VQ44_FLOWERS

Đề bài

VQ44. ĐÈN HOA

Tên chương trình: FLOWERS.???

Để chuẩn bị một ngày lễ lớn lần thứ k thành phố quyết định dùng đèn LED kết thành k bông hoa trang trí dọc đường phố chính, mỗi bông hoa được kết từ một loại bóng LED cùng màu. Để có hiệu ứng ánh sáng tốt nhất Công ty Chiếu sáng được yêu cầu tạo ra được càng nhiều hoa khác màu càng tốt. Trong kho của Công ty có n bộ đèn LED mỗi bộ lắp được một đèn và bộ thứ i có màu a_i.

Hãy chỉ ra k màu của các bộ đèn cần chọn sao cho số lượng hoa khác màu lắp được là nhiều nhất. Nếu có nhiều cách khác nhau thì đưa ra cách tùy chọn.

Dữ liệu: Vào từ file văn bản FLOWERS.INP:

- Dòng đầu tiên chứa 2 số nguyên n và k ($1 \le k \le n \le 10^5$),
- Dòng thứ 2 chứa n số nguyên $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản FLOWERS.OUT trên một dòng k số nguyên – màu của các đèn được chọn.

Ví dụ:

FLOWERS.INP
10 4 8 8 8 8 8 8 8 8 2 1

FLOWERS.OUT
1 2 8 8

Lời giải

- Tóm tắt: Cho n đèn có màu khác nhau, yêu cầu chọn ra k đèn sao cho số lượng đèn có màu khác nhau là lớn nhất.

- Thuật toán: Đầu tiên, với mỗi loại đèn có màu khác nhau, chúng ta sẽ chọn 1 cái. Nếu chưa đủ k đèn, chúng ta sẽ chọn tiếp từ các đèn còn lại.

- Độ phức tạp thuật toán: O(k)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
```

```cpp
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
        freopen("TASK.INP", "r", stdin);
        freopen("TASK.OUT", "w", stdout);
    }

    int n, k;
    cin >> n >> k;
    map<int, int> m;

    for (int i = 1; i <= n; i++) {
        int x;
        cin >> x;
        m[x]++;
    }

    for (auto it : m) {
        cout << it.first << " ";
        k--;
        if (k == 0) break;
        m[it.first]--;
    }

    if (k > 0) {
        for (auto it : m) {
            for (int i = 0; i < it.second; i++) {
                cout << it.first << " ";
                k--;
                if (k == 0) break;
            }
            if (k == 0) break;
        }
    }

    return 0;
}
```

Point2D

Đề bài

Point2D

Tập Điểm

Cho một tập N điểm (x,y) trên mặt phẳng Oxy. Bạn hãy sắp xếp và in ra các điểm tăng dần theo x, nếu x bằng nhau thì sắp xếp các điểm giảm dần theo y.

Input:

- Dòng đầu tiên chứa số nguyên N, $(1 \leq N \leq 10^6)$ là số lượng điểm trên mặt phẳng.
- N dòng tiếp theo gồm 2 số nguyên x, y $(-10^9 \leq x, y \leq 10^9)$ là toạ độ của các điểm trên mặt phẳng.

Output: Các điểm đã được sắp xếp , mỗi điểm được in trên một hàng.

Ví dụ:

Input	Output
5 1 2 1 3 2 4 2 9 4 1	1 3 1 2 2 9 2 4 4 1

Lời giải

- Tóm tắt: Cho n điểm (x, y) sắp xếp các điểm tăng dần theo x, nếu bằng sắp xếp giảm dần theo y.
- Thuật toán: Sử dụng hàm sort của c++ kết hợp hàm so sánh của pair<int, int>
- Độ phức tạp thuật toán: O(n log n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

bool cmp(pair<int, int> x, pair<int, int> y){
    if (x.first == y.first) return x.second > y.second;
    return x.first < y.first;
```

```
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    vector<pair<int, int>> v;

    int n;
    cin >> n;
    for (int i=1; i<=n; i++){
        int x, y;
        cin >> x >> y;
        v.push_back({x, y});
    }
    sort(v.begin(), v.end(), cmp);
    for (auto it : v){
        cout << it.first << " " << it.second << "\n";
    }
}
```

VS14_Gifts

Đề bài

QUÀ TẶNG *Tên chương trình: GIFTS.CPP*

Nhân dịp năm mới Steve quyết định mua tặng 2 người bạn thân của mình mỗi người một món quà. Trong cửa hàng lưu niệm có n mặt hàng khác nhau, mặt hàng thứ i có giá a_i, $i = 1 \div n$. Với tổng số tiền trong túi là x, Steve quyết định sẽ mua 2 món quà khác nhau có tổng giá trị lớn nhất và tất nhiên – không vượt quá khả năng chi trả của mình.

Ví dụ, có 6 mặt hàng với giá nêu ở trên và số tiền có thể chi tối đa là 18, Steve sẽ chọn các món quà thứ nhất và thứ ba. Tổng số tiền cần chi sẽ là 5 + 10 = 15.

