

# 代码面试最常用的10大算法

发表于 2014-04-10 11:34 | 108934次阅读 | 来源 ProgramCreek | 460 条评论 | 作者 XWang

Java

面试


算法

排序

二叉树

归并排序

职业生涯

 摘要：面试也是一门学问，在面试之前做好充分的准备则是成功的必须条件，而程序员在代码面试时，常会遇到编写算法的相关问题，比如排序、二叉树遍历等等。

在程序员的职业生涯中，算法亦算是一门基础课程，尤其是在面试的时候，很多公司都会让程序员编写一些算法实例，例如快速排序、二叉树查找等等。

本文总结了程序员在代码面试中最常遇到的10大算法类型，想要真正了解这些算法的原理，还需程序员们花些功夫。

## 1.String/Array/Matrix

在Java中，String是一个包含char数组和其它字段、方法的类。如果没有IDE自动完成代码，下面这个方法大家应该记住：

```
toCharArray() //get char array of a String
Arrays.sort() //sort an array
Arrays.toString(char[] a) //convert to string
charAt(int x) //get a char at the specific index
length() //string length
length //array size
substring(int beginIndex)
substring(int beginIndex, int endIndex)
Integer.valueOf()//string to integer
String.valueOf()/integer to string
```

String/arrays很容易理解，但与它们有关的问题常常需要高级的算法去解决，例如动态编程、递归等。

下面列出一些需要高级算法才能解决的经典问题：

- Evaluate Reverse Polish Notation
- Longest Palindromic Substring
- 单词分割
- 字梯
- Median of Two Sorted Arrays
- 正则表达式匹配
- 合并间隔
- 插入间隔
- Two Sum
- 3Sum
- 4Sum
- 3Sum Closest
- String to Integer
- 合并排序数组
- Valid Parentheses
- 实现strStr()
- Set Matrix Zeroes




CSDN官方微信

扫描二维码,向CSDN吐槽

微信号：CSDNnews



程序员移动端订阅下载



每日资讯快速浏览

微博关注

 软件研发

CSDN研发频道

北京 朝阳区

+ 加关注

做一枚全栈工程师，到底值不值？：http://t.cn/RcYt01r

8月18日 16:27 转发(1) 评论

移动 H5 首屏秒开优化方案探讨：http://t.cn/RcYlgZ6

8月18日 15:30 转发 评论

## 相关热门文章

## 热门标签

- Hadoop

▪ Java

▪ Swift

▪ OpenStack

▪ ERP

▪ CRM

▪ Ubuntu
- AWS

▪ Android

▪ 智能硬件

▪ VPN

▪ IE10

▪ JavaScript

▪ NFC
- 移动游戏

▪ iOS

▪ Docker

▪ Spark

▪ Eclipse

▪ 数据库

▪ WAP

## 下载专辑

 资源优选



download.csdn.net

十五个Docker的优质资源

2017上半年  
软考各科目  
真题及答案  
解析下载

2017年上半年软考各科目考试真题及答案解析！



```
        }

    }

}
```

队列（Queue）

```
class Queue{
    Node first, last;

    public void enqueue(Node n){
        if(first == null){
            first = n;
            last = first;
        }else{
            last.next = n;
            last = n;
        }
    }

    public Node dequeue(){
        if(first == null){
            return null;
        }else{
            Node temp = new Node(first.val);
            first = first.next;
            return temp;
        }
    }

}
```

值得一提的是，**Java**标准库中已经包含一个叫做**Stack**的类，链表也可以作为一个队列使用（**add()**和**remove()**）。（链表实现队列接口）如果你在面试过程中，需要用到栈或队列解决问题时，你可以直接使用它们。

在实际中，需要用到链表的算法有：

- [插入两个数字](#)
- [重新排序列表](#)
- [链表周期](#)
- [Copy List with Random Pointer](#)
- [合并两个有序列表](#)
- [合并多个排序列表](#)
- [从排序列表中删除重复的](#)
- [分区列表](#)
- [LRU缓存](#)

