

常用的数据结构

复杂数据结构



它们是学好算法的基石，只有把它们的性质牢牢掌握了，才能在接下来的课程里游刃有余。

学算法的好处

学习算法的过程其实是一个提高思维能力的过程。

如何应对算法面试

遇到新问题不要慌张，先想出最直观的解法；

直观解法大多数情况下不是最优的，却可以帮你打通思路；

理清解决问题的各个步骤后，根据问题核心来优化某些步骤即可。

后续课程安排

接下来的5节课里将——讲解面试中最常考的也最核心的算法介绍，本课时分享排序算法。

排序算法 / Sort



基本的排序算法

- 冒泡排序 / Bubble Sort
- 插入排序 / Insertion Sort

常考的排序算法

- 归并排序 / Merge Sort
- 快速排序 / Quick Sort
- 拓扑排序 / Topological Sort

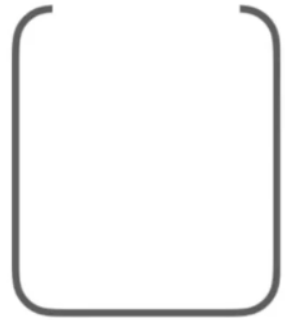
其他排序算法

- 堆排序 / Heap Sort
- 桶排序 / Bucket Sort

P1 P2 P5 P7 P8

冒泡排序的算法思想

每一轮，从杂乱无章的数组头部开始，每两个元素比较大小并进行交换；
直到这一轮当中最大或最小的元素被放置在数组的尾部；
然后，不断地重复这个过程，直到所有元素都排好位置。



```
void sort(int[] nums) {  
    boolean hasChange = true;  
  
    for (int i = 0; i < nums.length - 1 && hasChange; i++) {  
        hasChange = false;  
  
        for (int j = 0; j < nums.length - 1 - i; j++) {  
            if (nums[j] > nums[j + 1]) {  
                swap(nums, j, j + 1);  
                hasChange = true;  
            }  
        }  
    }  
}
```

空间复杂度：O(1)

假设数组的元素个数是n，整个排序的过程中，直接在给定的数组里进行元素的两两交换。

时间复杂度：O(n^2)

▸ 情景一：给定的数组按照顺序已经排好

只需要进行n - 1次的比较，两两交换次数为0，时间复杂度是O(n)，这是最好的情况。

▸ 情景二：给定的数组按照逆序排列

需要进行n(n - 1) / 2次比较，时间复杂度是O(n^2)，这是最坏的情况。

▸ 情景三：给定的数组杂乱无章

在这种情况下，平均时间复杂度是O(n^2)。

与冒泡排序对比

在冒泡排序中，经过每一轮的排序处理后，数组后端的数是排好序的；

在插入排序中，经过每一轮的排序处理后，数组前端的数都是排好序的。

插入排序的算法思想

不断地将尚未排好序的数插入到已经排好序的部分。

```
void sort(int[] nums) {  
    for (int i = 1, j, current; i < nums.length; i++) {  
        current = nums[i];  
  
        for (j = i - 1; j >= 0 && nums[j] > current; j--) {  
            nums[j + 1] = nums[j];  
        }  
  
        nums[j + 1] = current;  
    }  
}
```

空间复杂度：O(1)

假设数组的元素个数是 n ，整个排序的过程中，直接在给定的数组里进行元素的两两交换。

时间复杂度：O(n^2)