Hãy xác định tổng số tiền Steve cần chi trả.

Dữ liệu: Vào từ thiết bị nhập chuẩn:

- Dòng đầu tiên chứa một số nguyên n và x ($2 \le n \le 10^5$, $2 \le x \le 10^9$),
- Dòng thứ 2 chứa n số nguyên $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra thiết bị xuất chuẩn một số nguyên – số tiền cần chi trả.

Ví dụ:

INPUT	OUTPUT
6 18 5 3 10 2 4 9	15

Lời giải

- Tóm tắt: Cho n món quà và x tiền, chọn ra 2 món quà sao cho không vượt quá x tiền và có tổng giá cao nhất

- Thuật toán: Sắp xếp lại mảng và sử dụng thuật toán hai con trỏ để duyệt.

- Độ phức tạp thuật toán: O(n log n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

int solve(vector<int> a, int x) {
    int n = a.size();
    int maxSum = -1;
    pair<int, int> ans = {-1, -1};

    vector<int> b = a;
    sort(b.begin(), b.end());

    int i = 0, j = n - 1;
    while (i < j) {
        int sum = b[i] + b[j];

        if (sum <= x) {
            if (sum > maxSum) {
                maxSum = sum;
                ans = {b[i], b[j]};
            }
            i++;
        } else {
            j--;
        }
    }

    return ans.first + ans.second;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}
    int n, x;
```

```
    cin >> n >> x;

    vector<int> v;
    for (int i=0; i<n; i++){
        int x;
        cin >> x;
        v.push_back(x);
    }

    cout << solve(v, x);

}
```

PasswordStrength

Đề bài

Hiện nay, đã có nhiều trường hợp các tài khoản cá nhân của người dùng bị tấn công bởi các hacker. Đôi khi chỉ cần bạn tham gia một trò chơi bất kỳ trên mạng xã hội và đăng nhập vào trò chơi đó bằng thông tin từ tài khoản cá nhân là đã cấp quyền tiết lộ thông tin.

Do đó, nếu người dùng có mật khẩu lỏng lẻo thì việc hacker xâm nhập và lấy cắp thông tin là càng có thể xảy ra. Hiểu được vấn đề an ninh mạng, nhiều nhà cung cấp dịch vụ đã và đang yêu cầu khách hàng của mình thay đổi mật khẩu thường xuyên với các chuẩn thiết lập mật khẩu như phải đủ độ dài ký tự, phải chứa ký tự in hoa, ký tự đặc biệt, con số…

Vì vậy, việc xây dựng một mật khẩu mạnh để bảo vệ an toàn cho các tài khoản cá nhân hiện nay là vô cùng quan trọng.

Trên thực tế có rất nhiều công thức để xác định tính mạnh yếu của mật khẩu, nhưng một trong những công thức cơ bản là:

score = BaseScore + (Num_Excess*Bonus_Excess) + Num_Upper*Bonus_Upper) + (Num_Numbers*Bonus_Numbers) + (Num_Symbols*Bonus_Symbols) + Bonus_Combo + Bonus_FlatLower + Bonus_FlatNumber

Trong đó:

Biến	Giá trị	Điều kiện
BaseScore	40	
Bonus_Excess	3	
Bonus_Upper	4	
Bonus_Number	5	
Bonus_Symbols	5	

Bonus_Combo	25	Nếu có chữ cái in hoa, kí tự đặc biệt và chữ số trong mật khẩu.
	15	Nếu mật khẩu có chứa các kí tự thuộc 2 trong 3 tập (chữ cái in hoa, kí tự đặc biệt, chữ số)
	0	Ngược lại
Bonus_FlatLower	-15	Nếu mật khẩu chứa toàn kí tự là chữ cái thường
	0	Ngược lại
Bonus_FlatNumber	-35	Nếu mật khẩu đều là kí tự chữ số
	0	Ngược lại
Number_Execess	Hiệu giữa độ dài mật khẩu và 8	
Number_Upper	Số lượng chữ cái in hoa trong mật khẩu	
Number_Numbers	Số lượng chữ số trong mật khẩu	
Number_Symbols	Số lượng kí tự đặc biệt trong mật khẩu	

Lưu ý:

+ Tập kí tự đặc biệt là gồm các kí tự [! , @ , # , $, % , ^ , & , * , ? , _ , ~]

+ Nếu mật khẩu chứa các có độ dài < 8 hoặc có một trong các dấu [. , \ , / , ‘ ‘, ’,’] thì là một mật khẩu không hợp lệ .

+ Nếu **score < 50** là mật khẩu yếu, **50 ≤ score < 75** là mật khẩu vừa, **75 ≤ score < 100** là mật khẩu mạnh, **score ≥ 100** là mật khẩu rất mạnh.

Bạn hãy viết một chương trình để xác định xem độ mạnh của mật khẩu?

Dữ liệu: Vào từ thiết bị nhập chuẩn gồm một dòng chứa xâu **s** là mật khẩu cần kiểm tra ($1 \leq |s| \leq 100$).

Kết quả: Đưa ra thiết bị xuất chuẩn là các đánh giá về mật khẩu (‘KhongHopLe', ‘Yeu' , 'Vua' , 'Manh', 'RatManh') .

Lời giải

- Tóm tắt: Cho một password đánh giá độ mạnh của password đó

- Thuật toán: Dựa theo gợi ý của đề bài và xây dựng các hàm kiểm tra và tính điểm

- Độ phức tạp thuật toán: O(length(s))

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    vector<char> ban = {'.', '\\', '/', ' ', ','};
    vector<char> spec = {'!' , '@' , '#' , '$' , '%' , '^' , '&' , '*' , '?' ,
'_' , '~'};

    string s;
    getline(cin, s);
    if (s.length() < 8){
        cout << "KhongHopLe";
        return 0;
    }
    for (int i=0; i<s.length(); i++){
        if (find(ban.begin(), ban.end(), s[i]) != ban.end()){
            cout << "KhongHopLe";
            return 0;
        }
    }

    int BaseScore = 40;
    int Bonus_Excess = 3;
    int Bonus_Upper = 4;
    int Bonus_Number = 5;
    int Bonus_Symbols = 5;
    int Bonus_Combo = 0;
    int Bonus_FlatLower = 0;
    int Bonus_FlatNumber = 0;
    int Number_Execess = 0;
    int Number_Upper = 0;
    int Number_Numbers = 0;
    int Number_Symbols = 0;
    int Number_Lower = 0;
    Number_Execess = s.length() - 8;

    for (int i=0; i<s.length(); i++){
        if (s[i] >= 'A' && s[i] <= 'Z'){
            Number_Upper += 1;
```

```
        }
        if (s[i] >= 'a' && s[i] <= 'z'){
            Number_Lower += 1;
        }
        if (s[i] >= '0' && s[i] <= '9'){
            Number_Numbers += 1;
        }
        if (find(spec.begin(), spec.end(), s[i]) != spec.end()){
            Number_Symbols += 1;
        }
    }

    int check = (Number_Upper > 0) + (Number_Numbers > 0) + (Number_Symbols >
0);
    if (check == 3){
        Bonus_Combo = 25;
    }
    else if (check == 2){
        Bonus_Combo = 15;
    }

    if (Number_Lower == s.length()){
        Bonus_FlatLower = -15;
    }

    if (Number_Numbers == s.length()){
        Bonus_FlatNumber = -35;
    }

    int score = BaseScore + (Number_Execess*Bonus_Excess) +
(Number_Upper*Bonus_Upper)+ (Number_Numbers*Bonus_Number) +
(Number_Symbols*Bonus_Symbols) +Bonus_Combo + Bonus_FlatLower +
Bonus_FlatNumber;

    if (score < 50) cout << "Yeu";
    else if (score < 75) cout << "Vua";
    else if (score < 100) cout << "Manh";
    else cout << "RatManh";
}
```

