Contents

[Bài thực hành số 3 – Tuần 36 2](#_Toc137684606)

[**Bài tập 3.1.**Dãy Lucas được định nghĩa bởi Ln=Ln−1+Ln−2 và bắt đầu bởi L0=2, L1=1 . Viết hàm tính số Lucas thứ n. 2](#_Toc137684607)

[**Bài tập 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) 4](#_Toc137684608)

[**Bài tập 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 ci,j. Hãy tìm tổng chi phí nhỏ nhất có thể 11](#_Toc137684609)

[**Bài tập 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 16](#_Toc137684610)

[**Bài tập 3.5.** Tính Ckn 21](#_Toc137684611)

[**Bài tập 3.6.** Tính ước chung lớn nhất của hai số cho trước 26](#_Toc137684612)

[**Bài tập 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 29](#_Toc137684613)

[**Bài tập 3.8.**Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi. Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới. 36](#_Toc137684614)

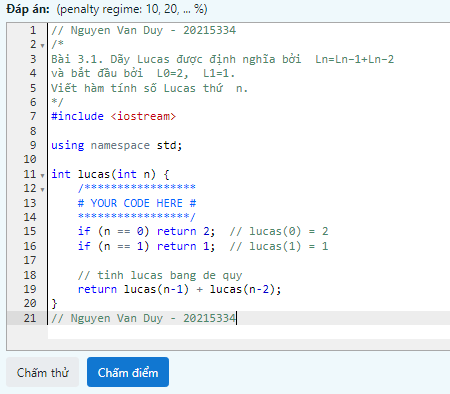
[**Bài tập 3.9.**Một y tá cần lập lịch làm việc trong N ngày, mỗi ngày chỉ có thể là làm việc hay nghỉ ngơi. Một lịch làm việc là tốt nếu không có hai ngày nghỉ nào liên tiếp và mọi chuỗi ngày tối đại làm việc liên tiếp đều có số ngày thuộc đoạn [K1,K2]. Hãy liệt kê tất cả các cách lập lịch tốt, với 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 41](#_Toc137684615)

[**Bài tập 3.10.** 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 46](#_Toc137684616)

[**Bài tập 3.11.** 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. 50](#_Toc137684617)

# Bài thực hành số 3 – Tuần 36

## **Bài tập 3.1.**Dãy Lucas được định nghĩa bởi Ln=Ln−1+Ln−2 và bắt đầu bởi L0=2, L1=1 . Viết hàm tính số Lucas thứ n.



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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 3.1. Dãy Lucas được định nghĩa bởi Ln=Ln−1+Ln−2

và bắt đầu bởi L0=2, L1=1.

Viết hàm tính số Lucas thứ n.

\*/

#include <iostream>

using namespace std;

int lucas(int n) {

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

# YOUR CODE HERE #

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

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

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

// tinh lucas bang de quy

return lucas(n-1) + lucas(n-2);

}

// Nguyen Van Duy - 20215334

## **Bài tập 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, màn hình, số

Mô tả được tạo tự động

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

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, Phông chữ, ảnh chụp màn hình, hàng

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, Hình chữ nhật, Song song

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

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 chunky 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)

\*/

#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);

}

//# Thuật toán quay lui

void TRY(int k){

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

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

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

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

# YOUR CODE HERE #

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

if (0 < xx && xx <= n && 0 < yy && yy <= n && mark[xx][yy] != 1) {

// neu (xx yy) van nam trong o co

// va

// vi tri do chua duoc di

X[k] = xx; // luu toa do x cua quan ma

Y[k] = yy; // luu toa do y cua quan ma

mark[xx][yy] = 1; // danh dau o co da duoc di

} else {

// backtrack

continue;

}

if (k == n\*n) {

// ghi nhan ket qua

print\_sol();

} else {

// tim buoc di tiep theo

TRY(k+1);

}

// bo danh dau

mark[xx][yy] = 0;

}

}

int main(){

cin >> n;

mark[1][1] = 1;

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

TRY(2);

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 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 ci,j. Hãy tìm tổng chi phí nhỏ nhất có thể

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, màn hình, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

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 ci,j.

Hãy tìm tổng chi phí nhỏ nhất có thể

\*/

#include <bits/stdc++.h>

using namespace std;

#define MAX 100

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++){

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

# YOUR CODE HERE #

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

if(mark[i] == 0 && c[x[k-1]][i] != 0){

x[k] = i; // luu thanh pho da di

mark[i] = 1; // danh dau thanh pho i da duoc di qua

curr += c[x[k-1]][i]; // cap nhat chi phi khi di qua thanh pho i

if(k == n){

// neu da di het n thanh pho

if(c[x[k]][1] != 0){

// va

// thanh pho cuoi cung di ve lai duoc thanh pho ban dau

best = min(best, curr + c[x[k]][1]); // cap nhat ket qua

}

}else{

int g = curr + (n-k+1)\*cmin; // chi phi toi thieu neu tiep tuc di

if(g < best){

// neu co the nho hon chi phi tot nhat cua hien tai

// thi tiep tuc

TRY(k+1);

