

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÀI TẬP MÔN HỌC
PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

Sinh viên: Đỗ Phương Duy - 23520362

Sinh viên: Nguyễn Nguyên Khang - 22520623

Ngày 26 tháng 10 năm 2024



Mục lục

1 Bài 1: Quản lý đơn hàng	3
1.1 Pseudocode	3
1.1.1 Lớp Product	3
1.1.2 Lớp Order	3
1.2 Áp dụng các bộ test	4
2 Bài 2: Dãy con có tổng lớn nhất	5
2.1 Code trâu độ phức tạp $O(n^2)$:	5
2.2 Code full độ phức tạp $O(n)$:	5
2.3 Trình sinh test và so test giữa "codetrau" và "codefull":	6
2.4 Các trường hợp đặc biệt:	7



1 Bài 1: Quản lý đơn hàng

1.1 Pseudocode

1.1.1 Lớp Product

```
1 Class Product:
2     Function __init__(name, price, amount, discount):
3         Set self.name = name
4         Set self.price = price
5         Set self.amount = amount
6         Set self.discount = discount
```

1.1.2 Lớp Order

```
1 Class Order:
2     Function __init__(products, IsRegularCustomer, ShippingFee):
3         Set self.products = products
4         Set self.IsRegularCustomer = IsRegularCustomer
5         Set self.ShippingFee = ShippingFee
6
7     Function Tinh_Phi_Van_Chuyen(total):
8         If total < 1000000:
9             Return total + self.ShippingFee
10        Else:
11            Return total
12
13    Function Chiet_Khau(total, status):
14        If status is True:
15            Return total * 0.9
16        Else:
17            Return total
18
19    Function Tinh_Chi_Phi_Co_Giam_Gia():
20        Set total_price = 0
21        For each product in self.products:
22            total_price += product.price * product.amount * (1 - product.discount)
23
24        total_price = self.Tinh_Phi_Van_Chuyen(total_price)
25        total_price = self.Chiet_Khau(total_price, self.IsRegularCustomer)
26        Return total_price
27
28    Function Tinh_Chi_Phi_Khong_Giam_Gia():
29        Set total_price = 0
30        For each product in self.products:
31            total_price += product.price * product.amount
32
33        total_price = self.Tinh_Phi_Van_Chuyen(total_price)
34        total_price = self.Chiet_Khau(total_price, self.IsRegularCustomer)
35        Return total_price
```



1.2 Áp dụng các bộ test

• Unit Test:

- Áp dụng vào hàm `Tinh_Phi_Van_Chuyen`: nhằm kiểm tra tính đúng đắn khi cộng phí vận chuyển
- Các trường hợp:
 - * Tổng giá trị dưới 1 triệu: kiểm tra nếu hệ thống cộng thêm phí vận chuyển.
 - * Tổng giá trị từ 1 triệu trở lên: kiểm tra nếu hệ thống miễn phí vận chuyển.
- Áp dụng vào hàm `Chiet_Khau`: kiểm tra tính đúng đắn khi nhận đầu vào là giá trị 1 đơn hàng và tình trạng khách hàng (thường xuyên hoặc không thường xuyên)
- Các trường hợp:
 - * Khách hàng thường xuyên → kiểm tra nếu hệ thống áp dụng chiết khấu 10
 - * Khách hàng không thường xuyên → kiểm tra nếu hệ thống không áp dụng chiết khấu.

• White Box Test:

- `Tinh_Chi_Phi_Co_Giam_Gia`:
 - * Trường hợp có sản phẩm áp dụng giảm giá (discount khác 0). (đảm bảo có giảm giá)
 - * Trường hợp tất cả sản phẩm không có giảm giá (discount = 0). (đảm bảo không có giảm giá)
 - * Trường hợp khách hàng là khách hàng thường xuyên và không thường xuyên để kiểm tra đường đi của chiết khấu.
 - * Các đường đi khác nhau của phí vận chuyển khi tổng giá trị đơn hàng trước giảm giá dưới 1 triệu và trên 1 triệu.
- `Tinh_Chi_Phi_Khong_Giam_Gia`:
 - * Trường hợp có sản phẩm áp dụng giảm giá (discount khác 0). (đảm bảo không có giảm giá dù thông tin sản phẩm có thông tin giảm giá)
 - * Trường hợp tất cả sản phẩm không có giảm giá (discount = 0). (đảm bảo không có giảm giá)
 - * Trường hợp khách hàng là khách hàng thường xuyên và không thường xuyên để kiểm tra đường đi của chiết khấu.
 - * Các đường đi khác nhau của phí vận chuyển khi tổng giá trị đơn hàng trước giảm giá dưới 1 triệu và trên 1 triệu.

