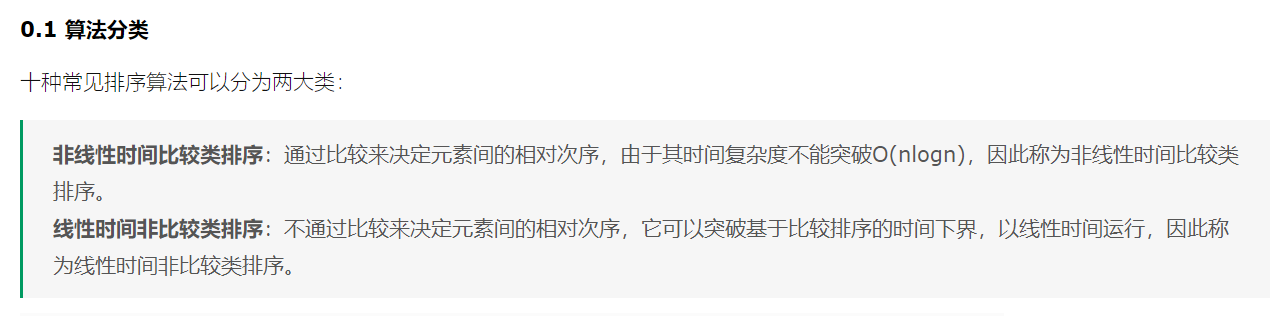
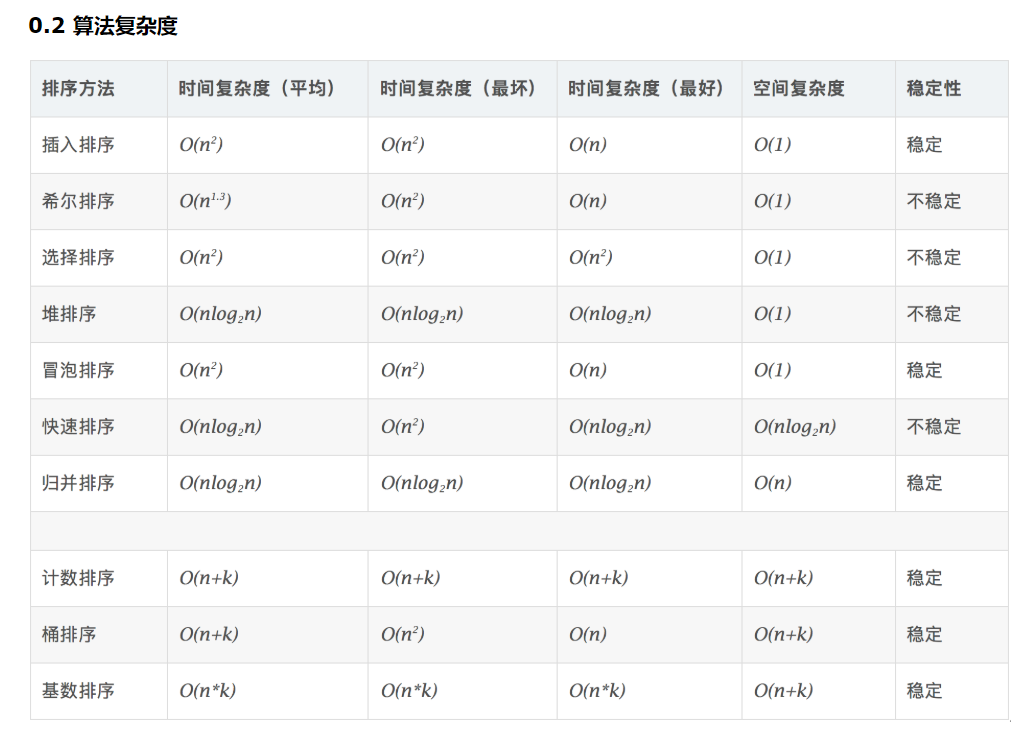
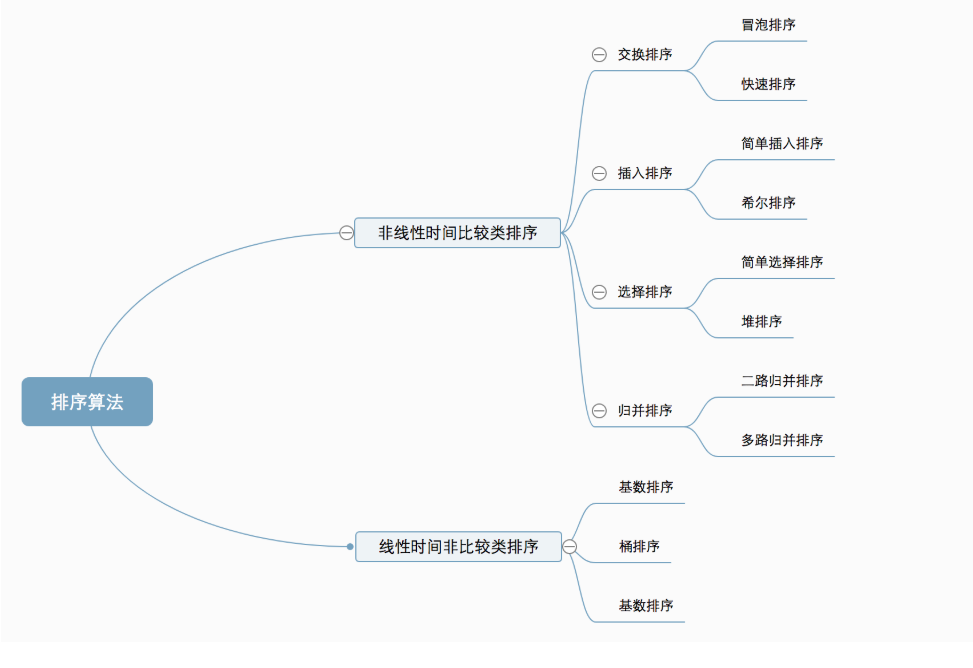
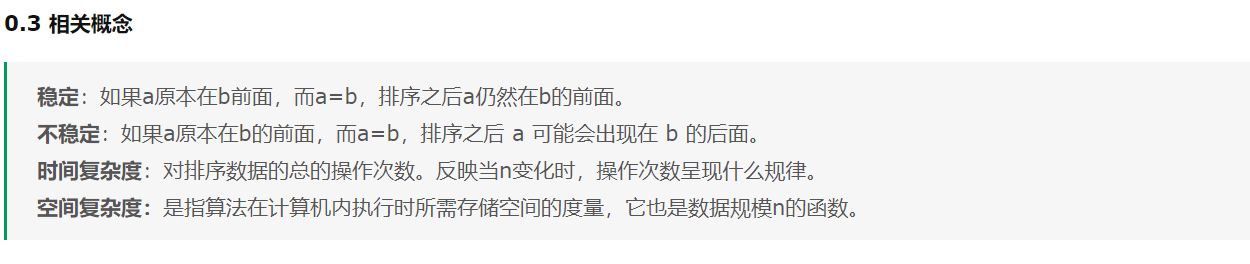
本文出自：

