公告

计蒜之道2015程序设计大赛 初赛题解

2015年7月19日
蒜头
6 Comments

第一场

搜狗输入法的分词算法

字符串处理的简单题目。需要注意一些细节，比如 0k0k0 后面的那个 0k0 是不能被提取出来的。还有 0k005 只能提取出 0k0，因为 005 是有不合法的前导 0 的，不是一个合法的数字。

工作区的颜值选择

子问题 1

题目中对于 $n \leq 2$，$m \leq 3$ 规模很小。但是如果直接枚举每个工位上的颜值，复杂度为 k^6，会超时。考虑将格子黑白染色，即将 $i+j$ 为奇数的染为白色，剩余的染为黑色。很容易发现：黑格子与黑格子，白格子与白格子之间是相互独立的，代价只产生在黑白格子之间。也就是我们如果确定了白格子，黑格子由于相互独立，可以贪心的选择颜值大小。

枚举白格子复杂度为 $O(k^3)$，贪心选择黑格子的复杂度为 $O(k)$ 。总复杂度为 $O(k^4)$。

子问题 2

题目中 n 等于 2，可以用动态规划来解决。

$dp_{i,j,z}$ 表示，工位 (1, i) 颜值为 j，工位 (2, i) 颜值为 z 的最小花费。

转移如下：

定义 $cost_{x,y,z}$ 为工位 (x, y) 上选择颜值为 z 付出的代价

$$dp_{i,j,z} = \min_{a,b\in[-k,k]}(dp_{i-1,a,b} + |a+j| + |b+z|) + |j+z| + cost_{1,i,j} + cost_{2,i,z}$$

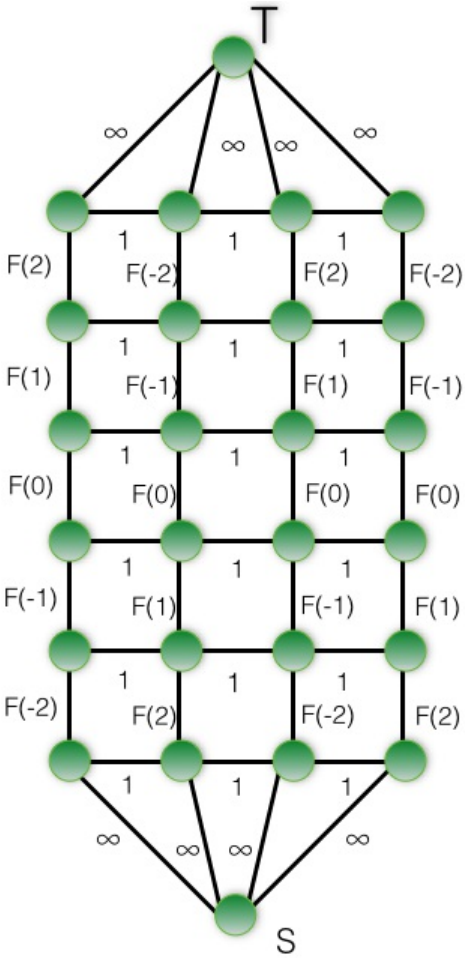
上式转移复杂度为 O(k^2)，动态规划总复杂度为 O(mk^4)，会超时。考虑优化，对于上面的转移方程，如果只枚举 a，那么转移方程，变为

$$dp_{i,j,z} = \min_{b\in[-k,k]}(dp_{i-1,a,b} + |a+j| + |b+z|) + |j+z| + cost_{1,i,j} + cost_{2,i,z}$$

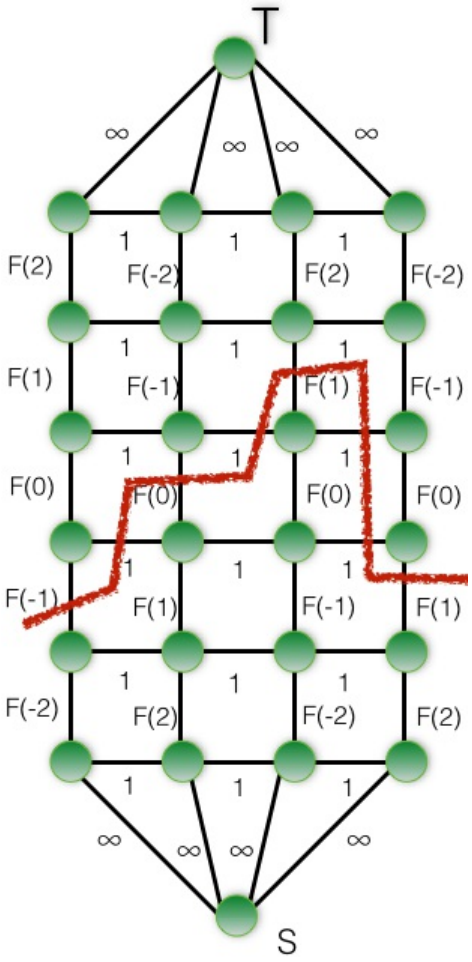
式子中的变量为 $dp_{i-1,a,b} + |b+z|$，转移即为寻找该变量的最小值，这是动态规划经典的绝对值最值寻找问题，直接将绝对值展开，分两段区间维护最小值即可。复杂度降为 O(mk^3)。

