

Russia contest 8

Created: 4/30/2025 15:02

Updated: 4/30/2025 15:06

Exported: 4/30/2025 15:05


Bài A. Đạo văn mã nguồn

 **Tên file vào:** chuẩn đầu vào

 **Tên file ra:** chuẩn đầu ra

 **Giới hạn thời gian:** 1 giây


 **Giới hạn bộ nhớ:** 256 MB

 Hai nhân viên của một công ty nổi tiếng, **Alice** và **Bob**, đã đưa ra hai đoạn mã để sửa một lỗi nghiêm trọng.

Alice nghi ngờ rằng Bob đã **đạo văn** mã của cô, bằng cách **thêm vào các ký tự không làm thay đổi chức năng** để khiến đoạn mã trông khác biệt hơn.

Ngôn ngữ lập trình của công ty là một ngôn ngữ **kỳ lạ giống Malbolge**, nên đoạn mã là một **xâu ký tự chữ cái thường** (`a` đến `z`).

- Mã của Alice là chuỗi `t`,
- Mã của Bob là chuỗi `s`.

 **Bàn phím của Bob bị hỏng**, nên anh ta chỉ có thể **gõ 2 ký tự mỗi lần**, tức là chỉ có thể **chèn hai ký tự bất kỳ (có thể giống hoặc khác nhau)** vào bất kỳ vị trí nào trong chuỗi.

 Bây giờ, để kiểm tra Bob có đạo văn hay không, ta cần xác định:

Liệu có thể từ chuỗi `s`, bằng cách xóa đi một số lần các cặp ký tự đứng liền nhau, thu được chuỗi `t` hay không?

Dữ liệu vào

- Dòng 1: Chuỗi `s` ($1 \leq |s| \leq 2 \cdot 10^5$) – chuỗi của Bob.
- Dòng 2: Chuỗi `t` ($1 \leq |t| \leq |s|$) – chuỗi của Alice.

Dữ liệu ra

- In `YES` nếu có thể thu được chuỗi `t` từ chuỗi `s` bằng cách xóa các **cặp ký tự đứng liền nhau** bao nhiêu lần tùy ý.
- In `NO` nếu không thể.

 Các cách viết như `yEs`, `YES`, `Yes`, `yes` đều được chấp nhận.

Ví dụ

Input	Output
<code>sobaka` baka`</code>	<code>YES`</code>
<code>sobabka` baka`</code>	<code>NO`</code>
<code>abacaba` aca`</code>	<code>YES`</code>

Tóm tắt đề bài

- Từ chuỗi `s`, bạn được phép **xóa một số lần bất kỳ** các **cặp ký tự liền nhau**.

- Hỏi bạn có thể **biến chuỗi** `s` **thành chuỗi** `t` được hay không?
- Trả lời `YES` hoặc `NO`.

Chấm điểm theo từng Subtask

Subtask	Số điểm	Giới hạn bổ sung	Cần hoàn thành các subtasks	Ghi chú
1	20	`	s	$\leq 10`$
2	23	`t` chỉ chứa ký tự ``a``	Subtask 1	Lỗi đầu tiên
3	27	`	s	$\leq 1000` và `$
4	30	Không giới hạn thêm	Subtask 1-3	Lỗi đầu tiên

Bạn muốn mình viết luôn lời giải hay ý tưởng không?

Bài B. SpamGPT-4

 **Tên file vào:** chuẩn đầu vào

 **Tên file ra:** chuẩn đầu ra

 **Giới hạn thời gian:** 1 giây

 **Giới hạn bộ nhớ:** 256 MB

Mô tả bài toán

Để kiểm tra độ bền của hai con bot spam hàng đầu của công ty **“LinkedOut”**, nhóm phát triển quyết định cho **chúng tự nhắn tin qua lại**, và theo dõi xem chúng có thể hoạt động ổn định bao lâu.

- Cả **hai bot bắt đầu tại giây 0** và **gửi một tin nhắn cho nhau**.
- Sau đó:
 - Bot 1 gửi tin nhắn mới **cứ mỗi a giây** (tức là vào các thời điểm $0, a, 2a, \dots$)
 - Bot 2 gửi tin nhắn mới **cứ mỗi b giây** (tức là vào các thời điểm $0, b, 2b, \dots$)

 Ngoài ra, **mỗi khi nhận được một tin nhắn**, bot sẽ **gửi lại phản hồi sau đúng 1 giây**.

