Nội dung

[BÀI THỰC HÀNH SỐ 5 – TUẦN 17 2](#_Toc186408664)

[BÀI TẬP TRÊN LAB 2](#_Toc186408665)

[Bài 5.1: Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn. 2](#_Toc186408666)

[Bài 5.2: Bài toán cái túi: 4](#_Toc186408667)

[Bài 5.3: Dãy ngoặc đúng 7](#_Toc186408668)

[Bài 5.4: Bài toán người du lịch 10](#_Toc186408669)

[Bài 5.5: Năm nhuận 13](#_Toc186408670)

[Bài 5.6: Một lớp có n sinh viên. Sinh viên thứ i có điểm tổng kết là ai theo thang điểm 10. 14](#_Toc186408671)

[Bài 5.7: Sau đại dịch, thầy trò Đường Tăng muốn xin tiền của các nhà giàu để chia cho các nhà nghèo. 16](#_Toc186408672)

[BÀI TẬP VỀ NHÀ 18](#_Toc186408673)

[Bài 5.8: Cắt hình chữ nhật 18](#_Toc186408674)

[Bài 5.9: Xây tháp 22](#_Toc186408675)

# BÀI THỰC HÀNH SỐ 5 – TUẦN 17

## BÀI TẬP TRÊN LAB

### Bài 5.1: Đoạn code sau liệt kê tất cả các hoán vị n số. Hãy tìm và sửa các lỗi cú pháp như hướng dẫn.

Lỗi: A computer screen with white and green text

Description automatically generated

Lời giải: Comment trong code

Output:

A computer screen with text

Description automatically generated

#include <stdio.h>

int x[100], mark[100], n;

void print(){

    for (int i = 1; i <= n; ++i) printf("%d ", x[i]);

    //ở đây cần in xuống dòng nên lỗi viết sai

    printf("\n");

}

void process(int i) {

    if (i > n){

        //ở đây cần gọi hàm nên viết sai

        print();

        return;

    }

    for (int j = 1; j <= n; ++j)

        if (!mark[j]){

            mark[j] = 1;

            x[i] = j;

            process(i+1);

            mark[j] = 0;

        }

}

int main() {

    n = 5;

    process(1);

    return 0;

}

### Bài 5.2: Bài toán cái túi:

Cho một cái túi có sức chứa M và n đồ vật. Đồ vật thứ i có khối lượng mi và giá trị vi. Cần chọn ra một số đồ vật để bỏ vào túi sao cho tổng khối lượng không quá M và tổng giá trị là lớn nhất có thể. Đoạn code sau đây giải bài toán cái túi bằng phương pháp duyệt nhánh cận. Hãy tìm và sửa các lỗi cú pháp.

Lỗi:

A computer screen shot of a program code

Description automatically generated

Lời giải: Comment trong code

Output:

A screen shot of a computer

Description automatically generated

#include <iostream>

using namespace std;

int n, M, m[100], v[100];

int x[100], best, sumV, sumM, All[100];

//một số lỗi gọi sai tên khi không viết hoa (all -> All)

void init(){

    All[n+1] = 0; // Khởi tạo All[n+1] bằng 0

    for (int i = n; i >= 1; --i){

        All[i] = All[i+1] + v[i];

    }

}

void print() {

    cout << best;

}

void process(int i){

    //ham void không được trả về giá trị

    if (sumV + All[i] <= best || sumM > M) return;

    if (i > n){

        best = sumV;

        return; //ham void không được trả về giá trị

    }

    process(i+1);

    sumM += m[i];

    sumV += v[i];

    process(i+1);

    sumM -= m[i];

    sumV -= v[i];

}

int main() {

    cin >> n >> M;

    for (int i = 1; i <= n; ++i)

        cin >> m[i] >> v[i];

    init();

    process(1);

    print();

    return 0;

}

### Bài 5.3: Dãy ngoặc đúng

Viết một chương trình nhận vào một dãy dấu ngoặc và kiểu tra xem dãy dấu ngoặc đóng mở đúng chưa.