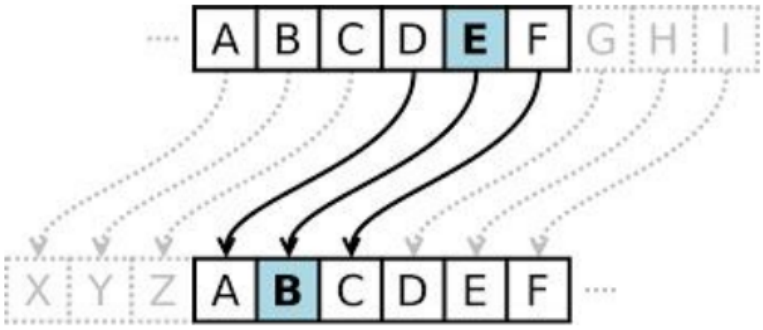
CaesarCipher

Đề bài

Trong mật mã học, **mật mã Caesar**, (còn được gọi là **Caesar code** hoặc **Caesar shift**), là một trong những kỹ thuật mã hóa đơn giản nhất và được biết đến rộng rãi nhất. Nó là một loại mật mã thay thế, trong đó mỗi chữ cái trong **plaintext** được thay thế bằng một bức thư một số số cố định (**key**) của các vị trí lên/xuống bảng chữ cái. Phương pháp này được đặt theo tên của Julius Caesar, người sử dụng nó trong thư riêng của ông.

Mã hóa cũng có thể được biểu diễn thông qua số học mô-đun bằng cách gán các ký tự bằng các con số, theo tuần tự, **A = 0, B = 1, ..., Z = 25**. Mã hóa một chữ cái *x* bằng phép dịch chuyển **k** vị trí (**key**) có thể mô tả bằng biểu thức toán học sau:

E(x)=(x+k) mod 26



Dữ liệu:

- Dòng đầu tiên số tự nhiên K cho biết độ dịch chuyển (0 <= K <= 10^9)
- Dòng tiếp theo là một chuỗi ký tự hoa có độ dài bé hơn bằng 10000 ký tự.

