和 $a[3]$ 互换。数列变化: 10,40,30,20,60,50 \rightarrow 10,20,30,40,60,50
- 第3趟: $i=2$ 。找出 $a[3...5]$ 中的最小值, 由于该最小值大于 $a[2]$, 该趟不做任何处理。
- 第4趟: $i=3$ 。找出 $a[4...5]$ 中的最小值, 由于该最小值大于 $a[3]$, 该趟不做任何处理。
- 第5趟: $i=4$ 。交换 $a[4]$ 和 $a[5]$ 的数据。数列变化: 10,20,30,40,60,50 \rightarrow 10,20,30,40,50,60

3、选择排序的时间复杂度和稳定性

- 选择排序时间复杂度
 - 选择排序的时间复杂度是 $O(N^2)$ 。
 - 假设被排序的数列中有 N 个数。遍历一趟的时间复杂度是 $O(N)$, 需要遍历多少次呢? $N-1$! 因此, 选择排序的时间复杂度是 $O(N^2)$ 。

- 选择排序稳定性
 - 选择排序是稳定的算法，它满足稳定算法的定义。
 - 算法稳定性 – 假设在数列中存在 $a[i]=a[j]$ ，若在排序之前， $a[i]$ 在 $a[j]$ 前面；并且排序之后， $a[i]$ 仍然在 $a[j]$ 前面。则这个排序算法是稳定的！

4、代码实现

```
using namespace std;

/*排序思想：
每一次从无序组的数据元素中选出最小（或最大）的一个元素，存放在无序组的起始位置，无序组元素
减少，有序组元素增加，直到全部待排序的数据元素排完。

*/

void selectionSort(int arr[], int n){
    for(int i = 0 ; i < n ; i ++){
        // 寻找[i, n)区间里的最小值
        int minIndex = i;
        for( int j = i + 1 ; j < n ; j ++ ){
            if( arr[j] < arr[minIndex] ){
                minIndex = j;
                swap( arr[i] , arr[minIndex] );
            }
        }
    }
}

int main(int argc, const char * argv[]) {

    int a[10] = {10,9,8,7,6,5,4,3,2,1};
    selectionSort(a,10);
    for( int i = 0 ; i < 10 ; i ++ )
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```

5、推荐阅读

[冒泡排序、插入排序、选择排序](#)

[推荐十大经典排序算法，再也不用担心面试了！](#)