TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÀI TẬP KIỂM TRA TÍNH ĐÚNG ĐẮN VÀ HIỆU NĂNG CỦA CHƯƠNG TRÌNH BẰNG BỘ TEST

MÔN: PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

Nhóm 9

Sinh viên thực hiện:

Họ và tên	MSSV
Bùi Ngọc Thiên Thanh	23521436
Nguyễn Thái Sơn	23521356

Thành phố Hồ Chí Minh, 2024

Mục lục

1 Hệ thống tính toán chi phí đơn hàng

1.1 Mã giả (Pseudocode):

```
Function TinhChiPhi(Order order):
    // Tính tổng giá thành không áp dụng giảm giá
    totalBeforeDiscount = TinhTongKhongGiamGia(order)

    // Nếu tổng giá >= 1 triệu, miễn phí vận chuyển
    if totalBeforeDiscount >= 1000000:
        shippingFee = 0
    else:
        shippingFee = order.ShippingFee

    // Tính tổng giá trị sau khi áp dụng giảm giá của từng sản phẩm
    totalAfterDiscount = TinhTongCoGiamGia(order)

    // Nếu khách hàng là khách hàng thường xuyên, áp dụng chiết khấu 10
    if order.IsRegularCustomer:
        finalCost = totalAfterDiscount * 0.9 + shippingFee
    else:
        finalCost = totalAfterDiscount + shippingFee

    return finalCost


Function TinhTongKhongGiamGia(Order order):
    total = 0
    For each product in order.ProductList:
        total += product.price * product.quantity
    return total


Function TinhTongCoGiamGia(Order order):
```

```
    total = 0
    For each product in order.ProductList:
        discountedPrice = product.price * (1 - product.discount)
        total += discountedPrice * product.quantity
    return total
```

1.2 Phân tích Kiểm thử

1.2.1 Unit Test

- **Phạm vi**: Kiểm tra từng hàm riêng lẻ.
- **Các hàm cần kiểm tra**:
  - `TinhTongKhongGiamGia()`: Đảm bảo hàm tính tổng giá thành trước khi áp dụng giảm giá đúng với từng sản phẩm.
  - `TinhTongCoGiamGia()`: Đảm bảo hàm tính tổng giá sau khi áp dụng giảm giá từng sản phẩm chính xác.
  - `TinhChiPhi()`: Kiểm tra tính chính xác của việc tính tổng chi phí của đơn hàng, bao gồm giảm giá sản phẩm, chiết khấu cho khách hàng thường xuyên, và phí vận chuyển.

1.2.2 White Box Test

- **Mục tiêu**: Đảm bảo bao phủ tất cả các nhánh và logic của mã.
- **Đặc điểm của các test case**:
  - **Nhánh với tổng giá trị ≥ 1 triệu và khách hàng thường xuyên**: Kiểm tra miễn phí vận chuyển và chiết khấu 10%.
  - **Nhánh với tổng giá trị < 1 triệu và khách hàng thường xuyên**: Kiểm tra tính phí vận chuyển và chiết khấu 10%.
  - **Nhánh với tổng giá trị ≥ 1 triệu và không phải khách hàng thường xuyên**: Kiểm tra miễn phí vận chuyển nhưng không có chiết khấu.

- **Nhánh với tổng giá trị < 1 triệu và không phải khách hàng thường xuyên**: Kiểm tra tính phí vận chuyển mà không có chiết khấu.

1.2.3 Black Box Test

- **Mục tiêu**: Kiểm tra tính chính xác của kết quả đầu ra dựa trên các tình huống đầu vào khác nhau mà không phụ thuộc vào cấu trúc mã.
- **Đặc điểm của các test case**:
  - **Các đầu vào biên**: Đảm bảo thử với các trường hợp đặc biệt như tổng giá trị trước giảm giá xấp xỉ 1 triệu (ví dụ: 999,999 và 1,000,001) để xác minh tính đúng đắn của điều kiện miễn phí vận chuyển.
  - **Khách hàng thường xuyên và không thường xuyên**: Đảm bảo kết quả đầu ra chính xác khi thay đổi loại khách hàng.
  - **Giảm giá sản phẩm khác nhau**: Kiểm tra với các sản phẩm có nhiều mức giảm giá khác nhau (0%, 50%, 100%) và xác nhận việc tính tổng đúng khi áp dụng giảm giá.

