# Môn thi: Tin học

Thời gian làm bài:***180*** *phút, không kể thời gian giao đề*

Ngày thi: 11/10/2021

#### Tổng quan bài thi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên bài** | **File nguồn** | **File Input** | **File Output** | **Thời gian** |
| Thuật toán sắp xếp | tripsort.\* | tripsort.in | tripsort.out | 1 giây |
| Số phi đối xứng | numbers.\* | numbers.out | numbers.out | 1 giây |
| Hệ thống định vị GPS | gps.\* | gps.in | gps.out | 1 giây |

**Dấu \* là Pas hoặc Cpp tương ứng với ngôn ngữ lập trình Pascal hoặc C++**

**Hãy lập trình giải các bài toán sau:**

Bài 1. (6 điểm) Thuật toán sắp xếp

Nam vừa mới làm quen với môn Cấu trúc dữ liệu và Giải thuật, và một trong những kiến thức cơ bản đầu tiên mà Nam được học là thuật toán sắp xếp quen thuộc: thuật toán sắp xếp nổi bọt (bubble sort).

Thao tác cơ bản trong thuật toán sắp xếp nổi bọt là so sánh hai phần tử liền kề, nếu số đứng trước lớn hơn số đứng sau thì hai phần tử đó sẽ được tráo đổi vị trí. Sau khi hiểu rõ thuật toán nổi bọt, Nam liền đặt ngay câu hỏi trong đầu: tại sao lại phải xét hai phần tử liền kề; nếu xét ba phần tử liến tiếp có được không? Từ đó, Nam "phát minh" được một thuật toán "sắp xếp" một mảng số nguyên A gồm N phần tử như sau:

done = false

while (done == false): done = true

for (i = 1; i <= N - 2; i++): if (A[i] > A[i + 2]):

done = false

đảo đoạn A[i], A[i + 1], A[i + 2]

trong đó A[i] là phần tử thứ i của mảng A, các phần tử của A được đánh chỉ số từ 1. Ví dụ, với A = [5, 6, 6, 4, 3], thuật toán của Nam sẽ lần lượt thực hiện:

* Ở vòng while đầu tiên
  + Xét 5, 6, 6; không có gì xảy ra, A = [5, 6, 6, 4, 3]
  + Xét 6, 6, 4; thấy 6 > 4, đảo ngược 3 số này, A = [5, 4, 6, 6, 3]
  + Xét 6, 6, 3; thấy 6 > 3, đảo ngược 3 số này, A = [5, 4, 3, 6, 6]
* Ở vòng while thứ hai:
  + Xét 5, 4, 3; thấy 5 > 3, đảo ngược 3 số này, A = [3, 4, 5, 6, 6]
  + Xét 4, 5, 6; không có gì xảy ra, A = [3, 4, 5, 6, 6]
  + Xét 5, 6, 6; không có gì xảy ra, A = [3, 4, 5, 6, 6]
* Ở vòng while thứ ba: A giữ nguyên, done = true, thuật toán kết thúc.

**Yêu cầu:** Cho mảng A. Bạn hãy giúp Nam kiểm tra xem thuật toán của Nam có sắp xếp được mảng A tăng dần hay không. Nếu không, hãy xác định chỉ số i đầu tiên mà A[i] > A[i + 1] (trên mảng A sau khi đã được "sắp xếp" bởi thuật toán của Nam).

**Input:** đọc từ file **tripsort.in**

Dòng đầu tiên chứa số nguyên dương T (T <= 10) là số lượng test. T nhóm dòng sau, mỗi nhóm dòng mô tả một test. Mỗi test có định dạng:

* Dòng đầu tiên chứa số nguyên dương N.
* Dòng tiếp theo chứa N số nguyên không âm A[1], A[2], ..., A[N] (0 <= A[i] <= 10^9).

**Output:** ghi ra file **tripsort.out**

Với mỗi test, theo đúng thứ tự được cho trong input, in ra trên một dòng: "OK" nếu thuật toán của Nam sắp xếp được mảng A; nếu không in ra chỉ số i đầu tiên mà A[i] > A[i + 1] sau khi đã áp dụng thuật toán của Nam trên mảng A.