▸ 情景一：给定的数组按照顺序已经排好

只需要进行 $n - 1$ 次的比较，两两交换次数为0，时间复杂度是 $O(n)$ ，这是最好的情况。

▸ 情景二：给定的数组按照逆序排列

需要进行 $n(n - 1) / 2$ 次比较，时间复杂度是 $O(n^2)$ ，这是最坏的情况。

▸ 情景三：给定的数组杂乱无章

在这种情况下，平均时间复杂度是 $O(n^2)$ 。



分治的思想

归并排序的核心思想是分治，把一个复杂问题拆分成若干个子问题来求解。

归并排序的算法思想

把数组从中间划分成两个子数组；

一直递归地把子数组划分成更小的子数组，直到子数组里面只有一个元素；

*/** 归并排序的主体函数 */*

```
void sort(int[] A, int lo, int hi) {  
    if (lo >= hi) return;  
  
    int mid = lo + (hi - lo) / 2;  
  
    sort(A, lo, mid);  
    sort(A, mid + 1, hi);  
  
    merge(A, lo, mid, hi);  
}
```

```
void merge(int[] nums, int lo, int mid, int hi) {  
    int[] copy = nums.clone();  
  
    int k = lo, i = lo, j = mid + 1;  
  
    while (k <= hi) {  
        if (i > mid) {  
            nums[k++] = copy[j++];  
        } else if (j > hi) {  
            nums[k++] = copy[i++];  
        } else if (copy[j] < copy[i]) {  
            nums[k++] = copy[j++];  
        } else {  
            nums[k++] = copy[i++];  
        }  
    }  
}
```

时间复杂度：T(n)

归并算法是一个不断递归的过程，假设数组的元素个数是n。

时间复杂度是T(n)的函数： $T(n) = 2 * T(n/2) + O(n)$

怎么解这个公式呢？

对于规模为n的问题，一共要进行 $\log(n)$ 层的大小切分；

每一层的合并复杂度都是 $O(n)$ ；

所以整体的复杂度就是 $O(n \log n)$ 。

空间复杂度：O(n)

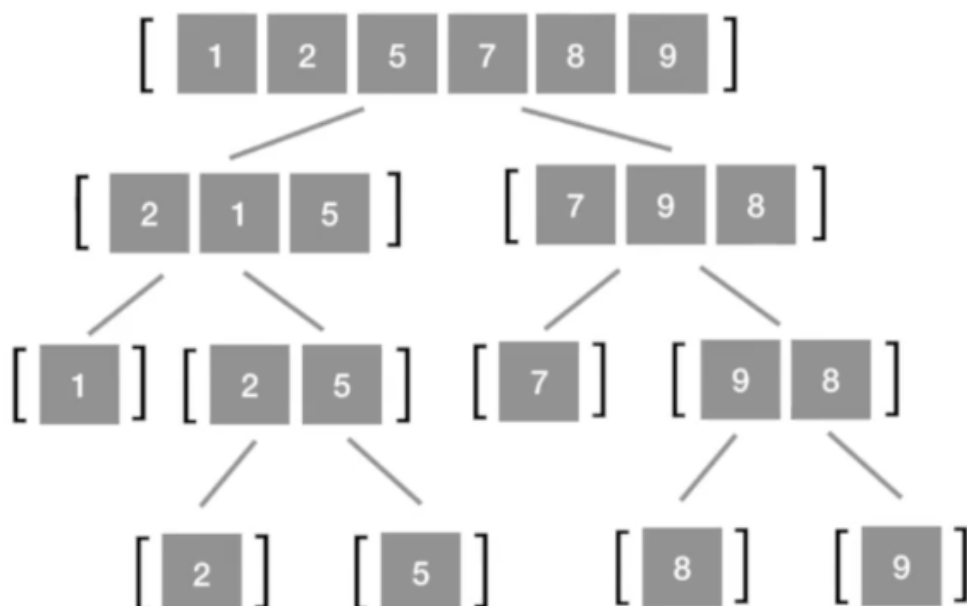
由于合并n个元素需要分配一个大小为n的额外数组，合并完成之后，这个数组的空间就会被释放。

