知识精炼（二）

主讲人：邓哲也

BZOJ 3939 Cow Hopscotch

奶牛们发明了一种新的跳格子游戏。

游戏在一个 R * C 的网格上进行，每个格子有一个取值在 1-K 之间的整数标号。奶牛开始在左上角的格子，目的是通过若干次跳跃后到达右下角的格子。

只有满足以下三个条件时，奶牛才可以从 A 跳到 B。

1. B 在 A 的右方
2. B 在 A 的下方
3. B 的标号和 A 的标号不同

求从左上角的格子到右下角的格子一共有几种不同的方案。

$2 \leqslant R,C \leqslant 750,\ 1 \leqslant K \leqslant R * C$

BZOJ 3939 Cow Hopscotch

样例输入：

4 4 4

1 1 1 1

1 3 2 1

1 2 4 1

1 1 1 1

样例输出：

5

BZOJ 3939 Cow Hopscotch

一个显然的思路，f[i][j] 表示从左上角 (1, 1) 跳到 (i, j) 的方案数。

$f[i][j] = \sum f[x][y]$ $(x<i,\ y<j,\ id[x][y]!=id[i][j])$

是一个前缀和的形式。

我们只要枚举第一维，然后第二维维护一个 $g[y]=\sum f[x][k]$ $(x<i, k\leqslant y)$

BZOJ 3939 Cow Hopscotch

但是还没有限制只有 id[x][y] != id[i][j] 才能转移。

只需要对每种标号都维护一个前缀和。

然后把 id[x][y] == id[i][j] 转移过来的 f 之和从答案里剪掉即可。

BZOJ 3939 Cow Hopscotch

由于一共有 R * C 种标号，每一个都需要维护一个前缀和数组，朴素的维护会超时。

考虑到总共只有 R * C 个位置，我们只要对每种标号维护一颗动态开节点的线段树，就可以在 O(log C) 的时间内查询和维护前缀和了。

BZOJ 3939 Cow Hopscotch

```
struct segnode{
	int lc, rc, sum;
}t[10000000];
int tot, rt[N * N];
void upd(int x){
	t[x].sum = (t[t[x].lc].sum + t[t[x].rc].sum) %
mo;
}
```

BZOJ 3939 Cow Hopscotch

```
void modify(int &x, int p, int v, int l, int r){
    if(!x) x = ++ tot;
    if(l == r){
        t[x].sum = (t[x].sum + v) % mo;
        return;
    }
    int mid = (l + r) >> 1;
    if(p <= mid) modify(t[x].lc, p, v, l, mid);
    else modify(t[x].rc, p, v, mid + 1, r);
    upd(x);
}
```

BZOJ 3939 Cow Hopscotch

```
int ask(int x, int ql, int qr, int l, int r){
    if(ql > qr || !x) return 0;
    if(ql <= l && r <= qr) return t[x].sum;
    int mid = (l + r) >> 1, ret = 0;
    if(ql <= mid) ret = (ret + ask(t[x].lc, ql, qr, l,
    mid)) % mo;
    if(mid < qr)  ret = (ret + ask(t[x].rc, ql, qr,
    mid + 1, r)) % mo;
    return ret;
}
```

BZOJ 3939 Cow Hopscotch

```
scanf("%d%d%d", &n, &m, &K);
for(int i = 1;i <= n;i ++) for(int j = 1;j <= m;j ++)
scanf("%d", &a[i][j]);
f[1][1] = 1;
modify(rt[a[1][1]], 1, 1, 1, m);
modify(rt[0], 1, 1, 1, m);
for(int i = 2;i <= n;i ++){
    for(int j = 2;j <= m;j ++){
        f[i][j] = (ask(rt[0], 1, j - 1, 1, m) -
        ask(rt[a[i][j]], 1, j - 1, 1, m) + mo) % mo;
        f[i][j] = (f[i][j] + mo) % mo;
    }
    for(int j = 2;j <= m;j ++) modify(rt[a[i][j]], j, f[i][j],
    1, m), modify(rt[0], j, f[i][j], 1, m);
}
printf("%d\n", f[n][m]);
```

下节课再见