Kết quả:

Kết quả là chuỗi ký tự (**cipher text**) cho biết kết quả của phép mã hóa Caesar.

Ví dụ:

INPUT		OUTPUT
10 THE END		DRO OXN

Lời giải

- Tóm tắt: Cho shift và string s, in ra dữ liệu đã được encrypt bằng caesar cipher
- Thuật toán: Sử dụng thuật toán caesar cipher trong tài liệu https://en.wikipedia.org/wiki/Caesar_cipher
- Độ phức tạp thuật toán: O(length(s))

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

string cipher(string str, int shift) {
    string result = "";

    for (int ch : str) {
        if (ch >= 'A' && ch <= 'Z') {
            ch = (ch - 'A' + shift) % 26 + 'A';
        }
        else if (ch >= 'a' && ch <= 'z') {
            ch = (ch - 'a' + shift) % 26 + 'a';
        }
        result += ch;
    }

    return result;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int shift;
    string s;
    cin >> shift;
    cin.ignore();
    getline(cin, s);
    string ans = cipher(s, shift);
    cout << ans;

}
```

ReversingEncryption

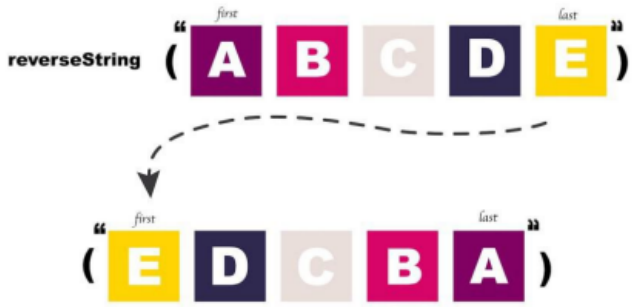
Đề bài

Một chuỗi s (**plaintext**) có độ dài n có thể được mã hóa bằng thuật toán sau:

+ Duyệt trên tất cả các ước của ntheo thứ tự giảm dần từ n đến 1.

+ Với mỗi ước số d, đảo ngược chuỗi con s [1… d] (tức là chuỗi con bắt đầu ở vị trí 1 và kết thúc ở vị trí d).

Ví dụ, thuật toán trên được áp dụng cho chuỗi s= "ATTN2020" dẫn đến những thay đổi sau: "0202NTTA" → "2020NTTA" → "0220NTTA" → "0220NTTA" (rõ ràng là thao tác dảo ngược cuối cùng không thay đổi chuỗi vì d = 1).



Bây giờ, bạn sẽ được cung cấp chuỗi đã được mã hóa t (**ciphertext**) .

Nhiệm vụ của bạn là giải mã chuỗi t này, tức là, để tìm một chuỗi s sao cho sau khi thực hiện thuật toán trên cho kết quả là chuỗi t. Luôn đảm bảo chuỗi s này luôn tồn tại và là duy nhất.

Dữ Liệu:

Đầu vào là chuỗi t là **ciphertext** gồm các chữ cái và chữ số (|t| <= 300).

Kết quả:

Kết quả là chuỗi ký tự ban đầu (**plaintext**)

Ví dụ:

INPUT		OUTPUT
0220NTTA		**ATTN2020**

Lời giải

- Tóm tắt: Cho string s đã được mã hóa bằng thuật toán trong đề bài, yêu cầu tìm chuỗi ban đầu.

- Thuật toán: Làm ngược lại với thuật toán mã hóa

- Độ phức tạp thuật toán: O(length(s))

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

string rev(string s){
    string tmp = s;
    for (int i=0; i<s.length(); i++){
        tmp[s.length() - i -1] = s[i];
    }
    return tmp;
}

string split(string s, int k){
    // cout << k << "\n";
    string tmp1, tmp2;
    for (int i=0; i<k; i++){
        tmp1 += s[i];
    }
    for (int i=k; i<s.length(); i++){
        tmp2 += s[i];
    }
    return rev(tmp1) + (tmp2);
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    string s;
    cin >> s;
    int n = s.length();
    vector<int> v;
    for (int i=1; i<=n ;i++){
        if (n % i == 0){
            v.push_back(i);
        }
    }

    for (int i = 0; i < v.size(); i++){
        s = split(s, v[i]);
    }
```

```
    cout << s;
}
```

Messages

Đề bài

Thầy Sơn cần gửi một văn bản quan trọng tới một đối tác của mình. Văn bản là một xâu *S* các chữ cái la tinh in thường. Để bảo mật nội dung văn bản, thầy Sơn gửi 2 bức thư. Bức thư thứ nhất là phần đầu Sb của xâu *S*, bức thư thứ 2 là phần cuối Se của *S*. Hai bức thư *Sb* và *Se* đảm bảo đầy đủ nội dung của *S*, tuy nhiên có thể một phần cuối của *Sb* có thể được viết lặp lại trong phần đầu của *Se*, song số kí tự được viết lặp lại không biết trước.