3.树&堆

这里的树通常是指二叉树。

```
class TreeNode{
    int value;
    TreeNode left;
    TreeNode right;
}
```

下面是一些与二叉树有关的概念：

- 二叉树搜索：对于所有节点，顺序是：left children <= current node <= right children；
- 平衡vs.非平衡：它是一 棵空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树；
- 满二叉树：除最后一层无任何子节点外，每一层上的所有结点都有两个子结点；
- 完美二叉树（Perfect Binary Tree）：一个满二叉树，所有叶子都在同一个深度或同一级，并且每个父节点都有两个子节点；
- 完全二叉树：若设二叉树的深度为h，除第 h 层外，其它各层 (1~h-1) 的结点数都达到最大个数，第 h 层所有的结点都连续集中在最左边，这就是完全二叉树。

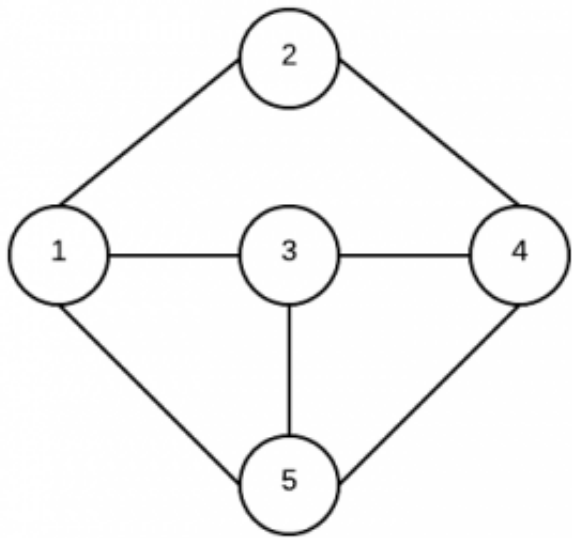
堆（Heap）是一个基于树的数据结构，也可以称为优先队列（ [PriorityQueue](#) ），在队列中，调度程序反复提取队列中第一个作业并运行，因而实际情况中某些时间较短的任务将等待很长时间才能结束，或者某些不短小，但具有重要性的作业，同样应当具有优先权。堆即为解决此类问题设计的一种数据结构。

下面列出一些基于二叉树和堆的算法：

- [二叉树前序遍历](#)
- [二叉树中序遍历](#)
- [二叉树后序遍历](#)
- [字梯](#)
- [验证二叉查找树](#)
- [把二叉树变平放到链表里](#)
- [二叉树路径和](#)
- [从前序和后序构建二叉树](#)
- [把有序数组转换为二叉查找树](#)
- [把有序列表转为二叉查找树](#)
- [最小深度二叉树](#)
- [二叉树最大路径和](#)
- [平衡二叉树](#)

## 4.Graph

与Graph相关的问题主要集中在深度优先搜索和宽度优先搜索。深度优先搜索非常简单，你可以从根节点开始循环整个邻居节点。下面是一个非常简单的宽度优先搜索例子，核心是用队列去存储节点。



第一步，定义一个GraphNode

```
class GraphNode{
    int val;
    GraphNode next;
    GraphNode[] neighbors;
    boolean visited;

    GraphNode(int x) {
        val = x;
    }
}
```

```
GraphNode(int x, GraphNode[] n) {
    val = x;
    neighbors = n;
}

public String toString() {
    return "value: " + this.val;
}
}
```

第二步，定义一个队列

```
class Queue{
    GraphNode first, last;

    public void enqueue(GraphNode n){
        if(first == null){
            first = n;
            last = first;
        }else{
            last.next = n;
            last = n;
        }
    }

    public GraphNode dequeue() {
        if(first == null){
            return null;
        }else{
            GraphNode temp = new GraphNode(first.val, first.neighbors);
            first = first.next;
            return temp;
        }
    }
}
```

