Michal Korbela
septima
Gym. J.J. Bánovce nad Bebravou
úloha 7

Najskôr sa pozrime na jednu zaujímavosť:

keď sa pozrieme na nejaké políčko plánika(zatiaľ bez prekážok) tak vieme, že sa počet ciest naň rovná počtu kombinácii z opakovaním... ale to teraz nepotrebujeme, ale vieme, že sa rovná súčtu políčka nad a naľavo od neho. Z prekážkami to bude to isté, akurát prekážky rátame akože je tam 0(čo si budem musieť prerobiť plánik).

Potom ak stúpim na nejaké políčko, tak viem sa rozhudnúť v konštantnom čase či pôjdem hore alebo doľava.

Ja pre krajšie riešenie si počty ciest vypočítam od dolného pravého rohu. Potom v ľavom hornom dostanem počet všetkých ciest.

Potom keď idem z ľavého horného rohu, tak sa pozriem doprava a dole a viem usúdiť podľa počtu ciest či pôjdem dole alebo doprava(či je počet ciest väčší alebo menší ako číslo dole)

Ak pôjdem doprava, tak v nasledujúcom ťahu musím pripočítať počet ciest smerom dole v predchádzajúcom ťahu.

Ak však na začiatku je číslo väčšie ako je celkový počet ciest, tak taká cesta potom nemôže existovať a vypíše sa neexituje.

Avšak vyskytol sa mi problém v maximálnej kapacite long long, tak potom keď sčítavam tak ak je to väčšie alebo rovné 10^{17} tak potom tam hodím 10^{17} – lebo väčšie číslo nikdy potrebovať nebudem.

```
#include <iostream>

using namespace std;

long long maxim=100000000000000000;              //maximálne číslo

long long pole[1047][1047];                      //zapamätáme si mriežku

long long r,s;

int chod(long long min, long long temp,long long x,long long y){          //rekurzívna funkcia, ktorá vypisuje a zisťuje cestu

if(x==r && y==s){ cout<<endl;                 // ak sme na konci, tak vypíš nový riadok
return 0;
};


if(pole[x+1][y]+temp>=min && pole[x+1][y]!=0 ){ // ak musím ísť dole – cesta je menšia alebo rovná ako počet ciest smerom dole
cout<<"D";
chod(min,temp,x+1,y);          // chod dole
```

```
}
else if(pole[x+1][y]+temp<=min && pole[x][y+1]!=0){     // inak chod vpravo
cout<<"P";
chod(min,temp+pole[x+1][y],x,y+1);
}


else                                // inak sa zacykli ak nemáš cestu, čo nikdy nenastane(pre kontrolu)
while(true)
cout<<"l";




}









int main(){

long long q;


cin>>r>>s>>q;                       // načítanie mriežky a otázok

for(int i=1; i<=r; i++)
for(int j=1; j<=s; j++){
int temp=0;;
cin>>temp;
if(temp==1) pole[i][j]=0;           // ak je tam prekážka, tak tam daj 0 nie 1
else pole[i][j]=-1;                 // inak voľné miesto označ -1
}




pole[r][s]=1;                       // do políčka r x s vedie práve 1 cesta

for(int i=r; i>0; i—)               // vypočítaj počty ciest
for(int j=s; j>0; j--){

if(pole[i][j]!=0){
if(i==r && j==s) continue;          //ak sme ešte na začiatku
                                    // inak
```

```
else if(i==r) pole[i][j]=pole[i][j+1];  //ak sme na kraji, tak aby sme nešahali mimo poľa, tak je tam 0 tak ju tam umelo dať

else if(j==s) pole[i][j]=pole[i+1][j];  //tak isto
else if(pole[i+1][j]+pole[i][j+1]<maxim)      // ak je vštko OK
pole[i][j]=pole[i+1][j]+pole[i][j+1];             // ak by sme prekročili limit long long tak tam dajme maxim
else pole[i][j]=maxim;
}
}




for(int i=1; i<=q; i++){                    // načítaj otázku a vypíš cestu
long long temp;
cin>>temp;
if(pole[1][1]<temp){ cout<<"neexistuje"<<endl; continue;}          //ak taká cesta neexituje
chod(temp, 0,1,1);                          // funkcia, ktorá vypisuje cestu a cestuje po pláne

}


}
```

Čo sa týka zložitostí:

čas - potrebujeme pre načítanie r*s a vypočítať počty ciest O(r*s)

a pre každú cestu ktorú vypisujeme potrebujeme q(r+s) času, takže dokopy O(r*s+q(r+s))

pamäť okrem poľa ktoré potrebujem na mriežku a parametrov vo fukcii rekurzívnej nepotrebujem nič iné, takže O(r*s+r+s) a spolu O(r*s)