

wjw 的旅馆plus

题解

很明显的区间询问和修改问题

修改操作就是区间赋值，将一个区间的数字全部赋值为某个数

询问操作就是询问区间内有多少个不同的数字

那么显然比较麻烦的就是询问，观察数据范围可以发现， $T \leq 30$ ，很明显是可以直接状压的

那么区间维护问题就解决了，直接 $sumv[id]$ 表示区间内每个数字的出现情况，状态压缩即可

但是注意这道题还有一个坑，那就是空房间 0 也当做是一个旅行社，这一步在 *lazy* 判断时需要注意，如果 *lazy* 初始化为 0 则会发生错误

这里比较简单的做法就是将 *lazy* 初始化为 -1，并且在 *pushdown* 中判断

也可以将房间 0 直接 +1 当成 1 来算，后面输入的所有旅行社编号 +1 即可，但是这样的话需要考虑到 $T + 1$ 后会达到 31， $1 \ll 31$ 会爆 *int*，所以要开 *long long*

wjw 的打字计划

题解

一看数据就知道可以搜，但是暴力搜索有 9^9 直接 *TLE* 了

但是这道题是没有办法进行剪枝或者优化的，那么其实虽然是搜索，但本质是枚举

枚举所有方案有个优化，叫折半枚举，标准的空间换时间的思路

枚举前 5 个数字会导致的所有局面

再枚举后 5 个数字会导致的所有局面

上下匹配就可以将复杂度降低到 $2 * (9^5)$

当然这道题还有第二种做法

7 4 0 1 2 3 观察每个数字占位的情况可以发现，按这个顺序可以直接将这几个数字的次数给计算出来

接下来枚举剩下的几个数字，然后解个方程就可以了，具体的可以看第二份标程，非常直观

标程

```
1 // 折半枚举
2 #include<bits/stdc++.h>
3 #define mod 1000000007
4 #define m1 998244353
5 #define m2 469762049
6 using namespace std;
7 long long ans=0;
8 char ch[5][45]=
9 {
```

```

10 "...1222233334..4555566667777888899990000",
11 "...1...2...34..45...6.....78..89..90..0",
12 "...122223333444455556666...7888899990..0",
13 "...12.....3...4...56..6...78..8...90..0",
14 "...122223333...455556666...7888899990000",
15 };
16 int arr[6][6];
17 map<pair<int,int>,string> num;
18 string best="999999999";
19 void init()
20 {
21     ios::sync_with_stdio(false);
22     for (int i=0;i<=4;i++)
23         for (int j=0;j<=3;j++)
24             cin>>arr[i][j];
25 }
26
27 void dfs1(int now,string s)
28 {
29     if (now==5)
30     {
31         long long hash1,hash2;
32         hash1=hash2=0;
33         for (int i=0;i<=4;i++)
34             for (int j=0;j<=3;j++)
35             {
36                 hash1=(hash1*101+arr[i][j])%m1;
37                 hash2=(hash2*101+arr[i][j])%m2;
38             }
39         if (num[make_pair(hash1,hash2)]=="") num[make_pair(hash1,hash2)]=s;
40         num[make_pair(hash1,hash2)]=min(num[make_pair(hash1,hash2)],s);
41         return;
42     }
43     for (int k=0;k<=9;k++)
44     {
45         int flag=0;
46         for (int i=0;i<=4;i++)
47             for (int j=0;j<=3;j++)
48                 if (ch[i][now*4+j]!='.')
49                 {
50                     arr[i][j]-=k;
51                     if (arr[i][j]<0) flag=1;
52                 }
53         if (!flag) dfs1(now+1,s+(char)('0'+k));
54         for (int i=0;i<=4;i++)
55             for (int j=0;j<=3;j++)
56                 if (ch[i][now*4+j]!='.')
57                 {
58                     arr[i][j]+=k;
59                 }
60         if (flag) break;
61     }
62     return;
63 }
64
65 void dfs2(int now,string s)

