


原

曲神的hu测 T2.Van（左偏树+dp） T3.Gay

2018年03月31日 19:42:33

Coco\_T\_

阅读数：173

 版权声明：本文为博主原创文章，未经博主允许不得转载。 [https://blog.csdn.net/wu\\_tongtong/article/details/79771237](https://blog.csdn.net/wu_tongtong/article/details/79771237)

版权属于yhzq，想要引用此题（包括题面）的朋友请联系博主

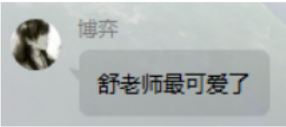
## T2.Van

### 题目背景

博弈退役以后，变了很多。



经过一个寒假的洗礼，他似乎变♂弯♂了♂



### 题目描述

博弈发现了一种很符合他性格的弯♂序♂列♂，一个序列被称作弯序列当且仅当以下两个条件中的一个（或两个）成立：

- 1) 除了首项，所有的奇项都比前项小且所有的偶项都比前项大。
- 2) 除了首项，所有的奇项都比前项大且所有的偶项都比前项小。

当然一个弯序列肯定是满足♂足♂不♂了♂博弈的。他又找到了一种序列叫作k弯曲序列当且仅当它的最长弯子序列（不一定是连续子序列）的长度不超过k。现在有一个序列 $a_1 \dots a_n$ ，每次可以花费1的代价使得a中的某一项增加或减少1。博弈的目的是花费最少的代价让它成为k弯曲序列。

### 输入格式

第一行包含两个正整数，分别表示数列 $a_1 \dots a_n$ 的长度n和k。下面一行有n个自然整数，用空格隔开，表示 $a_i$ 。

### 输出格式

输出一个整数，表示最小代价。

### 输入输出样例

#### 输入样例1

9 3  
1 3 4 7 10 9 7 5 6

[https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

输出样例1

1

样例解释1

把最后一项减小1，得到序列 **1 3 4 7 10 9 7 5 5**，它的最长弯曲子序列之一是 **1 10 5**

输入样例2

见 下发文件\样例\Van\van2.in

输出样例2

见 下发文件\样例\Van\van2.out [https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

数据范围

测试点编号	$n$ 的规模	$k$ 的规模	$a_i$ 的范围
1	$n \leq 20000$	$k = 1$	$a_i \leq 50000$
2	$n \leq 200$	$k = 2$	$a_i \leq 50000$
3	$n \leq 2000$	$k = 2$	$a_i \leq 50000$
4	$n \leq 20000$	$k = 2$	$a_i \leq 50000$
5	$n \leq 20000$	$k = 2$	$a_i \leq 50000$
6	$n \leq 20000$	$k = 2$	$a_i \leq 50000$
7	$n \leq 20000$	$k = 2$	$a_i \leq 50000$
8	$n \leq 20000$	$k = 2$	$a_i \leq 50000$
9	$n \leq 200$	$k = 3$	$a_i \leq 50000$
10	$n \leq 2000$	$k = 3$	$a_i \leq 50000$
11	$n \leq 20000$	$k = 3$	$a_i \leq 50000$
12	$n \leq 20000$	$k = 3$	$a_i \leq 50000$
13	$n \leq 20000$	$k = 3$	$a_i \leq 50000$
14	$n \leq 20000$	$k = 3$	$a_i \leq 50000$
15	$n \leq 20000$	$k = 3$	$a_i \leq 50000$
16	$n \leq 100$	$k \leq 10$	$a_i \leq 50000$
17	$n \leq 1000$	$k \leq 10$	$a_i \leq 50000$
18	$n \leq 1000$	$k \leq 10$	$a_i \leq 50000$
19	$n \leq 1000$	$k \leq 10$	$a_i \leq 50000$
20	$n \leq 1000$	$k \leq 10$	$a_i \leq 50000$

分析：  
这样把博弈黑出翔真的好吗。。。

首先我们要明确怎么计算一个序列的最长Van序列  
这就和花匠那道题有异曲同工之妙了  
我们只需要找到序列的所有折点即可

**k=1**

这种情况下我们只能把序列中的每一个数变成一样的  
显然都变成中位数最优

“中位数定理”的玄学证明

## k=2

这种情况下我们需要使整个序列单调不升或者不降  
于是我就想到了这道题  
mmp, 完全是原题好伐。。。不讲

## k=3

枚举中间折点, 转化成k=2的情况

## k<=10

注意到这个数据的n很小, 那么我们可以预处理使每个区间上升或下降的最小代价  
然后就可以dp了  
 $f[i][j][k]$ 表示第*i*位, 已经有了*j*段, 最后一段是上升还是下降

