排序

笔记本： 数据结构与算法
创建时间： 2021/4/12 20:47 **更新时间：** 2021/4/13 9:13

排序

快速排序

```
class Solution {
public:

    void QuickSort(vector<int>& nums, int l, int r){
        if(l >= r) return;

        int key = nums[l];
        int i = l, j = r;

        while(i < j){
            while(nums[j] >= key && i < j) j--;
            while(nums[i] <= key && i < j) i++;
            if(i < j)
                swap(nums[i], nums[j]);
        }

        swap(nums[l], nums[i]);
        QuickSort(nums, l, i-1);
        QuickSort(nums, i+1, r);
    }

    vector<int> sortArray(vector<int>& nums) {
        QuickSort(nums, 0, nums.size()-1);
        return nums;
    }
};
```

另一种写法：

```
class Solution {
public:

    void QuickSort(vector<int>& nums, int l, int r){
```

```
        if(l >= r) return;

        int key = rand()%(r-l+1)+l;
        swap(nums[key], nums[r]);

        int i = l-1;
        for(int j = l; j <= r-1; ++j){
            if(nums[j] <= nums[r]){
                swap(nums[++i], nums[j]);
            }
        }
        swap(nums[i+1], nums[r]);

        QuickSort(nums, l, i);
        QuickSort(nums, i+2, r);
    }

    vector<int> sortArray(vector<int>& nums) {

        QuickSort(nums, 0, nums.size()-1);
        return nums;
    }
};
```

归并排序

```
class Solution {
public:

    vector<int> data;
    void MergeSort(vector<int>& nums, int l, int r) {
        if(l >= r) return;

        int mid = (l+r)>>1;
        MergeSort(nums, l, mid);
        MergeSort(nums, mid+1, r);

        int i = l, j = mid+1;
        int cnt = 0;
        while(i <= mid && j <= r){
            if(nums[i] > nums[j]){
                data[cnt++] = nums[j++];
            }else{
                data[cnt++] = nums[i++];
            }
        }
        while(i <= mid){
            data[cnt++] = nums[i++];
        }
```

```
        while(j <= r){
            data[cnt++] = nums[j++];
        }

        for(int k = l; k <= r; ++k){
            nums[k] = data[k-l];
        }
    }

    vector<int> sortArray(vector<int>& nums) {
        data.resize(nums.size());
        MergeSort(nums, 0, nums.size()-1);
        return nums;
    }
};
```

堆排序

```
class Solution {
public:

    void HeapAdjust(vector<int>& nums, int s, int n){
        int tmp = nums[s];
        for(int i = 2*s+1; i <= n; i = i*2+1){
            if(i+1 <= n && nums[i+1] > nums[i]) i++;
            if(tmp > nums[i]) break;
            nums[s] = nums[i];
            s = i;
        }
        nums[s] = tmp;
    }

    void HeapSort(vector<int>& nums, int n){
        for(int i = n/2; i >= 0; --i){
            HeapAdjust(nums, i, n);
        }

        for(int i = n; i > 0; --i){
            swap(nums[i], nums[0]);
            HeapAdjust(nums, 0, i-1);
        }
    }

    vector<int> sortArray(vector<int>& nums) {

        HeapSort(nums, nums.size()-1);
        return nums;
    }
};
```