2009

ADVENTopLab

ADVENTop

[USACO 第二章通关总结]

HTTP://hi.baidu.com/adventop

USACO 第二章通关总结

By ADVENTop

原来这一章我用 Pascal 一直做到倒数第 4 题,,从某种意义上讲,因为第一章的基础打的牢固,所以第二章并不比第一章吃力,相反倒有几分轻松,比起第一章,第二章最大的不同是加入了初等图论和 DP 难度的加深.并且巩固第一章学过的算法,每道题目的难度恰到好处.新的算法不会学起来吃力,老的算法也能更加理解,此外本章对数学分析能力的要求加强,意味着编程必定要有一定的数学功底.主要学会的新算法有:**Floodfill,DP(初级),最短路,位运算的强化.**

第二章题目总体类型分布:

Chapter 1	DONE	2009.08.14	Getting started
Section 2.1	DONE	2009.08.10	TEXT Graph Theory
	DONE	2009.08.10	TEXT Flood Fill Algorithms
	DONE	2009.08.10	PROB The Castle DFS(Floodfill)
	DONE	2009.08.10	PROB Ordered Fractions 枚举
	DONE	2009.08.11	PROB Sorting A Three-Valued Sequence 贪心
	DONE	2009.08.11	PROB Healthy Holsteins DFS
	DONE	2009.08.11	PROB Hamming Codes 枚举
Section 2.2	DONE	2009.08.12	TEXT Data Structures
	DONE	2009.08.12	TEXT Dynamic Programming
	DONE	2009.08.12	PROB Preface Numbering 枚举
	DONE	2009.08.12	PROB Subset Sums DP
	DONE	2009.08.12	PROB Runaround Numbers 模拟
	DONE	2009.08.12	PROB Party Lamps DFS
Section 2.3	DONE	2009.08.13	PROB The Longest Prefix DP
	DONE	2009.08.14	PROB Cow Pedigrees DP
	DONE	2009.08.14	PROB Zero Sum DFS
	DONE	2009.08.15	PROB Money Systems DP
	DONE	2009.08.15	PROB Controlling Companies DFS
Section 2.4	DONE	2009.08.15	TEXT Shortest Paths
	DONE	2009.08.15	PROB The Tamworth Two 模拟
	DONE	2009.08.15	PROB Overfencing BFS(Floodfill)
	DONE	2009.08.16	PROB Cow Tours floyd
	DONE	2009.08.16	PROB Bessie Come Home floyd

	DONE	2009.08.17	PROB Fractions to Decimals 数学

◎第一节:

第一节与上一章是一个衔接,考查的内容新增了 Floodfill,其他的题目则更多的是一个巩固的过程.主要涉及了:DFS,贪心,位运算.

2.1.1:这是一个新的算法,Floodfill,满水填充法,要用到搜索实现,而本题是使用回溯搜索技术,有很多的细节处理,比如要判断 4 个方向,我就使用减法从大到小挨个减去,这样可以覆盖所有的情况,最后查找的时候,也是从左下角,而且要先判断北极,再判断东,因为所谓最西,在同一经度下北比东要靠西.主要是细枝末节,从这一点,第二章比第一章更进一步.

2.1.2:我使用枚举来做顺序的分数的,使用动态数据结构来完善,避免最后排序,事实上有一种分治策略,程序不过 10 行,通过这道题目,我才了解了分治是什么.也学会了分治精妙的思想.

2.1.3:这是原本文章中的一道题目,主要是贪心分析,我的做法笨拙,就是模拟出来,其实大可不必这样做,而是直接分析把结果计算出来,预处理 O(n),计算 O(1)!可以说很精妙.

2.1.4:这是一道典型的回溯搜索技术题,每种状态有选与不选 2 种,主要是分析,编程不是难点.

2.1.5:这是枚举题目,考点不在枚举,在于判断,于是位运算的优点再一次显现出来,使我对位运算更加理解.其实需要掌握的就是运算手段.

◎第二节:

第二节有点第二章的意思了,不论是什么算法,都不太容易编写,编程与算法的要求增高,从这一节开始,DP 正式被引入,之前第一章的数字三角形其实可以用 BFS 去做的,也就是说,第二章 DP 的要求加大,主要涉及:DFS,枚举,模拟,DP

2.2.1:我用到了筛法枚举,对于每个数字进行隔离判断并统计,不过还可以数学分析,其实不难,就是理解了法则,然后模拟.

2.2.2:这是一道 DP,属于背包的变种,**F[i][j]=f[i-1][j]+f[i-1][j-num[i]];**方程我想了好长时间,也算是我的第一个自己想出来的 DP.

2.2.3:这就是模拟,应该是考查对于 mod 运算的概念,其实 mod 比循环链表容易,性价比高.