Đề bài: <http://codeforces.com/group/Ir5CI6f3FD/contest/269186/problem/H>

Mã nguồn dưới đây là của một sinh viên, khi submit bị lỗi runtime (Exit code is -1073741819). Sử dụng công cụ debug ở trên, hãy tìm và sửa các lỗi trong mã nguồn.

Output ban đầu:

A computer screen with text

Description automatically generated

A screenshot of a computer

Description automatically generated

Cách sửa lỗi: Code đang bị lỗi chính ở phần kiểm tra ngăn xếp nên code không ra kết quả.

Sửa lỗi chi tiết trong comment ở từng dòng code mới.

Output mới:

A screen shot of a computer

Description automatically generated

#include <iostream>

#include <stack>

using namespace std;

int par(string str) {

    int a = str.length();

    stack<char> S;

    char x; //biến y không cần thiết

    for (int i = 0; i < a; i++) {

        x = str[i];

        if (x == '(' || x == '[' || x == '{') {

            S.push(x);

        } else {

            // Kiểm tra ngăn xếp rỗng trước khi gọi top()

            if (S.empty()) return 0; // Nếu ngăn xếp rỗng mà gặp dấu đóng, trả về 0

            if (x == ')') {

                if (S.top() == '(') {

                    S.pop();

                } else return 0;

            } else if (x == ']') {

                if (S.top() == '[') {

                    S.pop();

                } else return 0;

            } else if (x == '}') {

                if (S.top() == '{') {

                    S.pop();

                } else return 0;

            }

        }

    }

    // Kiểm tra ngăn xếp rỗng sau khi duyệt hết chuỗi

    //S.top() != NULL không đúng, vì top() trả về giá trị của phần tử đầu ngăn xếp, không phải là con trỏ

    if (!S.empty()) { // Nếu ngăn xếp không rỗng, trả về 0

        return 0;

    } else {

        return 1;

    }

}

int main() {

    int n;

    string str;

    cin >> n;

    for (int i = 0; i < n; i++) {

        cin >> str;

        cout << par(str) << endl;

    }

    return 0;

}

### Bài 5.4: Bài toán người du lịch

Ảnh có chứa văn bản

Mô tả được tạo tự động

Dưới đây là solution của một bạn sinh viên, khi submit bị sai kết quả. Hãy sử dụng hướng dẫn phía trên và thuật toán trực tiếp (được cho phía dưới) để tìm ra một test sai.

Output ban đầu:

A computer screen with text

Description automatically generated

Kết quả 100000 là sai.

Cách sửa lỗi: Trong code cũ có 1 phần chưa tính đến đồ thị vô hướng và khai báo mặc định đang không an toàn. Testcase khi không tìm được đường đi.

Output mới:

A computer screen shot of a computer code

Description automatically generated

#include <bits/stdc++.h>

using namespace std;

int m, n, Smin = INT\_MAX; // Sử dụng INT\_MAX để khởi tạo giá trị lớn nhất

long long S = 0;

int cmin = INT\_MAX; // Sử dụng INT\_MAX để khởi tạo giá trị lớn nhất

int x[100];

int c[100][100];

vector<int> flag(100, false);

void TRY(int k) {

    for (int i = 2; i <= n; i++) {

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

            flag[i] = true;

            x[k] = i;

            S += c[x[k - 1]][i];

            if (k == n) {

                if (S + c[i][1] < Smin && c[i][1] != -1)

                    Smin = S + c[i][1];

            } else if (S + cmin \* (n - k + 1) < Smin) {

                TRY(k + 1);

            }

            flag[i] = false;

            S -= c[x[k - 1]][i];

        }

    }

}

int main() {

    int a, b, cost;

    cin >> n >> m;

    for (int i = 1; i <= n; i++)

        for (int j = 1; j <= n; j++)

            c[i][j] = (i == j) ? 0 : -1; // Khởi tạo ma trận chi phí

    for (int i = 0; i < m; i++) {

        cin >> a >> b >> cost;

        c[a][b] = cost;

        c[b][a] = cost; // Nếu đồ thị vô hướng

        if (cost < cmin)

            cmin = cost;

    }

    x[1] = 1;

    flag[1] = true;

    TRY(2);

    cout << Smin;

    return 0;}

### Bài 5.5: Năm nhuận

Một năm được coi là nhuận nếu hoặc nó chia hết cho 4 nhưng không chia hết cho 100, hoặc nó chia hết cho 400. Cho một danh sách các năm, kiểm tra xem có tồn tại năm nhuận trong danh sách đó hay không.