}

}

mark[i] = 0; // bo danh dau

curr -= c[x[k-1]][i]; // xoa chi phi da di

}

}

}

int main() {

input();

x[1] = 1;

TRY(2);

cout << best;

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 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, đồ điện tử, ảnh chụp màn hình, 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

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

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

\*/

#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]

int lis(int i) {

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

# YOUR CODE HERE #

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

if (mem[i] != -1) {

// tra ve ket qua da duoc giai

return mem[i];

}

int res = 1; //# luu loi giai cua bai toan

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

// neu day con truoc do < a[i]

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

// res = day con dai nhat phia truoc + 1

res = max(res, lis(j) + 1);

}

}

mem[i] = res; // luu loi giai cua bai toan con da duoc giai

return res;

}

//# 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;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.5.** Tính Ckn

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, Phông chữ, số

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, số

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 3.5. Tính Ckn

\*/

#include <iostream>

using namespace std;

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){

if (k > n) return 0;

//# Khử đệ quy

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

# YOUR CODE HERE #

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

long long int giaiThua\_n = 1; // = n!

long long int giaiThua\_nk = 1; // = (n-k)!

long long int giaiThua\_k = 1; // = k!

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

giaiThua\_n \*= i;

if (i <= n-k) {

giaiThua\_nk \*= i;

}

if (i <= k) {

giaiThua\_k \*= i;

}

}

// binom(n, k) = n! / ((n-k)! \* k!)

return giaiThua\_n / (giaiThua\_nk \* giaiThua\_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;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.6.** Tính ước chung lớn nhất của hai số cho trước

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, số, Phông chữ

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 3.6. Tính ước chung lớn nhất của hai số cho trước

\*/

#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

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

# YOUR CODE HERE #

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

// lap den khi a = 0 hoac b = 0

while(a && b) {

// so lon = so lon % so be

if (a > b) {

a = a % b;

} else {

b = b % a;

}

}

// tra ve so khac 0 trong hai so a va b

return a ? a : b;

}

int main() {

int a, b;

cin >> a >> b;

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

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 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, màn hình, phần mềm

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, màn hình, số

Mô tả được tạo tự động



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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Hình chữ nhật, sách

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

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

\*/

#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

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

# YOUR CODE HERE #

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

// luu cac gia tri o ngan dau tien stack

int i = top.i, j = top.j, old\_L = top.old\_L;

s.pop();

x[i] = j; // ghi gia tri j vao vi tri thu i

if (j) {

// neu j = 1 thi so bit 1 o cuoi day tang them 1

L = old\_L + 1;

} else {

// neu j = 0 thi khong co bit 1 nao o cuoi day

L = 0;

if (old\_L + 1 < k) {

// dua vao stack 1 state (ghi gia tri 1 vao vi tri thu i)

// thay the so 0 o vi tri hien tai thanh so 1

s.push(state(i, 1, old\_L));

}

}

// dua vao stack 1 state (ghi gia tri 0 vao vi tri thu i+1)

// vi tri tiep theo se nhan gia tri 0

s.push(state(i + 1, 0, L));

}

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.8.**Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không. Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi. Hãy chỉ ra một cách cân thỏa mãn. Quy cách in ra đã được tích hợp trong mã nguồn dưới.

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

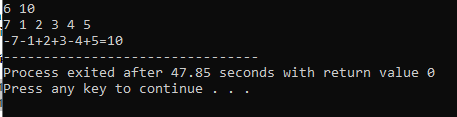
Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động



// Nguyen Van Duy - 20215334

/\*

Bài 3.8. Bạn đang muốn kiểm tra xem một vật cho trước có đúng nặng M như người ta nói hay không.

Có một cân thăng bằng và n quả cân. Quả thứ i nặng mi. Hãy chỉ ra một cách cân thỏa mãn.

Quy cách in ra đã được tích hợp trong mã nguồn dưới.

\*/

#include <bits/stdc++.h>

using namespace std;

struct state{

int i, j;

state(int \_i = 0, int \_j = 0): i(\_i), j(\_j) {}

};

int main() {

int n, M;

cin >> n >> M;

int m[n+1];

for (int i = 1; i <= n; ++i) cin >> m[i];

int x[n+1];

stack<state> s;

//# sum of selected weights

int sum = 0;

s.push(state(1, -1));

while (!s.empty()){

state &top = s.top();

if (top.i > n){

if (sum == M){

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

if (x[i] == -1) cout << '-' << m[i];

if (x[i] == 1) cout << '+' << m[i];

}

cout << "=" << M;

exit(0);

}

s.pop();

continue;

}

//# Khử đệ quy

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

# YOUR CODE HERE #

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

// luu cac gia tri o ngan dau tien stack

int i = top.i, j = top.j;

s.pop();

// neu truong hop truoc do da tung su dung qua can i

if (x[i] == 1) {

// huy bo trang thai truoc do

sum -= m[i]; // tinh lai sum

}

x[i] = j; // ghi gia tri j vao vi tri thu i (1 hoac -1)

if (j == 1) {

// su dung qua can thu i

sum += m[i];

} else {

// j = -1

// khong su dung qua can i

s.push(state(i, 1)); // dua vao stack 1 state (su dung qua can i)

}

// dua vao stack 1 state (khong su dung qua can i)

s.push(state(i + 1, -1));

}

cout << -1;

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.9.**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 [K1,K2]. 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

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

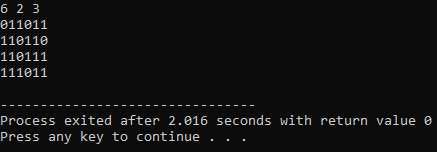
Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động



// Nguyen Van Duy - 20215334

/\*

Bài 3.9. 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 [K1,K2].

