Chương 2. KIỂU MẢNG

2.1. Định nghĩa mảng

Mảng là một tập hữu hạn các phần tử có cùng kiểu dữ liệu và được lưu trữ liên tiếp trong bộ nhớ (RAM). Mảng có thể được xem như là một tập các biến cùng kiểu dữ liệu với nhau. Mỗi phần tử của mảng được xác định và truy xuất bởi chỉ số tương ứng trong mảng. Số phần tử của mảng được gọi là kích thước mảng.

Mảng được sử dụng khi chương trình cần lưu trữ nhiều giá trị có cùng kiểu. Mảng giúp cho việc thao tác trên một tập dữ liệu được tiện lợi hơn. Bên cạnh đó mảng còn được sử dụng để lưu trữ dữ liệu tra cứu nhằm tăng hiệu quả xử lý của thuật toán.

Mảng được chia thành 2 loại: mảng một chiều và mảng nhiều chiều.

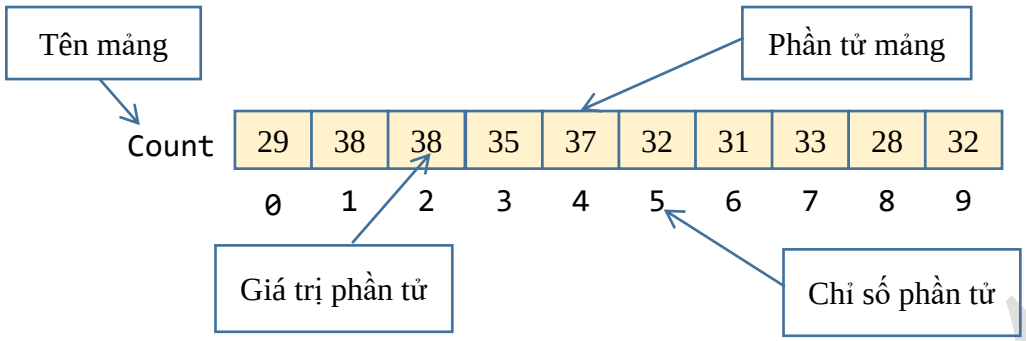
2.2. Mảng một chiều

Mảng một chiều được sử dụng khi chương trình cần lưu trữ một tập giá trị cùng kiểu có dạng dãy. Các phần tử của mảng được đánh chỉ số bắt đầu từ 0, nghĩa là mảng có kích thước n thì các phần tử được đánh chỉ số từ 0 đến $n-1$.

Xét bảng dữ liệu về danh sách các lớp như sau:

STT	Tên Lớp	Phòng học	Số HS
1	10A1	001	29
2	10A2	002	38
3	10A3	003	38
4	10A4	004	35
5	10A5	005	37
6	10B1	006	32
7	10B2	101	31
8	10B3	102	33
9	10B4	103	28
10	10B5	104	32

Cột dữ liệu về số lượng học sinh của từng lớp trong bảng trên tạo thành mảng một chiều được minh họa như hình bên dưới, danh sách dữ liệu về số lượng học sinh được đặt tên `Count`.



Mỗi ô trong dãy biểu diễn một phần tử của mảng. Phần tử thứ 0 có giá trị 29 cho biết số học sinh của lớp thứ 0 là 29, tương tự phần tử thứ 1 có giá trị 38, …

2.2.1. Khai báo mảng

Cú pháp: `<kiểu dữ liệu> <tên_mảng>[<kích thước mảng>];`

> ***Lưu ý:*** Mảng *nên được khai báo toàn cục* khi có kích thước lớn (cỡ 10^3 phần tử trở lên).

Ví dụ 2.2.1: Khai báo 2 mảng lưu giá trị 2 dãy số nguyên $a_0, a_1, \ldots, a_{n-1}$ và $b_0, b_1, \ldots, b_{m-1}$ với $(n, m \leq 10^3)$.

```
int a[1000], b[1000];
```

Ví dụ 2.2.2: Khai báo mảng lưu trữ điểm trung bình học kì của các học sinh trong một lớp.

```
double GPA[100];
```

Ví dụ 2.2.3: Khai báo mảng số nguyên a có kích thước 1000 và khởi tạo tất cả phần tử của mảng có giá trị 0.

```
int a[1000] = {0};
```

> ***Lưu ý:*** Khi khai báo và khởi tạo như sau `int a[1000] = {10};` thì chỉ có phần tử `a[0]` có giá trị 10, các phần tử còn lại có giá trị 0.

