ĐẠI HỌC QUỐC GIA TPHCM

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BỘ MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Báo cáo

Đề tài: Bài tập nhóm 3

Môn học: Phân tích và thiết kế thuật toán

Sinh viên thực hiện:

Nguyen Minh Huy(23520634)

Do Quang Luc(23520902)

Giáo viên hướng dẫn:

Nguyen Thanh Son

Ngày 31 tháng 10 năm 2024

Mục lục

Lời cảm ơn

Cảm ơn nhóm 3 đã cung cấp một số kiến thức khá là bổ ích

1 Giới thiệu

1.1 Tổng Quan

Đây là bài báo cáo về bài tập của nhóm 3

1.2 Tiến trình công việc

- Ngày 1: Nghiên cứu và đọc qua các tài liệu tham khảo
- Ngày 2 2: Đặt ra vấn đề và tìm hướng giải quyết
- Tuần 3: Đánh giá, code, và viết bài báo cáo

2 Tính Tổng Chi Phí Đơn Hàng

2.1 Mô tả bài toán

Giả sử bạn đang phát triển một hệ thống quản lý đơn hàng cho một cửa hàng trực tuyến. Hệ thống cần tính toán tổng chi phí của một đơn hàng dựa trên các yếu tố sau:

- **Thông tin đơn hàng**:
  - Danh sách sản phẩm: Mỗi sản phẩm có giá, số lượng và phần trăm giảm giá.
  - Phí vận chuyển: Thêm phí nếu tổng giá trị đơn hàng nhỏ hơn 1 triệu.
  - Loại khách hàng: Khách hàng thường xuyên sẽ được chiết khấu 10% trên tổng giá trị đơn hàng.
- **Yêu cầu**: Viết hàm `TinhChiPhi(Order order)` để tính tổng chi phí của đơn hàng sau khi đã áp dụng các quy tắc giảm giá, chiết khấu và vận chuyển.
- **Order** gồm:

- Danh sách các sản phẩm: Mỗi sản phẩm có giá gốc, số lượng và phần trăm giảm giá.
- IsRegularCustomer: Biến boolean xác định khách hàng thường xuyên hay không.
- ShippingFee: Phí vận chuyển.

2.2 Quy tắc

- Có 1 function tính tổng giá thành khi áp dụng giảm giá và 1 function không áp dụng giảm giá.
- Nếu giá trị đơn hàng trước khi áp dụng giảm giá lớn hơn hoặc bằng 1 triệu, miễn phí vận chuyển.
- Nếu khách hàng là khách hàng thường xuyên, chiết khấu 10% trên tổng giá trị đơn hàng sau khi áp dụng các giảm giá của sản phẩm.

2.3 Mã giả

Algorithm 1 Tính Tổng Chi Phí Đơn Hàng

```
function CalculateTotalWithoutDiscount(Order)
    total ← 0
    for product in Order.productList do
        total ← total + (product.price × product.quantity)
    end for
    return total
end function
function CalculateTotalWithDiscount(Order)
    total ← 0
    for product in Order.productList do
        discountedPrice ← product.price × (1 - product.discountPercentage / 100)
        total ← total + (discountedPrice × product.quantity)
    end for
    return total
end function
function CalculateOrderCost(Order)
    totalBeforeDiscount ← CalculateTotalWithoutDiscount(Order)
    totalAfterDiscount ← CalculateTotalWithDiscount(Order)
    if totalBeforeDiscount ≥ 1000000 then
        finalCost ← totalAfterDiscount
    else
        finalCost ← totalAfterDiscount + Order.shippingFee
    end if
    if Order.isRegularCustomer then
        finalCost ← finalCost ×0.9
    end if
    return finalCost
end function
```

2.4 Kiểm thử

2.4.1 Unit Test

Kiểm thử từng hàm riêng biệt để đảm bảo tính chính xác của các phần:

- **CalculateTotalWithoutDiscount**: Kiểm tra tổng giá trị sản phẩm trước giảm giá.
- **CalculateTotalWithDiscount**: Kiểm tra tổng giá trị sản phẩm sau khi áp dụng giảm giá của từng sản phẩm.

- **CalculateOrderCost**: Kiểm tra tổng chi phí cuối cùng với các trường hợp miễn phí vận chuyển và chiết khấu cho khách hàng thường xuyên.

2.4.2 White Box Test

Kiểm thử mã nguồn bằng cách kiểm tra tất cả các nhánh điều kiện:

- Trường hợp tổng giá trị đơn hàng trước giảm giá lớn hơn hoặc bằng 1 triệu để kiểm tra miễn phí vận chuyển.
- Trường hợp khách hàng là khách hàng thường xuyên để kiểm tra chiết khấu 10%.