Yêu cầu: Cho hai xâu *Sb* và *Se*, hãy xác định một xâu *S* có thể là nội dung của bức thư sao cho độ dài của xâu *S* là ngắn nhất.

Dữ liệu:

Dòng đầu chứa xâu *Sb*, dòng thứ hai chứa xâu *Se*. Mỗi xâu có độ dài không quá 500, gồm các chữ cái la-tinh thường

Kết quả:

Dòng đầu tiên là độ dài của xâu S ban đầu. Dòng thứ hai xuất ra xâu S.

Ví dụ:

INPUT		OUTPUT
ATTN2020 **2020K15**		**11** **ATTN2020K15**

Lời giải

- Tóm tắt: Cho 2 string, nối 2 tring bằng phần đầu và phần cuối sao cho độ dài nhỏ nhất

- Thuật toán: Duyệt trâu các substring của 2 string sau đó thử nối và tìm ra string nhỏ nhất

- Độ phức tạp thuật toán: O(max(length(sb), length(se))**2)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
        freopen("TASK.INP", "r", stdin);
        freopen("TASK.OUT", "w", stdout);
    }

    string sb, se;
    cin >> sb >> se;
    int lsb = sb.length();
    int lse = se.length();

    string ans = "";
    int lans = lsb + lse;

    for (int i = 0; i < lsb; i++) {
        for (int j = 0; j <= lse; j++) {
            if (sb.substr(i) == se.substr(0, lse - j)) {
                string tmp = sb + se.substr(lse - j);
                if (tmp.length() < lans) {
                    lans = tmp.length();
                    ans = tmp;
                }
            }
        }
    }

    if (lans == lsb + lse) {
        cout << lans << "\n" << sb + se;
    } else {
        cout << lans << "\n" << ans;
    }

    return 0;
}
```

Binary Search 1

Đề bài

Tìm kiếm nhị phân

Bài toán:

Thuật toán tìm kiếm nhị phân là một thuật toán tìm kiếm đơn giản nhưng có tốc độ tìm kiếm cực kỳ nhanh. Ý tưởng của thuật toán là: với một mảng N phần tử đã được sắp xếp (tăng dần/hoặc giảm dần tùy theo người ra đề) và số nguyên x cần tìm, ta sẽ thực hiện tìm kiếm ở miền có khả năng xuất hiện x sau mỗi lần lặp.

Yêu cầu: Hãy cài đặt thuật toán tìm kiếm nhị phân để tìm vị trí của phần tử x trong mảng có N phần tử và đếm số lần duyệt qua các phần tử.

Input :

- Dòng đầu tiên là số nguyên N dương (0 < N < 32000)
- Dòng tiếp theo chứa N số nguyên Ai là các phần tử của mảng đã được sắp xếp tăng dần và không trùng nhau
- Dòng cuối cùng là số nguyên x cần tìm

Output:

- Dòng đầu tiên là vị trí của x được tìm thấy trong mảng. Nếu không tìm thấy thì xuất ra -1
- Dòng tiếp theo là số lần duyệt qua các phần tử để tìm được x. Nếu không tìm thấy thì không cần xuất ra dòng này

Ví dụ:

INPUT	OUTPUT	Giải thích
8 1 3 4 5 10 12 15 20 4	2 3	- 4 được tìm thấy ở vị trí 2 - để tìm được 3 cần phải duyệt qua các phần tử ở vị trí 3, 1 và 2.
4 1 2 3 4 -9	-1	- không tìm thấy -9 trong mảng

Lời giải

- Tóm tắt: Tìm vị trí của x trong mảng đã sắp xếp và số luật duyệt qua
- Thuật toán: Sử dụng thuật toán binary search
- Độ phức tạp thuật toán: O(log n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
```

```cpp
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> v;
int ans = 0;
int bs(vector<int> A, int n, int T){
    int L = 0;
    int R = n - 1;
    while (L <= R){
        int m = (L + R) >> 1;
        // cout << m << "\n"
        if (A[m] < T){
            L = m + 1;
        }
        else if (A[m] > T){
            R = m - 1;
        }
        else return m;
        ans += 1;
    }
    return -1;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n;
    cin >> n;
    for (int i=0; i<n; i++){
        int x;
        cin >> x;
        v.push_back(x);
    }
    int k;
    cin >> k;
    int res = bs(v, v.size(), k);
    if (res == -1){
        cout << res;
    }
    else{
        cout << res  << "\n";
        cout << ans + 1;
    }
}
```

Binary Search 2

Đề bài

Tìm kiếm nhị phân 2

Bài toán:

Thuật toán tìm kiếm nhị phân là một thuật toán tìm kiếm đơn giản nhưng có tốc độ tìm kiếm cực kỳ nhanh. Tuy nhiên, đa phần các hướng dẫn và khóa học trên mạng đều hướng dẫn người học cài đặt trên mảng số nguyên trong khi trong thực tế chỉ cần mảng có thể sắp xếp được thì ra có thể áp dụng thuật toán tìm kiếm nhị phân.

