Meno: Michal Korbela
trieda: sexta
škola: Gymnázium J.J. Bánovce
úloha č.4

Keď sa pozrieme, tak v 0. kroku máme všetky základné prvky.
V 1. kroku vieme spraviť prvky, ktoré potrebujú len základné prvky.
V 2. kroku vieme spraviť len prvky, ktoré potrebovali aspoň jeden prvok, ktorý sme vyrobili v 1. ťahu a 2. mohol byť aj z 0. ťahu – základný.
V 3. ťahu zas môžeme vyrobiť len prvky, ktoré potrebovali prvok vyrobený v 2. ťahu – keby nepotrebovali, tak sa dajú vyrobiť už skôr.
Atd .

Preto môžeme uvažovať:
Ak vyrobíme nejaký prvok, tak potom sa pozrieme na všetky prvky, ktoré tento prvok potrebovali k svojej výrobe. Ak sa dajú vyrobiť, tak ich vyrobíme a ak nie, tak sa budú vyrobiť v budúcich ťahoch.

Keďže každé pravidlo potrebuje max 2 rôzne prvky na výrobu, tak sa ho pokúsime vyrobiť max 2 krát, čiže konštantný počet.

Takže keďže každé pravidlo použijeme konštantný počet krát, tak časová zložitosť bude O(m) – m je počet pravidiel
Ale keďže musíme načítať ešte počiatočné prvky, tak zložitosť bude O(m+n) – n je počet základných prvkov- musíme ich načítať a keď budeme vypisovať, tak určite vypíšeme všetky základné, ktoré sme aj načítali a aj všetky tie, ktoré sme museli vyrobiť pomocou nejakého pravidla – tam je už zložitosť zahrnutá.

Keď už vyrobíme nejaký prvok, tak jeho minimálny počet krokov na výrobu potrebuje o 1 viac krokov ako ten, ktorý bol vyrobený v predchádzajúcom ťahu.

-či je vyrobený
-koľko krokov potrebujeme na výrobu tohto prvku
-pravidlá na jeho výrobu
-práve vyrobené prvky

Pamäť – pre každý prvok si zapamätáme konštantný počet údajov O(k) – k - počet prvkov
a ešte si zapamätáme pravidlá, takže O(k+m)

```
#include<iostream>
#include<cstdio>
#include<vector>
using namespace std;

struct pol{
int c;
```

```
int b;
};


int mina=210000;
vector< vector< pol > > prvky;          //práve vyrobené prvky
vector<pol>::iterator it;

bool zakl[200009];                      //na začiatku základné prvky neskôr všetky vyrobené
int kroky[200009];                      //počet krokov pre daný prvok
int prvk[200009];                       //práve vyrobené prvky
int poc=0;









int main(){
int k,m,n;


prvky.resize(200009);



cin>>k>>n>>m;



for(int i=0; i<n; i++){
int temp;
cin>>temp;
zakl[temp]=true;                                //načítame základné prvky
kroky[temp]=0;                                  //na ich výrobu potrebujeme 0 krokov
prvk[poc++]=temp;                               //a sú to aj práve vyrobené prvky v 0 ťahu
}



for(int i=0; i<m; i++){

int a,b,c;
cin>>a>>b>>c;                                   //načítame pravidlá
pol temp;                                       //vytvoríme štruktru
temp.c=c;
temp.b=b;
prvky[a].push_back(temp);                       //a prvok sa dá vyrobiť buď ak je práve
vyrobený jeden z nich
```

```
temp.b=a;
prvky[b].push_back(temp);

}

int zac=0,kon=poc-1,krok=1;

while(kon+1<k){                                  //pokiaľ nie sú vyrobené všetky prvky


for(int i=zac; i<=kon; i++){                     //pokiaľ nie sú minuté všetky práve vyrobené
prvky
for(it=prvky[prvk[i]].begin(); it<prvky[prvk[i]].end(); it++){        //pozrieme sa na všetky pravidlá,
ktoré potrebujú práve vyrobený prvok

        if(zakl[(*it).b]==true && kroky[(*it).b]<krok && zakl[(*it).c]!=true) { //pokiaľ je už druhý
prvok vyrobený a nebol vyrobený  tomto ťahu a prvok ktorý chceme vyrobiť ešte nieje vyrobený
        prvk[poc++]=(*it).c;                     //označíme prvok ako práve vyrobený
        kroky[(*it).c]=krok;                     //zapíšeme mu kroky
        zakl[(*it).c]=true;                      //označíme prvok ako vyrobený
        }
}

}
zac=kon+1;                                       //pozrieme sa na ďalšie práve vyrobené
prvky
kon=poc-1;
krok++;                                          //zvýšime počet krokov

}

for(int i=1; i<=k; i++) printf("%d\n",kroky[i]);          //vypíšeme









}
```