2.2.2. Truy xuất phần tử mảng

Phần tử của mảng được truy xuất thông qua tên mảng và chỉ số của phần tử tương ứng.

Mỗi phần tử của mảng tương đương với một biến trong chương trình.

Cú pháp: `<tên_mảng>[chỉ_số];`

Ví dụ 2.2.4:

```
a[1] = 10;

cout<<GPA[1];

++b[0];

cout<<b[0];
```

> ***Lưu ý:*** chỉ số phần tử của mảng phải là một giá trị nguyên thuộc phạm vi từ `0` đến `<size>-1`, nghĩa là `0 ≤ chỉ_số ≤ <size>-1`, với `<size>` là kích thước mảng.

2.2.3. Một số ví dụ minh họa

Ví dụ 2.2.5: Cho số nguyên n và dãy số nguyên $a_0, a_1, \dots, a_{n-1} (n \leq 10^6)$. Tính tổng các phần tử chẵn của dãy.

Input: Dòng đầu chứa số nguyên n. Dòng tiếp theo chứa dãy $a_0, a_1, \dots, a_{n-1}$.

Output: Một số nguyên là tổng tìm được.

```
#include <iostream>

using namespace std;
#define maxN 1000000 //định nghĩa hằng số

int a[maxN], n;

void ReadData()
{
    cin>>n;
    for (int i = 0; i < n; ++i)
        cin>>a[i];
}

int Solve()
{
    int s = 0;
    for (int i = 0; i < n; ++i)
        if (a[i] % 2 == 0)
            s = s + a[i];
    return s;
}
```

```
int main()
{
    ReadData();
    int res = Solve();
    cout<<res;
    return 0;
}
```

Lưu ý: Nên chia các công việc xử lý thành các hàm, mỗi hàm thực hiện một tác vụ độc lập của thuật toán.

Ví dụ 2.2.6: Cho 2 số nguyên n, m và 2 dãy số nguyên $a_0, a_1, \dots, a_{n-1} (n \leq 10^6)$ và $b_0, b_1, \dots, b_{m-1} (m \leq 10^6)$. Tính độ lệch giữa 2 phần tử lớn nhất của 2 dãy.

Input:

Dòng đầu chứa hai số nguyên n, m.

Dòng thứ hai chứa dãy $a_1, a_2, \dots, a_n$.

Dòng thứ ba chứa dãy $b_1, b_2, \dots, b_m$.

Output: Một số nguyên là kết quả tìm được.

```
#include <iostream>
#include <cmath>

using namespace std;
#define maxN 1000000

int a[maxN], b[maxN], n, m;

void ReadData()
{
    cin>>n>>m;
    for (int i = 0; i < n; ++i)
        cin>>a[i];
    for (int i = 0; i < m; ++i)
        cin>>b[i];
}

int MaxVal(int a[], int n) //truyền mảng là tham số cho hàm
{
    int res = a[0];
    for (int i = 1; i < n; ++i)
        if (res < a[i])
            res = a[i];
    return res;
}
```

haipt

```
int main()
{
    ReadData();
    cout<<abs(MaxVal(a, n) - MaxVal(b, m));
    return 0;
}
```

Lưu ý

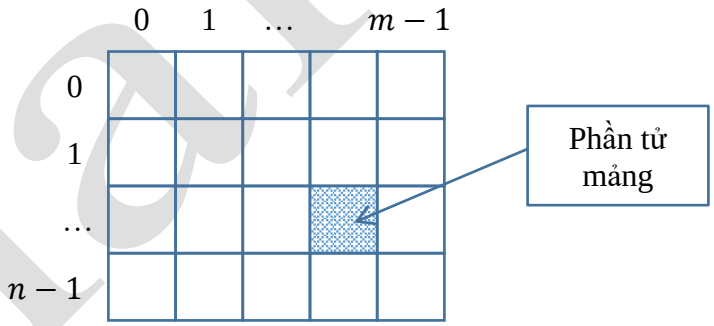
- Nếu sử dụng mảng là tham số cho hàm thì khi khai báo không cần chỉ định kích thước. Khi mảng là tham số của hàm thì mảng có vai trò là tham biến.
- Sử dụng mảng là tham số cho hàm giúp hàm có tính tái sử dụng trong trường hợp thực hiện các thao tác tương tự trên các mảng khác nhau.

