Bài thực hành số 3

Mục lục

[Bài thực hành số 3 1](#_Toc138161180)

[Bài 3.1 2](#_Toc138161181)

[Bài 3.2 3](#_Toc138161182)

[Bài 3.3 5](#_Toc138161183)

[Bài 3.4 7](#_Toc138161184)

[Bài 3.5 9](#_Toc138161185)

[Bài 3.6 12](#_Toc138161186)

[Bài 3.7 13](#_Toc138161187)

[Bài 3.9 15](#_Toc138161188)

[Bài 3.10 16](#_Toc138161189)

[Bài 3.11 17](#_Toc138161190)

[Bài 3.12 19](#_Toc138161191)

## Bài 3.1

A screenshot of a computer

Description automatically generated

// Nguyen Duong Anh 20210082

int lucas(int n) {

    if(n == 0) return 2;

    else if(n==1) return 1;

    return lucas(n-1) + lucas(n-2);

}

// Nguyen Duong Anh 20210082

## Bài 3.2

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing screenshot, text, software, display

Description automatically generated

A picture containing text, screenshot, rectangle

Description automatically generated

// Nguyen Duong Anh 20210082

#include<stdio.h>

#include<stdlib.h>

#include <iostream>

using namespace std;

int n;

int X[100], Y[100]; //# Lưu tọa độ các bước di chuyển của quân mã

int mark[100][100]; //# Đánh dấu vị trí các ô mà quân mã đã di chuyển qua

//# Mảng hx, hy mô tả 8 vị trí quân mã có thể di chuyển kể từ vị trí hiện tại

const int hx[] = {1, 1, 2, 2, -1, -1, -2, -2};

const int hy[] = {2, -2, 1, -1, 2, -2, 1, -1};

//# In ra dãy các di chuyển tìm được

void print\_sol(){

    for (int j = 1; j <= n \* n; ++j)

        printf("(%d %d)\n", X[j], Y[j]);

    exit(0);

}

bool check(int xx, int yy){

    if(mark[xx][yy] == 1) return false;     // đi qua rồi

    if(xx < 1 || xx > n || yy < 1 || yy > n) return false; // đi ra ngoài tọa độ 1-n

    return true;

}

//# Thuật toán quay lui

void TRY(int k){

    for(int i = 0; i < 8; i++){

        int xx = X[k-1] + hx[i];    // di chuyển theo trục x

        int yy = Y[k-1] + hy[i];    // di chuyển theo trục y

        if(check(xx,yy)){

            mark[xx][yy] = 1;

            X[k] = xx;

            Y[k] = yy;

            if(k == n\*n ) print\_sol();

            else TRY(k+1);

            mark[xx][yy] = 0;

        }

    }

}

int main(){

    cin >> n;

    mark[1][1] = 1;

    X[1] = Y[1] = 1;

    TRY(2);

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.3

A screenshot of a computer

Description automatically generated with medium confidence

A picture containing text, software, computer icon, multimedia software

Description automatically generated

// Nguyen Duong Anh 20210082

#include<stdio.h>

#include<stdlib.h>

#include <iostream>

using namespace std;

int n;

int X[100], Y[100]; //# Lưu tọa độ các bước di chuyển của quân mã

int mark[100][100]; //# Đánh dấu vị trí các ô mà quân mã đã di chuyển qua

//# Mảng hx, hy mô tả 8 vị trí quân mã có thể di chuyển kể từ vị trí hiện tại

const int hx[] = {1, 1, 2, 2, -1, -1, -2, -2};

const int hy[] = {2, -2, 1, -1, 2, -2, 1, -1};

//# In ra dãy các di chuyển tìm được

void print\_sol(){

    for (int j = 1; j <= n \* n; ++j)

        printf("(%d %d)\n", X[j], Y[j]);

    exit(0);

}

bool check(int xx, int yy){

    if(mark[xx][yy] == 1) return false;     // đi qua rồi

    if(xx < 1 || xx > n || yy < 1 || yy > n) return false; // đi ra ngoài tọa độ 1-n

    //cout << xx << " " << yy << endl;

    return true;

}

//# Thuật toán quay lui

void TRY(int k){

    for(int i = 0; i < 8; i++){

        int xx = X[k-1] + hx[i];    // di chuyển theo trục x

        int yy = Y[k-1] + hy[i];    // di chuyển theo trục y

        if(check(xx,yy)){

            mark[xx][yy] = 1;

            X[k] = xx;

            Y[k] = yy;

            if(k == n\*n ) print\_sol();

            else TRY(k+1);

            mark[xx][yy] = 0;

        }

    }

}

int main(){

    cin >> n;

    mark[1][1] = 1;

    X[1] = Y[1] = 1;

    TRY(2);

