## BFS+time, castle.cpp

#include <iostream>

#include <queue>

using namespace std;

int pintu[100010][12];

char jumlahdetik[100010][12];

queue<int> qpintu;

queue<long long> detik;

int nruang,periode;

int main(){

cin >> nruang >> periode;

for(int i = 1; i<=nruang; i++){

for(int j = 0; j<periode; j++){

cin >> pintu[i][j];

jumlahdetik[i][j] =0;

}

}

qpintu.push(1);

detik.push(0);

jumlahdetik[1][0] = 1;

long long cdetik = 0;

long long ans;

while(!qpintu.empty()){

int cpintu = qpintu.front();

cdetik = detik.front();

qpintu.pop();

detik.pop();

if(cpintu == nruang){

ans = cdetik;

break;

}

if(jumlahdetik[pintu[cpintu][cdetik%periode]][(cdetik+1)%periode]<=0){ // blm di visit

qpintu.push(pintu[cpintu][cdetik%periode]);

detik.push(cdetik+1);

jumlahdetik[pintu[cpintu][cdetik%periode]][(cdetik+1)%periode] = 1;

}

if(jumlahdetik[cpintu][(cdetik+1)%periode]<=0){ // blm di visit

qpintu.push(cpintu);

detik.push(cdetik+1);

jumlahdetik[cpintu][(cdetik+1)%periode] = 1;

}

}

cout << ans << endl;

return 0;}

## 

## faktorisasiprima.cpp

#include <iostream>

#include <vector>

#include <queue>

#include <cmath>

using namespace std;

int n; vector<bool> sieve; queue<int> primes;

queue<int> resultprime; queue<int> resultindex;

int main(){

cin >> n;

if(n==1){

cout << 1 << endl;

return 0;

}

for(int i = 0; i <= 1000000; i++) sieve.push\_back(true);

for(int i = 2; i<=1000; i++){

if(sieve[i]){

for(int j = i; i+j<=1000000; j+=i){

sieve[i+j] = false;

}

}

}

for(int i = 2; i<= 1000000; i++){

if(sieve[i]){

primes.push(i);

//cout << primes.front() << endl;

}

}

int currentprimecount = 0;

int currentprime;

while(n>1){

currentprime = primes.front();

if(n%currentprime == 0){

currentprimecount++;

n/=currentprime;

} else {

resultprime.push(currentprime);

resultindex.push(currentprimecount);

currentprimecount = 0;

primes.pop();

}

}

resultprime.push(currentprime);

resultindex.push(currentprimecount);

while(resultprime.size()>1){

if(resultindex.front()>0){

cout << resultprime.front();

if(resultindex.front()>1) cout << "^" << resultindex.front();

cout << " x ";

}

resultprime.pop();

resultindex.pop();

}

if(resultindex.front()>0){

cout << resultprime.front();

if(resultindex.front()>1) cout << "^" << resultindex.front();

cout << endl; }

return 0;}