USACO2012-2013

By-Winniechen

CATALOG

Warming up!

- 13 Mar The Cow Run
- 13 Mar Necklace
- 12 Nov Balanced Cow Breeds
- 13 Jan Seating
- 13 Feb Route Designing

Junior!

- 13 Jan Island Travels
- 12 Dec First!
- 12 Dec Gangs of Istanbull
- 13 Feb Taxi
- 13 Feb Partitioning the Farm

Senior!

- 13 Jan Cow Lineup!
- 12 Nov Balanced Tree
- 12 Nov Concurrently Balanced Strings
- 13 Open Photo
- 13 Open Yin and Yang

WARMING UP!


我们先来练练手！
13 Mar The Cow Run
13 Mar Necklace
12 Nov Balanced Cow Breeds
13 Jan Seating
13 Feb Route Designing

Description

John养了一只叫Joseph的奶牛。一次她去放牛，来到一个非常长的一片地，上面有N块地方长了茂盛的草。我们可以认为草地是一个数轴上的一些点。Joseph看到这些草非常兴奋，它想把它们全部吃光。于是它开始左右行走，吃草。John和Joseph开始的时候站在0位置。Joseph的移动速度是一个单位时间一个单位距离。不幸的是，草如果长时间不吃，就会腐败。我们定义一堆草的腐败值是从Joseph开始吃草到吃到这堆草的总时间。Joseph可不想吃太腐败的草，它请John帮它安排一个路线，使得它吃完所有的草后，总腐败值最小。John的数学很烂，她不知道该怎样做，你能帮她么？

13 MAR THE COW RUN

Sample Input	Sample Output
4 -2 -12 3 7	50

13 MAR THE COW RUN

题解：

区间DP裸题，现将草按位置排序，之后f[i][j]表示左边吃到i，右边吃到j，牛在i的最短路径，和g[i][j]表示左边吃到i，右边吃到j，牛在j的最短路径。

转移：

```
f[j][i]=min(f[j+1][i-1]+(n-i+1)*(a[j+1]-a[j])
,g[j+1][i-1]+(n-i+1)*(a[k]-a[j]));
g[j][i]=min(f[j][i-1]+(a[k]-a[j])*(n-i+1)
,g[j][i-1]+(a[k]-a[k-1])*(n-i+1));
```

13 MAR THE COW RUN

Description

给你一个长度为n的字符串A，再给你一个长度为m的字符串B，求至少在A中删去多少个字符才能使得B不是A的子串。注：该题只读入A和B，不读入长度，先读入A，再读入B。数据保证A和B中只含小写字母。

13 MAR NECKLACE

Sample Input	Sample Output
ababaa aba	1

13 MAR NECKLACE

题解：

KMP+DP

可以很显然的得出DP方程

f[i][j]表示主串匹配到i，模式串匹配到j主串中没有模式串的最少删除量。

常规KMP+DP的转移操作。

如果不会KMP右转转到：

https://www.cnblogs.com/Winniechen/p/9144323.html

13 MAR NECKLACE

Description

Farmer John usually brands his cows with a circular mark, but his branding iron is broken, so he instead must settle for branding each cow with a mark in the shape of a parenthesis -- (. He has two breeds of cows on his farm: Holsteins and Guernseys. He brands each of his cows with a parenthesis-shaped mark. Depending on which direction the cow is facing, this might look like either a left parenthesis or a right parenthesis. FJ's N cows are all standing in a row, each facing an arbitrary direction, so the marks on the cows look like a string of parentheses of length N. Looking at this lineup, FJ sees a remarkable pattern: if he scans from left to right through just the Holsteins (in the order they appear in the sequence), this gives a balanced string of parentheses; moreover, the same is true for the Guernseys! To see if this is truly a rare event, please help FJ compute the number of possible ways he could assign breeds to his N cows so that this property holds. There are several ways to define what it means for a string of parentheses to be "balanced". Perhaps the simplest definition is that there must be the same total number of ('s and)'s, and for any prefix of the string, there must be at least as many ('s as)'s. For example, the following strings are all balanced: () (()) ()(()()) while these are not:)(())(((()))

13 NOV BALANCED COW BREEDS

Description

给一个只有左右括号的字符串，然后用H,G两种字符来标记这个序列，所有标记H的括号可以组成一个正确的括号序列，所有G标记的括号也组成一个正确的括号序列，然后输出这种标记方案的总数mod2012的值。