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

Code mới:

#include <iostream>

using namespace std;

int main() {

    int n;

    cin >> n;

    bool found = false;

    while (n--) {

        int a;

        cin >> a;

        // Điều kiện kiểm tra năm nhuận

            if ((a % 400 == 0) || (a % 4 == 0 && a % 100 != 0)) {

            found = true;

            break; // Thoát khỏi vòng lặp khi tìm thấy năm nhuận

        }

    }

    cout << (found ? "Yes" : "No") << endl; // In kết quả dùng toán tử 3 ngôi

    return 0;

}

Lợi ích: Cho điều kiện dễ lên trước vì nó sẽ được kiểm tra trước.

Dùng break thoát vòng lặp, hạn chế vét cạn.

Dùng toán tử điều kiện, loại bỏ câu lệnh if-else.

### Bài 5.6: Một lớp có n sinh viên. Sinh viên thứ i có điểm tổng kết là ai theo thang điểm 10.

Để đánh giá chất lượng dạy học, giảng viên muốn biết có bao nhiêu bạn đạt điểm A, B, C, D, F.  Quy đổi thang điểm được cho như sau:

a < 4: F

4 ≤ a < 5.5: D

5.5 ≤ a < 7: C

7 ≤ a < 8.5: B

8.5 ≤ a: A

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

Code mới:

#include <bits/stdc++.h>

using namespace std;

char cal(double a) {

    //dùng else if để giảm số lần kiểm tra điều kiện

    if (a < 4) return 'F';

    else if (a < 5.5) return 'D';

    else if (a < 7) return 'C';

    else if (a < 8.5) return 'B';

    else return 'A';

}

int main() {

    int n;

    cin >> n;

    vector<int> grades(5, 0); // Mảng lưu trữ số lượng các hạng: A, B, C, D, F

    while (n--) {

        double a;

        cin >> a;

        switch (cal(a)) {

            case 'A': ++grades[0]; break;

            case 'B': ++grades[1]; break;

            case 'C': ++grades[2]; break;

            case 'D': ++grades[3]; break;

            case 'F': ++grades[4]; break;

        }

    }

    cout << grades[0] << " " << grades[1] << " " << grades[2] << " " << grades[3] << " " << grades[4];

    return 0;

}

Lợi ích:

dùng else if để giảm số lần kiểm tra điều kiện

điểm số có thể là số thực nên dùng double

mảng lưu trữ số lượng các hạng: A, B, C, D, F giúp code dễ đọc hiểu hơn

dùng switch case thay vì if else tăng hiệu suất kiểm tra điều kiện

### Bài 5.7: Sau đại dịch, thầy trò Đường Tăng muốn xin tiền của các nhà giàu để chia cho các nhà nghèo.

Họ sẽ vào n thôn, thôn thứ i có ki nhà. Mỗi thôn họ sẽ quyết định xin tiền hay cho tiền, phụ thuộc vào đánh giá của họ về mức độ giàu nghèo ở đây. Nếu thôn i giàu, họ sẽ đi từng nhà trong số ki nhà này và xin ai,j tiền của nhà thứ j. Nếu thôn i nghèo, họ sẽ đi từng nhà trong số ki nhà này và phát ai,j tiền cho nhà thứ j. Hãy tính số tiền ít nhất họ phải mang theo để đảm bảo có thể phát đủ cho người nghèo (tức số tiền luôn không bị âm).

Mã nguồn sau giải quyết bài toán đó, hãy tinh chỉnh nó để tăng hiệu suất chương trình.