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.4

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

// Nguyen Duong Anh 20210082

#include <bits/stdc++.h>

using namespace std;

int a[1000], n;

int mem[1000]; //# mảng ghi nhớ lời giải các bài toán con đã được giải

void init(){

    memset(mem, -1, sizeof(mem));

}

//# Quy hoạch động,

//# Hàm lis(i) trả về độ dài dãy con tăng dài nhất kết thúc bởi a[i]

void incsubseq(){

    mem[0] = 1;

    for(int i=1; i<n; i++){

        mem[i] = 1;

        for(int j=0; j<i; j++){

            if(a[j] < a[i]){

                if(mem[i] < mem[j] + 1){

                    mem[i] = mem[j] + 1;

                }

            }

        }

    }

}

int lis(int i) {

    return mem[i];

}

//# Truy vet loi giai

void trace(int i){

    for(int j = 0; j < i; j++){

        if (a[j] < a[i] && mem[i] == 1 + mem[j]){

            trace(j);

            break;

        }

    }

    cout << a[i] << " ";

}

int main(){

    init();

    cin >> n;

    for(int i = 0; i < n; i++) cin >> a[i];

    incsubseq();

    int res = 1, pos = 0;

    for(int i = 1; i < n; i++){

        if (res < lis(i)){

            res = lis(i);

            pos = i;

        }

    }

    cout << res << endl;

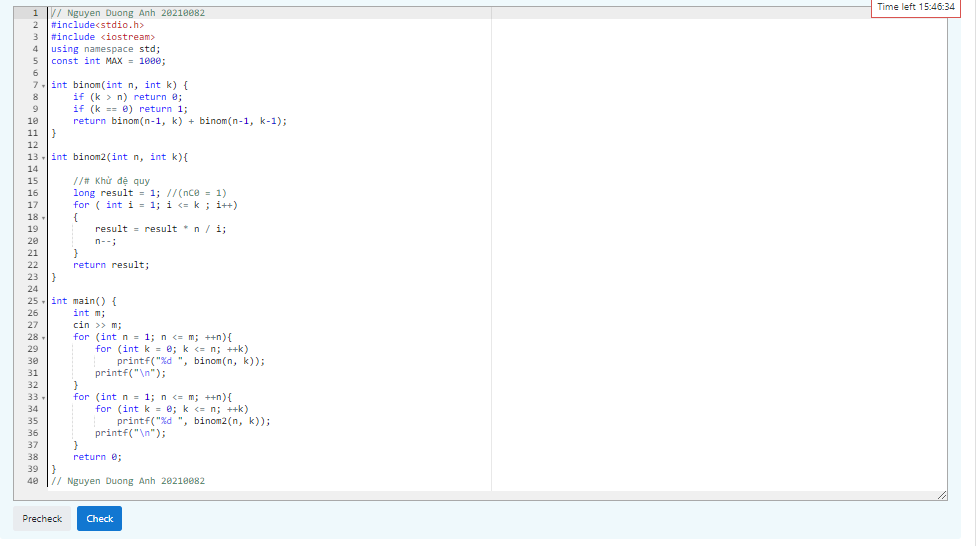
    trace(pos);

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.5



A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated with low confidence

// Nguyen Duong Anh 20210082

#include<stdio.h>

#include <iostream>

using namespace std;

const int MAX = 1000;

int binom(int n, int k) {

    if (k > n) return 0;

    if (k == 0) return 1;

    return binom(n-1, k) + binom(n-1, k-1);

}

int binom2(int n, int k){

    //# Khử đệ quy

    long result = 1; //(nC0 = 1)

    for ( int i = 1; i <= k ; i++)

    {

        result = result \* n / i;

        n--;

    }

    return result;

}

int main() {

    int m;

    cin >> m;

    for (int n = 1; n <= m; ++n){

        for (int k = 0; k <= n; ++k)

            printf("%d ", binom(n, k));

        printf("\n");

    }

    for (int n = 1; n <= m; ++n){

        for (int k = 0; k <= n; ++k)

            printf("%d ", binom2(n, k));

        printf("\n");

    }

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.6

A screenshot of a computer

Description automatically generated with medium confidence

// Nguyen Duong Anh 20210082

#include <iostream>

using namespace std;

int gcd(int a, int b){

    if (b == 0) return a;

    return gcd(b, a % b);

}

int gcd2(int a, int b){

    //# Khử đệ quy

    while(b != 0){

        int tmp = a % b;

        a = b;

        b = tmp;

    }

    return a;

}

int main() {

    int a, b;

    cin >> a >> b;