Sample Input	Sample Output
(())	6

题解：

DP

括号匹配的一般形式，如果一部分匹配，并且整体括号匹配，那么剩下的部分一定会匹配。（手画一下）

那么其实本质上，我们需要的就是括号匹配的方案数。

设状态f[i][j]表示枚举到i，其中选择的左括号比右括号多j

转移：

f[i][j]=f[i-1][j-x]+f[i-1][j];

压掉一维，随便弄弄就好了

13 NOV BALANCED COW BREEDS

同Hotel，详情见上一个PPT

13 NOV SEATING

Description

在一条蛋疼的河两旁有一些景点。河左边有N个景点，河右边有M个景点。在河上搭着一些桥沟通两岸的景点（桥是无向的），但是在河的同一边的景点之间都有参天的大树阻拦着，无法通行。我们从河的上游往下游按顺序把景点编上号。目前我们要设计1条路线，可以从河岸任意一边的任意一个景点开始到河岸任意一边的任意一个景点结束。这条路线必须要满足其中没有桥交叉。我们对于交叉的定义如下：对于a<->x和b<->y两座桥，若(a<b && x>y) || (a>b && x<y) || (a=b && x=y)，则这两座桥交叉。满足上面的条件之后，开始提问了：已知每个景点都有一个美丽值，你设计的路线必须满足其美丽值最大。那么最大的美丽值是多少？

13 FEB ROUTE DESIGNING

Sample Input	Sample Output
3 2 4 1 1 5 2 2 1 1 2 1 3 1 2 2	8

13 FEB ROUTE DESIGNING

题解：

贪心+DP

我们考虑，暴力的话是f[i][j]时间复杂度O(n^2)

那么转换一下枚举方式，用边作为转移状态

将边按左端点大小为第一关键字，右端点大小为第二关键字，这种情况下，我们可以发现，因为没有重边，所以点x更新到的点一定在更新x点的位置之下，而y同样

这种情况下，之间设状态f[i],g[i]分别表示左右端的最大值，之后按边的顺序枚举即可！

13 FEB ROUTE DESIGNING

JUNIOR！


难度有所提升，请认真思考！
13 Jan Island Travels
12 Dec First!
12 Dec Gangs of Istanbull
13 Feb Taxi
13 Feb Partitioning the Farm

Description

给你一张r*c的地图，有’S’，’X’，’.’三种地形，所有判定相邻与行走都是四连通的。我们设’X’为陆地，一个’X’连通块为一个岛屿，’S’为浅水，’.’为深水。刚开始你可以降落在任一一块陆地上，在陆地上可以行走，在浅水里可以游泳。并且陆地和浅水之间可以相互通行。但无论如何都不能走到深水。你现在要求通过行走和游泳使得你把所有的岛屿都经过一边。Q：你最少要经过几个浅水区？保证有解。

13 JAN ISLAND TRAVELS

Sample Input

```
5 4
XX.S
.S..
SXSS
S.SX
..SX
```

Sample Output

```
3
```

题解：

状压DP+最短路

乍一眼看过去应该是斯坦纳树，但是不需要那么麻烦，我们只需要求出任意两个岛之间的最短路径，用Dij或者Spfa处理出来，之后DP一下就可以了。

f[i][S]表示在i岛上，状态为S，之后转移一下，常规的DP操作，处理岛的编号很麻烦，需要读懂题…记住，是岛的编号，而不是土地的编号

13 JAN ISLAND TRAVELS

Description

给定n个总长不超过m的互不相同的字符串，现在你可以任意指定字符之间的大小关系。问有多少个串可能成为字典序最小的串，并输出这些串。n <= 30,000 , m <= 300,000

13 DEC FIRST!

Sample Input	Sample Output
4 omm moo mom ommnom	2 omm mom

13 DEC FIRST!

题解：

Trie树+拓扑排序

我们每次贪心的给出字符之间的关系，之后判断是否有环即可。

给出关系：Trie树同一层的存在其他字符小于该字符。

判环还不是随便判…

13 DEC FIRST!

Description

n头牛结成了m个帮派，现在它们争夺一块草地。每个单位时间内会有一头牛来。如果草地上还没有牛或者只有自己帮派的牛，他会留在这里。但如果已经有别的帮派的牛，它们会打一架，这使得当前牛和草地上的一头牛去找农夫思考人生。问如何安排来的牛的编号顺序，能使一 号帮派最后有最多的牛留在草地上，如果不为0，还要输出字典序最小的一组方案。

