2020/07/30 CF Mashup Training

Results

Table

=	Penalty	*	A	B	C	D	E	F	G	H	I	J	K	L	M
			+ 00:10	+2 00:44	+1 02:19	+4 04:46			+3 03:13	+ 03:47				+1 00:27	+ 00:58
16	2726		N	O	P	Q	R	S	T	U	V	W	X	Y	Z
			+1 01:26	+6 02:33		+5 01:53	+ 00:58	+ 03:53				+3 03:39	+ 04:58	+1 00:42	

Submit Distribution in Members

Solved	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Pantw															√		√	√	√					√	√	
Withinlover				√								√	√	√									√			
Gary	√	√	√				√	√																		

(√ for solved, O for upsolved, - for tried but not solved)

Solutions

A - Writing Code (543A)

- Solved by ***Gary***

题意

n个人，每个人填一行需要a_i的代价，问n个人按顺序填满m行(每个人可以填任意行)代价不超过b的方案数

解法

简单的dp

```
while(n--){
    int x=read();
    for(int i=1;i<=m;i++){
        for(int j=x;j<=b;j++){
```

```
            (f[i][j]+=f[i-1][j-x])%=mod;
        }
    }
}
```

B - Destroying Roads (543B)

- Solved by ***Gary***

题意

连通图问最多删除多少遍可以仍使得$distance(s1,t1)\le l1,distance(s2,t2)\le l2$

$1\le n,m \le 5\times 10^3$

解法

bfs求出所有点对的最短路，两种情况即$s1 \to t1$和$s2\to t2$是否有重合的路径

1. 没有则删除最短路以外所有点
2. 有则n^2枚举重合的道路，删去路径以外所有边

所有情况取最大值即可

C - Remembering Strings (543C)

- Solved by ***Gary***

题意

n个长m的串$(1\le n,m \le 20)$修改每一个位置的串需要$v_{i,j}$的代价，问最小的代价使得每个串只要有一位满足它与其余串的该位都不相同

解法

状压表示已经处理过的串，枚举状态每次只更改最小的没有被处理的串，对于该串，枚举每一位，要么更改这一位的字母，要么更改所有串这一位上与它字母相同的串并且保留更改代价最小的，这样的贪心可以使总代价最小

D - Road Improvement (543D)

- Solved by ***Withinlover***

题意

给定一颗无根树，树边可以黑白染色。对于选定的中心节点 x 一个优秀的染色方案应当满足树上任意一点到中心节点的路径上黑色的树边至多只有一个。

对于每个节点 n 求出以该节点作为中心节点时的染色方案数。

$n \leq 2e5$

解法

考虑换根dp

当中心节点确定时，显然可以通过简单的树形DP解决。

方程为 $dp[x] = \prod(dp[son[x]] + 1)$

不难发现，当根节点移动时，仅会影响一条边上的两个节点的 dp 数组，而这个可以 $O(1)$ 的进行转移。再模意义下 $dp[son[x]] + 1$ 的值可能为 0，此时逆元不存在，用前缀和后缀计算一下就可以了。

总复杂度 $O(n)$

E - Paths and Trees (545E)

- Solved by ***QwQ***

题意

解法

F - Soldier and Traveling (546E)

- Solved by ***QwQ***

题意

解法

G - Mike and Frog (547A)

- Solved by ***Gary*** & ***Pantw***

题意

$h1=(h1*x1+y1)%m,h2=(h2*x2+y2)%m$,问多少轮后恰好$h1=a1□h2=a2$;

$1\le m le 2\times 10^6$

解法

找到第一次出现a1□a2的位置，再找出运算出现循环的长度，可以保证这些值如果存在会在m次内找到

枚举h1循环了多少次后，判断时候恰好都相同(感觉会在m次循环内重合，不太会证)

有一个细节是a1或a2有可能在循环节出现前出现，也就是不在循环中，需要特判一下

H - Mike and Feet (547B)

- Solved by ***Gary***

题意

一个长n的序列a□问长度为i的所有子串串内最小值的最大值□$(1\le i \le n)$

$1\le n \le 2\times 10^5$

解法

对每个数维护其两侧第一个比它小的数的位置l和r,可以发现长度r-l-1以内的子串的答案都会被a_i更新一遍 所以只需要两次单调栈维护出l和r□在线段树上更新区间最大值，最后在线段树上查询n次

I - Mike and Foam (547C)

- Solved by ***QwQ***

题意

解法

J - Mike and Fish (547D)

- Solved by ***QwQ***

题意

解法

K - Mike and Friends (547E)

