* **全局排序**，O(n\*lg(n))
* **局部排序**，只排序 TopK 个数，O(n\*k)
* **堆**，TopK 个数也不排序了，O(n\*lg(k))
* ~~分治法，每个分支“都要”递归，例如：快速排序，O(n\*lg(n))~~
* ~~减治法，“只要”递归一个分支，例如：二分查找 O(lg(n))，随机选择 O(n)~~
* **随机选择**+partition
* **比特位图计数**

随机选择并排序/基于快速排序：

1. #include<iostream>
2. **using** **namespace** std;
4. **int** findK(**int** nums[],**int** k,**int** start,**int** end) {
5. **int** low=start;
6. **int** high=end;
7. **int** temp=nums[low];//枢纽点
8. **while**(low<high) {
9. **while**(low<high&&nums[high]<=temp) {
10. high--;
11. }
12. nums[low]=nums[high];
13. **while**(low<high&&nums[low]>=temp) {
14. low++;
15. }
16. nums[high]=nums[low];
17. }
18. nums[high]=temp;
20. **if**(high==k-1) {
21. **return** temp;
22. } **else** **if**(high>k-1) {
23. **return** findK(nums,k,start,high-1);
24. } **else** {
25. **return** findK(nums,k,high+1,end);
26. }
28. }
30. **int** findKthLargest(**int** nums[],**int** k,**int** len) {
31. **return** findK(nums,k,0,len);
32. }
34. **int** main() {
35. **int** nums[]= {97,76,99,102,3,5,888};
36. cout<<findKthLargest(nums,6,7)<<endl;
37. **return** 0;
38. }