 Tin nhắn **gửi ngay lập tức**, đến nơi **ngay lập tức**.

Tức là nếu bot 1 gửi tin lúc thời điểm t , thì:

- Đến $t + 1$ nó **nhận được phản hồi** từ bot 2
- Đến $t + 2$, nó **phản hồi lại** tin nhắn đó

 Các bot có thể **xử lý đồng thời nhiều tác vụ** — tức là có thể:

- Vừa gửi tin mới
- Vừa gửi phản hồi cho nhiều tin khác nhau
- Cùng một thời điểm vẫn làm được nhiều việc

Dữ liệu vào

Một dòng duy nhất gồm 3 số nguyên:

$a \ b \ T$

Trong đó:

- a : chu kỳ gửi tin mới của bot 1
- b : chu kỳ gửi tin mới của bot 2
- T : thời gian hoạt động của hệ thống ($1 \leq a, b, T \leq 10^9$)

Dữ liệu ra

In ra 2 số nguyên:

- Số lượng **tin nhắn mà bot 1 đã gửi** tính đến giây T (bao gồm tin gửi lúc T nếu có)
- Số lượng **tin nhắn mà bot 2 đã gửi** tính đến giây T

 Bao gồm cả tin nhắn mới và các phản hồi được gửi đi.

Ví dụ

Input	Output
<code>1 2 5</code>	<code>18 15</code>
<code>4 3 6</code>	<code>11 11</code>
<code>17 10 193</code>	<code>1596 1590</code>

Tóm tắt đề bài

- Tại thời điểm 0, cả 2 bot gửi **tin đầu tiên** cho nhau.
- Sau đó, mỗi bot sẽ:

- Gửi **tin mới** theo chu kỳ ``a`` (hoặc ``b``)
- **Trả lời lại** tất cả các tin đã nhận **sau đúng 1 giây**

👉 Yêu cầu: **đến thời điểm T (bao gồm cả giây T), mỗi bot đã gửi bao nhiêu tin nhắn tổng cộng?**

🧩 Chấm điểm theo từng Subtask

Subtask	Số điểm	Giới hạn bổ sung	Cần hoàn thành các subtasks	Ghi chú
1	15	<code>`a, b, T ≤ 1000`</code>	-	Full test nhỏ
2	18	<code>`T ≤ 50,000`</code>	Subtask 1	Lỗi đầu tiên
3	21	<code>`a = b = 1`</code>	Subtask 1	Lỗi đầu tiên
4	21	<code>`a ≤ T ≤ b < 2a`</code>	Subtask 1	Lỗi đầu tiên
5	25	Không giới hạn	Subtask 1-4	Lỗi đầu tiên

Bạn muốn mình giải thích luôn thuật toán hay viết code không?

📄 Bài C. Có `n` cái ghế...

📁 **Tên file vào:** chuẩn đầu vào

📁 **Tên file ra:** chuẩn đầu ra

🕒 **Giới hạn thời gian:** 1 giây

💾 **Giới hạn bộ nhớ:** 256 MB

🪑 Mô tả bài toán

🧑 Vlad cuối cùng cũng được lên làm **team lead**, nhưng vì quá bận rộn nên anh phải **ngủ lại văn phòng**.

Tuy nhiên, văn phòng không đủ điều kiện tốt nên Vlad buộc phải **ngủ trên các ghế văn phòng**.

- Có n **cái ghế**. Ghế thứ i có:
 - Chiều **cao** là h_i
 - Chiều **rộng** là w_i

📏 Vlad có chiều cao bằng H , nên anh muốn chọn **một dãy các ghế xếp thành hàng** sao cho **tổng chiều rộng $\geq H$** để có thể nằm lên được.

😓 Tuy nhiên, nếu các ghế có độ **chênh lệch độ cao** lớn thì nằm rất **khó chịu**.
→ Ta định nghĩa **độ khó chịu** của dãy ghế là **giá trị lớn nhất của hiệu tuyệt đối giữa hai ghế liên kế** trong dãy.

→ Nếu chỉ chọn **1 ghế**, thì độ khó chịu là 0 .