13 DEC GANGS OF ISTANBULL

Sample Input	Sample Output
5 3	YES
2	1
1	1
2	3
	2
	3
	1

题解：

贪心

我们可以思考如何处理出第一问的解。

每次用牛数量次大帮派的牛去抵消牛数量最大的那个，那么我们就可以得到最少的牛剩余，之后用把1号帮派的牛放入，剩余的数量就是答案。

而第二问则很暴力，每次枚举加入牛所在的帮派，之后用剩余的牛的验证是否还是最大…

13 DEC GANGS OF ISTANBULL

Description

Bessie在农场上为其他奶牛提供出租车服务。这些奶牛已经在沿着长度为M（1<= M <= 1,000,000,000）的栅栏上不同的地点聚集等候。不幸的是，他们已经厌倦了他们当前所在的位置并且每只奶牛都想要沿着栅栏去别的地方走走。 Bessie必须赶到这些奶牛的起始位置，并把他们带到它们的目的地。Bessie的车很小，所以她只能一次只能搭载一头奶牛。奶牛可以在同一时刻完成上车和下车。为了节省燃气，Bessie想以尽可能少的燃料完成这次任务。N只奶牛的起始位置和结束为止都是已知的（1 <= N <=100000），请确定Bessie的最少行程。Bessie意识到，要使所得到的行程最短，Bessie可能将在沿途中让奶牛上车或下车而并不一定将一头奶牛从起点直接送到中点。Bessie的起点是围栏的最左端，位置记为0。终点在篱笆的最右边，位置记为M。

13 FEB TAXI

Sample Input	Sample Output
2 10 0 9 6 5	12

在0号站接第1个客户，从0号站出发到6号站（7站），放下第1个客户，接第2个客户，送到5号站再回来（2站），再接第1个客户，送到9号点（3站），共12站。（因为送完所有客户之后Bessie已在终点，故无需再走）

13 FEB TAXI

题解：

贪心+排序

因为牛都是一样的，我们可以理解为当两头牛相遇在某一起点，另一头牛代替原牛走到最近的终点（然后就消失了，此终点失效），再回头载上个起点的牛往前走。发现每次回头的是当前最小的终点到当前最小的起点的距离，又由于从0出发要走到m，于是我们将0加入终点中，m加入起点中，（当作两段回头的距离），从小到大排序，ans累加起点减去终点（取绝对值）。

13 FEB TAXI

Description

给出一个n*n的矩阵，用k条水平或竖直直线分割矩阵，最小化区域和最大值。

13 FEB PARTITIONING THE FARM

Sample Input	Sample Output
3 2 1 1 2 1 1 2 2 2 4	4

13 FEB PARTITIONING THE FARM

题解：
二分答案+状压DP
横向状压，纵向贪心验证答案
具体实现没有去写，细节上应该会很多，我大概说了一嘴，之后JZYshuraK写了一个，大概想写的话可以去看JZYshuraK的blog

SENIOR !


难度较大，注意听讲
13 Jan Cow Lineup!
12 Nov Balanced Tree
12 Nov Concurrently Balanced Strings
13 Open Photo
13 Open Yin and Yang

Description

给你一个长度为n(1<=n<=100,000)的自然数数列，其中每一个数都小于等于10^9，现在给你一个k，表示你最多可以删去k类数。数列中相同的数字被称为一类数。设该数列中满足所有的数字相等的连续子序列被叫做完美序列，你的任务就是通过删数使得该数列中的最长完美序列尽量长。