• Black Box Test:

- `Tinh_Chi_Phi_Co_Giam_Gia` và `Tinh_Chi_Phi_Khong_Giam_Gia`:
 - * Trường hợp giỏ hàng có sản phẩm với các mức giảm giá khác nhau (0%, 10%, 20%,...).
 - * Trường hợp tổng giá trị đơn hàng khác nhau để kiểm tra phí vận chuyển: tổng giá trị < 1 triệu hoặc ≥ 1 triệu.
 - * Trường hợp các loại khách hàng: thường xuyên và không thường xuyên



2 Bài 2: Dãy con có tổng lớn nhất

2.1 Code trâu độ phức tạp $O(n^2)$:

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int N = 1e6 + 7;

int n, a[N], f[N];
signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    freopen("btn3.inp", "r", stdin);
    freopen("btn3_trau.out", "w", stdout);
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    f[0] = 0;
    for (int i = 1; i <= n; i++)
        f[i] = f[i - 1] + a[i];
    int res = -0x34;
    for (int i = 1; i <= n; i++)
        for (int j = i; j <= n; j++)
        {
            res = max(res, f[j] - f[i - 1]);
        }
    cout << res << "\n";
    return 0;
}
```

2.2 Code full độ phức tạp $O(n)$:

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int N = 1e6 + 7;

int n, a[N], f[N];
signed main()
{
    ios_base::sync_with_stdio(false);
```



```
cin.tie(0);cout.tie(0);
freopen("btn3.inp", "r", stdin);
freopen("btn3_full.out", "w", stdout);
cin >> n;
for (int i = 1; i <= n; i++)
    cin >> a[i];
f[0] = 0;
for (int i = 1; i <= n; i++)
    f[i] = f[i - 1] + a[i];
//Thuat toan Kandane's
int m = 0, res = -0x34;
for (int i = 1; i <= n; i++)
{
    m = min(m, f[i - 1]);
    res = max(res, f[i] - m);
}
cout << res << "\n";
return 0;
}
```

2.3 Trình sinh test và so test giữa "codetrau" và "codefull":

```
#include <bits/stdc++.h>
using namespace std;
// Tên chương trình
const string NAME = "btn3";
// Số test kiểm tra
const int NTEST = 100;

mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());
#define rand rd

// Viết lại hàm random để sử dụng cho thuận tiện. Hàm random này sinh ngẫu nhiên số trong ph
long long Rand(long long l, long long h) {
    assert(l <= h);
    return l + rd() * 1LL * rd() % (h - l + 1);
}

int n, a[1000006];
int main()
{
    srand(time(NULL));
    for (int iTest = 1; iTest <= NTEST; iTest++)
    {
        ofstream inp((NAME + ".inp").c_str());
        // Code phần sinh test ở đây
        n = rand() % 1000 + 1;
        for (int i = 1; i <= n; i++)
```



```
{
    a[i] = rand() % 1000000;
    int x = rand() % 2;
    if (x % 2 == 1) a[i] *= -1;
}
cout << n << "\n";
for (int i = 1; i <= n; i++)
    cout << a[i] << " ";

inp.close();
// Nếu dùng Linux thì "/" + Tên chương trình
system((NAME + "_full.exe").c_str());
system((NAME + "_trau.exe").c_str());
// Nếu dùng linux thì thay fc bằng diff
if (system(("fc " + NAME + ".out " + NAME + ".ans").c_str()) != 0)
{
    cout << "Test " << iTest << ": WRONG!\n";
    return 0;
}
cout << "Test " << iTest << ": CORRECT!\n";
}
return 0;
}
```

2.4 Các trường hợp đặc biệt:

- Dãy toàn số âm: kết quả là phần tử lớn nhất
- Dãy một phần tử: kết quả là phần tử duy
- Dãy có giá trị rất lớn (tiệm cận với giới hạn int)
- Dãy luân phiên âm dương: kiểm tra tính ổn định của thuật toán

Tài liệu