📖 Dữ liệu vào

- Dòng 1: hai số nguyên n và H — số ghế và chiều cao của Vlad ($1 \leq n \leq 2 \cdot 10^5$; $1 \leq H \leq 10^9$)
- Dòng 2: n số nguyên h_1, h_2, \dots, h_n — chiều cao của từng ghế
- Dòng 3: n số nguyên w_1, w_2, \dots, w_n — chiều rộng của từng ghế


Đảm bảo rằng: tổng tất cả $w_i \geq H$ (luôn có cách chọn đủ rộng để nằm được)

📖 Dữ liệu ra

- In ra **độ khó chịu nhỏ nhất** có thể đạt được, trong số tất cả các dãy ghế có **tổng chiều rộng $\geq H$**

Tóm tắt đề bài

- Có n cái ghế, mỗi cái có **chiều cao** và **chiều rộng**
- Bạn cần chọn 1 dãy ghế sao cho:
 - **Tổng chiều rộng $\geq H$**
 - **Độ khó chịu** là nhỏ nhất (hiệu lớn nhất giữa hai ghế liên kế trong dãy là nhỏ nhất có thể)

 **Lưu ý:** Thứ tự sắp xếp các ghế trong dãy là **quan trọng** – có thể ảnh hưởng tới độ khó chịu.

Ví dụ

Ví dụ 1:

Input :

4 7

1 4 1 2

1 4 2 3

Output :

2

→ Chọn ghế số 2 và 4, tổng rộng là $4 + 3 = 7$. Nếu xếp 4-2 thì hiệu độ cao là $|4 - 2| = 2$.

Ví dụ 2:

Input:

5 6

1 3 5 4 2

5 4 3 2 1

Output:

1

→ Chọn ghế số 2, 4, 3 (tổng rộng: $4+2+3 = 9$). Độ khó chịu: $\max(|3-4|, |4-5|) = \max(1,1) = 1$

Chấm điểm theo từng Subtask

Subtask	Điểm	Giới hạn bổ sung	Yêu cầu các subtask trước?	Ghi chú
1	10	$n \leq 100$	-	Full test nhỏ
2	20	$n \leq 1000$	Subtask 1	Lỗi đầu tiên
3	15	$w_i = 1$, $n \leq 10^5$	Subtask 1	Lỗi đầu tiên
4	19	$h_i \leq 30$, $n \leq 10^5$	Subtask 1	Lỗi đầu tiên
5	36	Không giới hạn	Subtask 1-4	Lỗi đầu tiên

Bạn muốn mình tóm tắt ý tưởng giải hay viết code cho bài này luôn?

Bài D. Chuyển giao trách nhiệm

 **Tên file vào:** chuẩn đầu vào

 **Tên file ra:** chuẩn đầu ra

 **Giới hạn thời gian:** 2.5 giây

 **Giới hạn bộ nhớ:** 256 MB



Mô tả bài toán

Trong quá trình chuẩn bị kỳ thi, **mỗi người ra đề** sẽ phụ trách một số phần của các bài toán.

Có:

- n **lập trình viên** (người ra đề)
- n **bài toán**

Bài toán thứ i gồm:

- c_i phần tử (module con)
- mỗi phần tử có **độ phức tạp** w_i



Luật phân công

- Mỗi người thứ i được **giao chính xác** c_i **phần tử**, nhưng các phần tử này có thể đến từ **nhiều bài toán khác nhau**.

Phân phối diễn ra theo thứ tự:

1. Người thứ nhất nhận c_1 phần tử, người thứ hai nhận c_2 phần tử, v.v...
2. Các phần tử được lấy **theo vòng tròn từ các bài toán còn lại**: bài 1 \rightarrow bài 2 \rightarrow ... \rightarrow bài n \rightarrow quay lại bài 1...
 - Nếu bài nào hết phần tử thì **bỏ qua**.
3. Người mới bắt đầu phân phối **từ bài tiếp theo sau bài cuối cùng mà người trước đó dừng lại** (cũng theo vòng tròn).



Nhiệm vụ

Tính tổng độ phức tạp **mà mỗi người được giao**, sau khi việc phân phối hoàn tất.

Dữ liệu vào

- Dòng đầu: Số nguyên n ($1 \leq n \leq 500.000$)
- Sau đó là n dòng, dòng thứ i gồm 2 số nguyên:
 - c_i : số phần tử mà người thứ i sẽ được giao
 - w_i : độ phức tạp của mỗi phần tử thuộc bài toán i

Dữ liệu ra

- In ra n số nguyên — tổng độ phức tạp các phần tử mà từng người nhận được.
-

Tóm tắt đề bài

- Mỗi bài toán có c_i phần tử, mỗi phần tử có độ khó w_i
- Mỗi người được giao đúng c_i phần tử, **không phải toàn bộ bài i** , mà có thể đến từ nhiều bài
- Phân phối phần tử theo **vòng tròn qua các bài toán**, bỏ qua bài nào đã hết
- Mỗi người bắt đầu từ nơi người trước đó dừng

→ Cần tính tổng độ khó mỗi người nhận được.