<https://www.cnblogs.com/onepixel/articles/7674659.html>









1.bubble sort

Function bubbleSort(arr){

Var len = arr.length;

For(var I = 0;I < len - 1;I ++)

For(var j = 0;j < len - 1;j ++)

If(arr[j] > arr[j + 1]){

//swap

Var temp = arr[j + 1];

Arr[j + 1] = arr[j];

Arr[j] = temp;

}

Return arr;

}

2.selection Sort

Function selectionSort(arr){

Var len = arr.length;

Var minIndex,temp;

For(var I = 0;I < len - 1;I ++){

minIndex = I;

For(var j = I + 1;j < len;j ++)

If(arr[j] < arr[minIndex]){

minIndex = j;

}

Temp = arr[i];

Arr[i] = arr[minIndex];

Arr[minIndex] = temp;

}

Return arr;

}

3.Insertion Sort

Function insertionSort(arr){

Var len = arr.length;

Var preIndex, current;

For(var I = 1;I < len;I ++){

preIndex = I - 1;

Current = arr[i];

While(preIndex >= 0 && arr[preIndex] > current){

Arr[preIndex + 1] = arr[preIndex];

preIndex -- ;

}

Arr[preIndex + 1] = current;

}

Return arr;

}

4.Shell Sort

//InsertionSort with different interval

Function shellSort(arr){

Var len = arr.length;

Var temp;

Var gap = 1;

While(gap < len/3){//this function has been delicately designed

Gap = gap \* 3 + 1;

}

For(gap;gap > 0;gap = Math.floor(gap / 3)){

For(var I = gap;I < len;I ++){

Temp = arr[i];

For(var I = gap;I < len;I ++){

Temp = arr[i];

For(var j = I - gap;j > 0 && arr[j] > temp;j -= gap)

Arr[j + gap] = temp;

}

}

Return arr;

}

5.Merge Sort

Function mergeSort(arr){

Var len = arr.length;

If(len < 2)

Return arr;

Var middle = Math.floor(len / 2);

Var left = arr.slice(0,middle);

Var right = arr.slice(middle);

Return merge(mergeSort(left),mergeSort(right));

}

Function merge(left, right){

Var result = [];

While(left.length > 0 && right.length > 0){

If(left[0] <= right[0]

Result.push(left.shift());

Else

Result.push(right.shift());

}

While(left.length)

Result.push(left.shift());

While(right.length)

Result.push(right.shift());

Return result;

}

6.Quick Sort

<https://blog.csdn.net/qq_36528114/article/details/78667034>

//this time we choose C++ code

Recursive:

Void QuickSort(int \* array,int left,int right){

If(left >= right)

Return;

Int index = partition(array,left,right);

QuickSort(array,left,index - 1);

QuickSort(array,index + 1,right);

}

//Now we have 3 methods to deal with partition()

Method1:

Int partition(int \* array,int left,int right){

Int & key = array[right];

While(left < right){

While(left < right && array[left] <= key)

++left;

While(left < right && array[right] <= key)

--right;

Swap(array[left],array[right]);

}

Swap(array[left],array[right]);

Return left;

}

Other Two methods may need you browse the website above.

Optimize:

Int GetMid(int \* array,int left,int right){

Int mid = left + ((right - left) >> 1);

If(array[left] <= array[right]){

If(array[mid] < array[left])

Return left;

Else if(array[mid] > array[right])

Return right;

Else

Return mid;

}

Else{

If(array[mid] < array[right])

Return right;

Else if(array[mid] > array[left])

Return left;

Else

Return mid;

}

}

Non-recursive:

Void QuickSortNotR(int \* array,int left,int right){

Stack<int> s;

S.push(left);

S.push(right);

While(!s.empty()){

Int right = s.top();

S.pop();

Int left = s.top();

S.pop();

Int index = partition(array,left,right);

If((index - 1) > left){

S.push(left);

S.push(index - 1);

}

If((index + 1) < right){

S.push(index + 1);

S.push(right);

}

}

}

7.Heap Sort

Build a heap

Insert:

Put it in the last and let it "float"

8.Counting Sort

9.Bucket Sort

10.Radix Sort

Sort from the low bit to high bit using buckets