Code mới:

#include <bits/stdc++.h>

using namespace std;

int main() {

    int n;

    cin >> n;

    int ans = 0, sum = 0;

    while (n--) {

        int k, t;

        cin >> k >> t;

// đưa điều kiện ra ngoài vòng lặp

        if (t == 1) {

            while (k--) {

                int a;

                cin >> a;

                sum += a;

            }

        } else {

            while (k--) {

                int a;

                cin >> a;

                sum -= a;

            }

        }

// đưa phép tính ra ngoài vòng lặp

        ans = max(ans, -sum);

    }

    cout << ans;

    return 0;

}

Lợi ích:

Bỏ bớt tính toán trong vòng lặp, tăng hiệu suất, giảm tính toán.

## BÀI TẬP VỀ NHÀ

### Bài 5.8: Cắt hình chữ nhật

Đề bài: <http://codeforces.com/group/Ir5CI6f3FD/contest/276073/problem/G>

Phong là một nhà điêu khắc, ông có một tấm đá cẩm thạch hình chữ nhật kích thước W×H. Ông ta muốn cắt tấm đá thành các miếng hình chữ nhật kích thước W1×H1, W2×H2, …, WN×HN. Ông ta muốn cắt đến tối đa các mẫu kích thước có thể. Tấm đá có những vân đá cho nên không thể xoay khi sử dụng, có nghĩa là không thể cắt ra miếng B×A thay cho miếng A×B trừ khi A=B. Các miếng phải được cắt tại các điểm nguyên trên hàng cột và mỗi nhát cắt phải cắt đến hết hàng hoặc hết cột. Sau khi cắt sẽ còn lại những mẩu đá còn thừa bỏ đi, nghĩa là những mẩu đá không thể cắt thành miếng kích thước cho trước nào.

Yêu cầu: Hãy tìm cách cắt sao cho còn ít nhất diện tích đá thừa bỏ đi.

Input

Dòng đầu tiên chứa hai số nguyên: W và H.

Dòng thứ hai chứa một số nguyên N. N dòng tiếp theo mỗi dòng chứa hai số nguyên Wi và Hi.

Output

Kết quả ghi ra duy nhất một số nguyên là tổng diện tích nhỏ nhất các miếng thừa bỏ đi.

Scoring

* 1 ≤ W ≤ 600, 1 ≤ H ≤ 600, 0 < N ≤ 200, 1 ≤ Wi ≤ W and 1 ≤ Hi ≤ H.
* Có 50% số test ứng với W ≤ 20, H ≤ 20 và N ≤ 5.

Output ban đầu:

A computer screen shot of a computer code

Description automatically generated

Sửa lỗi: Thiếu giá trị trả về cho cin.tie()

Sửa lỗi: Tránh khai báo lại các biến w và h trong hàm main

Sửa lỗi: Vòng lặp cắt ngang phải dừng trước i

Sửa lỗi: Vòng lặp cắt dọc phải dừng trước j

Sửa lỗi: Thêm return 0 vào cuối hàm main để chỉ ra rằng chương trình đã kết thúc thành công

Output mới:

A computer screen with white text

Description automatically generated

#include <bits/stdc++.h>

using namespace std;

int w, h;

int table[601][601];

void init() {

    for (int i = 1; i <= h; i++) {

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

            table[i][j] = i \* j;

        }

    }

}

int main() {

    ios::sync\_with\_stdio(false);

    cin.tie(nullptr); // Sửa lỗi: Thiếu giá trị trả về cho cin.tie()

    cin >> w >> h; // Sửa lỗi: Tránh khai báo lại các biến w và h trong hàm main

    int m;

    cin >> m;

    init();

    for (int i = 0; i < m; i++) {

        int tmp1, tmp2;

        cin >> tmp1 >> tmp2;

        table[tmp2][tmp1] = 0;

    }

    // dp

    for (int i = 1; i <= h; i++) {

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

            int minWaste = table[i][j];