13 JAN COW LINEUP!

Sample Input	Sample Output
9 1 1 2 7 3 7 7 3 7 5 7	4

13 JAN COW LINEUP!

题解：

单调队列+离散化

先将权值离散化，之后我们考虑答案究竟是什么

如果确定右端点，则左端点的最小值一定可以确定，即：最小的L满足[L,R]中只有K+1种数（即删除K个数后只有一种数）显然，L单调递增，之后维护一下就处理出来了！

13 JAN COW LINEUP!

Description

Fascinated by his experience with balanced parentheses so far, Farmer John is curious if you can help him solve one final problem. As it turns out, FJ's farm is in the shape of a giant tree of N pastures (1 <= N <= 40,000), each of which he has labeled with either (or). For example: '('--'('--')'--'('--')' | | ')' ')'--'('--'(' | | ')' '('--')'--')'--')'--'(' Recall that since his farm is a tree, this means that certain pairs of pastures are connected by corridors so that there is one unique path between any given pair of pastures. FJ believes that some of these paths represent balanced strings of parentheses. In particular, he would like to know, among all such balanced strings represented by paths through the tree, what is the maximum nesting depth one can find. The nesting depth of a balanced string of parentheses is the maximum, over all prefixes of the string, of the excess number of ('s within the prefix. For example, the string ()()() has nesting depth 1, but the string ((()))() has nesting depth 3, as we can see clearly if we count excess ('s for every prefix of the string: ((()))() 12321010 For the example farm above, the deepest string is ((())) with a depth of 3, and can be obtained by taking the path from A to B below: '('--'('--')'--'('--')' | | ')' ')'--'('--'(' < A | | ')' '('--')'--')'--')'--'(' ^C ^B Note that this is different than the longest balanced string; for instance (())(()), starting at A and ending at C, has length 8. Your task is to output the nesting depth of the deepest balanced path in the tree.