```

```

66 {
67     if (now==10)
68     {
69         long long hash1,hash2;
70         hash1=hash2=0;
71         for (int i=0;i<=4;i++)
72             for (int j=0;j<=3;j++)
73             {
74                 hash1=(hash1*101+arr[i][j])%m1;
75                 hash2=(hash2*101+arr[i][j])%m2;
76             }
77         if (num[make_pair(hash1,hash2)]=="") return;
78         best=min(best,num[make_pair(hash1,hash2)]+s);
79         return;
80     }
81     for (int k=0;k<=9;k++)
82     {
83         for (int i=0;i<=4;i++)
84             for (int j=0;j<=3;j++)
85                 if (ch[i][now*4+j]!='.')
86                 {
87                     arr[i][j]+=k;
88                 }
89         dfs2(now+1,s+(char)('0'+k));
90         for (int i=0;i<=4;i++)
91             for (int j=0;j<=3;j++)
92                 if (ch[i][now*4+j]!='.')
93                 {
94                     arr[i][j]-=k;
95                 }
96     }
97     return;
98 }
99
100 void work()
101 {
102     dfs1(0,"");
103     memset(arr,0,sizeof(arr));
104     dfs2(5,"");
105     cout<<best<<endl;
106 }
107 int main()
108 {
109     init();
110     work();
111 }
112

```

```

1 // 解方程
2
3 #include<bits/stdc++.h>
4 #define INF 0x3f3f3f3f
5 #define eps 1e-8
6 #define re register
7 #define LL long long

```

```

8  #define MOD 1000000007
9  #define MAXN 10
10 using namespace std;
11 int n=5,m=4,a[MAXN][MAXN],num[MAXN];
12 int main(){
13     for(int i=1;i<=n;i++)
14         for(int j=1;j<=m;j++)
15             scanf("%d",&a[i][j]);
16     while(a[1][2]>a[5][2]){
17         a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[2][4]--,a[3][4]--,a[4][4]-
18 -,a[5][4]--;
19         num[7]++;
20     }
21     while(a[1][1]>a[1][2]){
22         a[1][1]--,a[2][1]--,a[3][1]--,a[3][2]--,a[3][3]--,a[1][4]--,a[2][4]-
23 -,a[3][4]--,a[4][4]--,a[5][4]--;
24         num[4]++;
25     }
26     while(a[5][2]>a[3][2]){
27         a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[5][1]--,a[5][2]--,a[5][3]-
28 -,a[5][4]--;
29         a[2][1]--,a[3][1]--,a[4][1]--,a[2][4]--,a[3][4]--,a[4][4]--;
30         num[0]++;
31     }
32     while(a[3][4]>a[3][2]){
33         a[1][4]--,a[2][4]--,a[3][4]--,a[4][4]--,a[5][4]--;
34         num[1]++;
35     }
36     while(a[5][2]>a[4][4]){
37         a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[3][1]--,a[3][2]--,a[3][3]-
38 -,a[3][4]--,a[5][1]--,a[5][2]--,a[5][3]--,a[5][4]--;
39         a[2][4]--,a[4][1]--;
40         num[2]++;
41     }
42     while(a[1][1]>a[2][1]){
43         a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[3][1]--,a[3][2]--,a[3][3]-
44 -,a[3][4]--,a[5][1]--,a[5][2]--,a[5][3]--,a[5][4]--;
45         a[2][4]--,a[4][4]--;
46         num[3]++;
47     }
48     for(int fiv=0;fiv<=9;fiv++){
49         for(int eigh=min(a[4][1],a[2][4]);eigh>=0;eigh--){
50             int six=a[4][1]-eigh,nin=a[2][4]-eigh;
51             if(fiv+eigh+six+nin!=a[5][2] || six>9 || nin>9)continue;
52             printf("%d%d%d%d%d%d%d%d\n",num[1],num[2],num[3],num[4],fiv,six,num[7],eigh,nin,num[0]);
53             return 0;
54         }
55     }
56     return 0;
57 }

```

wjw 的湖泊建造

题解

容易看出来，如果 n 是奇数，最优搭建策略是一个三角形，如果 n 是偶数，最优搭建策略是一个底部为 2 的三角形。

所以直接对于给定的 n ，计算最多能容纳多少水，如果够用则直接输出答案，如果不够用需要二分一下，或者直接 $O(1)$ 计算答案。

注意一个坑是卡了 `long long`

一个 hack 点是 $n = 8 \times 10^9$ ，用 `long long` 会爆掉，需要用 `u11` 以及合适的写法（先除后乘）来避免爆掉