2.2.4:DFS+位运算,也是一道巩固题目,但是仔细想想,这道题不容易,要找着规律,必须用手模拟,然后数学归纳.

◎第三节:

本节有几道不错的 DP,应该是对上一节的扩充,并且难度不小,这是我比较喜欢的一节题目,数学分析,和逻辑推理性很强.主要涉及:DFS,DP

2.3.1:看似字符串的模式匹配,实则是一道 DP,我的程序本来就是 DP,但是我并没有看出,方法是不断更新当前的值,隐含着 DP.

```
for(int i=0;i<lent;i++)
    {
        for(int j=0;j<=len;j++)
            if(i+dl[j]>lent) continue; //超界判断
            else
            {
                f=false;
                for(int k=0;k<dl[j];k++)
                    if(core[i+k]!=pi[j][k])   {f=true;break;}
                if (!f) maxn=(dl[j]+i>maxn?dl[j]+i:maxn);   //更新最大值
            }
        if(i+1>maxn) break; //退出条件
    }
    fout<<maxn<<endl;
```

2.3.2:是一道有一点难度的 DP,我的程序是 4 重循环的,**dp[i][j]=Sigma {(p!=i-1)?2*dp[i-1][q]*dp[p][j-q-1]:dp[i-1][q]*dp[p][j-q-1]}(1<=p<i;2*i-3<=q<j),**思想就是从上一个状态递推,然后更新最大值,最后累加,细节就是 mod 运算,这是我非常喜欢的题目,不是因为难,而是这是我第一个自己写出的 DP!

```
dp[1][1]=1;
    for(int i=2;i<=k;i++)
        for(int j=3;j<=n;j+=2) //结点只能为奇数
        if((i<8)&&(((1<<i)-1)<j)) break; //局部剪枝
        else
        {
            if (j<(2*i-1)) continue; //剪枝
            for(int p=1;p<i;p++)
                for(int q=2*i-3;q<j;q+=2)
                {
                    temp=((p!=i-1)?2*dp[i-1][q]*dp[p][j-q-1]:dp[i-1][q]*dp[p][j-q-1]);
                    dp[i][j]=(temp+dp[i][j]>dp[i][j]?(temp+dp[i][j])%m:dp[i][j]);
                }
        }
    fout<<dp[k][n]<<endl;
```

2.3.3:DFS,由于没有把计算结果迭代,而是最后计算,不得不使用了惰性计算技术,如果迭代的话,程序会短不少.

2.3.4:经典的背包,但是方程略有不同,**f[i][j]=f[i-1][j]+f[i][j-num[i]],**注意第二个多项式不是 i-1 而是 i,为什么会这样,我总结了一下,首先先前那题,所有被选的物品(数字)只能选一次,而这个

可以选多次(货币),因此是 i,因为之前也可以选择.伪多项式降维后:**f[j]=f[j]+f[j-num[i]];**

2.3.5:开始我做麻烦了,没有想好,最后总结了一下,按层次 DFS,并且记录访问过的结点就没有问题.

◎第四节:

这一节就有图论了,当然是最基本的最短路问题,有的比较基本,有的则布满陷阱.主要涉及:BFS,模拟,数学.

2.4.1:基本的模拟却有很多的思想，第二章的模拟题明显比第一章要难，难点不在过程，而在退出条件，数学分析完状态后，加上最长的条件即可，当然还有用 6 维数组以及最小公倍数判断的，都不错。

2.4.2:BFS 的 Floodfill,填充时需要记录状态，这样能避免重复搜索。这样 floofill 的 dfs 与 bfs 版本都有了。

2.4.3:Floyd,主要是设计算法，如何合并，然后不要忘记一个条件“包含”，并不难求解,就是需要自己构造出可能的情况.**分析清楚问题的情况,往往能决定算法的健壮性,但是分析通常需要有充足的数学基底,同时不下手模拟看看结果,有时是分析不出问题的**。

2.4.4:简单的 Floyd+贪心，因为有些路径是没有用的，只需要最短的，不难。

2.4.5:这是数学模拟题，就是模拟除法，然后记录余数，只要再次出现就循环，因为我的方法所记录的余数实际是下一个数的，所以最后要整理，才能输出。

★章末总结：

本章循序渐进，并不急着拔苗助长，而是有层次的增加难度，但是无论是多难的题目，都要经过缜密的思考，理清思想模式，但是不要落入套路的陷阱，我的做错的题目主要原因就是落入了套路陷阱，即使做过的题目也应当再次尝试新的方式方法，争取找到思想的“捷径”，这也是本章的意义，所谓巩固提高，不是把做过的再做一遍，而是在此基础上，使用更好的。这样才能练就一身硬功，总有一天可以草木皆兵器。

2009/8/19