2.3. Mảng hai chiều

Mảng 2 chiều (bảng) được sử dụng khi chương trình cần lưu trữ một tập dữ liệu có cùng kiểu và có dạng bảng gồm các dòng và cột. Dòng và cột của mảng 2 chiều được đánh chỉ số là các số nguyên liên tiếp bắt đầu từ 0 (zero-base index).

Mảng 2 chiều có thể được xem như là mảng 1 chiều mà mỗi phần tử là một mảng 1 chiều.

Hình ảnh minh họa mảng 2 chiều kích thước $n \times m$ (n dòng, m cột).



Dòng được đánh số thứ tự liên tiếp từ trên xuống và bắt đầu từ 0, cột được đánh thứ tự liên tiếp từ trái qua phải và bắt đầu từ 0.

Mỗi phần tử của bảng tương đương một biến trong chương trình và được xác định thông qua chỉ số dòng, cột của nó trong bảng.

2.3.1. Khai báo mảng

Cú pháp:

```
<kiểu dữ liệu> <tên_mảng>[<kích thước dòng>][<kích thước cột>];
```

Ví dụ 2.3.1:

Khai báo mảng 2 chiều kích thước $n \times m (n, m \leq 10^3)$ lưu trữ các số nguyên. Dòng được đánh thứ tự từ 0 đến $n-1$, cột được đánh thứ tự từ 0 đến $m-1$.

```
#define maxN 1000
int a[maxN][maxN];
```

Lưu ý: Mảng nhiều chiều có cú pháp khai báo tổng quát như sau

```
<kiểu dữ liệu> <tên_mảng>[<size_1>][<size_2>]…[<size_k>];
```

Trong đó `size_1`, `size_2`, …, `size_k` là kích thước của k chiều tương ứng.

2.3.2. Truy xuất phần tử mảng

Phần tử của mảng được truy xuất thông qua tên mảng cùng với cặp chỉ số dòng, cột của phần tử tương ứng.

Cú pháp: `<tên_mảng>[chỉ_số_dòng][chỉ_số_cột];`

Ví dụ 2.3.2:

```
a[1][1] = 10;

cin>>a[i][j];

cout<<a[10][5];
```

2.3.3. Một số ví dụ minh họa

Ví dụ 2.3.3: Cho bảng số nguyên kích thước $n \times m (n, m \leq 10^3)$. Tìm vị trí của phần tử lớn nhất bảng.

Input: Dòng đầu chứa 2 số nguyên n, m. Dòng thứ i trong n dòng tiếp theo chứa dãy gồm m số nguyên $a_{i1}, a_{i2}, \ldots, a_{im}$ mô tả các phần tử trên dòng thứ i của mảng.

Output: Hai số nguyên x, y tương ứng với chỉ số dòng và chỉ số cột của phần tử lớn nhất bảng. Nếu có nhiều kết quả thì tìm x, y bất kỳ. Chỉ số của mảng được tính từ 1.

```
#include <iostream>

using namespace std;
#define maxN 1000
```

```
int a[maxN+1][maxN+1], n, m; //Tăng kích thước thêm 1 vì chỉ số tính từ 1

void ReadData()
{
    cin>>n>>m;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            cin>>a[i][j];
}

void Solve()
{
    int x = 1, y = 1;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            if (a[i][j] > a[x][y])
                x = i, y = j; //câu lệnh kép

    cout<<x<<" "<<y;
}

int main()
{
    ReadData();
    Solve();
    return 0;
}
```

Ví dụ 2.3.4: Cho bảng vuông các số nguyên kích thước $n \times n (n \leq 10^3)$. Cho biết các phần tử của bảng có đối xứng qua đường chéo chính hay không. Đường chéo chính của bảng là đường chéo bắt đầu từ ô $(1,1)$ đến ô (n,n).

Input: Dòng đầu chứa số nguyên n. Mỗi dòng trong n dòng tiếp theo chứa dãy gồm n số nguyên mô tả các phần tử trên bảng.