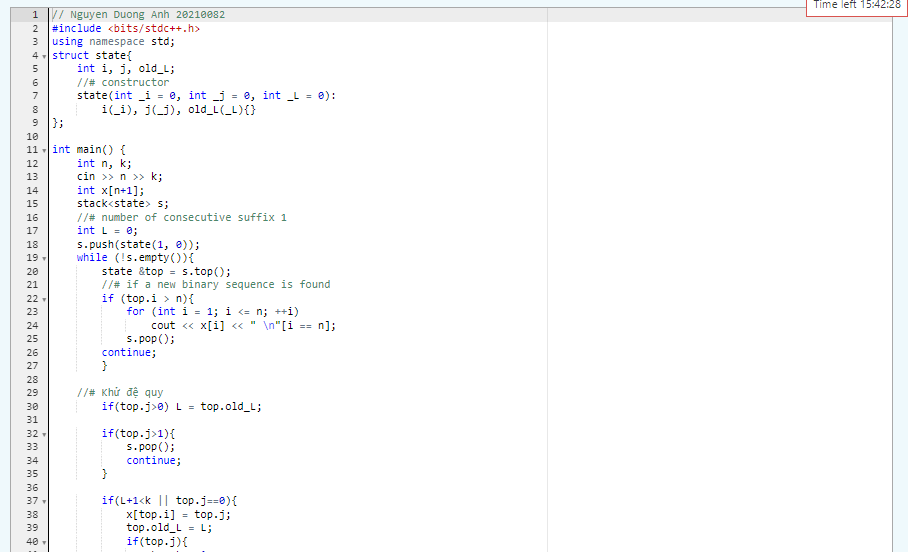
    cout << gcd(a, b) << endl << gcd2(a, b);

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.7



A screenshot of a computer

Description automatically generated

A picture containing text, screenshot

Description automatically generated

// Nguyen Duong Anh 20210082

#include <bits/stdc++.h>

using namespace std;

struct state{

    int i, j, old\_L;

    //# constructor

    state(int \_i = 0, int \_j = 0, int \_L = 0):

        i(\_i), j(\_j), old\_L(\_L){}

};

int main() {

    int n, k;

    cin >> n >> k;

    int x[n+1];

    stack<state> s;

    //# number of consecutive suffix 1

    int L = 0;

    s.push(state(1, 0));

    while (!s.empty()){

        state &top = s.top();

        //# if a new binary sequence is found

        if (top.i > n){

            for (int i = 1; i <= n; ++i)

                cout << x[i] << " \n"[i == n];

            s.pop();

        continue;

        }

    //# Khử đệ quy

        if(top.j>0) L = top.old\_L;

        if(top.j>1){

            s.pop();

            continue;

        }

        if(L+1<k || top.j==0){

            x[top.i] = top.j;

            top.old\_L = L;

            if(top.j){

                L = L + 1;

            } else {L = 0;}

            s.push(state(top.i+1,0));

        }

        top.j++;

    }

    return 0;

}

// Nguyen Duong Anh 20210082

## Bài 3.9

// Nguyễn Dương Ánh 20210082

#include<stdio.h>

#include<stdlib.h>

#include <iostream>

using namespace std;

int n,k1,k2;

int \*schedule;  // mảng để chứa giá trị ngày làm việc (1) và ngày nghỉ (0)

int streak = 0; // chuỗi ngày làm việc liên tiếp

void print\_sol(){

    for(int i=0; i<n;i++){

        cout<<schedule[i]<<" ";

    }

    cout<<"\n";

}

bool check(int prev, int current, int dayNumber){

    if (prev == 0 && current == 0 ) return false;                               // không được có 2 ngày nghỉ liên tiếp

    else if (prev == 1 && streak < k1 && current == 0) return false;            // chưa đủ số ngày làm tối thiểu

    else if (streak >= k2 && current ==1) return false;                         // quá số ngày làm tối đa

    else if (dayNumber == n-1 && current == 1 && streak <k1-1) return false;    // đến ngày cuối nhưng chưa đủ số ngày làm tối thiểu

    return true;

}

// ## Thuật toán quay lui

void TRY(int k){

    for(int i = 0; i < 2; i++){

        int current = i;

        int prevStreak = streak;

        // cout<< "dayNumber: "<<k<<" current: "<< current << " streak: "<<streak<<"\n";

        if(check(schedule[k-1],current, k)){

            if(i==1) streak ++;

            else if(i==0) streak =0;

            // cout<< "dayNumber: "<<k<<" current: "<< current << " streak: "<<streak<<"\n";

            schedule[k] = current;

            if(k==n-1) print\_sol();     // nếu thử đến cuối rồi thì in ra

            else TRY(k+1);

            // khôi phục chuỗi trước đó

            streak = prevStreak;

        }

    }

}

int main(){

    cin>> n>>k1>>k2;

    // cấp phát động cho schedule[]

    schedule=(int\*)malloc(n \* sizeof(int));

    for (int i=0; i<=1; i++){

        schedule[0] = i;

        streak = i;

        TRY(1);

    }

    return 0;