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <cstring>
4  #define N 20010
5  #define M 1010
6  using namespace std;
7  int tot, a[N], n, k;
8  struct heap {
9      int ch[N][2], dis[N], val[N];
10     int merge(int x, int y) {
11         if (x * y == 0) return x + y;
12         if (val[x] < val[y]) swap(x, y);
13         ch[x][1] = merge(ch[x][1], y);
14         if (dis[ch[x][0]] < dis[ch[x][1]])
15             swap(ch[x][0], ch[x][1]);
16         dis[x] = dis[ch[x][1]] + 1;
17         return x;
18     }
19     int pop(int x) {
20         return merge(ch[x][0], ch[x][1]);
21     }
22     void init(int x, int v) {
23         ch[x][0] = ch[x][1] = dis[x] = 0;
24         val[x] = v;
25     }
26 } h;
27 struct seqs {
28     int l, r, root, insum, innum, sum;
29     int cost() {
30         int v = h.val[root];
31         return (v * innum - insum) + (sum - insum - v * (r - l + 1 - innum));
32     }
33     void merge(seqs now) { //now must behind this
34         r = now.r, root = h.merge(root, now.root);
35         innum += now.innum, insum += now.insum, sum += now.sum;
36         while (innum * 2 > r - l + 2)
37             innum--, insum -= h.val[root],
38             root = h.pop(root);
39     }
40 } seq[N];
41 void add(int pos, int v, int &cost) {
42     seq[++tot].root = seq[tot].l = seq[tot].r = pos;
43     seq[tot].innum = 1, seq[tot].insum = seq[tot].sum = v;
44     h.init(pos, v), cost += seq[tot].cost();
45     while (tot > 1 && h.val[seq[tot].root] < h.val[seq[tot - 1].root]) {
46         cost -= seq[tot].cost() + seq[tot - 1].cost();
47         seq[tot - 1].merge(seq[tot]), tot--;
48         cost += seq[tot].cost();
49     }
50 }
```