Bài toán: Tạo ra các mô hình học máy có khả năng xử lý sự bất ổn thường gặp trong dự đoán rủi ro tín dụng.
Mục tiêu: Cải thiện độ tin cậy của việc đánh giá rủi ro tín dụng bằng cách đảm bảo rằng các mô hình dự đoán vẫn duy trì hiệu suất tốt ngay cả khi dữ liệu nền tảng thay đổi.
Input:
Dữ liệu khách hàng (Customer Data): Đây là tập dữ liệu chứa thông tin về các khách hàng đã nộp đơn xin vay tiền. Các thông tin này bao gồm nhiều đặc điểm khác nhau như:

- Thông tin cá nhân: Tuổi, giới tính, tình trạng hôn nhân, số người phụ thuộc, v.v.
- Thông tin kinh tế: Thu nhập, nghề nghiệp, loại hình công việc, tài sản, v.v.

- Lịch sử tín dụng: Thông tin về các khoản vay trước đó, hạn mức tín dụng, lịch sử thanh toán, v.v.
- Thông tin về khoản vay hiện tại: Số tiền vay, mục đích vay, thời gian vay, v.v.

Dữ liệu bổ sung: Các tập dữ liệu phụ khác có thể bao gồm thông tin về lịch sử thanh toán chi tiết, thông tin tín dụng từ các nguồn khác, v.v.
Output:

- **Xác suất vỡ nợ (Probability of Default):** Đối với mỗi khách hàng trong tập dữ liệu kiểm tra (test dataset), mô hình sẽ dự đoán xác suất mà khách hàng đó sẽ không thể trả nợ khoản vay của họ. Giá trị này thường nằm trong khoảng từ 0 đến 1.
- **File .CSV kết quả dự đoán:**là một tệp CSV chứa ID của khách hàng và xác suất vỡ nợ tương ứng.

2 Dãy con có tổng lớn nhất

Cho dãy số nguyên $a_1, a_2, \ldots, a_n$. Một dãy con liên tiếp của dãy a là $a_l + a_{l+1} + \ldots + a_r$. Nhiệm vụ của bài toán là tìm dãy con liên tiếp có tổng lớn nhất, tức là tìm một cặp (l, r) mà $1 \leq l \leq r \leq n$ sao cho tổng $a_l + a_{l+1} + \ldots + a_r$ là lớn nhất.

2.1 Yêu cầu 1: Code trâu với độ phức tạp $O(n^2)$ hoặc $O(n^3)$

Để giải quyết bằng code trâu, ta có thể duyệt tất cả các dãy con liên tiếp của dãy và tính tổng của mỗi dãy để tìm ra tổng lớn nhất.

```
def max_subarray_brute_force(arr):
n = len(arr)
max_sum = float('-inf')

for i in range(n):
```

```
        for j in range(i, n):
            current_sum = sum(arr[i:j+1])
            max_sum = max(max_sum, current_sum)

    return max_sum
```

2.2 Yêu cầu 2: Code tối ưu với độ phức tạp $O(n)$

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

```
def max_subarray_optimized(arr):
max_sum = float('-inf')
current_sum = 0

for num in arr:
    current_sum = max(num, current_sum + num)
    max_sum = max(max_sum, current_sum)

return max_sum
```

2.3 Yêu cầu 3: Trình sinh và so sánh kết quả giữa code trâu và code tối ưu

```
import random

def generate_test_case(n, value_range=(-10000, 10000)):
return [random.randint(value_range[0], value_range[1]) for _
    in range(n)]

def compare_results(n):
test_case = generate_test_case(n)

brute_result = max_subarray_brute_force(test_case)
optimized_result = max_subarray_optimized(test_case)

```

```
    assert brute_result == optimized_result, f"Mismatch! Brute:
        {brute_result}, Optimized: {optimized_result}"
    print("Test passed!")

    for i in range(5):
    compare_results(100)
```

2.4 Các trường hợp đặc biệt của test case

1. **Dãy chỉ có một phần tử**: Đây là trường hợp đơn giản nhất, chỉ cần trả về chính giá trị phần tử đó.
2. **Dãy có tất cả phần tử âm**: Ở đây, thuật toán Kadane sẽ chọn phần tử lớn nhất, vì không có dãy con nào khác có tổng lớn hơn.
3. **Dãy có tất cả phần tử dương**: Tổng của toàn bộ dãy sẽ là đáp án.
4. **Dãy có tổng hợp âm và dương xen kẽ**: Đây là trường hợp cần kiểm tra xem liệu thuật toán có nhận ra các dãy con tối ưu hay không.