- Solved by ***QwQ***

题意

解法

L - Regular Bridge (550D)

- Solved by ***Withinlover***

题意

给定 k 构造一个每个节点的度数都是 k 且包含至少一条割边的图。

解法

尝试了半天，宣判 $k = 2 * s$ 的情况无解

对于 $k = 2 * s - 1$ 的情况，可以构造 4 个包含 s 个点的完全图，然后添加两个点作为割边的两端。剩下的凭感觉对称的构造上去就可以了

M - Brackets in Implications (550E)

- Solved by ***Withinlover***

题意

给定一个仅包含 $\{0, 1, \to \}$ 的离散公式，询问是否可以添加若干个括号使得这个公式的计算结果为 0。

解法

末尾为 1，不管怎么加括号答案都是 1 ，无解

末尾为 0，考虑构造 $1 \to 0$ 倒数第二位是 1 就直接做完了。如果不是 1 的话考虑构造 $(0\to 0)

$\to 0$，第二个 0 可以迭代的拆成 $1\to 0$，即 $(0\to (1 \to 0))\to 0$，然后就做完了

$n = 1$ 和 $n = 2$ 的情况特殊一点，特判一下

N - GukiZ hates Boxes (551C)

- Solved by ***QwQ***

题意

解法

O - GukiZ and Binary Operations (551D)

- Solved by ***Pantw***

题意

给 n, k, l, m，问有多少方案构造数列 $\{a_i\}, 1\le i \le n$，满足以下条件：

1. $\forall i,\,\, 0 < a_i < 2^l$
2. $(a_1\,\mathrm{and}\,a_2)\,\mathrm{or}\,(a_2\,\mathrm{and}\,a_3)\,\mathrm{or}\,\dots\,\mathrm{or}\,(a_{n-1}\,\mathrm{and}\,a_n)=k$

方案数模 m 输出。

$2\le n\le 10^{18},\quad 0\le k\le 10^{18}, \quad 0\le l\le 64, \quad 1\le m\le 10^9+7$

解法

对 k 按位考虑。

对于 k 的某一位分析。记 $\{a_i\}$ 中元素的对应位构成的数列为 $\{d_i\}$，

- 若 k 的这一位上是 0，那么 $\{d_i\}$ 中无连续两项为 1。
- 若 k 的这一位上是 1，那么 $\{d_i\}$ 中有连续两项为 1。

那么我们需要计算出长为 n 的 $0/1$ 数列中无连续两项为 1 的方案数。

这就使我们想起 NOIP2016 提高组初赛第三大题第一小题。

简而言之，我们定义无连续两项为 1 的 $0/1$ 数列是合法序列。

令 f_i 为长为 i 的合法序列数量。

容易得到关系式 $f_i = f_{i - 1} + f_{i - 2}$ 及边界条件 $f_1 = 2, f_2 = 3$。

> 关于这个关系式的意义：考虑最后这位是否为 1。若该位为 1 那么方案数是 f_{i-2}（因为 1 不能相邻），否则方案数是 f_{i-1}。

$\{a_i\}$ 的各元素各位之间均独立，根据计数乘法原理，答案显然是 $(f_n)^c(2^n-f_n)^{l-c}$。这里 c 是 k 的二进制表示中 1 的个数。

如果 $k\ge 2^l$ 那么显然无解，记得特判输出 0。

$\{f_n\}$ 的形式显然是 Fibonacci 数列，我们可以以 $\Theta(n+l)$ 的时间复杂度做完这道题了。

（如果你已经会矩阵快速幂了就可以跳过以下部分了 2333）

这已经挺快了，但是还远不够快。

考虑以下关系式：

$$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}\begin{bmatrix}f[i-1]\\f[i-2]\end{bmatrix}=\begin{bmatrix}f[i]\\f[i-1]\end{bmatrix}$$

记

$$A=\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}$$

容易得到：

$$A^n\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}f[n]\\f[n-1]\end{bmatrix}$$

由矩阵乘法的结合律，我们可以用快速幂的方法，以 $\Theta(\log n)$ 的时间复杂度求出 A^n。

这样一来，这个题就可以以 $\Theta(l+\log n)$ 的时间复杂度通过了。

P - GukiZ and GukiZiana (551E)

- Solved by ***QwQ***

题意

解法

Q - Vanya and Scales (552C)