第三步，使用队列进行宽度优先搜索

```
public class GraphTest {

    public static void main(String[] args) {
        GraphNode n1 = new GraphNode(1);
        GraphNode n2 = new GraphNode(2);
        GraphNode n3 = new GraphNode(3);
        GraphNode n4 = new GraphNode(4);
        GraphNode n5 = new GraphNode(5);

        n1.neighbors = new GraphNode[] {n2, n3, n5};
        n2.neighbors = new GraphNode[] {n1, n4};
        n3.neighbors = new GraphNode[] {n1, n4, n5};
        n4.neighbors = new GraphNode[] {n2, n3, n5};
        n5.neighbors = new GraphNode[] {n1, n3, n4};

        breathFirstSearch(n1, 5);
    }

    public static void breathFirstSearch(GraphNode root, int x){
```

```
        if(root.val == x)
            System.out.println("find in root");

        Queue queue = new Queue();
        root.visited = true;
        queue.enqueue(root);

        while(queue.first != null){
            GraphNode c = (GraphNode) queue.dequeue();
            for(GraphNode n: c.neighbors){

                if(!n.visited){
                    System.out.print(n + " ");
                    n.visited = true;
                    if(n.val == x)
                        System.out.println("Find "+n);
                    queue.enqueue(n);
                }
            }
        }
    }
}
```

输出结果：

```
value: 2 value: 3 value: 5 Find value: 5
value: 4
```

实际中，基于**Graph**需要经常用到的算法：

- [克隆Graph](#)

### 5.排序

不同排序算法的时间复杂度，大家可以到**wiki**上查看它们的基本思想。

Algorithm	Average Time	Worst Time	Space
Bubble sort	n^2	n^2	1
Selection sort	n^2	n^2	1
Insertion sort	n^2	n^2	
Quick sort	n log(n)	n^2	
Merge sort	n log(n)	n log(n)	depends

**BinSort**、**Radix Sort**和**CountSort**使用了不同的假设，所有，它们不是一般的排序方法。

下面是这些算法的具体实例，另外，你还可以阅读：[Java开发者在实际操作中是如何排序的](#)。

- [归并排序](#)
- [快速排序](#)
- [插入排序](#)

### 6.递归和迭代

下面通过一个例子来说明什么是递归。

问题：

这里有n个台阶，每次能爬1或2节，请问有多少种爬法？

步骤1：查找n和n-1之间的关系

为了获得n，这里有两种方法：一个是从第一节台阶到n-1或者从2到n-2。如果f(n)种爬法刚好是爬到n节，那么f(n)=f(n-1)+f(n-2)。

步骤2：确保开始条件是正确的

f(0) = 0;  
f(1) = 1;

```
public static int f(int n){
    if(n <= 2) return n;
    int x = f(n-1) + f(n-2);
    return x;
}
```

递归方法的时间复杂度指数为n，这里会有很多冗余计算。

```
f(5)
f(4) + f(3)
f(3) + f(2) + f(2) + f(1)
f(2) + f(1) + f(2) + f(2) + f(1)
```

该递归可以很简单地转换为迭代。

```
public static int f(int n) {

    if (n <= 2){
        return n;
    }

    int first = 1, second = 2;
    int third = 0;

    for (int i = 3; i <= n; i++) {
        third = first + second;
        first = second;
        second = third;
    }

    return third;
}
```

在这个例子中，迭代花费的时间要少些。关于迭代和递归，你可以去 [这里](#)看看。

## 7.动态规划

动态规划主要用来解决如下技术问题：

- 通过较小的子例来解决一个实例；
- 对于一个较小的实例，可能需要许多个解决方案；
- 把较小实例的解决方案存储在一个表中，一旦遇上，就很容易解决；
- 附加空间用来节省时间。

