Mục lục nội dung

[Bài thực hành số 03 – Tuần 38 3](#_Toc168152316)

[BÀI TẬP TRÊN LAP 3](#_Toc168152317)

[**Bài 3.1.**  Dãy Lucas được định nghĩa bởi Ln = Ln-1 + Ln-2 với L0 = 2, L1 = 1. Hãy viết hàm tính số Lucas thứ n. 3](#_Toc168152318)

[**Bài 3.2**. Trên bàn cờ vua kích thước n\*n có một quân mã đang ở ô (1, 1). Hãy đưa ra một dãy các di chuyển của mã sao cho mỗi ô trên bàn cờ đều được đi qua đúng 1 lần (ô (1, 1) được xem là đã đi qua). 3](#_Toc168152319)

[**Bài 3.3**. Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố i sang thành phố j là cij. Hãy tìm tổng chi phí nhỏ nhất có thể 3](#_Toc168152320)

[**Bài 3.4**. Cho dãy a có  n phần tử. Một dãy con của  a là dãy thu được bằng cách xóa đi một số phần tử của  a và giữ nguyên thứ tự các phần tử còn lại (có thể không xóa phần tử nào). Hãy tìm dãy con tăng dài nhất của a. 3](#_Toc168152321)

[**Bài 3.5**. Tính hệ số tổ hợp C(n, k) 6](#_Toc168152322)

[**Bài 3.6.**Tìm ước chung lớn nhất của hai số nguyên a, b cho trước. 10](#_Toc168152323)

[**Bài 3.7.**Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n không có k bit 1 nào liên tiếp 10](#_Toc168152324)

[**BÀI TẬP VỀ NHÀ** 10](#_Toc168152325)

[**Bài tập 3.9:** Lập lịch cho y tá 10](#_Toc168152326)

[**Bài tập 3.10:** Khoảng cách Hamming 16](#_Toc168152327)

[**Bài tập 3.11:** Đoàn người du lịch 19](#_Toc168152328)

[**Bài tập 3.12:** Đếm số đường đi đơn độ dài k 24](#_Toc168152329)

Mục lục hình ảnh

[Hình 1 Bài 3.1 Tính số Lucas thứ n 3](#_Toc168151562)

[Hình 2 Bài 3.2. Quân mã đi tuần 8](#_Toc168151563)

[Hình 3 Bài 3.3 Nhánh cận người đi du lịch 12](#_Toc168151564)

[Hình 4 Bài 3.4 Dãy con tăng dài nhất 15](#_Toc168151565)

[Hình 5 Bài 3.5 Tính hệ số tổ hợp C(n,k) 20](#_Toc168151566)

[Hình 6 Bài 3.6 Tìm ước chung lớn nhất 22](#_Toc168151567)

[Hình 7 Bài 3.7 Liệt kê xâu nhị phân không có k bit 1 nào liên tiếp 28](#_Toc168151568)

[Hình 8 Bài 3.9 32](#_Toc168151569)

[Hình 9. Bài 3.10 36](#_Toc168151570)

[Hình 10. Bài 3.11 42](#_Toc168151571)

[Hình 11. Bài 3.12 44](#_Toc168151572)

# Bài thực hành số 03 – Tuần 38

# BÀI TẬP TRÊN LAP

## **Bài 3.1.**  Dãy Lucas được định nghĩa bởi Ln = Ln-1 + Ln-2 với L0 = 2, L1 = 1. Hãy viết hàm tính số Lucas thứ n.

Ảnh có chứa văn bản, hàng, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Hình 1 Bài 3.1 Tính số Lucas thứ n

int lucas(int n) {

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (n==0) return 2; //L0 = 2

if (n==1) return 1; //L1=1

else return lucas(n-1) + lucas(n-2); //L(n)=L(n-1)+L(n-2)

}

## **Bài 3.2**. Trên bàn cờ vua kích thước n\*n có một quân mã đang ở ô (1, 1). Hãy đưa ra một dãy các di chuyển của mã sao cho mỗi ô trên bàn cờ đều được đi qua đúng 1 lần (ô (1, 1) được xem là đã đi qua).