Ví dụ

Ví dụ 1:

Input :

```
3
3 1
2 10
4 100
```

Output :

```
111 11 301
```

Chi tiết:

- Người 1 nhận 3 phần tử: từ bài 1, 2, 3 → mỗi phần tử lần lượt độ khó 1, 10, 100 → tổng 111
- Người 2 nhận 2 phần tử tiếp theo: từ bài 1 và 2 → mỗi phần tử độ khó 1, 10 → tổng 11
- Người 3 nhận 4 phần tử cuối: từ bài 1, 3, 1, 3 → mỗi phần tử độ khó 1, 100, 1, 100 → tổng 301

Chấm điểm theo từng Subtask

Subtask	Số điểm	Giới hạn bổ sung	Cần hoàn thành	Ghi chú
1	10	$n \leq 1000$, $c_i \leq 1000$	-	Full test nhỏ
2	10	$n = 2$	-	Lỗi đầu tiên
3	13	$c_i \leq 2$ với mọi i	-	Lỗi đầu tiên
4	17	$n \leq 1000$	1, 2	Lỗi đầu tiên

Subtask	Số điểm	Giới hạn bổ sung	Cần hoàn thành	Ghi chú
5	29	<code>`n ≤ 200.000`</code>	4	Lỗi đầu tiên
6	21	Không giới hạn	1-5	Lỗi đầu tiên

Bạn muốn mình tiếp tục với ý tưởng giải bài này hay viết code luôn?

Bài E. Trình thực thi nhanh

 **Tên file vào:** chuẩn đầu vào

 **Tên file ra:** chuẩn đầu ra

 **Giới hạn thời gian:** 3 giây

 **Giới hạn bộ nhớ:** 256 MB

Mô tả bài toán

Sinh viên năm nhất Misha đang học một ngôn ngữ lập trình mới.

Ngôn ngữ này chỉ làm việc trên **mảng số nguyên không âm** có độ dài ``n``.

Misha tạo ra:

- Mảng ``a``
- Sao chép ``a`` sang ``b`` ban đầu (``b = a``)

Sau đó, Misha lập trình 4 thao tác sau:

Các hàm có thể thực hiện:

1. ``shift``: dịch vòng mảng ``a`` sang trái ``d`` đơn vị:

```
a ← [a[d], ..., a[n-1], a[0], ..., a[d-1]]
```

2. ``xor``:

```
b[i] ← a[i] ⊕ b[i]    (xor từng phần tử)
```

3. ``and``:

```
b[i] ← a[i] & b[i]    (and từng phần tử)
```

4. ``or``:

```
b[i] ← a[i] | b[i]    (or từng phần tử)
```

Mô tả chương trình Misha viết

Misha có:

- Một **chuỗi thao tác gồm** ``m`` **phép toán** (chỉ gồm: ``xor``, ``and``, ``or``)
- Một số nguyên ``p`` – số lần lặp lại chuỗi thao tác

Quy tắc thực thi:

- Trong mỗi vòng lặp:
 - Với mỗi phép toán trong chuỗi:
 - Gọi ``shift``, rồi gọi phép toán tương ứng
- Thực hiện điều đó ``p`` **lần**

Ví dụ với:

```
m = 3, chuỗi = [or, xor, and], p = 2
```

→ chương trình hoạt động như sau:

```
repeat 2 times {  
    shift, or  
    shift, xor  
    shift, and  
}
```

Dữ liệu vào

- Dòng 1: 4 số nguyên ``n m d p``
 - ``n``: độ dài mảng
 - ``m``: số thao tác trong chuỗi
 - ``d``: số lượng phần tử dịch trái khi shift
 - ``p``: số lần lặp chuỗi thao tác
- Dòng 2: ``n`` số nguyên ``a0, a1, ..., an-1`` — mảng ``a``, cũng chính là mảng ``b`` ban đầu
- Dòng 3: ``m`` chuỗi – mỗi chuỗi là 1 trong: ``xor``, ``and``, ``or``