```
51 void solve_1() {
52     sort(a + 1, a + 1 + n);
53     int ans = 0, mid = a[n / 2];
54     for (int i = 1; i <= n; i++) ans += abs(a[i] - mid);
55     printf("%d\n", ans);
56 }
57 void solve_2() {
58     int ans = 0x3f3f3f3f, cost;
59     cost = 0, tot = 0;
60     for (int i = 1; i <= n; i++) add(i, a[i], cost);
61     ans = min(ans, cost);
62     cost = 0, tot = 0;
63     for (int i = 1; i <= n; i++) add(i, -a[i], cost);
64     ans = min(ans, cost);
65     printf("%d\n", ans);
66 }
67 void solve_3() {
68     static int fwd1[N], fwd2[N], bac1[N], bac2[N];
69     int ans = 0x3f3f3f3f, cost;
70     cost = tot = 0;
71     for (int i = 1; i <= n; i++) add(i, a[i], cost), fwd1[i] = cost;
72     cost = tot = 0;
73     for (int i = 1; i <= n; i++) add(i, -a[i], cost), fwd2[i] = cost;
74     cost = tot = 0;
75     for (int i = n; i >= 1; i--) add(n - i + 1, a[i], cost), bac1[i] = cost;
76     cost = tot = 0;
77     for (int i = n; i >= 1; i--) add(n - i + 1, -a[i], cost), bac2[i] = cost;
78     for (int i = 1; i <= n + 1; i++)
79         ans = min(ans, fwd1[i - 1] + bac1[i]),
80         ans = min(ans, fwd2[i - 1] + bac2[i]);
81     printf("%d\n", ans);
82 }
83 void solve_4() {
84     k--;
85     static int fwd[M][M], bac[M][M], f[M][15], g[M][15];
86     for (int i = 1; i <= n; i++) {
87         int cost = 0; tot = 0;
88         for (int j = i; j <= n; j++) add(j, a[j], cost), fwd[i][j] = cost;
89         cost = 0, tot = 0;
90         for (int j = i; j <= n; j++) add(j, -a[j], cost), bac[i][j] = cost;
91     }
92     memset(f, 0x3f, sizeof f), memset(g, 0x3f, sizeof g);
93     f[0][0] = g[0][0] = 0;
94     for (int i = 1; i <= n; i++)
95         for (int j = 1; j <= k; j++)
96             for (int l = 0; l < i; l++)
97                 f[i][j] = min(f[i][j], g[l][j - 1] + fwd[l + 1][i]),
98                 g[i][j] = min(g[i][j], f[l][j - 1] + bac[l + 1][i]);
99     int ans = 0x3f3f3f3f;
100     for (int i = 1; i <= k; i++) ans = min(ans, f[n][i]);
101     for (int i = 1; i <= k; i++) ans = min(ans, g[n][i]);
102     printf("%d\n", ans);
103 }
104 main() {
105     freopen("van.in", "r", stdin);
106     freopen("van.out", "w", stdout);
107     scanf("%d%d", &n, &k);
108     for (int i = 1; i <= n; i++)
109         scanf("%d", &a[i]);
110     if (k == 1) solve_1();
111     else if (k == 2) solve_2();
112     else if (k == 3) solve_3();
113     else solve_4();
114 }
```

## T3.Gay

## 题目描述

外星人的星球上共有 $2^n - 1$ 个住宅区，每个住宅区住着 $n$ 户人家，门牌号分别为 $1, 2, \dots, n$ ，每户人家家里养着一个博弈。这时恰好是博弈变 $\delta$ 化 $\delta$ 的时间(具体见第二题)。每个住宅区都有至少一个博弈发生了病变。一个住宅区中，门牌号为 $u$ 的人家的博弈要么病变，要么不病变，一共 $2^n$ 种情况，再去掉都没病变的情况，一共 $2^n - 1$ 种。每种情况都会发生在恰好一个住宅区中。

这天外星人的每个住宅区都放出了一个爆炸性的新闻："你们这个住宅区里至少有一个博弈发生了病 $\delta$ 变 $\delta$ ！"

每个住宅区中每户人家都不知道自己的博弈到底是病变了还是正常，但是他能一眼看出某些人家的博弈有没有病变。由于这个社会里人与人之间的信任已经崩塌，一个人即使看出别人的博弈是否病变了也不愿告诉他。

可以用一个 $n$ 个结点的有向图来描述可见性， $v$ 到 $u$ 有一条有向边表示门牌号为 $v$ 的人家能看出门牌号为 $u$ 的家里的博弈是否病变了，没边则表示看不出。**每个人都知道这张有向图。**

那么一个残酷的逻辑链条就开始启动。对于每个住宅区：

1. 第一天，早上每户人家的主人会出门看看别人家的博弈，如果一个人能推断出自己家的博弈发生了病变，下午6点整，他就会掏出超离子垃圾处理器把他家的博弈熔化了。
2. 如果有多个人都在同一天推断出了，那么他们会在下午6点整同时熔化博弈。
3. 因为使用超离子垃圾处理器会发出巨大的噪音，所以每个人都能听到这个住宅区里的处理器使用情况。如果没听到，这个住宅区里的人第二天会继续早上出门看别人家的博弈，推断出自己家的博弈病变了下午就熔掉。如果还没有听到，第三天也会如此，依次类推。（所以如果一个人听到了噪音那么就不会再熔化自己的博弈）

作为一个外星博弈保护协会的人，想帮助当地居民调节矛盾的你想要向外星人们展示灾难性的后果，请计算出对于所有前 $2^{333}$ 天内有人使用过处理器的住宅区：

1. 熔化的时间之和。如一个住宅区在第 $k$ 天下午熔化了博弈，则熔化时间为 $k$ 。（多个人同时熔化只算一次）
2. 死亡的博弈的总数。

你只用输出对 $998244353$ 取模后的结果。

[https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

## 输入格式

第一行一个整数 $n$ ，含义如前所述。

接下来 $n$ 行每行 $n$ 个字符。这 $n$ 行中的第 $v$ 行第 $u$ 个字符为"1"表示 $v$ 能看出 $u$ 家的博弈是否病变了，如果字符为"0"表示看不出。保证只会出现"0"和"1"这两种字符，且对于任意一个满足 $1 \leq v \leq n$ 的 $v$ ，第 $v$ 行第 $v$ 列为"0"。

## 输出格式

一行输出两个整数分别表示熔化时间之和、死亡的博弈的总数。

## 输入输出样例

### 输入样例1