2.4.3 Black Box Test

Kiểm thử dựa trên đầu vào và đầu ra mà không xem mã nguồn:

- Đầu vào: Các bộ dữ liệu đơn hàng với giá trị, số lượng, và giảm giá khác nhau.
- Đầu ra: Tổng chi phí đơn hàng tính đúng với các quy tắc về giảm giá và chiết khấu.

2.5 Các Trường Hợp Kiểm Thử Đặc Biệt

Các trường hợp đặc biệt cần được kiểm tra:

- Tổng giá trị đơn hàng trước khi giảm giá đúng bằng 1 triệu để kiểm tra điều kiện miễn phí vận chuyển.
- Không có sản phẩm nào trong danh sách (danh sách rỗng).
- Tất cả sản phẩm đều có giảm giá 100%.
- Khách hàng không phải là khách hàng thường xuyên và có tổng giá trị đơn hàng rất nhỏ (kiểm tra thêm phí vận chuyển).

3 Bài 2

3.1 Mô tả Bài Toán

Cho một dãy số nguyên $a_1, a_2, \ldots, a_n$, yêu cầu tìm dãy con liên tiếp có tổng lớn nhất. Hai chỉ số (l, r) cần được xác định sao cho $1 \leq l \leq r \leq n$ và tổng $a_l + a_{l+1} + \cdots + a_r$ là lớn nhất.

3.2 Giải pháp Code Trâu với Độ Phức Tạp $O(n^2)$

Giải pháp này duyệt qua tất cả các dãy con liên tiếp trong mảng, tính tổng của từng dãy con và cập nhật tổng lớn nhất nếu tìm thấy một tổng lớn hơn.

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int ans = -9e18, n;
    cin >> n;
    vector<int> a(n + 5);
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) {
        int sum = 0;
        for (int j = i; j <= n; j++) {
            sum += a[j];
            if (ans < sum) ans = sum;
        }
    }
    cout << ans;
    return 0;
}
```

Listing 1: Code Trâu với độ phức tạp $O(n^2)$

3.2.1 Giải thích

- Biến `ans` lưu trữ tổng lớn nhất của dãy con tìm được.

- Vòng lặp lồng nhau tính tổng của từng dãy con bắt đầu từ phần tử thứ `i` đến phần tử thứ `j`.
- Kết quả cuối cùng là giá trị lớn nhất của tổng dãy con được lưu trong `ans`.

3.3 Giải pháp Tối ưu với Độ Phức Tạp $O(n)$

Sử dụng thuật toán Kadane để tìm dãy con có tổng lớn nhất với độ phức tạp $O(n)$.

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    int ans = -9e18, n, pref = 0;
    cin >> n;
    vector<int> a(n + 5);
    for (int i = 1; i <= n; i++) cin >> a[i];
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += a[i];
        ans = max(ans, sum - pref);
        pref = min(pref, sum);
    }
    cout << ans;
    return 0;
}
```

Listing 2: Thuật toán Kadane với độ phức tạp $O(n)$

3.3.1 Giải thích

- Biến `sum` tính tổng của các phần tử từ đầu đến phần tử hiện tại.
- Biến `pref` lưu tổng con nhỏ nhất để tìm điểm khởi đầu có lợi nhất.
- Biến `ans` lưu giá trị lớn nhất của tổng dãy con bằng cách lấy `sum` − `pref`.

3.4 Chương trình Sinh Bộ Test và So Sánh Kết Quả

```
Function GenerateTestCases(numCases, maxN, maxAbsValue):
    testCases = []
    For i = 1 to numCases:
        n = Random(1, maxN)
        array = []
        For j = 1 to n:
            value = Random(-maxAbsValue, maxAbsValue)
            array.append(value)
        testCases.append(array)
    Return testCases
```

Listing 3: Mã giả Sinh Bộ Test

3.5 Các Trường Hợp Kiểm Thử Đặc Biệt

Để đảm bảo tính chính xác và hiệu quả của các thuật toán, các bộ test đặc biệt cần được xem xét:

- Dãy chỉ chứa số âm, để kiểm tra nếu thuật toán trả về phần tử âm lớn nhất.
- Dãy có cả số dương và số âm, kiểm tra tổng lớn nhất nằm ở giữa.
- Dãy có độ dài lớn nhất gần sát $n = 10^5$.

3.6 Kết Luận

Thuật toán Kadane với độ phức tạp $O(n)$ là giải pháp tối ưu cho bài toán tìm dãy con có tổng lớn nhất. Giải pháp $O(n^2)$ mặc dù đúng nhưng chỉ thích hợp cho các mảng ngắn.