上面所列的爬台阶问题完全符合这四个属性，因此，可以使用动态规划来解决：

```
public static int[] A = new int[100];

public static int f3(int n) {
    if (n <= 2)
        A[n]= n;

    if(A[n] > 0)
        return A[n];

    else
        A[n] = f3(n-1) + f3(n-2); //store results so only calculate once!
    return A[n];
}
```

一些基于动态规划的算法：

- [编辑距离](#)
- [最长回文子串](#)
- [单词分割](#)
- [最大的子数组](#)

8.位操作

位操作符：

OR ( )	AND (&)	XOR (^)	Left Shift (<<)	Right Shift (>>)
1 0=1	1&0=0	1^0=1	0010<<2=1000	1100>>2=0011

从一个给定的数n中找位i（i从0开始，然后向右开始）

```
public static boolean getBit(int num, int i){
    int result = num & (1<<i);

    if(result == 0){
        return false;
    }else{
        return true;
    }
}
```

例如，获取10的第二位：

```
i=1, n=10
1<<1= 10
1010&10=10
10 is not 0, so return true;
```

典型的位算法：

- [Find Single Number](#)
- [Maximum Binary Gap](#)

9.概率

通常要解决概率相关问题，都需要很好地格式化问题，下面提供一个简单的例子：

有50个人在一个房间，那么有两个人是同一天生日的可能性有多大？（忽略闰年，即一年有365天）



算法：

```
public static double caculateProbability(int n){
    double x = 1;

    for(int i=0; i<n; i++){
        x *=  (365.0-i)/365.0;
    }

    double pro = Math.round((1-x) * 100);
    return pro/100;
}
```

结果：

calculateProbability(50) = 0.97

10.组合和排列

组合和排列的主要差别在于顺序是否重要。

例1：

1、2、3、4、5这5个数字，输出不同的顺序，其中4不可以排在第三位，3和5不能相邻，请问有多少种组合？

例2：

有5个香蕉、4个梨、3个苹果，假设每种水果都是一样的，请问有多少种不同的组合？

基于它们的一些常见算法

- 排列
- 排列2
- 排列顺序

来自：[ProgramCreek](#)

本文为CSDN编译整理，未经允许不得转载，如需转载请联系market#csdn.net(#换成@)

顶

468

踩

6

推荐阅读相关主题：[正则表达式](#)[动态规划](#)[快速排序](#)[数据结构](#)[解决方案](#)[排序算法](#)

