Google面试题精讲

七月算法　曹鹏

2015年6月12日

提纲

- 关于面试
- 一些例题
  - 例1 被3和5整除的数的和
  - 例2 合法单词
  - 例3 和0交换的序列排序
  - 例4 放到结尾的序列排序
  - 例5 BFS及其推广
  - 例6 单词对
- 总结

关于面试

- 各个公司有没有自己的题库
  - 题库里的题目来源
    - 员工
    - 网络
- 笔试和面试
  - 笔试：没有交流
  - 面试
    - 展现思路
    - 给面试官好印象

例1 被3和5整除的数的和

- 例1 给定一个数n, 求不超过n的所有的能被3或者5整除的数的和。例如: n = 9, 答案3 + 6 + 5 + 9 = 23。
  - 分析：数学问题
    - 被3整除的数， 3,6,9,…[n / 3] * 3
    - 被5整除的数， 5,10,15, [n / 5] * 5
    - 重复的数同时是3和5的倍数 15, 30, …[n / 15] * 15
    - 注意点？ n的范围

例1 续

- 等差数列求和公式
  - x是首项，y是项数, d是公差
  - (x + x + d * (y – 1)) * y / 2, 注意y = 0也适用
  - 关键是项数!
    - 加 x = 3, d = 3, y = n / 3
    - 加 x = 5, d = 5, y = n / 5
    - 减 x = 15, d = 15, y = n / 15

例2 合法字符串

- 例2 字符串只有可能有A、B、C三个字母组成，如果任何紧邻的三个字母相同，就非法。求长度为n的合法字符串有多少个？比如：ABBBCA是非法，ACCBCCA是合法的。
  - 分析：动态规划的思路——真的要枚举么？
    - dp[i][0]：长度为i的、最后两位不同的合法串的个数
    - dp[i][1]: 长度为 i的、最后两位相同的合法串的个数
    - 递推: dp[i][0] = (dp[i-1][0] * 2 + dp[i-1][1] * 2)

      dp[i][1] = dp[i-1][0]

例2 续

- 初值
  - dp[1][0] = 3, dp[1][1] = 0
- 结果
  - dp[n][0] + dp[n][1]
- 空间优化
  - dp[i][0,1]只与dp[i-1][0,1]相关，可以省掉高维
- 时间复杂度
  - O(n)

例3 和0交换的排序

- 例3 一个整数组里包含0-(n-1)的排列 (0到(n-1)恰好只出现一次)，如果每次只允许把任意数和0交换，求排好顺序至少交换多少次。（PAT 1067）
  - 分析： 组合数学里的圈。
    - 例如0占了1的位置，1占了2的位置，2占了0的位置。
    - 一个排列，总可以划分为若干个不相交的圈
    - 上例我们交换0和1，再交换0和2，则排好顺序了

例3 续

- 一个长度为m的圈，如果包含0，则交换(m -1)次可以恢复所有的数到原位
- 如果一个长度为m的圈不包含0，则交换(m+1) 次可以恢复所有的数到原位
  - 例如1在2的位置，2在3的位置，3在1的位置
  - 我们先交换0和任意一个数，例如交换0和1
    - 则变成1在0的位置，0在2的位置，2在3的位置，3在1的位置。

例3 续2 代码

```
#include <cstdio>
#include <cstring>
#include <string>

using namespace std;

bool mark[100005];
int a[100005];

int give(int x) {
int r = 0;
bool have = false;
    for (;!mark[x];++r) {
        if (x == 0) {
            have = true;
        }
        mark[x] = true;
        x = a[x];
    }
    return have?(r - 1):((r <= 1)?0:(r + 1));
}

int main() {
int n;
    scanf("%d",&n);
    for (int i = 0; i < n; ++i) {
        scanf("%d",a + i);
    }
    int answer = 0;
    for (int i = 0; i < n; ++i) {
        answer += give(a[i]);
    }
    printf("%d\n",answer);
    return 0;
}
```

例4 放到结尾的排序

- 例4 给定一个1-n的排列，每次只能把一个数放到序列末尾，至少几次能排好顺序？(O(n)时间内解决的问题（下））
  - 为什么要移动1?
    - 其他数都排好了，1自然就排好顺序了
  - 如果某一步把x移动到末尾，则我们必须把(x + 1), (x + 2).. n都移动到末尾——否则无法排序
  - 如何让x尽可能大?
  - 从1-(x-1)是必须是按照顺序出现的
  - 从开头扫描，检查x最大是多少。

