# 排序算法总结

# Sort algorithm summary:

## 快速排序quicksort

Quicksort is a divide-and-conquer algorithm, it is also a comparison sort.

Its central idea is to select a pivot and divide the whole array into two sub-arrays. The left sub-array’s elements should be less than this pivot, and the right sub-arrays elements should be greater than this pivot. For every array, we find the first smaller element from the right side, select the first greater element from the left side, exchange their position, and continue search until the current array become sorted.

Average Time complexity O(nlogn), in the worst case O(n^2)

Public void quicksort(int[] nums, int l, in r){

If(l>r) return;

//一般 l=0, r = nums.length()-1

int i = recur(nums, l, r);

//Tail call优化

//由于普通快速排序每轮选取「子数组最左元素」作为「基准数」，因此在输入数组

//完全倒序，partition() 的递归深度会达到N ，即 最差空间复杂度 为)O(N) 。

//每轮递归时，仅对 较短的子数组 执行哨兵划分 partition() ，就可将最差的递归深度//控制在O(logN)

if( i-l < r-i){

recur(nums, l, i-1);

l = i+1;

}

else{

recur(nums,i+1,r);

r=i-1;

}

}

Public int recur(int[] nums, int l, int r){

int i = l, j = r;

//确定此时的主元为l， the pivot here is ‘l’

while(i<j){

while(i<j && nums[l]>nums[j]) j--;

while(i<j && nums[l]< nums[i]) i++;

swap(nums[i], nums[j]);

}

swap(nums[l], nums[i]);

return i;

}

Public void swap(int[] nums, int i, int j) {

// 交换 nums[i] 和 nums[j]

int tmp = nums[i];

nums[i] = nums[j];

nums[j] = tmp;

}

## 归并排序Merge Sort