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

\*/

#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, k1, k2;

cin >> n >> k1 >> k2;

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){

if (x[n] && top.old\_L >= k1 || !x[n]) {

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

cout << x[i];

cout << endl;

}

s.pop();

continue;

}

//# Khử đệ quy

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

# YOUR CODE HERE #

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

// luu cac gia tri o ngan dau tien stack

int i = top.i, j = top.j, old\_L = top.old\_L;

s.pop();

x[i] = j; // ghi trang thai lam viec ngay i

if (j) {

// case: j = 1

L = old\_L + 1; // so ngay lam viec lien tiep tang them 1

if (L >= k1) {

// dua vao stack 1 state (nghi vao ngay tiep theo)

s.push(state(i + 1, 0, L));

} else {

// dua vao stack 1 state (di lam vao ngay tiep theo)

s.push(state(i + 1, 1, L));

}

} else {

// case: j = 0

L = 0; // reset so ngay lam viec truoc do

if (old\_L + 1 <= k2) {

// chuoi ngay lam viec truoc do chua toi da

// dua vao stack 1 state (di lam vao ngay hien tai)

s.push(state(i, 1, old\_L));

}

// dua vao stack 1 state (di lam vao ngay tiep theo)

s.push(state(i+1, 1, L));

}

}

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.10.** 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

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

Mô tả được tạo tự động

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, Phần mềm đa phương tiện

Mô tả được tạo tự động

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

Mô tả được tạo tự động

// Nguyen Van Duy - 20215334

/\*

Bài 3.10. 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

\*/

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

inline void print\_sol(const vector<int>& v) {

for (int i : v) {

cout << i;

}

cout << endl;

}

int main() {

int t;

cin >> t;

while (t--) {

int n, h;

cin >> n >> h;

vector<int> v(n);

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

v[i] = i < h ? 1 : 0;

}

vector<int> v\_end(n);

v\_end = v;

do {

next\_permutation(v.begin(), v.end());

print\_sol(v);

} while (v != v\_end);

}

return 0;

}

// Nguyen Van Duy - 20215334

## **Bài tập 3.11.** 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.

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, Phần mềm đa phương tiện

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

Mô tả được tạo tự động

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

// Nguyen Van Duy - 20215334

/\*

Bài 3.11. 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.

\*/

#include <iostream>

#include <vector>

#include <sstream>

#include <stack>

#include <cstring>

using namespace std;

#define MAX 100

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

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

int r;

inline void show\_curr() {

cout << "curr : " << curr << endl;

}

//# Đọc dữ liệu vào

inline void input(){

cin >> n >> r;

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

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

cin >> c[i][j];

}

struct state {

int i;

int j;

state(int i, int j) : i(i), j(j) {}

};

inline void show\_x(size\_t n) {

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

cout << x[i] << " ";

}

cout << " <- x[]" << endl;

}

inline void write(int i, int j) {

mark[j] = 1;

if (i == 1) {

x[i] = j;

return;

}

if (j) {

x[i] = j;

curr += c[x[i-1]][x[i]];

return;

} else {

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

mark[x[i]] = 0;

x[i] = j;

return;

}

}

int main() {

input();

cin.ignore();

while (r--) {

string line;

getline(cin, line);

stringstream ss(line);

vector<int> v;

v.push\_back(0);

while (!ss.eof()) {

int temp;

ss >> temp;

v.push\_back(temp);

}

stack<state> s;

memset(mark, 0, sizeof(mark));

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

curr = 0;

bool not\_found = true;

s.emplace(1, v[1]);

while (!s.empty()) {

state &top = s.top();

int i = top.i, j = top.j;

s.pop();

write(i, j);

// show\_x(v.size());

if (!j) continue;

if (i == v.size() - 2 && c[j][v.back()]) {

write(++i, v.back());

// cout << "--";

// show\_x(v.size());

// cout << ">> ";

// show\_x(v.size());

if (curr < best) {

best = curr;

not\_found = false;

}

write(i, 0);

// cout << "--";

// show\_x(v.size());

continue;

}

for (int k = 1; k < v.size() - 1; ++k) {

if (!mark[v[k]] && c[j][v[k]]) {

s.emplace(i+1, 0);

s.emplace(i+1, v[k]);

}

}

}

if (not\_found) {

cout << "--> " << 0 << endl << endl << endl;

} else {

cout << "--> " << best << endl << endl << endl;

}

}

return 0;

}

// Nguyen Van Duy - 20215334