}

// Nguyễn Dương Ánh 20210082

## Bài 3.10

// Nguyễn Dương Ánh 20210082

#include<iostream>

using namespace std;

const int MAX = 20;

int N, H;

int x[MAX];

int S[MAX]; // Xâu gốc để so sánh

int count = 0;

void input(){

    cin >> N >> H;

    for(int i=0; i<N; i++) S[i] = 0;

    for(int i=0; i<N; i++) x[i] = 0;

}

// Kiểm tra khoảng cách Hamming của 2 xâu

int check(int str1[], int str2[]){

    int lens = N;

    int count = 0;

    for(int i=0; i<lens; i++){

        if(str1[i] != str2[i]) count++;

    }

    return count;

}

bool check(int a, int i){

    return true;

}

void solution(){

    if(check(x,S) == H){

        for(int i=0; i<N; i++) cout << x[i];

        cout << endl;

    }

}

void TRY(int a){

    for(int i=0; i<=1; i++){

        x[a] = i;

        if(a == N-1) solution();

        else

            TRY(a+1);

    }

}

int main(){

    int k;

    cin >> k;

    while(k > 0){

        input();

        TRY(0);

        k--;

    }

}

// Nguyễn Dương Ánh 20210082

## Bài 3.11

// Nguyễn Dương Ánh 20210082

#include<bits/stdc++.h>

using namespace std;

const int MAX = 10000;

int n, r;

int cost[MAX][MAX];

int x[MAX];

bool visited[MAX];

vector<int> vt;

int min\_cost;

int sum\_cost;

int start, destination, numberOfPoint;

void input(){

    cin >> n >> r;

    for(int i=0; i<n; i++)

        for(int j=0; j<n; j++){

            cin >> cost[i][j];

        }

}

bool check(int a, int i){

    if(visited[vt[i]]) return false;

    if(cost[x[a-1]][vt[i]] == 0) return false;

    return true;

}

void solution(){

    if(cost[x[numberOfPoint-2]][destination] == 0) return;

    min\_cost = min(min\_cost, sum\_cost + cost[x[numberOfPoint-2]][destination]);

}

void TRY(int a){

    for(int i=1; i<numberOfPoint-1; i++){

        if(check(a, i)){

            visited[vt[i]] = true;

            sum\_cost += cost[x[a-1]][vt[i]];

            x[a] = vt[i];

            if(a == numberOfPoint-2) solution();

            else TRY(a+1);

            visited[vt[i]] = false;

            sum\_cost -= cost[x[a-1]][vt[i]];

        }

    }

}

int main(){

    string str;

    input(); getline(cin,str);

    while(r > 0){

        min\_cost = INT\_MAX;

        sum\_cost = 0;

        getline(cin, str);

        while (!str.empty()){

            stringstream convert(str.substr(0, str.find(" ")));

            int tmp = 0;

            convert >> tmp;

            vt.push\_back(tmp - 1);

            if (str.find(" ") > str.size()){

                break;

            } else {

                str.erase(0, str.find(" ") + 1);

            }

        }

        start = vt[0];

        destination = vt[vt.size()-1];

        numberOfPoint = vt.size();

        x[0] = start; x[numberOfPoint-1] = destination;

        for(int i=0; i<n; i++)

            visited[i] = false;

        TRY(1);

        if(min\_cost == INT\_MAX) cout << "0" << endl;

        else cout << min\_cost << endl;

        vt.erase(vt.begin(), vt.end());

        r--;

    }

}

// Nguyễn Dương Ánh 20210082

## Bài 3.12

// Nguyễn Dương Ánh 20210082

#include<bits/stdc++.h>

using namespace std;

const int MAX = 100;

int n, k;

int m;

vector<vector<int> > vt;

int x[MAX];

bool visited[MAX];

int res;

void input(){

    cin >> n >> k;

    cin >> m;

    vt.resize(n+1);

    for(int i=0; i<m; i++){

        int t1, t2;

        cin >> t1 >> t2;

        vt[t1-1].push\_back(t2-1);

        vt[t2-1].push\_back(t1-1);

    }

    for(int i=0; i<n; i++){

        visited[i] = false;

    }

    res = 0;

}

bool check(int a, int i){

    if(a == 0) return true;

    if(visited[i]) return false;

    int index = 0;

    for(int j=0; j<vt[x[a-1]].size(); j++){

        if(i == vt[x[a-1]][j]) index++;

    }

    if(index == 0) return false;

    return true;

}

void solution(){

    res++;

}

void TRY(int a){

    for(int i=0; i<n; i++){

        if(check(a, i)){

            visited[i] = true;

            x[a] = i;

            if(a == k) solution();

            else TRY(a+1);

            visited[i] = false;

        }

    }

}

int main(){

    input();

    TRY(0);

    cout << res / 2;

}

// Nguyễn Dương Ánh 20210082