Giải Bài Tập Nhóm 3

Nhóm 8

Bài 1: Hệ thống quản lý đơn hàng

Yêu cầu

Bài toán yêu cầu tính tổng chi phí cho một đơn hàng dựa trên các yếu tố: danh sách sản phẩm (bao gồm giá, số lượng, giảm giá), loại khách hàng, và phí vận chuyển.

Mã giả (Pseudocode)

Đầu vào:

- `Order`: chứa danh sách sản phẩm (mỗi sản phẩm có giá, số lượng và phần trăm giảm giá), loại khách hàng (có phải khách hàng thường xuyên không), và phí vận chuyển.

Quy tắc tính toán:

1. Tính tổng giá trị sản phẩm trước khi áp dụng giảm giá.
2. Nếu tổng giá trị sản phẩm $\geq$ 1 triệu, miễn phí vận chuyển.
3. Áp dụng giảm giá cho từng sản phẩm để có tổng giá trị sau giảm giá.
4. Nếu khách hàng là khách hàng thường xuyên, áp dụng thêm chiết khấu 10% trên tổng giá trị sau khi đã áp dụng giảm giá sản phẩm.

Mã giả:

```
Function CalculateTotalCost(Order):
    original_total = 0       // Tổng giá trị gốc trước khi áp dụng giảm giá
    discounted_total = 0     // Tổng giá trị sau khi áp dụng giảm giá

    // Tính tổng giá trước và sau khi giảm giá cho mỗi sản phẩm
    For each product in Order.ProductList:
        original_price = product.Price * product.Quantity
        discounted_price = original_price * (1 - product.Discount / 100)

        original_total += original_price
```

```
    discounted_total += discounted_price

// Miễn phí vận chuyển nếu tổng giá trị gốc >= 1 triệu
If original_total >= 1000000:
    Order.ShippingFee = 0

// Áp dụng chiết khấu cho khách hàng thường xuyên
If Order.IsRegularCustomer:
    discounted_total *= 0.9  // Giảm 10% cho khách hàng thường xuyên

// Tính tổng chi phí cuối cùng
total_cost = discounted_total + Order.ShippingFee
Return total_cost
```

Phân tích kiểm thử

1. Unit Test

Các thành phần nhỏ trong hàm `CalculateTotalCost` có thể được kiểm thử riêng lẻ để đảm bảo chúng hoạt động chính xác:

- **Kiểm tra tính tổng giá trị sản phẩm**:
  - Kiểm tra tính đúng đắn của việc tính giá trị trước và sau giảm giá.
- **Kiểm tra điều kiện miễn phí vận chuyển**:
  - Đảm bảo phí vận chuyển bằng 0 khi tổng giá trị trước giảm giá lớn hơn hoặc bằng 1 triệu.
- **Kiểm tra chiết khấu khách hàng thường xuyên**:
  - Kiểm tra nếu `IsRegularCustomer` là `true` thì chiết khấu được áp dụng đúng 10%.

2. White-box Test

Kiểm thử white-box dựa trên cấu trúc của hàm để đảm bảo mọi nhánh logic đều được chạy:

- **Các trường hợp miễn phí vận chuyển**:
  - **Trường hợp 1**: Tổng giá trị đơn hàng trước khi giảm giá < 1 triệu $\rightarrow$ Phí vận chuyển không miễn phí.
  - **Trường hợp 2**: Tổng giá trị đơn hàng trước khi giảm giá ≥ 1 triệu $\rightarrow$ Phí vận chuyển được miễn phí.
- **Các trường hợp giảm giá cho khách hàng thường xuyên**:

- **Trường hợp 3**: `IsRegularCustomer = True` → Áp dụng chiết khấu 10%.
- **Trường hợp 4**: `IsRegularCustomer = False` → Không áp dụng chiết khấu 10%.

3. Black-box Test

Kiểm thử black-box kiểm tra các trường hợp đầu vào và kết quả đầu ra để đảm bảo hàm xử lý đúng yêu cầu. Dưới đây là các test case cơ bản:

- **Test case 1**: Đơn hàng không có sản phẩm nào.
- **Test case 2**: Đơn hàng có một sản phẩm, không có giảm giá, không phải khách hàng thường xuyên, giá trị nhỏ hơn 1 triệu → Tính tổng cộng phí vận chuyển.
- **Test case 3**: Đơn hàng có một sản phẩm, có giảm giá, là khách hàng thường xuyên, giá trị sau giảm giá nhỏ hơn 1 triệu.
- **Test case 4**: Đơn hàng có nhiều sản phẩm, tổng giá trị gốc lớn hơn 1 triệu → Miễn phí vận chuyển và có áp dụng giảm giá cho khách hàng thường xuyên.

Bài 2: Dãy con có tổng lớn nhất

Yêu cầu

Bài toán yêu cầu tìm dãy con liên tiếp có tổng lớn nhất trong một dãy số nguyên.

Mã giả cho từng giải pháp

Giải pháp 1 (Code trâu):

```
#include<bits/stdc++.h>

using namespace std;

#define int long long

int a[201001];
int32_t main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }
    int ans = -1e18;
```

```
    for (int i = 1; i <= n; ++i) {
        int sum = 0;
        for (int j = i; j <= n; ++j) {
            sum += a[j];
            ans = max(ans, sum);
        }
    }
    cout << ans << '\n';
    return 0;
}
```

Giải pháp 2 (Code tối ưu):

```
#include<bits/stdc++.h>

using namespace std;

#define int long long

int a[201001];
int32_t main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
    }
    int ans = -1e18, sum = 0;
    for (int i = 1; i <= n; ++i) {
        sum += a[i];
        if (sum < 0) {
            sum = 0;
        }
        ans = max(ans, sum);
    }
    cout << ans << '\n';
    return 0;
}
```

Các trường hợp đặc biệt:

- **Test case 1**: Dãy toàn số dương → Tổng lớn nhất là tổng của toàn bộ dãy.
- **Test case 2**: Dãy toàn số âm → Tổng lớn nhất là phần tử lớn nhất.
- **Test case 3**: Dãy xen kẽ số dương và số âm → Kiểm tra liệu thuật toán có chọn được dãy con tối ưu.

- **Test case 4**: Dãy có các số 0 xen kẽ → Kiểm tra xem dãy con tối ưu có bỏ qua các số 0 nếu cần.
- **Test case 5**: Dãy chỉ có một phần tử → Tổng lớn nhất chính là giá trị của phần tử đó.
- **Test case 6**: Dãy rất dài với các giá trị ngẫu nhiên → Kiểm tra hiệu năng và tính đúng đắn khi xử lý các dãy lớn.