Output: Thông báo `YES` tương ứng với câu trả lời có, ngược lại thông báo `NO`.

```
#include <iostream>

using namespace std;
#define maxN 1000 //định nghĩa hằng số maxN có giá trị 1000

int a[maxN][maxN], n, m;

void ReadData()
{
    cin>>n;
    for (int i = 0; i < n; ++i)
```

haipt

```
        for (int j = 0; j < n; ++j)
            cin>>a[i][j];
}

bool Solve()
{
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < i; ++j)
            if (a[i][j] != a[j][i])
                return false;

    return true;
}

int main()
{
    ReadData();
    if (Solve())
        cout<<"YES";
    else cout<<"NO";
    return 0;
}
```

2.4. Mảng động với kiểu `vector` trong C++

Cách khai báo mảng như nội dung được trình bày ở trên được gọi là khai báo mảng tĩnh. Kích thước của mảng phải được xác định cố định ngay khi khai báo. Trong quá trình sử dụng, kích thước mảng không được thay đổi. Số phần tử của mảng không được vượt quá kích thước đã được chỉ định.

Ngôn ngữ C++ cung cấp kiểu dữ liệu `vector<kiểu phần tử>` thuộc thư viện `<vector>` có tính chất và cách sử dụng tương tự mảng tĩnh nhưng cho phép thay đổi kích thước (khi thêm mới hoặc hủy phần tử).

Ưu điểm của `vector` so với mảng tĩnh

- Không cần chỉ định kích thước khi khai báo.
- Không cần quản lý số phần tử.
- Không chứa các vùng nhớ dư thừa như mảng tĩnh nên tiết kiệm bộ nhớ và tiện lợi hơn trong quá trình sử dụng.

2.4.1. Khai báo vector

Cú pháp

```
vector<kiểu phần tử> <tên_vector>([<kích thước>],[<giá trị khởi tạo>]);
```

Ví dụ 2.4.1: Khai báo vector $v1$ các số nguyên.

```
vector<int> v1;
```

Ví dụ 2.4.2: Khai báo vector $v2$ các số nguyên có kích thước ban đầu là 100 và khởi tạo tất cả phần tử của vector có giá trị `0`.

```
vector<int> v2(100);
```

Ví dụ 2.4.3: Khai báo vector $v3$ các số nguyên có kích thước ban đầu là 100 và khởi tạo tất cả phần tử của vector có giá trị `-5`.

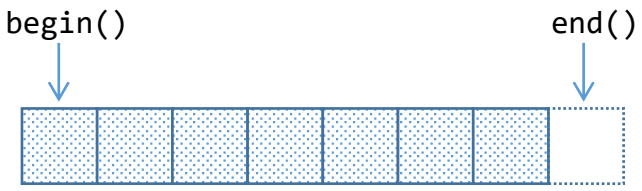
```
vector<int> v3(100, -5);
```

Ví dụ 2.4.4: Khai báo vector v các số nguyên `long long` có kích thước ban đầu là 10^5 và khởi tạo tất cả phần tử của vector có giá trị `1`.

```
vector<long long> v(100000, 1);
```

2.4.2. Một số phương thức hàm thông dụng trên vector

STT	Têm hàm	Diễn giải
1	`size()`	Trả về kích thước hiện tại của `vector`
2	`resize(<size>)`	Thay đổi kích thước của `vector` thành `<size>`
3	`push_back(<v>)`	Thêm phần tử có giá trị `<v>` vào cuối `vector`
4	`pop_back()`	Hủy phần tử cuối ra khỏi `vector`
5	`clear()`	Hủy tất cả phần tử ra khỏi `vector`
6	`begin()`	Trả về con trỏ đến địa chỉ phần tử đầu tiên của `vector`
7	`end()`	Trả về con trỏ đến địa chỉ sau phần tử cuối của `vector`



2.4.3. Một số ví dụ minh họa

Ví dụ 2.4.5: Cho dãy số nguyên $a_0, a_1, \dots, a_{n-1}$ và số nguyên x. Đếm số phần tử trong dãy có giá trị x.

Dữ liệu: Nhập từ bàn phím gồm 2 dòng

- Dòng đầu chứa 2 số nguyên $n, x (n \le 10^6; |x| \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_0, a_1, \dots, a_{n-1} (|a_i| \le 10^6)$.