子问题 3

考虑如下建图



将每一个工位拆为 2k+2 个点，之间连权值为 F(i) 的边，F(i) 表示对应格点填 i 时的价值，相邻对应点连权值为 1 的边源点汇点分别连向链的末端权值 ∞，考虑一个割的情况



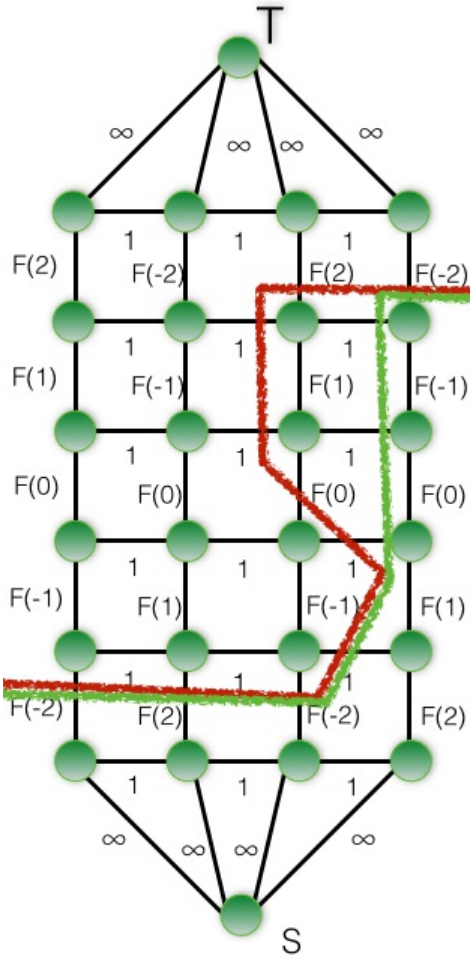
该割集对应的价值为

F(-1)+1+F(0)+1+F(1)+1+1+F(1)=

F(-1)+|-1+0|+F(0)+|0+1|+F(1)+|1+1|+F(1)

也就是一个割集对应一个解的价值，所以对上图求解最小割即可

要注意考虑到以下割集的情况



在该图中红色的割集一定不会是最小割，因为链之间的边权都相等 且都为1，这样的割集显然会没有绿色的割集价值小，或者可以将每条链上的边权加上一个比上图中∞小的∞’ 。这样也能保证每条链只被割一次，也就是不会出现上图红色的割集为最小割的情况，图中点数为 30×30×60 ，大概 5 万左右的点数，需要用效率较高的最大流算法才能通过所有测试数据。

第二场

人人都有极客精神

需要用到经典的日期处理逻辑，注意一些细节的处理，比如闰年判断等。还有，数据里面有 00010101 这种数据哦（人人可是一家历史悠久的公司 >_<）。

还要注意的一点是，输出是有可能等于 0 的，要仔细读题。

自建物流的无人机实验

子问题1 ，$n \le 10$：

暴力大法好，直接 O(2^n) 或O($2^n n$)枚举所有选点方案即可。

子问题2 ，$n \le 2000$：

首先，对于每个节点 x，只有在以 x 为根的子树内的点才会对最近公共祖先为 x 的点对数产生影响，所以我们在解决每个节点的时候，应该优先把它的所有子节点都解决了（因为它所有子节点为根的子树内的点都能对它产生影响）。

然后，如果两个点属于 x 同一个子节点为根的子树内，那么这两个点的最近公共祖先不为 x；反之，若它们属于不同子节点为根的子树，那么它们的最近公共祖先必定为 x。

我们在这个计算中把 x 这个点本身特殊处理成 x 的一个子树，因为它也满足其他子树内的点跟它的最近公共祖先为 x。

我们假设 x 的子节点分别为 s_1，s_2，…，s_k，cnt_x 表示以 x 为根的子树内选了多少个点，dp_x 表示最近公共祖先为 x 的点对对数。于是有

$$cnt_x = \sum_{i\in[1,k],i\in Z} cnt_{s_i}$$

然后我们可以得到最近公共祖先为 x 的点对对数等于 x 为根的子树内的点对总数 –x 所有子节点为根的子树内的点对总数

写成公式就是：

$$\begin{aligned} dp_x &= cnt_x\frac{cnt_x-1}{2} - \sum_{i\in[1,k],i\in Z} cnt_{s_i}\frac{cnt_{s_i}}{2} \\ &= \frac{cnt_x^2 - cnt_x - \sum_{i\in[1,k],i\in Z} cnt_{s_i}^2 + \sum_{i\in[1,k],i\in Z} cnt_{s_i}}{2} \\ &= \frac{(\sum_{i\in[1,k],i\in Z} cnt_{s_i})^2 - \sum_{i\in[1,k],i\in Z} cnt_{s_i}^2}{2} \end{aligned}$$