Yêu cầu: Hãy cài đặt thuật toán tìm kiếm nhị phân trên mảng chuỗi có N phần tử.

Input :

- Dòng đầu tiên là số nguyên N dương (0 < N < 2000)
- N dòng tiếp theo chứa các phần tử Ai trong mảng, mỗi phần tử là một chuỗi với độ dài không quá 10. Các phần tử này đã được sắp xếp theo thứ tự bảng chữ cái
- Dòng cuối cùng là chuỗi x cần tìm

Output:

- Dòng đầu tiên là vị trí của x được tìm thấy trong mảng. Nếu không tìm thấy thì xuất ra -1
- Dòng tiếp theo là số lần duyệt qua các phần tử để tìm được x. Nếu không tìm thấy thì không cần xuất ra dòng này

Ví dụ:

INPUT	OUTPUT	Giải thích
8 2024 Cau Giai Lieu Thuat Va du truc Va	5 2	- 'Va' được tìm thấy ở vị trí 5 - để tìm được 'Va' cần phải duyệt qua 2 vị trí là 3 và 5

Lời giải

- Tóm tắt: Giống bài Binary Search 1 nhưng tìm string
- Thuật toán: Sử dụng hàm Binary Search của bài trước và thay vector bằng string sau đó sort
- Độ phức tạp thuật toán: O(log n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

vector<int> v;
int ans = 0;
```

```cpp
int bs(vector<int> A, int n, int T){
    int L = 0;
    int R = n - 1;
    while (L <= R){
        int m = (L + R) >> 1;
        // cout << m << "\n"
        if (A[m] < T){
            L = m + 1;
        }
        else if (A[m] > T){
            R = m - 1;
        }
        else return m;
        ans += 1;
    }
    return -1;
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n;
    cin >> n;
    for (int i=0; i<n; i++){
        int x;
        cin >> x;
        v.push_back(x);
    }
    int k;
    cin >> k;
    int res = bs(v, v.size(), k);
    if (res == -1){
        cout << res;
    }
    else{
        cout << res  << "\n";
        cout << ans + 1;
    }
}
```

Linear Search 1

Đề bài

Bài toán:

Thuật toán tìm kiếm tuyến tính là một trong những thuật toán tìm kiếm đơn giản và dễ dàng cài đặt nhất. Ý tưởng của thuật toán là: với một mảng cho trước có N phần tử và phần tử cần tìm x, duyệt từ đầu mảng đến cuối mảng cho đến khi tìm được phần tử x.

Yêu cầu: Hãy cài đặt thuật toán tìm kiếm tuyến tính để tìm vị trí của x đầu tiên trong mảng có N phần tử và đếm số lần duyệt qua các phần tử.

Input :

- Dòng đầu tiên là số nguyên N dương (0 < N < 10000)
- Dòng tiếp theo chứa N số nguyên Ai là các phần tử của mảng
- Dòng cuối cùng là số nguyên x cần tìm

Output:

- Dòng đầu tiên là vị trí của x đầu tiên trong mảng nếu tìm được. Nếu không tìm thấy thì xuất ra -1
- Số lần duyệt qua các phần tử của mảng (từ đầu mảng đến cuối) để tìm được x
- Dòng thứ ba là vị trí của x đầu tiên trong mảng (đếm từ cuối đến đầu) nếu tìm được. Nếu không tìm thấy thì xuất ra -1
- Số lần duyệt qua các phần tử của mảng (từ cuối mảng đến đầu) để tìm được x
- Nếu như ở 2 dòng đầu tiên mà không tìm thấy thì không cần xuất ra 2 dòng tiếp theo

Ví dụ:

INPUT	OUTPUT	Giải thích
5 1 -4 3 2 6 -4	1 2 3 4	• -4 ở vị trí 1 tính từ đầu mảng • phải duyệt qua 2 lần để tìm được -4 • -4 ở vị trí 3 tính từ cuối mảng • phải duyệt qua 4 lần từ cuối mảng để tìm được -4
4 1 2 3 4 -9	-1	Không có phần tử -9 trong mảng

Lời giải

- Tóm tắt: Tìm vị trí của x khi duyệt từ đầu mảng và cuối mảng
- Thuật toán: Duyệt từ trái sang phải và phải sang trái của mảng
- Độ phức tạp thuật toán: O(n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

signed main(){
```

```
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    map<int, int> m, m2;
    bool flag = 0;
    int n;
    cin >> n;
    for (int i=0; i<n; i++) {
        int x;
        cin >> x;
        m[x] = i;
        m2[x] = 1;
    }
    int k;
    cin >> k;
    if (!m2[k]){
        cout << -1;
    }
    else{
        cout << m[k] << "\n" << m[k] + 1 << "\n";
        cout << n - 1 - m[k] << "\n" << n - m[k];
    }
}
```

Linear Search 2

Tìm kiếm tuyến tính 2

Bài toán:

Thuật toán tìm kiếm tuyến tính là một trong những thuật toán tìm kiếm đơn giản và dễ dàng cài đặt nhất. Ý tưởng của thuật toán là: với một mảng cho trước có N phần tử và phần tử cần tìm x, duyệt từ đầu mảng đến cuối mảng cho đến khi tìm được phần tử x.