Kết quả: Xuất ra màn hình số nguyên là số phần tử trong dãy có giá trị x.

```
#include <iostream>
using namespace std;
#define maxN 1000000 //định nghĩa hằng số maxN có giá trị 10^6

int a[maxN], n, x;

void ReadData()
{
    cin>>n>>x;
    for (int i = 0; i < n; ++i)
        cin>>a[i];
}

int Solve()
{
    int cnt = 0;
    for (int i = 0; i < n; ++i)
        if (a[i] == x)
            ++cnt;
    return cnt;
}

int main()
{
    ReadData();
    cout<<Solve();
    return 0;
}
```

Ví dụ 2.4.6: Cho dãy gồm n số nguyên có giá trị tuyệt đối không vượt quá 10^{12}. Đếm số phần tử của dãy là số Palindrome.

Dữ liệu: Nhập từ bàn phím gồm 2 dòng

- Dòng đầu tiên chứa số nguyên dương $n (n \le 10^5)$.
- Dòng thứ hai chứa dãy $a_0, a_1, \dots, a_{n-1} (|a_i| \le 10^{12})$.

haitpt

Kết quả: Xuất ra màn hình một số nguyên là số phần tử thỏa Palindrome.

```
#include <iostream>
#include <vector> //thư viện chứa kiểu vector
using namespace std;
typedef long long ll;
vector<ll> v;

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

bool checkPALIND(ll n)
{
    ll m1 = 0, m2 = n;
    while (n > 0)
    {
        m1 = m1*10 + n%10;
        n = n/10;
    }
    return (m1 == m2);
}

int Solve()
{
    int cnt = 0;
    for (int i = 0; i < v.size(); ++i)
        if (checkPALIND(v[i])) ++cnt;
    return cnt;
}

int main()
{
    ReadData();
    cout<<Solve();
    return 0;
}
```

Ví dụ 2.4.7: Cho dãy gồm $n(n \leq 10^5)$ số nguyên có giá trị tuyệt đối không vượt quá 10^6. Tìm số chính phương lớn nhất và lớn nhì dãy.

Dữ liệu: Nhập từ bàn phím gồm 2 dòng

- Dòng đầu tiên chứa số nguyên dương n.
- Dòng thứ hai chứa dãy $a_0, a_1, \dots, a_{n-1} (|a_i| \le 10^6)$.

Kết quả: Ghi 2 số nguyên $i, j (0 \le i, j < n)$ tương ứng với vị trí của số chính phương lớn nhất và lớn thứ nhì. Nếu không tìm được số chính phương ở vị trí nào thì ghi -1 ở vị trí tương ứng.

```
#include <iostream>
#include <vector>
#include <cmath> //thư viện chứa hàm sqrt

using namespace std;

vector<int> a;

void ReadData()
{
    int n;
    cin>>n;
    a.resize(n); //thay đổi kích thước của a từ 0 thành n
    for (int i = 0; i < n; ++i)
        cin>>a[i];
}

bool checkSQR(int n)
{
    if (n < 0) return false;
    int m = (int)sqrt(n); //ép kiểu: lấy phần nguyên của sqrt(n)
    return (m*m == n);
}

//tìm vị trí p của số chính phương lớn nhất thỏa p != id
int MaxSQR(int id)
{
    int p = -1;
    for (int i = 0; i < a.size(); ++i)
    {
        if (i != id && checkSQR(a[i]))
            if (p == -1 || a[i] > a[p])
                p = i;
    }
    return p;
}

int main()
{
    ReadData();
    int p1 = MaxSQR(-1); //vị trí số chính phương lớn nhất
```

haitpt

```
    int p2 = MaxSQR(p1); //vị trí số chính phương lớn nhì
    cout<<p1<<" "<<p2;
    return 0;
}
```

Ví dụ 2.4.8: Cho dãy số nguyên $a_0, a_1, \dots, a_{n-1}$. Tìm vị trí của phần tử dương chẵn lớn nhất dãy. Nếu dãy không tồn tại số dương chẵn thì ghi -1.

