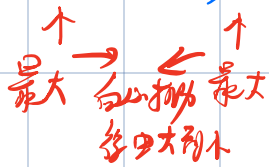
2是一道双指针的题

一个非递减顺序整数组 int nums;

返回每个数字的平方组成的新数组(也按非递减顺序排)

对于一个顺序组,其最大元素只能在两边{-5, 1, 2, 3}

从两头开始比较;



暴力解:平方后快排:

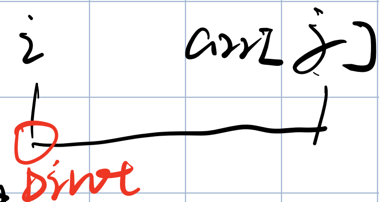
```
for(int i=0; i<nums.size(); i++)
{
    nums[i] = (nums[i])^2;
}
```

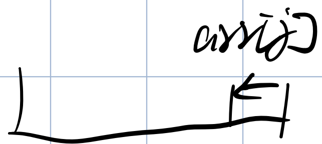
// 快速排序 (Quick Sort);

left: 0　　right: size-1　　非递归

```
void quickSort(vector<int>& arr, int left, int right)
{
    if (left >= right) return; // 排序终止条件

    int pivot = arr[left]; // 左一为基准
    int i = left, j = right;
    while (i<j)
    {
        // 从右向左找到第一个小于基准的元素
        while (i<j && arr[j] >= pivot)
        {
            j--;
        }
```





若 arr[j]大于基准则前置; (将右边大的放到右边)

```
        arr[i] = arr[j];
```

// 此时 arr[j] < pivot 往前放;

```
        while (i<j && arr[i] <= pivot)
        {
            i++;
        }
```

将左边小的

```
        arr[j] = arr[i];
```

// 此时 arr[i] > pivot;

```
        arr[i] = pivot;

        // 对基准左右部分递归
        quickSort(arr, left, i-1);
        quickSort(arr, i+1, right);

}
```

双指针法;

```
vector<int> result(nums.size());   // 声明大小;
int k = nums.size() - 1;
```

要考虑指针重合;

```
for (i=0, j=nums.size-1; i<=j;)
    if (nums[i] * nums[i] > nums[j] * nums[j])
    {                                   // 移动指针i
        result[k] = nums[i]^2;
        k--;   // 新数组的下标;
        i++;   // nums[i]^2大, 就走大的那个
    }
    else
    {                                   // 移动指针j;
        result[k] = nums[j]^2;
        k--;
        j--;
    }
```