



CƠ BẢN VỀ NGÔN NGỮ LẬP TRÌNH C++

6 SẮP XẾP

- Sắp xếp tăng dần trên mảng một chiều
- Sắp xếp giảm dần trên mảng một chiều
- Các bài tập vận dụng

A. Sắp xếp tăng dần

`int a[10000];`

+ Sắp xếp các phần tử: $a[0], a[1], \dots, a[n]$, ($n < 10000$) theo thứ tự tăng dần, ta thực hiện:

`sort(a, a + 1 + n);`

+ Sắp xếp các phần tử: $a[1], a[2], \dots, a[n]$ theo thứ tự tăng dần, ta thực hiện:

`sort(a + 1, a + 1 + n);`

Ví dụ: Viết chương trình thực hiện:

- Nhập n ($n < 100$) số nguyên từ bàn phím;
- Sắp xếp các số được nhập theo thứ tự tăng dần;
- Đưa dãy đã sắp xếp ra màn hình.

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int a[10000];
4  int n;
5  int main() {
6      cout<<"Nhập n: ";
7      cin>>n;
8      for(int i = 1; i <= n; ++i)  cin>>a[i];
9      sort(a+1, a+1+n);
10     cout<<"Day sau khi sap xep: "<<endl;
11     for(int i = 1; i <= n; ++i)  cout<<a[i]<<" ";
12 }
```

Hoặc

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int a[10000];
4  int n;
5  int main() {
6      cout<<"Nhập n: ";
7      cin>>n;
8      for(int i = 0; i < n; ++i)  cin>>a[i];
9      sort(a, a+n);
10     cout<<"Day sau khi sap xep: "<<endl;
11     for(int i = 0; i < n; ++i)  cout<<a[i]<<" ";
12 }
13
```



B. Sắp xếp giảm dần

int a[10000];

+ Sắp xếp các phần tử: $a[0], a[1], \dots, a[n]$, ($n < 10000$) theo **thứ tự giảm dần**, ta thực hiện:

`sort(a, a + 1 + n, greater < int > ());`

+ Sắp xếp các phần tử: $a[1], a[2], \dots, a[n]$ theo **thứ tự tăng dần**, ta thực hiện:

`sort(a + 1, a + 1 + n, greater < int > ());`

Ví dụ: Viết chương trình thực hiện:

- Nhập n ($n < 100$) số nguyên từ bàn phím;
- Sắp xếp các số được nhập theo **thứ tự giảm dần**;
- Đưa dãy đã sắp xếp ra màn hình.