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, máy tính

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 2 Bài 3.2. Quân mã đi tuần

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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);

}

//# Thuật toán quay lui

void TRY(int k){

if (k == n\*n + 1) print\_sol();

for(int i = 0; i < 8; i++){

//di chuyển mã đến ô mới

int xx = X[k-1] + hx[i];

int yy = Y[k-1] + hy[i];

//kiểm tra ô có thỏa mãn không

if (xx>0 && yy>0 && xx<=n && yy<=n && mark[xx][yy] == 0)

{

X[k] = xx;

Y[k] = yy;

mark[xx][yy] = 1; //đánh dấu đã qua ô này

TRY(k+1); //đi tiếp

mark[xx][yy] = 0; //backtrack

}

}

}

int main(){

cin >> n;

mark[1][1] = 1;

X[1] = Y[1] = 1;

TRY(2);

return 0;

}

## **Bài 3.3**. Một người xuất phát tại thành phố 1, muốn đi thăm tất cả các thành phố khác, mỗi thành phố đúng 1 lần và quay về 1. Chi phí để đi từ thành phố i sang thành phố j là cij. Hãy tìm tổng chi phí nhỏ nhất có thể

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động Ảnh có chứa văn bản, biên lai, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Mô tả được tạo tự động

Hình 3 Bài 3.3 Nhánh cận người đi du lịch

#include <bits/stdc++.h>

using namespace std;

#define MAX 100

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int n, c[MAX][MAX]; //# số thành phố và ma trận chi phí

int cmin = INT\_MAX; //# chi phí đi lại nhỏ nhất giữa hai thành phố khác nhau

int best = INT\_MAX; //# tổng chi phí nhỏ nhất cần tìm, ban đầu đặt bằng giá trị vô cùng lớn INT\_MAX = 2^31-1

int curr; //# tổng chi phí tới thời điểm hiện tại

int mark[MAX]; //# đánh dấu những thành phố đã đi

int x[MAX]; //# lưu giữ các thành phố đã đi

//# Đọc dữ liệu vào

void input(){

cin >> n;

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

for (int j = 1; j <= n; ++j){

cin >> c[i][j];

if (c[i][j] > 0) cmin = min(cmin, c[i][j]);

}

}

//# Thuật toán quay lui

void TRY(int k){

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

if (mark[i] == 0) //nếu chưa đi qua thành phố này

{

x[k]=i; //đi qua thành phố này

mark[i]=1; //đánh dấu thành phố

curr+=c[x[k-1]][x[k]]; //tăng chi phí

if (k==n) //nếu đã đi qua tất cả các thành phố

{

//cập nhật chi phí min

if (curr+c[x[k]][1]<best) best = curr+c[x[k]][1];

}

else

{

//nếu nhỏ hơn cận dưới thì mới tiếp tục, không thì cắt nhánh

if (curr+cmin\*(n-k+1)<best) TRY(k+1);

}

//backtrack

curr-=c[x[k-1]][x[k]];

mark[i]=0;

}

}

}

int main() {

input();

x[1] = 1;

curr = 0;

TRY(2);

cout << best;

return 0;

}

## **Bài 3.4**. Cho dãy a có  n phần tử. Một dãy con của  a là dãy thu được bằng cách xóa đi một số phần tử của  a và giữ nguyên thứ tự các phần tử còn lại (có thể không xóa phần tử nào). Hãy tìm dãy con tăng dài nhất của a.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, đại số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Hình 4 Bài 3.4 Dãy con tăng dài nhất

#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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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]

int lis(int i) {

if (mem[i] != -1) return mem[i]; // trả về kết quả

int maxLength = 1; //Số phần tử khởi tạo của dãy con

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

if (a[j] < a[i]) {

maxLength = max(maxLength, 1 + lis(j)); //cập nhật

}

}

mem[i] = maxLength; // Lưu kết quả

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];