于是我们在 cnt_X，也就是 $\sum_{i\in[1,k],i\in Z} cnt_{s_i}$ 相同的情况下，当然是 cnt_{s_i} 越平均 dp_X越大了。

或者换个角度说，如果有两种方案 *cnt* 和 *cnt'*，其中 $cnt_{S1} < cnt'_{S1} \le cnt'_{S2} < cnt_{S1}$，并且对于 i > 2 都有 $cnt_{Si} = cnt'_{Si}$，那么 *cnt'* 的方案得到的对数更大，是要优于 *cnt* 的方案的。

于是我们在 *i* 的子树中选点的时候，应该尽量朝着平均的方向选，也就是每次都贪心地选择一个未选满并且已选点数最少的子树在里面选择一个未被选过的点。如果子树里的点都选完了点对数还不够，那么显然是无解的。

子问题3 ，$n \le 200000$：

如果是暴力地找已选点数最少的子树或者暴力枚举子树中所有节点来找到一个未被选过的点，复杂度最差情况下会是O(n^2)，于是我们需要优化一下。
首先未选满的子树用优先队列或集合按已选点数大小存取子树的根，每次存取复杂度 O(log*n*)，由于选的次数不会超过 *n* 所以这一步总的复杂度不会超过 O(*n*log*n*) 。

至于如何在子树中选点，这里稍微详细说一下标程的做法。我们可以先 **DFS** 一次求出每个节点的 **DFS** 序，然后用并查集维护一个 *fp*[*x*]，代表 **DFS** 序大于等于 *x* 的节点中第一个还没被选的，每次并查集查该子树的根，查到的就是要选的点的 **DFS** 序，选择后将这个点的 *fp* 指向 **DFS** 序的下一个即可。

当然这里的实现方法不止一种，比如还可以先算出每个点为根的子树需要选多少个点，然后将每个点的选点数按子树们剩余未选的点数分配下去。

整个算法总的复杂度是 O(*n*log*n*)。

第三场

腾讯手机地图

注意输入格式中有提到输入的角度和高度全部都是整数，所以可以开一个长度为 359 的数组来保存每个 1 度区间的最大半径。对于区间 [*l*, *r*]，将数组的 [*l*, *r*-1] 这些元素更新最大值，最终扫一遍求面积总和即可。题目是 Special Judge，所以不用担心精度误差问题，差的不太多都可以通过判题。

商品推荐走马灯

子问题1

对于每次询问，暴力枚举所有子区间，再判断该子区间是否是回文串，将总和累加即可。时间复杂度O(mn^2)。

子问题2

首先用 **Manacher** 算法处理出 2n-1 个回文中心能够向两边延伸的最长回文长度。之后对于每组询问，在询问区间内枚举回文中心的位置，根据 **Manacher** 算出的结果，对于每个回文中心，通过 $sum_i = i \times a_0 + (i\text{-}1) \times a_1 + \ldots + a_i - 1$ 部分和预处理的方法可以将以该位置为中心的在区间范围内的所有回文串的权值总和求出。时间复杂度 O(mn)。

子问题3

题目询问的是一个区间内所有的回文串的权值和，因为考虑的对象是回文串，所以我们可以从回文中心下手。首先用 **Manacher** 算法处理出每一个回文中心能够向两边延伸的最长回文串长度；又题目是支持离线的，所以我们可以把询问离线，然后考虑每一个回文中心对询问的影响。

对于一个询问，我们要考虑的显然只有那些在区间内部的回文中心，并且在区间左半部分的回文中心在向两边延伸时先碰到区间左端点；在区间右半部分的回文中心在向两边延伸时先碰到区间右端点，所以可以把每一个询问区间从中间拆开成两个，分别统计答案就行了。

接下来以统计左半部分区间的答案为例，右半部分类似。按区间的右端点排序，把位于当前区间右端点左边的回文中心用一棵线段树维护起来以方便统计答案。具体的维护方法是考虑这个回文中心到线段树上每一个点构成的回文串的权值和，这个值是对应原串的一个区间和，若处理出前缀和数组，就可以方便地在线段树上进行区间更新了，同时询问答案就是在线段树上询问一段区间的和。