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int a[10000];
4  int n;
5  int main() {
6      cout<<"Nhập n: ";
7      cin>>n;
8      for(int i = 0; i < n; ++i)  cin>>a[i];
9      sort(a, a+n, greater<int>());
10     cout<<"Day sau khi sap xep: "<<endl;
11     for(int i = 0; i < n; ++i)  cout<<a[i]<<" ";
12 }
13
```



BÀI TẬP

**1. Số bé thứ k và số lớn thứ k .**

Cho dãy số a_1, a_2, \dots, a_n ($a_i \neq a_j$ với $i \neq j$). Cho số tự nhiên k ($1 \leq k \leq n$). Tìm số bé thứ k và số lớn thứ k của dãy số.

Dữ liệu cho trong file MINMAXK.INP gồm:

- Dòng đầu ghi hai số nguyên dương n và k ($1 \leq k \leq n \leq 10^5$).
- Dòng thứ hai ghi n số nguyên đôi một khác nhau a_1, a_2, \dots, a_n ($|a_i| \leq 10^9$).

Kết quả ghi ra file MINMAXK.OUT gồm hai dòng:

- Dòng đầu ghi số bé thứ k của dãy.
- Dòng hai ghi số lớn thứ k của dãy.

Ví dụ:

MINMAXK.INP	MINMAXK.OUT
6 4	4
1 4 3 2 6 5	3

**2. (Version easy) Kiểm tra hoán vị (1) - CheckPermuE.Cpp**

Cho N số nguyên A_1, A_2, \dots, A_N . Ta nói N số nguyên này là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$ nếu: Trong dãy A_1, A_2, \dots, A_N có đúng 1 số hạng bằng 1, đúng 1 số hạng bằng 2, ..., đúng 1 số hạng bằng N .

Ví dụ: Dãy gồm 5 số hạng: $[2, 1, 3, 5, 4]$ là một hoán vị của 5 số tự nhiên $1, 2, \dots, 5$.

Dãy gồm 5 số hạng: $[2, 2, 1, 3, 5]$ không phải là một hoán vị của 5 số tự nhiên $1, 2, \dots, 5$.

Yêu cầu: Kiểm tra xem dãy gồm N số nguyên A_1, A_2, \dots, A_N có phải là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$.

Dữ liệu cho trong file CheckPermuE.Inp gồm:

- Dòng đầu ghi số nguyên dương N ($N \leq 10^3$).
- Dòng thứ 2 ghi N số nguyên A_1, A_2, \dots, A_N ($|A_i| \leq 10^9$).

Kết quả ghi ra file CheckPermuE.Out ghi ra “Yes” nếu dãy đã cho là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$, ngược lại ghi ra “No”.

Ví dụ:

CheckPermuE.Inp	CheckPermuE.Out
3 1 3 2	Yes
3 1 2 2	No

**3☀️. (Version medium) Kiểm tra hoán vị (2) - CheckPermuM.Cpp**

Cho N số nguyên A_1, A_2, \dots, A_N . Ta nói N số nguyên này là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$ nếu: Trong dãy A_1, A_2, \dots, A_N có đúng 1 số hạng bằng 1, đúng 1 số hạng bằng 2, ..., đúng 1 số hạng bằng N .

Ví dụ: Dãy gồm 5 số hạng: $[2, 1, 3, 5, 4]$ là một hoán vị của 5 số tự nhiên $1, 2, \dots, 5$.

Dãy gồm 5 số hạng: $[2, 2, 1, 3, 5]$ không phải là một hoán vị của 5 số tự nhiên $1, 2, \dots, 5$.

Yêu cầu: Kiểm tra xem dãy gồm N số nguyên A_1, A_2, \dots, A_N có phải là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$.

Dữ liệu cho trong file **CheckPermuM.Inp** gồm:

- Dòng đầu ghi số nguyên dương N ($N \leq 10^5$).
- Dòng thứ 2 ghi N số nguyên A_1, A_2, \dots, A_N ($|A_i| \leq 10^9$)

Kết quả ghi ra file **CheckPermuM.Out** ghi ra “Yes” nếu dãy đã cho là một hoán vị của N số tự nhiên $1, 2, 3, \dots, N$, ngược lại ghi ra “No”.

Ví dụ:

CheckPermuM.Inp	CheckPermuM.Out
3 1 3 2	Yes
3 1 2 2	No