            // horizontal cut

            for (int k = 1; k < i; k++) { // Sửa lỗi: Vòng lặp cắt ngang phải dừng trước i

                minWaste = min(minWaste, table[k][j] + table[i - k][j]);

            }

            // vertical cut

            for (int k = 1; k < j; k++) { // Sửa lỗi: Vòng lặp cắt dọc phải dừng trước j

                minWaste = min(minWaste, table[i][k] + table[i][j - k]);

            }

            table[i][j] = minWaste;

        }

    }

    cout << table[h][w] << endl;

    return 0; // Sửa lỗi: Thêm return 0 vào cuối hàm main để chỉ ra rằng chương trình đã kết thúc thành công

}

### Bài 5.9: Xây tháp

Đề bài: <http://codeforces.com/group/Ir5CI6f3FD/contest/276073/problem/I>

Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of this tale have been forgotten. So now, in line with the educational nature of this contest, we will tell you the whole story:

The babylonians had n types of blocks, and an unlimited supply of blocks of each type. Each type-i block was a rectangular solid with linear dimensions (xi,yi,zi) . A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height. They wanted to construct the tallest tower possible by stacking blocks. The problem was that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block. This meant, for example, that blocks oriented to have equal-sized bases couldn't be stacked.

Your job is to write a program that determines the height of the tallest tower the babylonians can build with a given set of blocks.

Input

The input file will contain one or more test cases. The first line of each test case contains an integer n, representing the number of different blocks in the following data set. The maximum value for n is 30. Each of the next n lines contains three integers representing the values xi, yi and zi.

Input is terminated by a value of zero (0) for n.

Output

For each test case, print one line containing the case number (they are numbered sequentially starting from 1) and the height of the tallest possible tower in the format "Case case: maximum height = height"

Output ban đầu:

A computer screen shot of a program

Description automatically generated

Sửa lỗi:

Khởi tạo mảng đủ lớn để chứa tất cả các khối

Khởi tạo mảng đủ lớn để chứa chiều cao tối đa

Tìm chiều cao của tòa tháp với đỉnh là viên i

Sửa điều kiện vòng lặp 3 lần

Khởi tạo mảng maxh

Output mới:

A screenshot of a computer program

Description automatically generated

#include <bits/stdc++.h>

using namespace std;

typedef struct {

    int x, y, z;

} block;

int n;

block a[300]; // Khởi tạo mảng đủ lớn để chứa tất cả các khối

int maxh[300]; // Khởi tạo mảng đủ lớn để chứa chiều cao tối đa

void input() {

    cin >> n;

    if (n == 0) exit(0);

    int x, y, z;

    for (int i = 1; i <= n; i++) { // Sửa điều kiện vòng lặp

        cin >> x >> y >> z;

        a[3 \* i - 3].x = x;

        a[3 \* i - 3].y = y;

        a[3 \* i - 3].z = z;

        a[3 \* i - 2].x = y;

        a[3 \* i - 2].y = z;

        a[3 \* i - 2].z = x;

        a[3 \* i - 1].x = z;

        a[3 \* i - 1].y = x;

        a[3 \* i - 1].z = y;

    }

}

int dp(int i) { // Tìm chiều cao của tòa tháp với đỉnh là viên i

    if (maxh[i] != 0) return maxh[i];

    maxh[i] = a[i].z;

    for (int j = 0; j < 3 \* n; j++) { // Sửa điều kiện vòng lặp

        if ((a[i].x < a[j].x && a[i].y < a[j].y) ||

            (a[i].x < a[j].y && a[i].y < a[j].x)) {

            maxh[i] = max(maxh[i], a[i].z + dp(j));

        }

    }

    return maxh[i];

}

int main() {

    int cnt = 1;

    while (true) {

        int res = 0;

        input();

        memset(maxh, 0, sizeof(maxh)); // Khởi tạo mảng maxh

        for (int i = 0; i < 3 \* n; i++) { // Sửa điều kiện vòng lặp

            res = max(res, dp(i));

        }

        printf("Case %d: maximum height = %d\n", cnt++, res);

    }

    return 0;

}