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;

}

## **Bài 3.5**. Tính hệ số tổ hợp C(n, k)

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động 

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, đen và trắng

Mô tả được tạo tự động 

*Expected:*

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động

*Got:* Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, tài liệu

Mô tả được tạo tự động 

Hình 5 Bài 3.5 Tính hệ số tổ hợp C(n,k)

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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 c[1000][1000];

int binom2(int n, int k){

//# Khử đệ quy

for (int i=0;i<=n;i++)

{

c[i][0] = 1;

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

{

c[i][j] = c[i-1][j] + c[i-1][j-1];

//C(n,k) = C(n-1,k)+C(n-1,k-1)

}

}

return c[n][k];

}

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;

}

## **Bài 3.6.**Tìm ước chung lớn nhất của hai số nguyên a, b cho trước.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, biểu đồ

Mô tả được tạo tự động

Hình 6 Bài 3.6 Tìm ước chung lớn nhất

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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 (a!=b)

{

//tìm ước chung nhỏ nhất

if (a>b) a-=b;

if (a<b) b-=a;

}

return a;

}

int main() {

int a, b;

cin >> a >> b;

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

return 0;

}

## **Bài 3.7.**Sử dụng phương pháp khử đệ quy bằng stack, hãy liệt kê các xâu nhị phân độ dài n không có k bit 1 nào liên tiếp

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, số, màn hình

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, số

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, hình vuông

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, sách, Hình chữ nhật

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, hình vuông

Mô tả được tạo tự động Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, hình vuông

Mô tả được tạo tự động

Hình 7 Bài 3.7 Liệt kê xâu nhị phân không có k bit 1 nào liên tiếp

#include <bits/stdc++.h>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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's

int L = 0;

s.push(state(1, 0, 0)); // Push initial state with old\_L set to 0

while (!s.empty()) {

state &top = s.top();

// Nếu tìm thấy một dãy nhị phân mới

if (top.i > n) {

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

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

s.pop();

continue;

}

// Khôi phục giá trị L cũ nếu j > 0

if (top.j > 0) L = top.old\_L;

// Nếu đã duyệt qua tất cả các giá trị con

if (top.j > 1) {

s.pop();

continue;

}

// Chỉ tiếp tục nếu ràng buộc cho phép

if (L + 1 < k || top.j == 0) {

x[top.i] = top.j; // Gán giá trị hiện tại (0 hoặc 1) vào dãy

top.old\_L = L; // Lưu giá trị L hiện tại

L = top.j ? L + 1 : 0; // Cập nhật L cho trạng thái tiếp theo

s.push(state(top.i + 1, 0, L)); // Đẩy trạng thái tiếp theo vào stack

}

++top.j; // Chuyển sang giá trị tiếp theo cho vị trí hiện tại (0 sang 1)

}

return 0;

}

# **BÀI TẬP VỀ NHÀ**

Chụp ảnh kết quả của tất cả các test.

## **Bài tập 3.9:** Lập lịch cho y tá

Một y tá cần lập lịch làm việc trong *N* ngày, mỗi ngày chỉ có thể là làm việc hay nghỉ ngơi. Một lịch làm việc là tốt nếu không có hai ngày nghỉ nào liên tiếp và mọi chuỗi ngày tối đại làm việc liên tiếp đều có số ngày thuộc đoạn [*K*1,*K*2]. Hãy liệt kê tất cả các cách lập lịch tốt, với mỗi lịch in ra trên một dòng một xâu nhị phân độ dài *n* với bit 0/1 tương ứng là nghỉ/làm việc. Các xâu phải được in ra theo thứ tự từ điển

**Dữ liệu vào:**

Ghi 3 số nguyên *N*,*K*1,*K*2 (*N*≤200,*K*1<*K*2≤70)