用同样的方法处理一遍左半区间和右半区间，就能得出完整的答案。时间复杂度 O($m\log n$) 。

第四场

爱奇艺的自制节目

W 和 X 节目只能在对应演播室录制，主要问题在 Y 和 Z 节目如何分配演播室。由于数据规模不大，我们完全可以暴力枚举 Y 节目的分配方案，然后 Z 节目尽可能让两个演播室的时间平均即可。注意 Z 节目数量奇偶性等细节的处理。

外卖的餐厅展示

子问题1 ， $n \le 6$

直接暴力枚举每个格子放不放障碍，再进行动态规划或者匹配看时候能完美覆盖即可。 时间复杂度 O(n^2) 或O($2^{4n}n^2T$)。由于 n 不大，在本地打表直接提交也是可行的。

子问题2 ， $n \le 1000$

这时的数据范围就有点大了，我们需要仔细思考问题的本质。 如果每个格子已经确定了放不放障碍，看是否存在完美覆盖是一个经典问题。一 个方法是进行二分图匹配，而另外的一个方法是状态压缩动态规划，由于这里不需要计算完美覆盖的方案数，所以动态规划的方法很容易被忽略。动态规划很简单，开一个 $f[x][1<<4]$ 逐格转移，由这个格子有没有障碍而决定动态规划的两种转移方式，而我们可以发现 f 数组只需是一个布尔类型，就是说 f 数组总共就只有 1<<(1<<4) 也就是 65536 种。这样从中我们就发现了一点门路。

设 $f[x][1<<16]$ 表示当前已经转移到 x 格子，f 数组的状态是 y 的方案数量。y 是一个 16 位的数，里面存放着之前问题的(1<<4) 个 布尔值是 0 还是 1。然后 转移也是看这一个格子放不放障碍，放或不放都将对应着 y 的下一个状态，直接转移即可，具体的求下一个转移的过程就是再做一个和之前问题一模一样的动态规划。时间复杂度为O($2^{16}n$)。

子问题3 ， $n \le 10^{18}$

看到这么大的范围第一想法就是矩阵快速幂，但是如果直接沿用子问题 2 的思路矩 阵大小是 65536 肯定不可取。但是 65536 种状态肯定有冗余的，可以先做一下子问题 2 的动态规划筛选出有用状态再重新标号。可以发现居然只有 82 种有效状态。所以就可以很愉快的矩阵转移了。最后处理每一个询问时可以预先预处理出转移矩阵的 2^i。这样就能把单组询问的时间复杂度去掉一个 82 （虽然不一定有必要）。 综上，时间复杂度为 O($82^3\log n+82^27\log n$)。

PREVIOUS POST
计蒜客参展国际创新峰会 正筹办程序设计大赛

NEXT POST
计蒜之道2015程序设计大赛 初赛作弊

6 comments

zq说道：

2015年7月20日 下午1:32

第四场第一题线性python超时了==求比枚举快的做法。。

Reply

tiger说道：

2015年7月20日 下午2:31

尼玛，艹，第二场太贱了。

Reply

银月游侠说道：

2015年7月20日 下午10:45

原来还可以评论。

第二场第一题，题目没有考虑1582年才开始采用现代的闰年计算方法。

Reply

蒜头说道：

2015年7月21日 上午10:07

还真是这样呢啊，蒜头也是学到新知识了。感谢你的反馈。

Reply

银月游侠说道：

2015年7月21日 上午11:00

就是因为用了Java的GregorianCalendar跪了才发现的-_-||

Reply

过客说道：

2015年7月23日 下午9:15

求助，不知道为何程序过不了第三场第1题，貌似跟解说的一样啊

```
#include
#define PI 3.1415926536
using namespace std;

int t[360];

float loopsum(){
long long sum=0;
for(int i=0;i<359;i++){
sum+=t[i]*t[i];
}
return PI*sum/360.0;
}

void loop(int start,int end,int val){
for(int i=start;i<end;i++){
if(t[i+90]>T;
while(T–){
for(int i=0;i>n;
while(n–){
cin>>r>>a>>b;
loop(a,b,r);
}
cout<<loopsum()<<endl;
}
return 0;
}
```

Reply

评论

你的邮箱地址并不会被公开显示 Required fields are marked *

Name *

Email *

Website

在这里发表你的评论内容……

发表评论

搜索…

近期文章

计蒜之道2015程序设计大赛 电子科大学子夺冠

计蒜之道2015程序设计大赛 总决赛成绩

计蒜之道2015程序设计大赛 决赛题目

计蒜之道2015程序设计大赛 复赛晋级名单

计蒜之道2015程序设计大赛 复赛题解

文章归档

2015年八月

2015年七月

2015年六月

2015年五月

2015年四月

2015年三月

2015年一月

2014年十一月

2014年十月

2014年九月

2014年八月

2014年七月

2014年六月

2014年五月

2014年四月