Dữ liệu: Vào từ tập tin văn bản *input.txt* gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n (n \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_0, a_1, \dots, a_{n-1} (|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản *output.txt* vị trí của phần tử dương chẵn lớn nhất hoặc -1.

```
#include <iostream>
#include <vector>
#include <cstdio>

using namespace std;

vector<int> a;

void ReadData()
{
    int n;
    cin>>n;
    a.resize(n);
    for (int i = 0; i < n; ++i)
        cin>>a[i];
}

int Solve()
{
    int pos = -1;
    for (int i = 0; i < a.size(); ++i)
    {
        if (a[i] > 0 && a[i] % 2 == 0)
            if (pos == -1 || a[i] > a[pos])
                pos = i;
    }
    return pos;
}

int main()
{
    freopen("input.txt", "r", stdin); //mở file input.txt để đọc
    freopen("output.txt", "w", stdout); //mở file output.txt để ghi
    ReadData();
```

haipt

```
    int p = Solve();
    cout<<p;
    return 0;
}
```

Ví dụ 2.4.9: Cho dãy gồm $n(n \leq 10^5)$ số nguyên có giá trị tuyệt đối không vượt quá 10^6. Hãy tìm đường chạy dài nhất của dãy.

Dữ liệu: Nhập từ tập tin văn bản *input.txt* gồm 2 dòng

- Dòng đầu tiên chứa số nguyên dương n.
- Dòng thứ hai chứa dãy $a_0, a_1, \ldots, a_{n-1}(|a_i| \leq 10^6)$.

Kết quả: Ghi ra tập tin văn bản *output.txt* các phần tử thuộc đường chạy dài nhất. Nếu có nhiều kết quả thì in đường chạy nằm trái nhất.

```
#include <iostream>
#include <vector>
#include <cstdio> //thư viện chứa hàm freopen

using namespace std;
#define maxN 100000

//mỗi vector lưu một đường chạy, khai báo mảng các vector
vector<int> a[maxN];
int m = 0; //m: số phần tử của mảng vector

void ReadData()
{
    int n, x1 = (-1e6) - 10, x2; //1e6 = 10^6 = 1000000
    cin>>n;
    for (int i = 0; i < n; ++i)
    {
        cin>>x2;
        if (x2 >= x1)
            a[m].push_back(x2);
        else a[++m].push_back(x2);
        x1 = x2;
    }
}

int MaxRuns()
{
    int imax = 0;
    for (int i = 1; i <= m; ++i)
    {
        if (a[i].size() > a[imax].size())
            imax = i;
```

haitpt

```
    }
    return imax;
}

void WriteANS(int id)
{
    for (int i = 0; i < a[id].size(); ++i)
        cout<<a[id][i]<<" ";
}

int main()
{
    freopen("input.txt", "r", stdin);
    freopen("output.txt", "w", stdout);
    ReadData();
    int id = MaxRuns();
    WriteANS(id);
    return 0;
}
```

haipt

BÀI TẬP CHƯƠNG 2

Lưu ý: *các bài làm dưới đây được đặt tên theo định dạng* ***MANGXX.cpp****, trong đó* ***XX*** *là số hiệu bài làm. Ví dụ: MANG01.cpp, MANG02.cpp, …*

1) Cho dãy số nguyên $a_1, a_2, \dots, a_n$ và số nguyên $x(|x| \le 10^6)$. Hãy đếm số lượng phần tử của dãy là ước của x.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng chứa 2 số nguyên dương $n, x(n \le 10^6; |x| \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản số lượng phần tử của dãy là ước của x.

Ví dụ:

INPUT	OUTPUT
7 20 5 3 -4 2 6 9 10	4

2) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Tìm độ lệch nhỏ nhất giữa các phần tử nằm cạnh nhau.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n(n \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản độ lệch nhỏ nhất tìm được.

Ví dụ:

INPUT	OUTPUT
7 5 3 -4 2 6 9 10	1

3) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Hãy đếm số lượng phần tử của dãy là số nguyên tố.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n(n \le 10^4)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản một số nguyên là kết quả bài toán.

Ví dụ:

INPUT	OUTPUT
7 5 3 -4 2 6 9 10	3

4) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Tìm giá trị phân biệt nhỏ nhất và nhỏ thứ nhì dãy.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n (n \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n (1 \le a_i \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản giá trị phần tử nhỏ nhất và nhỏ nhì dãy trên cùng một dòng. Nếu không tồn tại kết quả nào thì ghi -1 tại vị trí của kết quả tương ứng.

Ví dụ:

INPUT	OUTPUT
7 5 2 2 7 10 8 2	2 5

5) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Tìm vị trí của phần tử nhỏ nhất và nhỏ thứ nhì dãy (hoặc nhỏ nhất đồng hạng) của dãy.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n (n \le 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n (|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản vị trí của 2 phần tử cần tìm tương ứng. Nếu có nhiều kết quả thì ghi các vị trí trái nhất.