**Kết quả:**

Ghi danh sách các lịch tìm được theo thứ tự từ điển

**Ví dụ:**

**Dữ liệu mẫu:**

6 2 3

**Kết quả mẫu:**

011011  
110110  
110111  
111011

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, văn bản, màu đen, Phần mềm đa phương tiện

Mô tả được tạo tự động

***Case 1:***

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động

***Case 2:***

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động

***Case 3:***

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Ảnh có chứa ảnh chụp màn hình, văn bản, màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

Hình 8 Bài 3.9

#include <iostream>

using namespace std;

int n,k1,k2,x[201];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void inp()

{

cin>>n>>k1>>k2;

}

void out()

{

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

{

cout<<x[i];

}

cout<<"\n";

}

bool check(int pos,int value, int count) //kiểm tra tại ngày pos, có được nghỉ hay không

{ //với số ngày đã làm việc là count

if (pos==1) return true; //ngày đầu tiên có thể nghỉ hoặc làm

else //các ngày khác

{

if (value == 0) //nếu như muốn nghỉ

{

return count>=k1; //phải thỏa mãn số ngày đã đi làm >= k1

}

if (value == 1) //nếu muốn đi làm tiếp

{

return count<k2; //phải thỏa mãn số ngày đi làm < k2

}

}

return false;

}

void Try(int pos, int count) //quay lui

{

for (int value=0;value<=1;value++)

{

if (check(pos,value,count)){

x[pos] = value;

int new\_count=(x[pos] == 1)?(count+1):0; //cập nhật số ngày đã đi làm

if (pos == n)

{

if (x[pos] == 0|| new\_count>=k1) out(); //đủ n ngày, in kết quả

}

else Try(pos+1,new\_count);

}

}

}

int main()

{

inp();

Try(1,0);

return 0;

}

## **Bài tập 3.10:** Khoảng cách Hamming

Khoảng cách Hamming giữa hai xâu cùng độ dài là số vị trí mà ký tự tại vị trí đó là khác nhau trên hai xâu. Cho *S* là xâu gồm *n* ký tự 0. Hãy liệt kê tất cả các xâu nhị phân độ dài *n*, có khoảng cách Hamming với *S* bằng *H*. Các xâu phải được liệt kê theo thứ tự từ điển

**Dữ liệu vào:**

Dòng đầu chứa *T* là số testcase

*T* dòng tiếp theo, mỗi dòng mô tả một testcase, ghi *N* và *H* (1≤*H*≤*N*≤16)

**Kết quả:**

Với mỗi testcase, in ra danh sách các xâu thỏa mãn. In ra một dòng trống giữa hai testcase

**Ví dụ:**

**Dữ liệu mẫu:**

2  
4 2  
1 0

**Kết quả mẫu:**

0011  
0101  
0110  
1001  
1010  
1100

0

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa ảnh chụp màn hình, màu đen

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động









Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 9. Bài 3.10

#include <iostream>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int n,h,x[20];

void inp()

{

cin>>n>>h;

}

void out()

{

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

{

cout<<x[i];

}

cout<<"\n";

}

void Try(int pos,int count) //quay lui-nhánh cận in ra xâu nhị phân độ dài n có h số 1

{

for (int i=0;i<=1;i++)

{

x[pos] = i;

int new\_count=count;

if (x[pos] == 1){

new\_count = count+1;

}

if (pos==n){

if (new\_count == h)out(); //nếu độ dài là n mà có h số 1 thì in ra

}

else if (n-pos>=h-new\_count) Try(pos+1,new\_count); //nếu vị trí trống còn lại không đủ cho số lượng số 1, cắt nhánh

}

}

int main()

{

int t;

cin>>t;

while(t--)

{

inp();

Try(1,0);

}

}

## **Bài tập 3.11:** Đoàn người du lịch

Superior là một hòn đảo tuyệt đẹp với *n* địa điểm chụp ảnh và các đường một chiều nối các điểm chụp ảnh với nhau. Đoàn khách tham quan có *r* người với sở thích chụp ảnh khác nhau. Theo đó, mỗi người sẽ đưa ra danh sách các địa điểm mà họ muốn chụp. Bạn cần giúp mỗi người trong đoàn lập lịch di chuyển sao cho đi qua các điểm họ yêu cầu đúng một lần, không đi qua điểm nào khác, bắt đầu tại điểm đầu tiên và kết thúc tại điểm cuối cùng trong danh sách mà họ đưa ra, và có tổng khoảng cách đi lại là nhỏ nhất.