**4☀️. (medium) Trò chơi SmartBowling – SmartBowling.Cpp**

Có N chai trong trò chơi SmartBowling được xếp thành một hàng ngang. Các chai được đánh số thứ tự từ 1 đến N (theo hướng từ trái sang phải). Độ vững chắc của chai thứ i ($i = 1, 2, \dots, N$) là V_i . Người chơi có 2 loại bi để chọn ném các chai. Loại bi **đỏ** và loại bi **xanh**. Khi chọn loại bi **đỏ** để ném, người chơi ném trúng chai nào thì độ vững chắc của chai đó giảm xuống 1. Cái chai sẽ bị đổ nếu độ vững chắc của cái chai đó bằng 0. Khi chọn loại bi **xanh** để ném, nếu ném trúng chai nào thì chai đó đổ ngay tức khắc.

Yêu cầu: Tìm số lần ném ít nhất để làm đổ N chai, biết rằng, mỗi lần ném, người chơi chọn 1 bi và đều ném trúng 1 chai; số lần chọn bi xanh không quá K lần.


Dữ liệu cho trong file **SmartBowling.Inp** gồm:

- Dòng đầu ghi hai số nguyên dương N và K .
- Dòng thứ 2 ghi N số nguyên dương V_1, V_2, \dots, V_N tương ứng là độ bền vững của N chai lúc bắt đầu chơi.

Kết quả ghi ra file **SmartBowling.Out** là số lần ném ít nhất để làm đổ N chai.

Ví dụ:



SmartBowling.Inp	SmartBowling.Out	Hình minh họa
5 1 1 4 2 9 3	11	

Giải thích:

- Chọn 1 bi xanh để ném trúng và làm đổ chai có độ bền vững là 9.
- Chọn 10 bi đỏ để ném 10 lần để làm đổ các chai còn lại.

Giới hạn:

- $1 \leq N, K \leq 5 \times 10^5$;
- $1 \leq V_i \leq 10^6$.



5☀. Hoán vị

Cho dãy số nguyên dương gồm n số hạng a_1, a_2, \dots, a_n và cặp chỉ số i, j ($1 \leq i \leq j \leq n$). Xét dãy con gồm các số hạng liên tiếp a_i, a_{i+1}, \dots, a_j . Nhận thấy dãy này gồm $k = j - i + 1$ số hạng.

Yêu cầu: Kiểm tra xem, dãy con a_i, a_{i+1}, \dots, a_j có phải là một hoán vị của k số $1, 2, \dots, k$ hay không?

Ví dụ: Dãy a_1, a_2, a_3, a_4, a_5 ($n = 5$) có giá trị tương ứng là: 1, 2, 1, 3, 7.

Với cặp chỉ số $(i, j) = (2, 4)$, ta có dãy: 2, 1, 3. Số số hạng của dãy bằng 3 và dãy này là một hoán vị của 3 số: 1, 2, 3.

Với cặp chỉ số $(i, j) = (1, 4)$, ta có dãy: 1, 2, 1, 3. Số số hạng của dãy bằng 4 và dãy này không là một hoán vị của 4 số: 1, 2, 3, 4.

Dữ liệu cho trong file PERMU.INP gồm:

- Dòng đầu ghi hai số nguyên dương n và k tương ứng là số các số hạng trong dãy và số cặp chỉ số.
- Dòng thứ hai ghi n số nguyên dương a_1, a_2, \dots, a_n ($a_i \leq 10^9$).
- k dòng cuối, mỗi dòng ghi cặp chỉ số i, j ($1 \leq i \leq j \leq n$).

Kết quả ghi ra file PERMU.OUT gồm k dòng, mỗi dòng ghi 1 nếu là hoán vị, ghi 0 nếu không phải là một hoán vị.

Ví dụ:



PERMU.INP	PERMU.OUT
6 4	1
1 2 3 1 4 6	0
1 3	1
5 6	1
4 4	
2 5	

Giới hạn:

- $n \leq 10^4$
- $k \leq 20$.

Một số bài tập tổng hợp



6☀️ (medium) Dịch trái – dịch phải – MoveLeftRight.Cpp

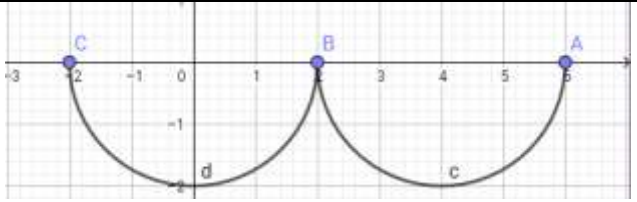
Có một con robot được đặt trên một trục số tại vị trí tọa độ X . Bạn có thể điều khiển con robot này dịch sang bên trái hoặc sang bên phải một khoảng cách D đơn vị. Tức là, nếu hiện tại robot ở vị trí có tọa độ U thì, nếu dịch trái, robot sẽ chuyển đến vị trí có tọa độ $U - D$, nếu dịch phải, robot sẽ dịch chuyển đến vị trí có tọa độ $U + D$.