```
2
01
00
```

### 输出样例1

```
5 3
```

### 样例解释1

门牌号为1的人能看见门牌号为2的人家里的博弈是不是病变了。

共有三个住宅区。

1. 1病了，2没病。第一天1发现2没得病，又知道他们中肯定有一个病了，所以第一天下午熔化了自己的博弈。熔化时间为1，死了1个博弈。
2. 1没病，2病了。第一天1发现2得了病，于是第一天两个人什么也没干。第二天2发现前一天1没有熔化，所以下午熔化了自己的博弈。熔化时间为2，死了1个博弈。
3. 1病了，2病了。2还是第二天熔化了博弈，1始终没有熔化。熔化时间为2，死了1个博弈。

所以 $1 + 2 + 2 = 5$ ， $1 + 1 + 1 = 3$ 。

[https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

输入样例2

2  
01  
10

输出样例2

4 4

样例解释2

懒得写了。。(加上样例2只是让排版更齐一点)

输入样例3

见 下发文件\样例\Gay\gay3.in

输出样例3

见 下发文件\样例\Gay\gay3.out [https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

数据范围

测试点编号	$n$ 的规模	其它限制
1	$n \leq 8$	无
2	$n \leq 8$	无
3	$n \leq 20$	无
4	$n \leq 20$	无
5	$n \leq 100$	无
6	$n \leq 100$	无
7	$n \leq 100$	每户人家都能看出其他每户人家的博弈有没有病变
8	$n \leq 3000$	每户人家都能看出其他每户人家的博弈有没有病变
9	$n \leq 3000$	无
10	$n \leq 3000$	无

注意：本题有 *spj*，对于每一个测试点，假如你答出了其中一问，你就会获得一半的分数。注意假如你要得出了第二问，需要随便输出一个第一个数才能得到分数。[https://blog.csdn.net/wu\\_tongtong](https://blog.csdn.net/wu_tongtong)

分析：  
题目来源：33IQ  
曲神表示：啊~原题是狗，我改成了博弈，没什么太大区别嘛  
。。。博弈被黑的最惨的一次。。。

算法一

最简单的还是完全图的20分

如果第一天，有人走出去看到其他所有人的博弈都正常，那么就可以确定ta的博弈病变了，于是ta就会怀着沉重的心情用炒栗子超离子垃圾处理器把ta  
如果第一天，有人走出去看见一家的博弈病变了，但是到了这一天没有人去融掉博弈，说明这个住宅区中一定还存在一个病变的博弈，就是自己的那乐位？那坨？  
反正第二天两个人会同时怀着沉重的心情用炒栗子超离子垃圾处理器把博弈融掉（嘶~~）

以此类推

因此有 $x$ 个病变博弈的社区需要 $x$ 天才能确定下来，进行融化  
因此两问答案都是 $\sum_{i=1}^n C(n, i) * i$

## 算法二

上一个算法也还是挺有启发性的，我们发现我们在分析自己的博弈有没有病变的时候的方法是：**假设没有病变**，然后看前一天是否应该有熔化。而这两个都适用的。

考虑状压dp，令 $dp_U$ 表示 $U$ 集合中的博弈病变时的熔化时间

然后我们接着分析：如果我的博弈没病，那么什么时候应该熔化呢？

于是枚举了所有可能的病变情况 $V$ ，那么如果他的博弈没有病变，最迟应该在

$\max dp_V$ 的时间有人熔化自己的博弈

所以他会在第 $\max dp_V + 1$ 天熔化

所以我们就知道怎么DP了！

对于一个状态，我们枚举这个状态中每个病变的博弈的主人，那么这个状态的熔化时间应该是这些病变的博弈主人中熔化时间的最小值，从中我们也可人同时熔化

那么一个人会枚举哪些可能的病变情况呢：首先对于他看到的博弈，他枚举的病变情况一定是和当前情况相同的，其次他自己一定是没有病变的，而至于到的博弈，他就只好枚举这些博弈的所有病变状况了

最后，如果有限时间内不会熔化怎么办？事实上是存在这种情况的，但是没关系，如果我们发现转移出现了环，所以这个环中的所有状态都是在有限时的

时间复杂度似乎是 $O(4^n n)$ ？

可以通过前两个测试点

## 算法三

有了算法二，我们可以把表打出来看看，然后就可以发现一个小小的规律：

对于病变状态 $U$ 和 $V$ ，如果 $U \subseteq V$ ，那么必然有 $dp_U \leq dp_V$

至于这一个规律的证明放在后面

所以就可以只枚举一种病变情况：首先对于所有他看到的博弈，他枚举的病变情况一定是和当前情况相同的，其次他自己一定是没有病变的，而至于那博弈，由刚才的规律，可以全部当成病变的

所以时间复杂度就可以降到 $O(2^n n)$ ，可以通过前四个测试点

## 算法四

为了取得更多的分，我们需要一个多项式算法

我们重新来考虑有限时间内熔化的条件：**转移无环**

实际上这是一个很强的条件

根据转移建一个新图，如果 $i$ 不能看到 $j$ ，那么就在 $i$ 到 $j$ 连条有向边

这个新图中可能有若干个强连通分量，如果一个病变状态中有病变的博弈的主人是在一个多于一个点的强连通分量中，那么这一个状态一定无法在有限（转移有环），否则一定能在有限时间内熔化（转移无环）

那么我们先把所有可以到达大小大于1的强连通分量的点连同和他们相关的边一起删掉，于是我们得到了一个DAG！

接着我们继续站在这张图的立场上来想怎么求一个状态的熔化时间：

首先这张图上有一些点被染黑了，黑点表示这个博弈有病，白点表示这个博弈没病

接着每一时刻，我们可以把一个黑点染白，然后把这个点连向的点的集合的一个子集染黑

若干轮之后，所有点都变白了

假设 $k$ 个点一度被染黑，那么熔化时间就是所有操作方案中最大的 $k$

看起来这个转化比较意识流，但是如果考虑怎么DP所有操作方案中最大的 $k$ ，我们会发现这个DP方法和算法二是等价的

如果再要解释的话，首先枚举博弈主人相当于枚举反转的黑点，算博弈主人熔化时间中的max体现在求最大的一度被染黑的点数，算状态的熔化时间中熔化时间取min体现在多次被染黑的点只被计算一次

而在这个模型下，算法三的小规律就很显然了

增加黑点对答案的贡献一定非负，所以问题又变成了：

一个DAG上有一些点被染黑了，接着每一时刻，我们可以把一个黑点染白，然后把这个点连向的点全部染黑。若干轮之后，所有点都变白了

假设有 $k$ 个点一度被染黑，求所有操作方案中最大的 $k$

这不就是求DAG上一个点集能直接或者间接到达的点数吗？

所以我们将判断一个病变状态在第几天熔化转化为了求DAG上一个集合能直接或间接到达的点数。这样第一问已经非常简单了。考虑算每一个点的贡献点能被 $i$ 个点到达（包括自己），而DAG上一共有 $N$ 个点，那么显然这个点的贡献就是 $(2^i - 1) \times 2^{N-i}$



接下来考虑第二问，一个人要在DAG上满足什么样的条件才会在第一天熔化呢？

在原来的DP中，我们是在取min的过程中得到同时熔化的人数的，而我们知道，在现在的模型中，算状态的熔化时间中对博弈主人熔化时间取min体现黑的点只被计算一次。

所以我们脑补一下可以得到：如果一个点第一个反转，且之后得到的状态无论怎么操作都无法再把这个点染黑，那么这个点对应着一个最早熔化的博弈

所以第二问也可以很简单的解决了，还是考虑算每一个点的贡献，如果一个点能被 $i$ 个点到达（包括自己），而DAG上一共有 $N$ 个点，那么显然这个点的贡献是 $2^{N-i}$

因为计算DAG中每一个点可以被多少点到达是 $O(nm)$ 的，而在这个DAG中 $m$ 是 $O(n^2)$ 的，所以时间复杂度是 $O(n^3)$ ，可以通过前六个点。

## 算法五

最后的技术含量就很低了，这个问题显然是可以压位的嘛，所以可以把时间复杂度搞到 $O(\frac{n^3}{32})$ ，可以通过所有测试点

