Kiểm tra tính đúng đắn và hiệu năng của chương trình bằng bộ test

Nguyễn Hữu Đặng Nguyên (23521045)
Trần Vạn Tấn (23521407)

CS112.P11.KHTN
University of Information Technology

October 12, 2024

Mục lục

1. Giới thiệu

2. Nội dung chi tiết
   - Các phương pháp kiểm thử
   - Test generation plan
   - Ví dụ

Định nghĩa

Thế nào là **kiểm tra tính đúng đắn và hiệu năng của chương trình?**

Định nghĩa

Kiểm Tra Tính Đúng Đắn

Xác minh rằng chương trình thực hiện đúng các bước được định nghĩa và cho ra kết quả chính xác cho mọi trường hợp đầu vào hợp lệ.

Kiểm Tra Hiệu Năng

Đánh giá khả năng thực thi của thuật toán, chủ yếu dựa vào thời gian và tài nguyên sử dụng khi xử lý các đầu vào khác nhau.

Ý nghĩa

Tại sao cần **kiểm tra tính đúng đắn và hiệu năng của chương trình**?

- Đảm bảo độ chính xác
- Đánh giá hiệu suất
- Phát triển bền vững

Các phương pháp kiểm thử

- **Unit test**
- **Black Box/White Box Test**
- **Ví dụ**

Unix test

Định nghĩa

- Kiểm tra các thành phần nhỏ cụ thể.
- Thường là một hàm, phương thức hoặc một lớp trong chương trình.

Ý nghĩa

Dùng để kiểm tra từng phần nhỏ của code trước khi tích hợp vào bài toán hoàn chỉnh.

Black Box/White Box Test

White Box Test

- Hiểu rõ cấu trúc bên trong và logic của chương trình.
- Đảm bảo tất cả các trường hợp đều được kiểm thử.

Black Box Test

- Không cần biết chi tiết bên trong của chương trình.
- Chỉ quan tâm và kiểm tra Input, Output.

Ví dụ

```
Input: d, m, y // d: ngày, m: tháng, y: năm
Output: Số thứ tự của ngày trong năm và ngày tiếp theo

//Khởi tạo arr với số ngày trong các tháng của năm không nhuận:
arr = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

Function leap(y): // Kiểm tra năm nhuận
    If (y % 400 == 0) or (y % 4 == 0 and y % 100 != 0):
        return True
    return False

Function dayOfYear(d, m, y):
    If leap(y):
        arr[1] = 29

    dayTh = d
    For i from 0 to m - 2:
        dayTh += arr[i]
    return dayTh

Function nextDay(d, m, y):
    If leap(y):
        arr[1] = 29

    If d < arr[m - 1]:
        d += 1
    Else If m < 12:
        d = 1, m += 1
    Else:
        d = 1, m = 1, y += 1
    return d, m , y
```

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Test Generation Plan

- Mục tiêu
- Các bộ test cần thiết
- Kiểm tra đúng đắn
- Chương trình sinh test
- Ví dụ

Mục tiêu

- Cố gắng đa dạng bộ test để tối đa hoá việc xuất ra kết quả sai.
- Đảm bảo khả năng tái tạo lại bộ test.

Các bộ test cần thiết

- Test nhỏ.
- Test lớn, đạt giới hạn của bài toán.
- Test đặc biệt.
- Test ngẫu nhiên.

Kiểm tra tính đúng đắn

- Kiểm tra thủ công.
- Sử dụng các thuật toán "trâu bò".
- In ra các bước đi để theo dõi tiến trình bài toán.

Chương trình sinh test

- **Các hàm hỗ trợ.**

Sinh số ngẫu nhiên

- Dùng *mt*19937 để sinh random.
- Lấy seed là thời gian hiện tại.

```
mt19937_64 seed(chrono::steady_clock::now().time_since_epoch().count());
ll curSeed = seed();
```

So khớp kết quả

- Dùng để so khớp file output của 2 code.

```
// Câu lệnh dùng để chạy code, ví dụ ở đây là code "ADN.cpp" và "ADN_2.cpp"
system("ADN.exe"); system("ADN_2.exe");

// File đầu ra của code "ADN.cpp" là "ADN.out".
// File đầu ra của code "ADN_2.cpp" là "ADN.ans".
if (("system fc ADN.out ADN.ans") != 0) -> Hai file output khac nhau
```

Chương trình sinh test

```cpp
mt19937 random(chrono::steady_clock::now().time_since_epoch().count());
int seed;

// sinh ngẫu nhiên các số trong đoạn từ l đến r
int rnd_seg(int l, int r) {
    assert(l <= r);
    int len = r - l + 1;
    int add = rng() % len;
    return (l + add);
}
int main() {
    // số test mà ta muốn sinh ra để kiểm tra
    int tests; cin >> tests;
    for (int test = 1; test <= tests; test++) {
        // Câu lệnh dùng để mở file, ví dụ ở đây là file "ADN.inp"
        // Reset thành file rỗng và sinh test vào input
        ofstream inp("ADN.inp");

        // Tạo seed ngẫu nhiên đại diện cho testcase hiện tại
        seed = random();
        rng = mt19937(seed);
```

```cpp
        // Input mẫu: Nhập một mảng a[] chứa n số (1 <= n <= 1000, 0 <= ai <= 10^6)
        // sinh n ngẫu nhiên thuộc khoảng [1, 1000]
        int n = rnd_seg(1, 1000);

        // Tùy theo cú pháp sau ofstream mà ta sẽ có cú pháp đưa vào file
        // Ở đây là "ofstream inp("ADN.inp");" nên ta sẽ dùng inp thay cho cin thông thường
        inp << n << '\n';
        for (int i = 1; i <= n; i++)
            inp << rnd_seg(0, 1000000) << ' ';
        inp << '\n';
        inp.close();

        // Giả sử ta có 2 file code cày trâu là ADN.cpp và ADN_2.cpp
        system("ADN.exe");
        system("ADN_2.exe");

        // File đầu ra của code "ADN.cpp" là "ADN.out".
        // File đầu ra của code "ADN_2.cpp" là "ADN.ans".
        if (("system fc ADN.out ADN.ans") != 0) {
            cout << "Test " << test << " WA!";
            cout << seed << '\n';
            return 0;
        } else {
            cout << "Test " << test << " AC!\n";
        }
```

VOI 23 Bài 1 - Chuỗi ADN

Đề bài

-Cho một chuỗi S độ dài N gồm 5 ký tự $A, T, G, X, ?$. Một đoạn con $[i, j]$ được xác định bởi vị trí phần tử bắt đầu i và kết thúc j $(1 \leq i \leq j \leq N)$ là dãy $S_i S_{i+1} \ldots S_j$.
-Một đoạn con **phức tạp** nếu như nó chứa ít nhất hai ký tự khác nhau.
Độ đa dạng của S được định nghĩa là số lượng đoạn con phức tạp.
-Tính độ đa dạng nhỏ nhất của chuỗi nêu trên khi thay mỗi ký tự chấm hỏi (?) bởi trong bốn ký tự A, T, G, X.

Giới hạn

- 40% số test thỏa mãn: $N \leq 20$.
- 24% số test khác thỏa mãn: $N \leq 5000$.
- 36% số test khác thỏa mãn: $N \leq 10^5$.

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Thank you for listening!