快排模板

分治，返回第几大数

```
int partion(vector<int> &nums,int l,int r){
    int i = l;
    int j = r;
    while(i < j){
        while(i < j && nums[j] >= nums[l]) j--;
        while(i < j && nums[i] <= nums[l]) i++;
        swap(nums[i], nums[j]);
    }
    swap(nums[i], nums[l]);
    return i;
}
```

```
void quickSort(vector<int> &nums,int l,int r){
    int i = l;
    int j = r;
    while(i < j){
        while(i < j && nums[j] >= nums[l]) j--;
        while(i < j && nums[i] <= nums[l]) i++;
        swap(nums[i], nums[j]);
    }
    swap(nums[i], nums[l]);
    quickSort(nums,l,i);
    quickSort(nums,i + 1, r);
}
```

数组转树

```
TreeNode *vectorToTreeNode(vector<string> input){
    if (input.size() == 0) {
        return NULL;
    }

    TreeNode *root = (TreeNode *)malloc(sizeof(TreeNode));
    root->val = stringToInteger(input[0]);

    deque<TreeNode *> queue;
    int index = 1;
    int front = 0;
    queue.push_back(root);
```

```
    while(index < input.size()){
        TreeNode *node = queue[front];
        front += 1;
        string value = input[index];
        index += 1;
        if (value != "#") {
            int leftValue = stringToInteger(value);
            TreeNode *p = new TreeNode();
            p->val = leftValue;
            node->left = p;
            queue.push_back(node->left);
        } else {
            node->left = NULL;
        }

        if (index >= input.size()){
            break;
        }

        value = input[index];
        index += 1;
        if (value != "#") {
            int rightValue = stringToInteger(value);
            TreeNode *p = new TreeNode();
            p->val = rightValue;
            node->right = p;
            queue.push_back(node->right);
        } else {
            node->right = NULL;
        }
    }
    return root;
}
```

单调栈

- 数组逆序，从后往前
- 栈顶元素与当前元素比较，如果是小于，维护一个递增的栈；如果是大于；维护递减的栈
- 栈里可以存数字，或者索引，只要维护好关系即可

```
int[] nextGreaterElement(int[] nums) {
    int n = nums.length;
    // 存放答案的数组
    int[] res = new int[n];
    Stack<Integer> s = new Stack<>();
    // 倒着往栈里放
```

```
    for (int i = n - 1; i >= 0; i--) {
        // 判定个子高矮
        while (!s.isEmpty() && s.peek() <= nums[i]) {
            // 矮个起开，反正也被挡着了。。。
            s.pop();
        }
        // nums[i] 身后的更大元素
        res[i] = s.isEmpty() ? -1 : s.peek();
        s.push(nums[i]);
    }
    return res;
}
```