例4 续

- 代码非常简单

```
int solution(vector<int> &a) {
int n = a.size(), want = 1;
    for (int i = 0; i < n; ++i) {
        if (a[i] == want) {
            ++want;
        }
    }
    // want .. n must be moved
    return n - want + 1;
}
```

例5 BFS及其推广

- 例5 给定一个矩阵X表示起点, Y表示终点,#表示墙, 从每个位置只能上下左右四个方向走, 不能走出矩阵,

(1) 问至少多少步？

(2) 如果允许最多拆3堵墙, 至少多少步？

```
.......
.......
..#.X..
....#..
..##...
..Y....
```

例5 续

- 分析
  - （1） 很简单，就是直接BFS
  - （2） 枚举拆墙？
    - $C(n^2,3)$　$O(n^6)$
    - BFS　$O(n^2)$
  - 重新构图？
    - 4层的有向图（0，1，2，3）
    - 每一层（相同）
      - 每个点（包括墙）到它的非墙邻居有边
      - 注意：墙有出边，无入边

例5 续2

- 第i层到第(i + 1)层 (i=0,1,2)
  - 第i层的**任意**位置的邻居如果是墙，则有一条从第i层该位置到第(i + 1)层对应墙位置的边。
  - 从第i层相当于“穿墙”到了第(i+1)层，虽然第(i+1)层该位置仍是墙，但是该位置可以出到别的位置。
  - 在这个“立体”图上做BFS
- 节点数$O(n^2)$，边数$O(n^2)$，时间复杂度$O(n^2)$

例6 单词对

- 例6 给定一个字典，找到两个单词，它们不包含相同的字母，且乘积尽可能大，允许预处理字典。
  - 分析：
    - 开放问题——没有标准答案
    - 需要和面试官交流、探讨
    - 打开思路
  - 方法1
    - 枚举， 至少$O(n^2)$，如何判断不包含相同字母？

例6 续

- 细节：如何判断包含相同字母？
  - 每个单词出现的字母适用bitmap？
  - 每个单词“签名”，表示出现哪些字母2^{26}
  - 两个单词签名是x和y
    - 如果x & y（按位与）非0，则包含相同的字母
- n多大可以接受

方法2 预处理字典？如果空间足够大!

- 如何表示每个单词？ 仍然签名x =$[0..2^{26}-1]$

例6 续2

- 如何表示字母集合？
  - 给定状态s在$[0..2^{26}-1]$中，表示**可以**出现哪些字母，我们想找到满足条件的最长单词，dp[s]
    - 可以的含义？可以出现，可以不出现。即s中为1的位(bit)表示的字母可以在这个单词中出现，也可以不在这个单词中出现，但s中为0的那些位一定不允许在单词中出现的最长单词的长度
    - 换言之，单词中出现的字母是状态s为1的位表示的字母集合的**子集**

例6 续3

- 如何计算dp[s]?
  - 初值
    - 全0
    - 对每个单词的签名x, 计算dp[x] = max(dp[x], length(word))
  - 更新?

for s = 1 to $2^{26} - 1$

对每一个比s少一个二进制1(bit)的状态s'

dp[s] = max(dp[s], dp[s'])

例6 续4

- 对dp[s]的更新的理解
  - s’ < s
  - s’已经计算好了
    - 例如我们想计算状态11010，
      - 状态01010, 10010, 11000已经计算好了
    - 子集的最优解
      - 要么是上面某个状态的子集的最优解（已经计算好了）
      - 要么是这个集合本身 （预处理过）

例6 续5

- 最终答案
  - 对每个单词签名x， 我们可选的字母集合是(~x)的子集——因为不能优相同的位
  - 所以求 max(length(x) * dp[(~x) & ((1 << 26) – 1))] 即可
- 时间复杂度：
  - 每个单词签名 O(length * n)
  - 计算dp[s] $O(2^{26} * 26)$
  - 最终求解 O(n)

例6 续6

- 空间复杂度 $O(2^{26})$
- 难点
  - 理解位运算
  - 理解dp
    - 当前集合的子集包括
      - 比当前集合少一个元素的全部子集
      - 当前集合本身
  - 我们可以把dp的整数换为实际单词，可以得到解

总结

- 一定要和面试官交流
  - 不要把面试当成笔试
  - 给面试官积极的情绪
  - 没有标准答案——开放问题
  - 多提假设
    - 函数头部要自己写出
- 无固定套路
- 多总结、思考、归纳
- Goode luck