12 NOV BALANCED TREE

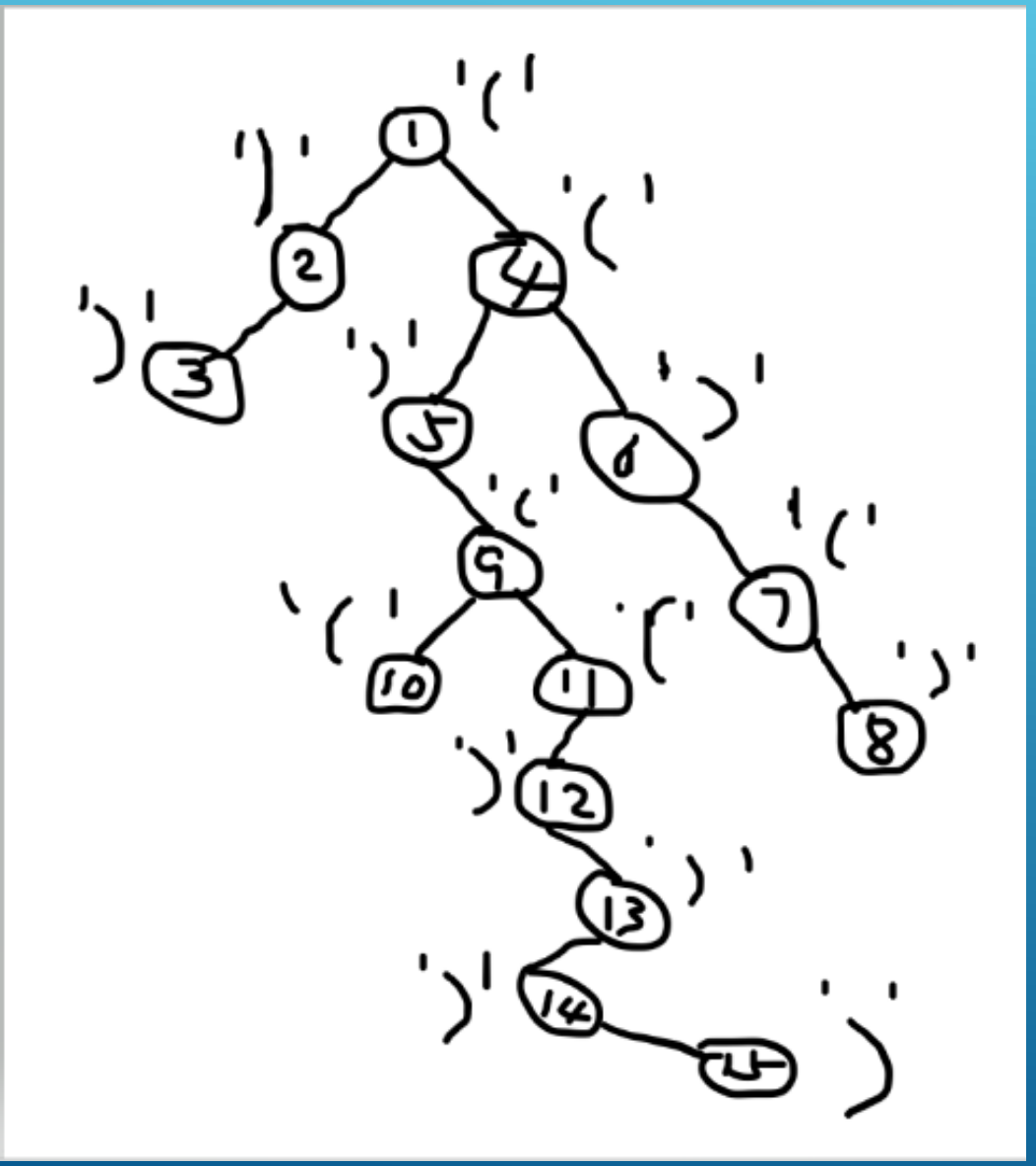
Description

给出一棵树，每个结点上有一个括号。问在所有括号平衡的路径中，最大嵌套深度为多少。

（最大嵌套深度的大概意思：()()()的最大嵌套深度为1，((()))的最大嵌套深度为3）

12 NOV BALANCED TREE

Sample Input



Sample Output

3

12 NOV BALANCED TREE

题解：

点分治+栈

这道题USACO官网上还另外一种解法，但是我并不太明白时间复杂度为什么是对的，所以就不讲了。

点分治应该是很显然的，以下是官方题解

“请注意，对于一些带有相同数量'（'和'）'的括号的字符串a1 a2 ...，总会存在一些嵌套深度至少等于a1的平衡子字符串...一个。因此，简单地找到具有相同数目的两个圆括号的路径的最大嵌套深度就足够了。对于在O（n）时间内通过特定顶点v的路径，很容易解决这个问题，其中n是v的子树的大小。如果选择v使得v的所有子树的大小至多为v的一半那么问题可以在O（N log N）中解决，通过递归地解决v的子树的问题。这样的一个顶点总是存在，并且可以通过重复地下降特定顶点的子树来找到，如果它的大小是大于树的一半大小。”

很显然，他可能说的是废话…（？！很容易O(n)时间处理处理出来？）（一半在讲点分治怎么实现）

12 NOV BALANCED TREE

真正的题解：
点分治+栈
其实本质上这道题还是树上统计路径问题，仔细思考一下可以发现，点分治可以做。
我们先想一个简单一点的版本的，我们求括号匹配的数量。那么就是求出以重心为根的子树中，括号匹配的前提下，左括号的剩余数量和右括号的剩余数量，之后容斥原理搞一搞就可以了。
但是…显然，这道题不是个计数问题，容斥原理不成立。

12 NOV BALANCED TREE

真正的题解：
点分治+栈
那么，我们每次模拟栈求出每个点到根节点的括号匹配前提下的左括号剩余和左括号剩余为i的情况下，最深的嵌套为多少，之后正向枚举每个子树，先求出左括号的剩余和最深嵌套，之后和前面所有的右括号剩余和最深嵌套合并一下，之后再将右括号剩余和最深嵌套合并到前缀中，记得记录枚举的边的顺序。
因为，还需要反向枚举一遍。

真正的题解：

点分治+栈

如何处理每一条路径的信息就需要用到栈来维护。

左括号具体实现如下：

```
void get_left(int x,int from,int now,int last,int sum)
{
    siz[x]=1;
    if(now||a[x]==-1)now-=a[x];else last++;sum=max(sum,now-last);
    if(!now)f[last]=1,mf[last]=max(mf[last],sum),maxx=max(last,maxx);
    for(int i=head[x];i!=-1;i=e[i].next)
    {
        int to1=e[i].to;
        if(to1!=from&&!vis[to1])
        {
            siz[x]+=siz[to1];
            get_left(to1,x,now,last,sum);
        }
    }
}
```

真正的题解：

点分治+栈

如何处理每一条路径的信息就需要用到栈来维护。

右括号具体实现如下：

```
void get_right(int x,int from,int now,int last,int sum)
{
    siz[x]=1;
    if(now||a[x]==1)now+=a[x];else last++;sum=max(sum,now-last);
    if(!now)g[last]=1,mg[last]=max(mg[last],sum),maxx=max(last,maxx);
    for(int i=head[x];i!=-1;i=e[i].next)
    {
        int to1=e[i].to;
        if(to1!=from&&!vis[to1])
        {
            siz[x]+=siz[to1];
            get_right(to1,x,now,last,sum);
        }
    }
}
```