Yêu cầu: Hãy cài đặt thuật toán tìm kiếm tuyến tính để tìm vị trí của tất cả x trong mảng có N phần tử và đếm số lần duyệt qua các phần tử.

Input :

- Dòng đầu tiên là số nguyên N dương (0 < N < 10000)
- Dòng tiếp theo chứa N số nguyên Ai là các phần tử của mảng
- Dòng cuối cùng là số nguyên x cần tìm

Output:

- Dòng đầu tiên là số lần x được tìm thấy trong mảng
- Mỗi dòng tiếp theo tương ứng với vị trí của x và số lần duyệt từ đầu mảng để tìm được x

Ví dụ:

INPUT	OUTPUT	Giải thích
8 1 3 -4 2 6 10 3 12 3	2 1 2 6 7	- 3 được tìm thấy ở 2 vị trí là 1 và 6, tương ứng phải duyệt 2 và 7 lần để tìm được
4 1 2 3 4 -9	0	- không tìm thấy -9 trong mảng

Đề bài

Lời giải

- Tóm tắt: Giống bài trước nhưng tìm ở mọi chỗ
- Thuật toán: Duyệt tùng phần tử và lưu lại vị trí
- Độ phức tạp thuật toán: O(n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;
```

```cpp
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    vector<int> v1(1000001), v2;
    int n;
    cin >> n;
    for (int i=1; i<=n; i++){
        int x;
        cin >> x;
        v1[i] = x;
    }
    int k;
    cin >> k;
    for (int i=1; i<=n; i++){
        if (v1[i] == k){
            v2.push_back(i);
            // cout << i-1 << " " << i << "\n";
        }
    }
    cout << v2.size() << "\n";
    for (auto i : v2){
        cout << i-1 << " " << i << "\n";
    }

}
```

Linear Search 3

Đề bài

Find MEX

Bài toán:

Cho trước một mảng có N phần tử. Hãy in ra MEX_i với $0 \leq i < n$.

Biết rằng, MEX_i là số nguyên nhỏ nhất lớn hơn hoặc bằng 0 mà không xuất hiện trong từ đầu mảng cho đến phần tử i.

Input :

- Dòng đầu tiên là N $(1 \leq N \leq 2 \times 10^5)$ thể hiện số phần tử của mảng
- Dòng tiếp theo chứ N số nguyên là các phần tử của mảng

Output:

- N số nguyên tương ứng với MEX_i

Ví dụ:

INPUT	OUTPUT	Giải thích
5 1 0 5 5 3	0 2 2 2 2	- Với i = 0, thì số nhỏ nhất chưa xuất hiện từ đầu mảng đến phần từ 0 là 0. - Với i = 1, thì số nhỏ nhất chưa xuất hiện từ đầu mảng đến phần từ 1 là 2. - Với i = 2, thì số nhỏ nhất chưa xuất hiện từ đầu mảng đến phần từ 2 là 2. - Với i = 3, thì số nhỏ nhất chưa xuất hiện từ đầu

Lời giải

- Tóm tắt: Với mọi vị trí tìm số nguyên lớn hơn hoặc bằng 0 nhỏ nhất mà chưa xuất hiện

- Thuật toán: Với mỗi vị trí lưu lại số và duyệt từ 0, nếu số đó chưa được đánh dấu thì in ra.

- Độ phức tạp thuật toán: O(n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;
```

```
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n;
    cin >> n;
    vector<int> v(1000001), v2(1000001);
    for (int i=1; i<=n; i++){
        cin >> v[i];
    }
    int k = 0;
    for (int i=1; i<=n; i++){
        v2[v[i]] = 1;
        for (k; k<=1000001; k++){
            if (v2[k] == 0){
                cout << k << " ";
                break;
            }
        }
    }

}
```

Linear Search 5

Đề bài

Max different

Bài toán:

Trọng số của một mảng được định nghĩa là sự khác biệt giữa phần tử lớn nhất và nhỏ nhất của mảng. Ví dụ: trọng số của mảng [3, 7, 1, 2] là (7 - 1) = 6, trọng số của mảng [2] là (2 - 2) = 0, trọng số của mảng trống là 0.

Cho một mảng A có độ dài N. Bạn phải chia các phần tử của A thành hai dãy con S1 và S2 (một trong S1, S2 có thể trống) sao cho:

- Mỗi phần tử của mảng A chỉ thuộc một trong hai dãy S1 hoặc S2.
- Tổng trọng số của hai dãy S1 và S2 là lớn nhất.

