K. Walking

题意

给定一个 $n\times m$ 的网格和初始点 (a,b)，求从初始点出发移动 t 步且始终不出界的情况下的所有走法。

题解

显然横轴坐标是独立的，可以分开考虑。

设 $f(s,n,a)$ 表示从一维坐标轴从 a 点出发走 s 步且始终处于 $[1,n]$ 范围内的情况下的所有走法。于是答案为

$$ \sum_{i=0}^t {t\choose i}f(i,n,a)f(t-i,m,b) $$

接下来考虑如何计算 $f(s,n,a)(s\in [0,t])$，$f(s,m,b)$ 的计算方式类同。

方案一：设 $\text{dp}(i,j)$ 表示走 i 步最后位于 j 点且始终为出界的方案数，不难得到一个 $O(nt)$ 的暴力解法。

方案二：不难发现，有

$$ \begin{equation}\begin{split} f(s,n,a)&=\sum_{i=1}^n \text{dp}(s,i) \\ &=\text{dp}(s-1,1)+\sum_{i=2}^{n-1} (\text{dp}(s-1,i-1)+\text{dp}(s-1,i+1))+\text{dp}(s-1,n)\\ &=2f(s-1,n,a)-\text{dp}(s-1,1)-\text{dp}(s-1,n) \end{split}\end{equation} $$ 于是问题转化为计算 $\text{dp}(s,1),\text{dp}(s,n)$。

对每个移动方案，定义非法序列，每当点进入 $(-\infty,0)$ 时非法序列末尾加上 L，每当点进入 $(n,+\infty)$ 时非法序列末尾加上 R。

对于 $\text{dp}(s,1)$，我们需要获得所有非法序列为空串的移动方案。设 $h(a,b,s)$ 表示从 a 移动 s 步到达 b 的方案数。

显然根据 a,b,s 奇偶性以及预处理组合数可以 $O(1)$ 计算 $h(a,b,s)$。

然后总移动方案为 $h(a,1,s)$，利用容斥，首先我们减去非法序列为 `L+.*` 和 `R+.*` (非法序列均用正则表达式表示)的移动方案。

设 $l(x)=-x,r(x)=2(n+1)-x$，根据简单组合数学知识，不难发现 `L+.*` 代表的方案为 $h(a,l(1),s)$，`R+.*` 代表的方案为 $h(a,r(1),s)$。

接下来我们需要补上减去非法序列为 `L+R+.*` 和 `R+L+.*` 的移动方案，分别为 $h(a,r(l(1)),s),h(a,l(r(1)),s)$，依次类推，有

$$ \text{dp}(s,1)=h(a,1,s)-h(a,l(1),s)-h(a,r(1),s)+h(a,r(l(1)),s)+h(a,l(r(1)),s)-h(a,l(r(l(1))),s)-h(a,r(l(r(1))),s)+\cdots $$

由于 $r(l(x))=2(n+1)+x$，且当 $\text{abs}(a-x)\gt s$ 时一定有 $h(a,x,s)=0$，所以上述容斥最多迭代 $O(\frac sn)$ 次。

于是方案二的总时间复杂度为 $O\left(\sum_{i=1}^t \frac in\right)\sim O\left(\frac {t^2}n\right)$

考虑根号分治，当 $n\lt \sqrt t$ 时采用方案一，否则采用方案二。总时间复杂度 $O(t\sqrt t)$

```
const int MAXN=5e5+5,MAXM=800,mod=998244353;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1)ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int frac[MAXN],invf[MAXN];
int C(int n,int m){
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
void init(){
    frac[0]=1;
    _for(i,1,MAXN)frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--)
    invf[i-1]=1LL*invf[i]*i%mod;
}
int ans1[MAXN],ans2[MAXN];
int dp[2][MAXM];
void solve1(int s,int n,int a,int *ans){
    int pos=0;
    mem(dp[pos],0);
    dp[pos][a]=1;
    ans[0]=1;
    _rep(i,1,s){
        pos=!pos;
        mem(dp[pos],0);
        _rep(j,1,n){
            dp[pos][j]=(dp[!pos][j-1]+dp[!pos][j+1])%mod;
            ans[i]=(ans[i]+dp[pos][j])%mod;
        }
    }
}
int cal(int s,int n,int a,int pos){
    int pos1=pos,pos2=pos,ans=0,d=abs(pos-a);
    if(d<=s&&(s-d)%2==0)
    ans=C(s,(s+d)/2);
    for(int j=1;;j++){
        if(j&1){
            pos1=-pos1;
            pos2=2*(n+1)-pos2;
        }
```

```
        else{
            pos1=2*(n+1)-pos1;
            pos2=-pos2;
        }
        int d1=abs(pos1-a),d2=abs(pos2-a),det=0;
        if(d1>s&&d2>s)break;
        if(d1<=s&&(s-d1)%2==0)
        det=(det+C(s,(s+d1)/2));
        if(d2<=s&&(s-d2)%2==0)
        det=(det+C(s,(s+d2)/2))%mod;
        if(j&1)
        ans=(ans+mod-det)%mod;
        else
        ans=(ans+det)%mod;
    }
    return ans;
}
void solve2(int s,int n,int a,int *ans){
    ans[0]=1;
    _rep(i,1,s)
    ans[i]=(2LL*ans[i-1]+mod-cal(i-1,n,a,1)+mod-cal(i-1,n,a,n))%mod;
}
void solve(int s,int n,int a,int *ans){
    if(1LL*n*n<s)
    solve1(s,n,a,ans);
    else
    solve2(s,n,a,ans);
}
int main()
{
    init();
    int t=read_int(),n=read_int(),m=read_int(),a=read_int(),b=read_int();
    solve(t,n,a,ans1);
    solve(t,m,b,ans2);
    int ans=0;
    _rep(i,0,t)
    ans=(ans+1LL*C(t,i)*ans1[i]%mod*ans2[t-i])%mod;
    enter(ans);
    return 0;
}
```