快速排序 / Quick Sort**快速排序的算法思想**

快速排序也采用了分治的思想；

把原始的数组筛选成较小和较大的两个子数组，然后递归地排序两个子数组；

在分成较小和较大的两个子数组过程中，如何选定一个基准值尤为关键。



```
/** 快速排序的主体函数 */  
  
void sort(int[] nums, int lo, int hi) {  
    if (lo >= hi) return;  
  
    int p = partition(nums, lo, hi);  
  
    sort(nums, lo, p - 1);  
    sort(nums, p + 1, hi);  
}
```

```
int partition(int[] nums, int lo, int hi) {  
    swap(nums, randRange(lo, hi), hi);  
  
    int i, j;  
  
    for (i = lo, j = lo; j < hi; j++) {  
        if (nums[j] <= nums[hi]) {  
            swap(nums, i++, j);  
        }  
    }  
  
    swap(nums, i, j);  
  
    return i;  
}
```

快速排序算法分析 / Algorithm Analysis



最优情况下的时间复杂度

$$T(n) = 2 * T(n/2) + O(n)$$

O(n)是怎么得出来的呢？

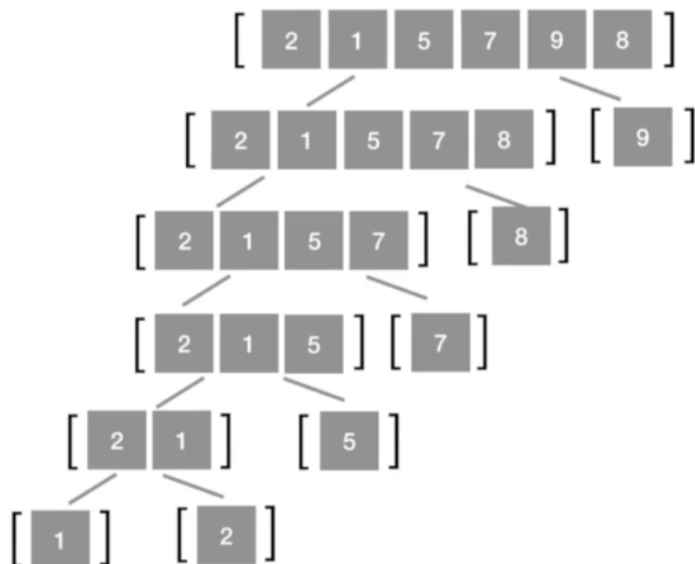
把规模大小为n的问题分解成n/2的两个子问题；

和基准值进行n-1次比较，n-1次比较的复杂度就是O(n)；

快速排序的复杂度也是O(nlogn)。

最复杂的情况

每次在选择基准值的时候；
都不幸选择了子数组里的最大或最小值；
其中一个子数组长度为1；
另一个长度只比父数组少1。



空间复杂度： $O(\log n)$

和归并排序不同，快速排序在每次递归的过程中；
只需要开辟 $O(1)$ 的存储空间来完成交换操作实现直接对数组的修改；
而递归次数为 $\log n$ ，所以它的整体空间复杂度完全取决于压堆栈的次数。

拓扑排序 / Topological Sort



应用场合

拓扑排序就是要将图论里的顶点按照相连的性质进行排序

前提

必须是有向图

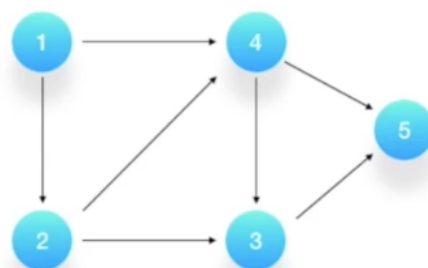
图里没有环

有一个学生想要修完5门课程的学分，这5门课程分别用1、2、3、4、5来表示，现在已知学习这些课程有如下的要求：

- 课程2和4依赖于课程1
- 课程3依赖于课程2和4
- 课程4依赖于课程1和2
- 课程5依赖于课程3和4

那么

这个学生应该按照怎样的顺序来学习这5门课程呢？



```
void sort() {
    Queue<Integer> q = new LinkedList();

    for (int v = 0; v < V; v++) {
        if (indegree[v] == 0) q.add(v);
    }

    while (!q.isEmpty()) {
        int v = q.poll();
        print(v);

        for (int u = 0; u < adj[v].length; u++) {
            if (--indegree[u] == 0) {
                q.add(u);
            }
        }
    }
}
```