# Subtask:

*Subtask 1 (50%):* 3 <= N <= 100

*Subtask 2 (50%):* 3 <= N <= 100000

# Ví dụ

|  |  |
| --- | --- |
| **tripsort.in** | **tripsort.out** |
| 2 | OK |
| 5 | 2 |
| 5 6 8 4 3 |  |
| 3 |  |
| 8 9 7 |  |

Bài 2. (7 điểm) Số phi đối xứng

Một xâu được gọi là đối xứng nếu đọc nó từ trái qua phải cũng giống với đọc từ phải qua trái. Một số nguyên được gọi là phi đối xứng nếu nó không chứa một xâu con liên tiếp nào là xâu đối xứng với độ dài lớn hơn 1. Ví dụ, 16276 là một số phi đối xứng, nhưng 17276 thì không phải bởi nó chứa xâu đối xứng 727.

**Yêu cầu:** Cho hai số nguyên không âm A và B (A <= B). Hãy đếm và in ra số lượng số phi đối xứng không nhỏ hơn A và không lớn hơn B.

**Input:** đọc từ file **numbers.in**

* Gồm một dòng chứa hai số nguyên A, B (0 <= A <= B <= 10^18).

**Output:** ghi ra file **numbers.out**

In ra đáp án trên một dòng.

# Subtask:

*Subtask 1 (25%):* B - A <= 100000

*Subtask 2 (75%):* không có ràng buộc gì thêm

# Ví dụ

|  |  |
| --- | --- |
| **numbers.in** | **numbers.out** |
| 123 321 | 153 |
| 123456789 987654321 | 167386971 |

Bài 3. (7 điểm) Hệ thống định vị GPS

Xe ô tô của Nam rất hiện đại và tiện nghi, nó có những hai hệ thống định vị GPS, đề phòng khi một cái hư thì cái còn lại vẫn hoạt động. Thật không may cho Nam, vào một ngày "đẹp trời", cả hai hệ thống định vị bị hỏng cùng một lúc!

Thành phố nơi Nam sống gồm có N địa điểm được đánh chỉ số từ 1 tới N. Có M con đường một chiều nối các địa điểm với nhau, con đường thứ i (1 <= i <= M) nối từ địa điểm A\_i tới địa điểm B\_i. Hai hệ thống định vị của Nam đã bị mất dữ liệu đúng về thời gian cần để đi qua các con đường. Với con đường thứ i (1 <= i <= M), hệ thống định vị thứ nhất, thứ hai cho rằng thời gian đi hết con đường đó lần lượt là P\_i và Q\_i.

Đã đến giờ đi làm, Nam cần phải lái xe di chuyển từ nhà mình nằm ở địa điểm 1 đi đến công ty nằm ở địa điểm N. Mỗi khi Nam quyết định đi qua một con đường (ví dụ từ X tới Y), mỗi hệ thống định vị sẽ phát âm thanh cảnh báo nếu như con đường này không thuộc một đường đi với tổng thời gian di chuyển ngắn nhất (theo quan điểm của hệ thống định vị đó) từ X tới N. Những âm thanh cảnh báo này khiến Nam khó chịu bởi chúng rất chói toi.

**Yêu cầu:** Bạn hãy giúp Nam xác định đường đi từ nhà đến công ty sao cho tổng số lượng âm thanh cảnh báo được phát ra là nhỏ nhất có thể.

**Input:** đọc từ file **gps.in**

* Dòng đầu tiên chứa hai số nguyên N, M (2 <= N <= 10000, 1 <= M <= 50000)
* M dòng sau, dòng thứ i (1 <= i <= M) chứa bốn số A\_i, B\_i, P\_i, Q\_i (1 <= A\_i <= N, 1 <= B\_i <= N, 1 <= P\_i, Q\_i <= 100000).
* Dữ liệu đảm bảo luôn tồn tại đường đi từ nhà Nam đến công ty.

**Output:** ghi ra file **gps.out**

In ra số lượng âm thanh cảnh báo ít nhất được phát ra.

**Ví dụ**

|  |  |  |
| --- | --- | --- |
| **gps.in** | **gps.out** | **Giải thích** |
| 5 7 | 1 | Nam nên đi theo đường đi: 1->2->4->5. |
| 3 4 7 1 |  | Hệ thống định vị thứ nhất sẽ phát âm thanh |
| 1 3 2 20 |  | cảnh báo khi Nam đi con đường 1->2. |
| 1 4 17 18 |  | Trên đoạn đường 2->4->5 còn lại cả hai hệ |
| 4 5 25 3 |  | thống không cảnh báo gì bởi đây là đường đi |
| 1 2 10 1 |  | ngắn nhất từ 2 đến 5 đối với cả hai hệ thống. |
| 3 5 4 14 |  |  |
| 2 4 6 5 |  |  |