# 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 $31$，$1<<31$ 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

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

```
"...1222233334..455556666777788899990000",
"...1...2...34..45...6......78..89..90..0",
"...122223333444455556666...7888899990..0",
"...12......3...4...56..6...78..8...90..0",
"...122223333...455556666...7888899990000",
};
int arr[6][6];
map<pair<int,int>,string> num;
string best="9999999999";
void init()
{
    ios::sync_with_stdio(false);
    for (int i=0;i<=4;i++)
        for (int j=0;j<=3;j++)
            cin>>arr[i][j];
}

void dfs1(int now,string s)
{
    if (now==5)
    {
        long long hash1,hash2;
        hash1=hash2=0;
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            {
                hash1=(hash1*101+arr[i][j])%m1;
                hash2=(hash2*101+arr[i][j])%m2;
            }
        if (num[make_pair(hash1,hash2)]=="") num[make_pair(hash1,hash2)]=s;
        num[make_pair(hash1,hash2)]=min(num[make_pair(hash1,hash2)],s);
        return;
    }
    for (int k=0;k<=9;k++)
    {
        int flag=0;
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            if (ch[i][now*4+j]!='.')
            {
                arr[i][j]-=k;
                if (arr[i][j]<0) flag=1;
            }
        if (!flag) dfs1(now+1,s+(char)('0'+k));
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            if (ch[i][now*4+j]!='.')
            {
                arr[i][j]+=k;
            }
        if (flag) break;
    }
    return;
}

void dfs2(int now,string s)
```

```
{
    if (now==10)
    {
        long long hash1,hash2;
        hash1=hash2=0;
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            {
                hash1=(hash1*101+arr[i][j])%m1;
                hash2=(hash2*101+arr[i][j])%m2;
            }
        if (num[make_pair(hash1,hash2)]=="") return;
        best=min(best,num[make_pair(hash1,hash2)]+s);
        return;
    }
    for (int k=0;k<=9;k++)
    {
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            if (ch[i][now*4+j]!='.')
            {
                arr[i][j]+=k;
            }
        dfs2(now+1,s+(char)('0'+k));
        for (int i=0;i<=4;i++)
            for (int j=0;j<=3;j++)
            if (ch[i][now*4+j]!='.')
            {
                arr[i][j]-=k;
            }
    }
    return;
}

void work()
{
    dfs1(0,"");
    memset(arr,0,sizeof(arr));
    dfs2(5,"");
    cout<<best<<endl;
}
int main()
{
    init();
    work();
}

```

```
// 解方程

#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define eps 1e-8
#define re register
#define LL long long
```

```cpp
#define MOD 1000000007ll
#define MAXN 10
using namespace std;
int n=5,m=4,a[MAXN][MAXN],num[MAXN];
int main(){
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            scanf("%d", &a[i][j]);
    while(a[1][2]>a[5][2]){
        a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[2][4]--,a[3][4]--,a[4][4]--,a[5][4]--;
        num[7]++;
    }
    while(a[1][1]>a[1][2]){
        a[1][1]--,a[2][1]--,a[3][1]--,a[3][2]--,a[3][3]--,a[1][4]--,a[2][4]--,a[3][4]--,a[4][4]--,a[5][4]--;
        num[4]++;
    }
    while(a[5][2]>a[3][2]){
        a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[5][1]--,a[5][2]--,a[5][3]--,a[5][4]--;
        a[2][1]--,a[3][1]--,a[4][1]--,a[2][4]--,a[3][4]--,a[4][4]--;
        num[0]++;
    }
    while(a[3][4]>a[3][2]){
        a[1][4]--,a[2][4]--,a[3][4]--,a[4][4]--,a[5][4]--;
        num[1]++;
    }
    while(a[5][2]>a[4][4]){
        a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[3][1]--,a[3][2]--,a[3][3]--,a[3][4]--,a[5][1]--,a[5][2]--,a[5][3]--,a[5][4]--;
        a[2][4]--,a[4][1]--;
        num[2]++;
    }
    while(a[1][1]>a[2][1]){
        a[1][1]--,a[1][2]--,a[1][3]--,a[1][4]--,a[3][1]--,a[3][2]--,a[3][3]--,a[3][4]--,a[5][1]--,a[5][2]--,a[5][3]--,a[5][4]--;
        a[2][4]--,a[4][4]--;
        num[3]++;
    }
    for(int fiv=0;fiv<=9;fiv++){
        for(int eigh=min(a[4][1],a[2][4]);eigh>=0;eigh--){
            int six=a[4][1]-eigh,nin=a[2][4]-eigh;
            if(fiv+eigh+six+nin!=a[5][2] || six>9 || nin>9)continue;

printf("%d%d%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]);
            return 0;
        }
    }
    return 0;
}
```

# 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`ull`以及合适的写法（先除后乘）来避免爆掉

---