```
1  #include <cstdio>
2  #include <bitset>
3  using namespace std;
4  typedef long long ll;
5  const int N = 3030;
6  const int mod = 998244353;
7  bitset <N> b[N];
8  int d[N], que[N], g[N][N], n, m, t;
9  int pw2[N], ans1, ans2;
10 char s[N];
11 main() {
12     freopen("gay.in", "r", stdin);
13     freopen("gay.out", "w", stdout);
14     scanf("%d", &n);
15     for (int i = 1; i <= n; ++i) {
16         scanf("%s", s + 1), s[i] = '1';
17         for (int j = 1; j <= n; ++j)
18             if (s[j] == '0')
19                 d[i]++, g[i][j] = 1;
20     }
21     pw2[0] = 1;
22     for (int i = 1; i <= n; ++i) pw2[i] = (pw2[i - 1] << 1) % mod;
23     for (int i = 1; i <= n; ++i) if (!d[i]) que[++t] = i;
24     for (int i = 1; i <= t; ++i)
25         for (int j = 1, now = que[i]; j <= n; ++j)
26             if (g[j][now] && !--d[j]) que[++t] = j;
27     for (int i = t; i; --i) {
28         int now = que[i];
29         b[now][now] = 1;
30         int cnt = b[now].count();
31         ans1 = (ans1 + 1ll * (pw2[cnt] - 1) * pw2[t - cnt]) % mod;
32         ans2 = (ans2 + pw2[t - cnt]) % mod;
33         for (int j = 1; j <= n; ++j)
34             if (g[now][j]) b[j] |= b[now];
35     }
36     printf("%d %d\n", ans1, ans2);
37 }
```



想对作者说点什么