**Dữ liệu vào:**

Dòng đầu chứa *n* và *r*

Tiếp theo là ma trận *n*×*n*mô tả chi phí đi lại giữa các địa điểm. Chi phí bằng 0 có nghĩa là không thể đi lại giữa hai địa điểm đó.

*r* dòng tiếp theo chứa danh sách các địa điểm mà người *r* đưa ra. Lưu ý là hành mỗi hành trình cần phải bắt đầu và kết thúc bởi hai đỉnh đầu và cuối của danh sách, còn các địa điểm còn lại có thể thăm theo bất kỳ thứ tự nào.

**Kết quả:**

Gồm *r* dòng ghi chi phí đi lại ít nhất của *r* người theo thứ tự đầu vào.

**Ví dụ:**

**Dữ liệu mẫu:**

6 3  
0 1 2 0 1 1  
1 0 1 1 1 0  
0 2 0 1 3 0  
4 3 1 0 0 0  
0 0 1 1 0 0  
1 0 0 0 0 0  
1 3 5  
6 3 2 5  
6 1 2 3 4 5

**Kết quả mẫu:**

5  
0  
7

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phần mềm đa phương tiện, phần mềm

Mô tả được tạo tự độngẢnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

Hình 10. Bài 3.11

#include <bits/stdc++.h>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define inf 10000

int n,r,c[101][101],cmin=inf,f,fopt,x[101],visited[101];

void inp() //nhập, tìm chi phí tối thiểu cmin

{

cin>>n>>r;

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

{

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

{

cin>>c[i][j];

if(c[i][j]>0) cmin=min(c[i][j],cmin);

}

}

}

void reset() //reset giá trị mảng lưu đường x, mảng visited

{

for (int i=0;i<=n;i++)

{

x[i]=0;

visited[i]=0;

}

}

void Try(int k,const vector<int>&a) //quay lui

{

for (int i=1;i<a.size()-1;i++) //xét các thành phố không phải đầu và cuối

{

if (!visited[a[i]] && c[x[k-1]][a[i]] != 0) //nếu chưa đi qua thành phố đó và có đường đi đến thành phố đó

{

x[k]=a[i]; //đi qua thành phố này

visited[a[i]]=1; //đánh dấu đã đi qua

f += c[x[k-1]][x[k]]; //tăng chi phí hiện tại

if (k == a.size()-2) //nếu đã đi đến thành phố gần cuối

{

if (c[x[k]][a.back()] != 0 && f + c[x[k]][a.back()] < fopt)

{

fopt = f + c[x[k]][a.back()]; //nếu có thể đi đến thành phố cuối cùng, cập nhật chi phí nếu có thể

}

}

else

{

if(f + cmin\*(a.size()-k) < fopt) Try(k+1,a); //nếu chi phí cận dưới lớn hơn chi phí tối ưu, cắt nhánh

}

f-=c[x[k-1]][x[k]]; //backtrack

visited[a[i]]=0;

}

}

}

int main()

{

inp();

cin.ignore();

vector<vector<int>> vt(r);

string line;

for (int i=0;i<r;i++) //nhập danh sách địa điểm

{

getline(cin,line);

stringstream ss(line);

int value;

while(ss>>value)

{

vt[i].push\_back(value);

}

}

for (int i=0;i<r;i++) //áp dụng thuật toán cho từng người

{

x[0] = vt[i][0];

visited[vt[i][0]]=1;

f=0;

fopt=inf;

Try(1,vt[i]);

cout<<((fopt<inf)?fopt:0)<<"\n";

reset(); //reset các mảng trạng thái sau khi xong

}

return 0;

}

## **Bài tập 3.12:** Đếm số đường đi đơn độ dài k

Cho đồ thị vô hướng *G*, hãy đếm số đường đi đi qua *k* cạnh và không đi qua đỉnh nào quá một lần.