Ví dụ:

INPUT	OUTPUT
7 5 2 2 7 10 8 2	2 3

6) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Tìm vị trí của số nguyên tố lớn nhất dãy. Nếu dãy không tồn tại số nguyên tố thì ghi -1. Nếu có nhiều số nguyên tố lớn nhất thì in ra vị trí nhỏ nhất.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n (n \le 10^4)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n (|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản vị trí của số nguyên tố lớn nhất dãy hoặc -1.

Ví dụ:

INPUT	OUTPUT
7 5 2 2 7 10 8 2	4

haipt

7) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Một đường chạy có độ dài $k(k \geq 1)$ là một dãy con dài nhất gồm các phần tử liên tiếp thỏa $a_i \leq a_{i+1} \leq \cdots \leq a_{i+k-1}$. Hãy xuất ra tất cả đường chạy của dãy, mỗi đường chạy trên một dòng.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n(n \leq 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \leq 10^6)$.

Kết quả: Ghi ra tập tin văn bản gồm nhiều dòng, mỗi dòng là một đường chạy tìm được.

Ví dụ:

INPUT	OUTPUT
7 5 2 2 7 10 8 12	5 2 2 7 10 8 12

8) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Một đường chạy có độ dài $k(k \geq 1)$ là một dãy con dài nhất gồm các phần tử liên tiếp thỏa $a_i \leq a_{i+1} \leq \cdots \leq a_{i+k-1}$. Hãy tìm đường chạy dài nhất.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n(n \leq 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \leq 10^6)$.

Kết quả: Ghi ra tập tin văn bản một số nguyên là độ dài đường chạy dài nhất.

Ví dụ:

INPUT	OUTPUT
7 5 2 2 7 10 8 12	4

9) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Hãy tìm một dãy con dài nhất gồm các phần tử liên tiếp là số chính phương. Số chính phương là số nguyên dương bằng bình phương của một số nguyên dương.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n(n \leq 10^6)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n(|a_i| \leq 10^6)$.

Kết quả: Ghi ra tập tin văn bản một số nguyên là độ dài của dãy con tìm được.

Ví dụ:

INPUT	OUTPUT
7 5 2 25 9 1 28 16	3

10) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Một phép dịch trái sẽ dời phần tử nằm ở vị trí i sang vị trí $i-1 (2 \le i \le n)$ và phần tử nằm ở vị trí đầu (vị trí 1) sẽ được dời đến cuối dãy (vị trí n). Ví dụ dãy $1, 2, 3, 4, 5$ sau 1 phép dịch trái sẽ trở thành $2, 3, 4, 5, 1$; nếu dịch trái 1 lần nữa sẽ trở thành $3, 4, 5, 1, 2$. Hãy cho biết dãy kết quả sau khi thực hiện m phép dịch trái đối với dãy $a_1, a_2, \dots, a_n$.

Yêu cầu: Cho dãy số nguyên $a_1, a_2, \dots, a_n$ và số nguyên m. Hãy cho biết trạng thái của dãy khi thực hiện m phép dịch trái.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu tiên chứa 2 số nguyên $n, m (1 \le n \le 10^6; 0 \le m \le 10^9)$.
- Dòng thứ hai chứa dãy gồm n số nguyên $a_1, a_2, \dots, a_n (|a_i| \le 10^6)$

Kết quả: Ghi ra tập tin văn bản dãy kết quả trên một dòng.

Ví dụ:

INPUT	**OUTPUT**
7 5 **5 2 25 9 1 28 16**	**28 16 5 2 25 9 1**

11) Cho dãy số nguyên $a_1, a_2, \dots, a_n$. Hãy chỉ ra một dãy con dài nhất gồm các phần tử nằm liên tiếp là dãy đối xứng.

Dữ liệu: Vào từ tập tin văn bản gồm 2 dòng

- Dòng đầu chứa số nguyên dương $n (n \le 100)$.
- Dòng tiếp theo chứa dãy $a_1, a_2, \dots, a_n (|a_i| \le 10^6)$.

Kết quả: Ghi ra tập tin văn bản 1 số nguyên là chiều dài của dãy con thỏa điều kiện.

Ví dụ:

INPUT	**OUTPUT**
7 **5 2 25 2 28 28 16**	**3**

haipt