真正的题解：
计算每个重心作为根的情况具体实现如下：

```
void calc(int x)
{
    int v=0,maxn=0,tot=0;
    for(int i=head[x];i!=-1;i=e[i].next)
    {
        f[0]=1,mf[0]=0;
        maxx=0;tms[++tot]=i;
        int to1=e[i].to;
        if(!vis[to1])
        {
            get_left(to1,x,0,0,0);
            for(int k=0;k<=maxx;k++)ans=max(ans,g[k]*f[k]*max(mf[k]+k,mg[k]+k));
            for(int k=1;k<=maxx;k++)f[k]=0,mf[k]=0;
            maxx=0;
            get_right(to1,x,a[x]==1,a[x]==-1,a[x]==1),maxn=max(maxx,maxn);ans=max(ans,mg[0]);
            if(a[x]==-1)g[1]=1,mg[1]=max(mg[1],1),maxx=max(maxx,1);
        }
    }
```

真正的题解：
计算每个重心作为根的情况具体实现如下：

```
    for(int i=0;i<=maxn;i++)g[i]=0,mg[i]=0;maxn=0;
    for(int j=tot;j;j--)
    {
        int i=tms[j],to1=e[i].to;f[0]=1,mf[0]=0;maxx=0;
        if(!vis[to1])
        {
            get_left(to1,x,0,0,0);
            for(int k=0;k<=maxx;k++)ans=max(ans,g[k]*f[k]*max(mf[k]+k,mg[k]+k));
            for(int k=1;k<=maxx;k++)f[k]=0,mf[k]=0;
            maxx=0;
            get_right(to1,x,a[x]==1,a[x]==-1,a[x]==1),maxn=max(maxx,maxn);ans=max(ans,mg[0]);
            if(a[x]==-1)g[1]=1,mg[1]=max(mg[1],1),maxx=max(maxx,1);
        }
    }
    for(int i=0;i<=maxn;i++)g[i]=0,mg[i]=0;maxn=0;
}
```

Description

[Brian Dean, 2012] Farmer John's cows are all of a very peculiar breed known for its distinctive appearance -- each cow is marked with a giant spot on its hide in the shape of a parenthesis (depending on the direction the cow is facing, this could look like either a left or a right parenthesis). One morning, Farmer John arranges his cows into K lines each of N cows (1 <= K <= 10, 1 <= N <= 50,000). The cows are facing rather arbitrary directions, so this lineup can be described by K length-N strings of parentheses S_1,..., S_k. Farmer John notes with great excitement that some ranges of his cows are "concurrently balanced", where a range i...j of cows is concurrently balanced only if each of the strings S_1,..., S_k is balanced in that range (we define what it means for a single string of parentheses to be balanced below). For instance, if K = 3, and we have S_1 =)()((()))(()) S_2 = ()(()()()(()) S_3 =)))(()()))(()) 1111 01234567890123 Then the range [3...8] is concurrently balanced because S_1[3...8] = ((())), S_2[3...8] = ()()(), and S_3[3...8] = (()()). The ranges [10...13] and [11...12] are also concurrently balanced. Given K length-N strings of parentheses, help Farmer John count the number of pairs (i,j) such that the range i...j is concurrently balanced. There are several ways to define what it means for a single string of parentheses to be "balanced". Perhaps the simplest definition is that there must be the same total number of ('s and)'s, and for any prefix of the string, there must be at least as many ('s as)'s. For example, the following strings are all balanced: () (()) ()(()()) while these are not:)(())(((()))

12 NOV CONCURRENTLY BALANCED STRINGS

Description

给出k个长度为n的括号序列，问有多少个区间在k个序列中对应的子串均平衡。

Sample Input	Sample Output
3 14)()((()))(()) ()(()()(()()))))(()())(())	3

12 NOV CONCURRENTLY BALANCED STRINGS

题解：

(DP？栈？神奇操作？)+(平衡树？)

一个串多少个括号匹配的子串有多少会不会啊？

我会！O(n^2)暴力枚举！（一拳打倒）

要100000可做的！

两种做法

(1)BY *fowuvu* 离线树状数组或者主席树二维数点具体实现我不太会

(2)权值线段树or平衡树动态维护前缀和的桶之后对应动态的进行括号匹配O(nlogn)

12 NOV CONCURRENTLY BALANCED STRINGS

题解：
那么一个会不会先不重要，我们可以先看一个性质。
其实是一样的，map有一个非常棒的性质：
map的第一维可以是一个vector，之后这个vector作为比较参数的时候，只要两个vector中有一个不同的那么就不同。利用这个性质可以做很多事情。（比如不用非要存一个结构体之后手写重载==号和!=号）
这个性质非常棒，这样我们考虑用map进行一个串匹配的时候是如果实现的。