Yêu cầu: Bạn được lệnh cho robot dịch chuyển đúng K lần, hãy tìm cách dịch chuyển robot đến gần gốc tọa độ nhất, tức là đến vị trí tọa độ Y sao cho $|Y|$ đạt giá trị nhỏ nhất.

Dữ liệu cho trong file **MoveLeftRight.Inp** gồm 3 số nguyên X, K, D .

Kết quả ghi ra file **MoveLeftRight.Out** là giá trị $|Y|$ nhỏ nhất với Y là vị trí cuối cùng của robot sau khi điều khiển đúng K lần dịch chuyển.

Ví dụ:

MoveLeftRight.Inp	MoveLeftRight.Out	Hình minh họa
6 2 4	2	 <p>Ban đầu robot ở vị trí A tọa độ 6. Lần dịch chuyển 1: Sang trái đến vị trí B tọa độ 2. Lần dịch chuyển 2: Sang trái đến vị trí C tọa độ -2, ($Y = -2$).</p>

Giới hạn:

- $-10^{15} \leq X \leq 10^{15}$;
- $1 \leq K \leq 10^{15}$;
- $1 \leq D \leq 10^{15}$;



7 (Medium) Tây du kí - NgoKhong.Cpp

Tác phẩm “Tây du kí” của Ngô Thừa Ân là một tác phẩm văn học kinh điển của Trung Hoa. Tác phẩm này đã được dựng thành phim mà nhiều bạn trẻ rất là yêu thích.




Tôm cũng vậy, rất thích nhân vật Tôn Ngộ Không với 72 phép biến hóa thần thông quảng đại. Trong đó có phép nhân đôi “*Tôn Ngộ Không*”. Với ý tưởng của phép thuật này, Tôm đã thiết kế ra một GameTNK với nội dung như sau: Ban đầu chỉ có một Tôn Ngộ Không có chiều cao là X . Khi người chơi chọn vào một Tôn Ngộ Không có chiều cao H và niệm thần chú thì Tôn Ngộ Không đó sẽ biến thành 2 Tôn Ngộ Không có chiều cao là $\lfloor \frac{H}{2} \rfloor$ (tức là phần nguyên của $H/2$). Tôn Ngộ Không sẽ biến mất nếu chiều cao của nó bằng 0.

Yêu cầu: Cho chiều cao X . Tính số lần cần niệm thầy chú để tất cả các Tôn Ngộ Không đều biến mất.

Dữ liệu cho trong file **NgoKhong.Inp** gồm một số nguyên dương X .

Kết quả ghi ra file **NgoKhong.Out** là số các lần niệm thần chú để tất cả các Tôn Ngộ Không biến mất.

Ví dụ:

NgoKhong.Inp	NgoKhong.Out	Giải thích
3	3	 Ban đầu chiều cao 3. Niệm thần chú lần 1: Được 2 Tôn Ngộ Không chiều cao $\lfloor 3/2 \rfloor = 1$.   Chọn mỗi Tôn Ngộ Không và niệm thần chú 1 lần cho 1 Tôn Ngộ Không, các Tôn Ngộ Không đều biến mất.

Giới hạn:

- Sub1: $X \leq 100$;
- Sub2: $X \leq 10^{12}$;

**8☀️ (Very Hard) Dãy con dư K – RemainK.Cpp**

Cho dãy số nguyên dương A_1, A_2, \dots, A_N và số nguyên dương K . Dãy con gồm các số hạng ở vị trí liên tiếp A_i, A_{i+1}, \dots, A_j là một dãy con **RemainK** nếu thỏa mãn:

$$(A_i + A_{i+1} + \dots + A_j) \% K = (j - i + 1).$$

Yêu cầu: Tính số cặp (i, j) để A_i, A_{i+1}, \dots, A_j là một dãy con **RemainK**.

Dữ liệu cho trong file **RemainK.Inp** gồm:

- Dòng đầu ghi số nguyên dương N và K .
- Dòng 2 ghi N số nguyên dương A_1, A_2, \dots, A_N .

Kết quả ghi ra file **RemainK.Out** là số cặp (i, j) để A_i, A_{i+1}, \dots, A_j là một dãy con **RemainK**.

Ví dụ:

RemainK.Inp	RemainK.Out	Giải thích
5 4 1 4 2 3 5	4	Ta có các dãy RemainK: [1] ứng với cặp chỉ số (1, 1). [4, 2] ứng với cặp chỉ số (2, 3). [1, 4, 2] ứng với cặp chỉ số (1, 3). [5] ứng với cặp (5, 5).

Giới hạn:

- $1 \leq N \leq 2 \times 10^5$;
- $1 \leq K \leq 10^6$;
- $1 \leq A_i \leq 10^9$.