**Dữ liệu vào:**

Dòng 1: Chứa hai số nguyên *n* và *k* (1≤*n*≤30, 1≤*k*≤10) với *n* là số đỉnh của *G*. Các đỉnh sẽ được đánh số từ 1 đến *n*

Dòng 2: Chứa số nguyên *m*(1≤*m*≤60) là số cạnh của *G*;

*m* dòng tiếp theo: Mỗi dòng chưa hai số nguyên là một cạnh của *G*

**Kết quả:**

Số lượng đường đi đơn độ dài *k*

**Ví dụ:**

**Dữ liệu mẫu:**

4 3  
5  
1 2  
1 3  
1 4  
2 3  
3 4

**Kết quả mẫu:**

6

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

Hình 11. Bài 3.12

#include <bits/stdc++.h>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nguyễn Kim Cường-20224939-738918-2023.2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int n, k, m, pass,res=0,x[101];

vector<vector<int>> ke;

vector<bool> visited;

void inp() {

cin >> n >> k;

cin >> m;

ke.resize(n + 1);

visited.resize(n + 1, false);

for (int i = 0; i < m; i++) { //nhập vào danh sách kề

int x, y;

cin >> x >> y;

ke[x].push\_back(y);

ke[y].push\_back(x);

}

}

void reset() //reset các mảng trạng thái

{

for (int i=0;i<=n;i++)

{

x[i]=0;

visited[i]=false;

}

}

void Try(int pos)

{

for (auto v : ke[x[pos-1]]) //xét mỗi cạnh kề của đỉnh vừa mới thăm

{

if (!visited[v]) //nếu chưa thăm đỉnh đó

{

x[pos] = v; //thăm

visited[v] = true; //đánh dấu đã thăm

pass+=1; //tăng số cạnh đã qua

if (pass == k) res++;//nếu đã đi qua k cạnh, tăng số đường đi lên 1

else Try(pos+1); //đệ quy

visited[v]=false; //backtrack

pass--;

}

}

}

int main()

{

inp();

for (int i=1;i<=n;i++) //lặp với từng đỉnh trong đồ thị

{

x[1]=i;

pass=0;

visited[i]=true;

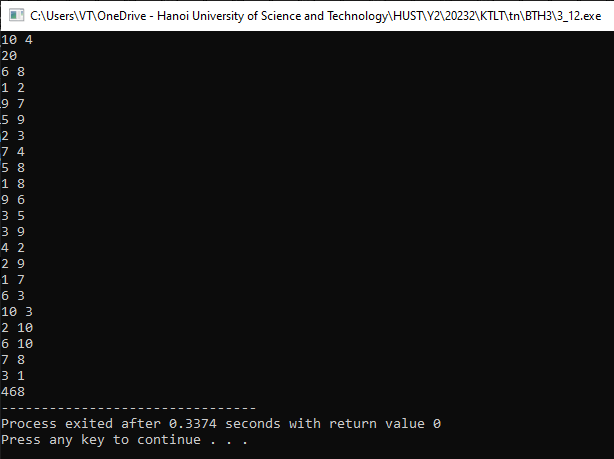
Try(2);

reset(); //reset giá trị sau khi tính xong số đường đi đơn từ đỉnh đó

}

cout<<res/2; //chia 2 vì mỗi đường đi được tính 2 lần

}

***Case 1:***

***Case 2:***

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

***Case 3:*** Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

***Case 4:***

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

***Case 5:***

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

***Case 6:***

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động

***Case 7:***

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

***Case 8:***

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động

***Case 9:***

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự độngẢnh có chứa văn bản, ảnh chụp màn hình, màn hình

Mô tả được tạo tự động