- Solved by ***Pantw***

题意

给定 w, m，有 $w^0, w^1, \dots, w^{100}$ 这些质量的砝码，每种仅 1 个。

问能否在天平左侧放一个质量为 m 的物品以及一些砝码，右侧放另一些砝码，使得天平两端平衡。

$2\le w\le 10^9,\,\, 1\le m \le 10^9$

解法

这个题可以考虑大力搜索。

容易想到，当 w^i 远大于 m 时，它不可能参与构成解。

亦即我们可以仅对与 m 数量级相当（或比其小）的砝码搜索。

当 $n=5$ 时，只需搜索到 $5^{13}=1,220,703,125$ 即可。

可见 $n>5$ 时，需要考虑的元素数量不超过 13。

从另一个视角来看，这相当于采用 w 进制凑出 m。每个数码只能是 1 或 -1 或 0。

（这对应三种情况：在右盘，在左盘，未使用）

当 $n=2$ 时，显然我们只用 0 与 1 即可表示足够大范围内的所有数。

当 $n=3$ 时，考虑 k 的三进制表示，将数码为 2 的位改为 -1 并进位即可表示出所有的数。

当 $n=4$ 时，考虑 k 的四进制表示，将数码为 3 的位改为 -1 并进位即可。如果碰见数码为 2 的位说明无解。

那么这其实就是一个比较 general 的 solution。

$n\ge 5$ 时搜索亦可，也可以直接同 $n=4$ 的情况一样处理。

时间复杂度是 $\Theta(m^{\ln 3/\ln w})$ 或 $\Theta(\log_w m)$。

R - Vanya and Triangles (552D)

- Solved by ***Pantw***

题意

平面上 n 个整点 (x_i, y_i)。问构成多少个不同的面积不为 0 的三角形。

$1\le n \le 2000,\,\, -100\le x_i, y_i\le 100$

解法

一看时限 4s。想想这是 Codeforces。就直接莽了

$\binom{n}{3}\le 1331334000\approx 1.3\times 10^9$

~~不会吧不会吧不会真的有人不会判三点不共线吧~~

题解说枚举直线然后把直线拿个 map 存下来，记录出现次数。

这样的话是 $\Theta(n^2\log n)$ 的。

S - Vanya and Brackets (552E)

- Solved by ***Pantw*** & ***Withinlover***

题意

给一个长不超过 5001 的串，串长为奇数，奇数位上都是 0-9 的数字，偶数位上是 + 或 *

最多可以加一个括号，要求最大化表达式值。

乘号的数量不超过 15。

解法

这个题直接猜括号的两侧要么贴着头尾要么贴着乘号。

然后直接用 py 的 eval 就莽过去了 2333。

Code as follows:

```
S = input()
N = len(S)
Ans = eval(S)
for i in range(0, N, 2):
    for j in range(i, N, 2):
        if (i == 0 or (i > 0 and S[i - 1] == '*')) and (j == N - 1 or (j <
N - 1 and S[j + 1] == '*')):
            SS = S[:i] + str(eval(S[i:j+1])) + S[j+1:]
            if int(eval(SS)) > Ans:
                Ans = int(eval(SS))
print(Ans)
```

T - Kyoya and Permutation (553B)

- Solved by ***QwQ***

题意

解法

U - Love Triangles (553C)

- Solved by ***QwQ***

题意

解法

V - Nudist Beach (553D)

- Solved by ***QwQ***

题意

解法

W - Case of Fugitive (555B)

- Solved by ***Withinlover***

题意

无限长的河流上有 n 块石头，每一块用一个区间 $[l, r]$ 表示，你有 m 块木板，每块的长度固定。

你需要依次将着 m 块石头用木板连起来，即1连2，2连3，3连4……

询问是否存在摆放木板的方案，如果有，输出方案。

解法

n 块石头之间需要 $n - 1$ 块木板，对木板长度的要求可以转换成 $n - 1$ 条形如 $[l, r]$ 的限制条件。木板的长度视为单点，就转换成了一个经典的贪心。

X - Case of Chocolate (555C)

- Solved by ***Pantw***

题意

给一个 $n\times n$ 尺寸的上三角方格。

有 q 个操作。每次操作选择一个副对角线上的格子，指定一个方向，从副对角线上的格子开始对这个方向上的格子全部染色，直至碰见边界，或者遇见在之前的查询中染过色的格子为止。

对每个操作需要输出在该次操作中被染色的格子数。

示意图如下（图源题面）：

解法

可以发现，经过一些操作后的剩余区域有以上几种形状。

每进行一次操作，可以看成将一个区域分为了两个这样的区域。

区域可由其右下角的斜边代表。

仔细观察可以发现，左上角的位置和右下角斜线的两个端点可以唯一确定一个这样的形状，右下角的第四种情况是 general case，其他三种均可看成这种形状的特例。

那么我们对每个区域需要保存的信息如下：

- L: 斜线左端点 x 坐标
- R: 斜线右端点 x 坐标
- Lb: left-bound，区域左边界对应的 x 坐标
- Ub: upper-bound，区域上边界对应的 y 坐标