相关文章

最新报道

▪ 主宰全球的10大算法

▪ [实战]查询无序列表中第K小元素

▪ IT旅途——程序员面试经验分享

▪ 简约而不简单！看Twitter上市后的入职面试题

▪ 组建团队：找人看重经验还是天资

▪ 高效的面试方式：结对编程

已有460条评论

还可以再输入500个字




有什么感想，你也来说说吧！

克己、欢迎您！

发表评论


- 最新评论

最热评论
- 

和尚要吐槽 2017-08-01 10:49

mark。。。怎么收藏？


▲▼

回复
- 

Michael0505 2017-06-20 17:04

MARK


▲▼

回复
- 

Michael0505 2017-06-20 16:50

MARK


▲▼

回复
- 

跌倒的小绵羊 2017-06-15 17:44

mark


▲▼

回复
- 

barney5 2017-05-30 15:38

good


▲▼

回复
- 

LZ\_MZ 2017-04-28 10:45

mark~


▲▼

回复
- 

妖瞳沐风2015 2017-04-17 11:10

mark


▲▼

回复
- 

nJcxs 2017-03-25 15:19

mark


▲▼

回复
- 

lighting8000 2017-03-24 01:03

mark ,thank you


▲▼

回复
- 

jpfs 2017-03-12 16:09

mark


▲▼

回复
- 

怎呼虹 2017-02-26 21:49

mark

▲▼

回复
- 

yangxi\_001 2016-12-29 15:06

mark

▲▼

回复



qq540156767 2016-12-26 13:48

mark

▲▼

回复



81bear 2016-12-02 09:32

mark

▲▼

回复



曹学亮 2016-10-30 11:04

很全面，学习了。

▲▼

回复



JY大魔王 2016-10-12 14:20

mark

▲▼

回复



笔尖下的日子 2016-10-07 17:42

mark

▲▼

回复



fengqiusuo 2016-09-06 10:23

mark

▲▼

回复



Hello\_Word 2016-08-26 10:03

mark

▲▼

回复



有流星飞过的日子 2016-08-08 12:00

mark

▲▼

回复



遛狗的猫咪 2016-08-06 19:30

mark

▲▼

回复



qwe132052 2016-07-09 12:04

mark

▲▼

回复



猿字号 2016-02-18 02:05

mark

▲▼

回复



invisiblescheme 2016-01-06 16:08

这怎么收藏啊

▲▼

回复




henu\_zyy 2016-01-06 08:55

谢谢分享


▲▼

回复



FutureTo2050 2015-11-23 23:23


mark



ZhanCF

2015-09-28 09:37

mark



wyhazq


2015-09-15 20:44

栈这里有点问题:

```
public void push(Node n){
    if(n != null){
        n.next = top;
        top = n; //永远都在替换栈顶元素
    }
}
```

应该改成:


```
public void push(Node n){
    if(n != null){
        if (top == null) {
            top = n;
        }
        else {
            top.next = n;
            top = n;
        }
    }
}
```



wyhazq

2015-09-15 21:04


是我理解错了，不好意思，楼主那样更好处理



gycg

2015-09-06 09:16


mark



wzhmmq

2015-07-22 20:20


很有用，感谢



wangtao198377

2015-04-14 14:51


mark



安翔

2015-04-03 10:19

mark



zhouziting

2015-01-16 13:51

代码面试最常用的10大算法 mark



木颜飞翔

2014-12-05 16:03

mark



macroscopicer 2014-11-01 20:49

Mark!

▲ ▼ 回复



lizhihaoweiei 2014-10-29 18:46

好东西。

▲ ▼ 回复



蜗牛菌 2014-09-24 23:26

mark

▲ ▼ 回复



CheerUpLin 2014-09-23 09:31

mark too

▲ ▼ 回复



想飞的鹰 2014-08-23 12:30

mark

▲ ▼ 回复



Anthony0859 2014-08-13 14:27

马

▲ ▼ 回复



lidechen2014 2014-08-07 07:24

mark!!!!!!

▲ ▼ 回复



进击的包子桑 2014-07-16 22:48

mark!!!!!!!!

▲ ▼ 回复



cjtn 2014-07-01 13:59

mark

▲ ▼ 回复



暮色已沉 2014-06-13 13:10

ding

▲ ▼ 回复



麻辣番茄丝 2014-05-30 10:48

mark

▲ ▼ 回复



从零开始\_ 2014-05-21 11:19

马克

▲ ▼ 回复



火云牌神 2014-05-20 20:00


ding!

▲ ▼ 回复




cheerfullchen 2014-05-18 18:46

好文

 fly\_gis 2014-05-08 10:54

Mark,值得收藏

 hongheqq 2014-05-07 22:04

mark

请您注意

- 自觉遵守：爱国、守法、自律、真实、文明的原则
- 尊重网上道德，遵守《全国人大常委会关于维护互联网安全的决定》及中华人民共和国其他各项有关法律法规
- 严禁发表危害国家安全，破坏民族团结、国家宗教政策和社会稳定，含侮辱、诽谤、教唆、淫秽等内容的作品
- 承担一切因您的行为而直接或间接导致的民事或刑事法律责任
- 您在CSDN新闻评论发表的作品，CSDN有权在网站内保留、转载、引用或者删除
- 参与本评论即表明您已经阅读并接受上述条款