12 NOV CONCURRENTLY BALANCED STRINGS

题解：

用map进行括号匹配个数查询的时候，我们可以动态的维护出一个前缀和，和一个L[i]表示前缀和为i的时候，最早的满足括号匹配性质的前缀和为i的位置。map中以前缀和为第一关键字，L[前缀和]作为第二关键字的第一关键字，数量为第二关键字的第二关键字，之后map中每次多一个括号的时候更新一下对于前缀和的最早位置，之后找一下map中存储的最早位置是否相同，如果相同ans+=map[L[sum]].second，并且map[L[sum]].second++，如果不同，之间修改

题解：
如何保证正确性的关键是每次改变L[i]，其实很简单，就是来了右括号那么就将i和L[sum]取min，来了左括号就之间赋值，之后将进行上一步的操作就可以了。
那么我们转换一个多个的，那个map的性质很棒，所以可以考虑之间存一个vector记录每个串的前缀和，之后剩下的操作几乎一模一样。就需要另外每次判断的用所有的L[sum]的最大值进行判断就可以了。
下页是具体实现。

12 NOV CONCURRENTLY BALANCED STRINGS

题解：

```
for(int i=0;i<n;i++)
{
    int left=0;
    for(int j=0;j<k;j++)
    {
        if(a[j][i]==1)R[j][++L[j]]=i+1;
        else L[j]--,R[j][L[j]]=min(R[j][L[j]],i+1);
        left=max(left,R[j][L[j]]);
    }
    if(left==n)continue;
    pair<int ,int >&tmp=mp[L];
    if(tmp.first==left)ans+=tmp.second++;
    else tmp=make_pair(left,1);
}
```

12 NOV CONCURRENTLY BALANCED STRINGS

Description

Farmer John has decided to assemble a panoramic photo of a lineup of his N cows (1 <= N <= 200,000), which, as always, are conveniently numbered from 1..N. Accordingly, he snapped M (1 <= M <= 100,000) photos, each covering a contiguous range of cows: photo i contains cows a_i through b_i inclusive. The photos collectively may not necessarily cover every single cow. After taking his photos, FJ notices a very interesting phenomenon: each photo he took contains exactly one cow with spots! FJ was aware that he had some number of spotted cows in his herd, but he had never actually counted them. Based on his photos, please determine the maximum possible number of spotted cows that could exist in his herd. Output -1 if there is no possible assignment of spots to cows consistent with FJ's photographic results.

13 OPEN PHOTO

Description

给你一个n长度的数轴和m个区间，每个区间里有且仅有一个点，问能有多少个点

Sample Input	Sample Output
5 3 1 4 2 5 3 4	1

13 OPEN PHOTO

题解：
乍一眼看过去应该是线段树嘛…不过线段树也可以做，但是时间复杂度是O(nlogn)，而且常数较大，并不推荐使用，而且，写起来比较麻烦。
单调队列+DP
转化一下就变成了JDFZOJ1057假期那道题了。（其实你们可以尝试一下用线段树写，应该没问题维护区间最大值和单点修改）
重点来了，怎么转化！

13 OPEN PHOTO

题解：
其实知道是DP之后方程可以很显然的写出来（并不显然）
f[i]表示在i这个点有一头牛包括i以前最多能放多少牛。
转移f[i]=f[j]+1；
我们考虑j有没有什么性质，发现j必定是连续的一段。
而连续的一段满足什么性质呢？
考虑针对每一个点进行分类，如果一个点是某一区间中的点，那么必定满足j<区间的左端点，并且，一定满足左端点大于等于上一个最靠右的不包含i的区间的左端点。这个用差分来实现(前缀最大值和后缀最小值)

13 OPEN PHOTO

题解：

据说可以用差分约束系统做，但是我不是很会，而且USACO卡spfa已经形成习惯…用spfa做实在是等死…

https://www.luogu.org/problemnew/solution/P3084

这里有一个人作死…之后你们可以去看看…

13 OPEN PHOTO

最后，我们用一个双倍经验来结束
同 采药人的路径
如果不会采药人的路径，右转转到
https://www.cnblogs.com/Winniechen/p/9093585.html

13 OPEN YIN AND YANG

THANKS FOR YOUR LISTENING!