直接包装成一个类，类里封装一个 Split 方法，在其中写出分裂时的行为即可。

每个区域用其右下角的斜线代表，这些斜线可以看成 $[1, n]$ 内的互不相交的闭区间。

我们可以用平衡树维护这样的区间信息，每次操作时取出相应区间将其一分为二，再插入回平衡树即可。

至于具体写法，考虑到 `std::set` 就是优秀的平衡树，我们直接用它来维护即可。

一些关键代码如下。

```
tuple<Interval, Interval, int> Split (int pos, Direction dir) const {
  if(dir == UP)
    return make_tuple(Interval(L, pos - 1, Lb, Ub), Interval(pos + 1, R,
pos + 1, Ub), (N + 1 - pos) - Ub + 1);
  else
    return make_tuple(Interval(L, pos - 1, Lb, N + 1 - pos + 1),
Interval(pos + 1, R, Lb, Ub), pos - Lb + 1);
}
bool operator < (const Interval &b) const {
```

```
  return this->L == b.L ? this->Lb < b.Lb : this->L < b.L;
}


int main() {
  S.insert(Interval(1, n, 1, 1));
  for {
    int x, y;
    char c;
    scanf("%d%d %c", &x, &y, &c);
    Interval itv(x, x, INF, INF);
    auto pos = S.insert(itv);
    auto it = pos.first;
    if(it == S.begin()) puts("0");
    else {
      --it;
      if(it->R < x) puts("0");
      else {
        auto tup = it->Split(x, c == 'U' ? Interval::UP : Interval::LEFT);
        S.erase(itv);
        S.erase(it);
        if(get<0>(tup).valid()) S.insert(get<0>(tup));
        if(get<1>(tup).valid()) S.insert(get<1>(tup));
        printf("%d\n", get<2>(tup));
      }
    }
  }
}
```

Y - Case of a Top Secret (555D)

- Solved by ***Pantw***

题意

平面上有横向一列 n 个钉子，坐标 $x_1, \dots, x_n$。

给出 m 组询问 (a_i, l_i)。询问如下：

初始时在编号为 a_i 的钉子下方用长为 l_i 的绳系一小重物，给予其向右的初速度。

在绳子与其他钉子重合时，绳子将绕过钉子，物体向回旋转。（角速度方向不变，横向速度方向改变）

示意图如下（图源题面）：

钉子视为无限细，所以在钉子上打转不消耗绳长。问最后物体绕哪根单独的钉子不断旋转。

$1\le n, m\le 2\times 10^5$

$-10^9\le x_i\le 10^9$

解法

这个题初步的想法是直接模拟。

先将钉子排序，然后每次根据绳长、当前位置及运动方向在序列上二分下一个绕过的钉子。

这样做的一个最大问题就是时间。

可以很容易地构造出围着两个钉子转 5×10^8 左右次的情况。

这样一来我们必须想办法加速我们的模拟过程。

如上图，假设我们的重物目前在左边的红色钉子，正向右摆，下一次要绕过的钉子是右边的红色钉子。三段长度如图所示标记。

那么当目前绳长 l 满足 $l-2L<l_1$ 时，我们可以直接用 l 除以 L 的余数 r 替代 l并根据 $\lfloor \cfrac{l}{L}\rfloor$ 的奇偶性判断变换后的所在位置。若其为偶数，那么变换后在左侧的钉子；否则在右侧的钉子。

下证这么做是正确的。

首先右侧红色钉是下一次要绕过的钉，那么显然有 $L\le l<L+l_2$

绕回时，绳剩余长度为 $l-L$如果 $L \le l-L < L + l_1$那么下一次将在左侧钉处绕向右。

那么整理一下式子，我们可以将 $l-2L<l_1$ 作为进入在左右侧红色钉之间循环的条件。当 $l<L$ 时，其自然退出这个循环。所以这样做是正确的。

这么做的时间复杂度应该是 $O(m\log n\log \max\limits_{i=1}^{n} |x_i|)$

具体证明目前没有想到比较简洁的证法，如果之后有了大概会补上。

Z - Case of Computer Network (555E)

- Solved by ***QwQ***

题意

解法

Comments

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:mian:cf_mashup:20200730&rev=1596180192

Last update: **2020/07/31 15:23**