Can’ın Matrisi Çözüm

Herhangi bir NxM ilk karede eğer hiç engel yok ise sol üst kareden sağ alt engele toplamda Kombinasyon(N+M-2 , N-1) farklı şekilde gidebiliriz. Bu kombinasyon işlemini hesaplarken 1 den 10^5'e kadar olan faktoriyelleri önceden hesaplayıp, mod'u alınmış bir sayıyı bölmek için de inverse modulo işlemini kullanıyoruz, daha fazla ayrıntı için koddan inceleyebilirsiniz.

Peki bir engel olduğu durumda ne yapabiliriz; Tüm durumdan istenmeyen durumu çıkartarak cevabı bulabiliriz, tüm durum bütün NxM için cevap olur. Eğer engel X,Y noktasında ise istenmeyen durum X,Y ye farklı gitme sayımız. K(X+Y-2, X-1) ve X,Y den en sona gitme sayımız K(N-X + M-Y, N-X) bu iki değerin çarpımı olur. Bu istenmeyen değeri tüm durumdan çıkartarak cevaba ulaşabiliriz.

İki engel olduğunu düşünelim. İstenmeyen durumu her iki engel için hesaplamamız gerekli, ilk olarak sadece 1. engelden geçtiğimiz durum sayısı ile ardından sadece 2. engelden geçtiğimiz durum sayısını hesaplarız, ardından bu durum sayılarının toplamını cevaptan çıkarttığımızda bazı durumları birden fazla kere çıkartmış oluruz. Bu durumlar hem 1den hem de 2den geçtiğimiz durumlardır, bunu yine aynı mantık ile hesaplayabiliriz. Bulduğumuz değeri de istenmeyen durumumuza eklersek cevabı buluruz. Cevap= Tüm Durum - 1. den geçilen - 2.den geçilen + hem 1. den hem 2. den geçilen.

İki'den daha fazla engel olduğu durumda, sırasıyla 1.'den geçilen 2.'den geçilen 3.'den geçilen ..... k.'dan geçilen, 1. ve 2.'den geçilen, 1. ve 3.'den geçilen ..... her engelin geçme ve geçilmeme durumlarına göre toplamda 2^k tane durumun hepsi için ayrı ayrı hesaplamamız lazım, eğer tek sayıda engelden geçerek en sona ulaşma sayımızı hesapladıysak bunu cevaptan çıkartıyoruz, eğer çift sayıda engelden geçerek en sona ulaştıysak bunu cevaba ekliyoruz, bu sayede istenen sonuca ulaşabiliyoruz. Bu işleme içerme dışarda prensibi denir.

Çözüm Kodu:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long int lint;
const int MOD = 1000000007;
const int MAX_N = 200005;
lint fact[MAX_N];
int n,m,k;
pair <int,int> d[MAX_N];


lint fast_pow(lint a, lint b) {
    if(b==0) return 1;
```

```
    if(b==1) return a;
    long long int tmp = fast_pow(a,b/2);
    if(b&1) return (((tmp*tmp)%MOD)*a)%MOD;
    return (tmp*tmp)%MOD;
}
void init_fact() {
    fact[0] = 1;
    for (int i = 1; i < MAX_N; i++)
    {
        fact[i] = (fact[i-1] * i) % MOD;
    }
}
lint bolme(lint a, lint b) {
    return (a * fast_pow(b,MOD-2)) % MOD;
}
lint comb(int a, int b) {
    return  bolme(bolme(fact[a],fact[b]),fact[a-b]);
}

lint cozum(int a, int b) { // a x b yerin cozumu
    if(a<=0 || b<=0) return 0;
    return comb(a+b-2, a-1);
}

lint calc(int state) {
    vector <pair<int,int> > v;
    v.push_back(pair<int,int>(1,1));
    v.push_back(pair<int,int>(n,m));
    for (int i = 0; i < k; i++)
        if(state & 1<<i) {
            v.push_back(pair<int,int>(d[i].first, d[i].second));
        }
    sort(v.begin(), v.end());
    for (int i = 1; i < v.size(); i++)
        if(v[i-1].second  > v[i].second) {
            return 0;
        }
    lint retVal = 1;
    for (int i = 1; i < v.size(); i++)
        retVal   =   (retVal   *   cozum(v[i].first-v[i-1].first   +   1,
v[i].second-v[i-1].second + 1) ) % MOD;
    return v.size()%2==0 ? retVal : -retVal;
}
int main() {
    init_fact();

    cin >> n >> m >> k;
    for (int i = 0; i < k; i++)
```

```
    {
        cin >> d[i].first >> d[i].second;
    }
    lint ans = 0;
    for (int i = 0; i < (1<<k); i++) {
        ans = (ans + calc(i) + MOD) % MOD;
    }
    cout << ans << endl;
}
```