Bạn in ra đáp án lớn nhất có thể của tổng trọng số của hai dãy S1 và S2.

Input :

- Dòng đầu tiên chứa T là số lượng test. Mô tả các trường hợp thử nghiệm T như sau:
- Với mỗi test case:
  - Dòng đầu tiên chứa N là kích thước của mảng A.
  - Dòng thứ hai chứa N số nguyên A[1], A[2],....., A[N] - biểu thị các phần tử của A.

Constraints:

- 1 <= T <= 10^4
- 2 <= N <= 10^5

Constraints:

- 1 <= T <= 10^4
- 2 <= N <= 10^5
- 1 <= A[i] <= 10^9
- Tổng của N tren tất cả các test case không quá 2×10^5

Output:

- Đối với mỗi trường hợp thử nghiệm, hãy in tổng trọng số tối đa có thể có của hai dãy con.

Ví dụ:

INPUT	OUTPUT	Giải thích
2 3 1 2 3 5 4 8 1 10 2	2 15	- Một phép chia có thể là [1, 3], [2] Ở đây tổng trọng số là = (3 - 1) + (2 - 2) = 2 - Ở testcase số 2, có thể chia thành 2 dãy [4, 1, 10] và [8, 2] thì tổng trọng số là (10 - 1) + (8 - 2) = 15

Lời giải

- Tóm tắt: Chia mảng thành 2 mảng sao cho tổng của hiệu số nhỏ nhất và lớn nhất của 2 mảng lớn nhất.
- Thuật toán: Sắp xếp lại mảng và lấy 2 số nhỏ nhất và lớn nhất.
- Độ phức tạp thuật toán: O(n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;
```

```cpp
signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int T;
    cin >> T;
    while (T--){
        vector<int> v;
        int n;
        cin >> n;
        for (int i=1; i<=n; i++){
            int x;
            cin >> x;
            v.push_back(x);
        }
        sort(v.begin(), v.end());
        int a = v[v.size()-1] + v[v.size()-2] - v[0] - v[1];
        int b = v[v.size()-1] - v[0];
        cout << max(a, b) << "\n";
    }

}
```

VW05p_Enrichement

Đề bài

Các nhà địa chất phát hiện một khu mỏ quặng dất hiếm, một thứ rất cần thiết cho công nghiệp chế tạo thiết bị điện tử. Khu mỏ có hình chữ nhật kích thước n×m ô. Trữ lượng quặng ở ô (i, j) được đánh giá là $a_{i,j}(1 \le i \le n, 1 \le j \le m)$. Cần xây dựng một xí nghiệp làm giàu quặng trước khi đưa ra thị trường. Do điều kiện địa hình, xí nghiệp phải xây dựng ngay trên khu mỏ. Xí nghiệp chiếm diện tích 3×3 ô. Dĩ nhiên không thể khai thác quặng dưới nền của xí nghiệp, vì vậy người ta muốn tìm vị trí đặt xí nghiệp sao cho tổng trữ lượng phải để lại là ít nhất.
Hãy xác định tổng trữ lượng nhỏ nhất phải để lại.

Dữ liệu vào :
Dòng đầu tiên chứa 2 số nguyên n và m $(3 \le n, m \le 10^6; 1 \le n \times m \le 10^6)$,
Dòng thứ i trong n dòng sau chứa m số nguyên a_{i1}, a_{i2}, ... ,a_{im} $(0 \le a_{i,j} \le 10^5, 1 \le j \le m)$.
Kết quả: Đưa ra một số nguyên – tổng trữ lượng nhỏ nhất phải để lại.
Ví dụ:

Input	Output
5 7 10 2 3 7 10 4 8 3 2 1 9 6 2 1 0 3 6 7 8 9 10 5 4 3 0 2 1 8 9 2 3 10 6 4 8	27

Lời giải

- Tóm tắt: Tìm mảng 3x3 có tổng lớn nhất
- Thuật toán: Sử dụng mảng tiền tố 2 chiều
- Độ phức tạp thuật toán: O(m*n)

Code mẫu

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;



signed main(){
```

```
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    vector<vector<int>> g(n, vector<int>(m));
    vector<vector<int>> ps(n + 1, vector<int>(m + 1, 0));

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> g[i][j];
        }
    }

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            ps[i][j] = g[i - 1][j - 1] + ps[i - 1][j] + ps[i][j - 1] - ps[i -
1][j - 1];
        }
    }

    int ans = 10000001;

    for (int i = 2; i < n; i++) {
        for (int j = 2; j < m; j++) {
            int sum = ps[i + 1][j + 1] - ps[i - 2][j + 1] - ps[i + 1][j - 2]
+ ps[i - 2][j - 2];

            ans = min(ans, sum);
        }
    }

    cout << ans;

}
```