Dữ liệu ra

- In ra mảng ``b`` cuối cùng sau khi thực thi toàn bộ chương trình

Ví dụ

Ví dụ 1:

Input:

5 3 2 2
1 0 1 0 1
or and or

Output:

1 0 1 1 1

Ví dụ 2:

Input:

6 3 2 3
1 2 3 4 5 6
xor and or

Output:

1 6 3 6 5 6

Tóm tắt đề bài

- Bắt đầu với mảng `a` và `b = a`
- Lặp lại `p` lần:
 - Với mỗi thao tác `op` trong danh sách thao tác:
 - Gọi `shift` trên `a`
 - Gọi `op` (xor, and, or) giữa `a` và `b`
- In ra `b` cuối cùng sau `p * m` lần thao tác

Chấm điểm theo từng Subtask

Subtask	Số điểm	Giới hạn bổ sung	Phụ thuộc	Ghi chú
1	11	$n \leq 1000$, $p \leq 100$	-	Full test nhỏ
2	14	$m = 1$	-	Lỗi đầu tiên
3	17	Các thao tác đều giống nhau	2	Lỗi đầu tiên
4	15	Không có phép xor nào	-	Lỗi đầu tiên
5	16	$a_i \leq 1$ với mọi i	-	Lỗi đầu tiên
6	27	Không giới hạn	1-5	Lỗi đầu tiên

Bạn muốn mình trình bày hướng giải hay viết luôn lời giải cho bài này?

F Bài F. Tính nhẩm

 **Tên file vào:** chuẩn đầu vào

 **Tên file ra:** chuẩn đầu ra

 **Giới hạn thời gian:** 4 giây

 **Giới hạn bộ nhớ:** 1024 MB

Mô tả bài toán

Cậu bé **Vova** (3 tuổi) đang luyện tính nhẩm biểu thức **số học mô-đun** $10^9 + 7$.

- Cậu viết ra một biểu thức toán học đẹp, gồm:
 - **``n`` số nguyên không âm** (mỗi số $< 10^9 + 7$)
 - **Các dấu ``+`` và ``*``** giữa các số
 - Sau đó, tính ra **kết quả cuối cùng** (mod $10^9 + 7$) và ghi lại

🤖 Khi Vova ngủ trưa, một số bạn xấu đã **thay đổi vài chữ số trong biểu thức** (chỉ chữ số, không thay đổi phép toán hoặc kết quả). Khi tỉnh dậy, cậu thấy:

- Một số chữ số bị thay đổi
- Các dấu ``+``, ``*``, và **kết quả cuối** vẫn giữ nguyên

🤔 Vova đặt câu hỏi:

*“Liệu có thể **khôi phục được biểu thức ban đầu**, bằng cách thay **không quá 2 chữ số** (ở tối đa 2 số khác nhau)?”*

Hãy giúp Vova xác định:

- Nếu **không thể** khôi phục, in **``NO``**
- Nếu **có thể**, in **``YES``**, rồi in:
 - Số lượng số bị thay đổi ($k \leq 2$)
 - Với mỗi số bị thay đổi: vị trí (tính từ trái sang phải, bắt đầu từ 1) và **giá trị ban đầu**

🔒 Yêu cầu:

- Tổng số chữ số bị thay đổi ≤ 2
- Không tạo ra số có **0 ở đầu**
- Tất cả số vẫn phải $< 10^9 + 7$

Dữ liệu vào

Một dòng chứa biểu thức:

- Dạng: ``A1 op A2 op A3 ... op An = B``
- Các phần tử và phép toán cách nhau bởi đúng **1 dấu cách**

Trong đó:

- Mỗi ``Ai``: số nguyên không âm, **không có số 0 ở đầu**, $< 10^9 + 7$
 - ``op`` là ``+`` hoặc ``*``
 - ``B``: kết quả cuối cùng theo mô-đun $10^9 + 7$
-

Dữ liệu ra

- Nếu không thể khôi phục đúng với ≤ 2 chữ số bị thay đổi \rightarrow in ``NO``
- Nếu có thể:

```
YES
k
pos1 value1
pos2 value2
```

(với ``k = 1`` hoặc ``2``)

Có thể có nhiều lời giải \rightarrow in **bất kỳ lời giải hợp lệ nào**

Ví dụ

Ví dụ 1:

Input:

$56 + 14 * 86 + 51 * 55 = 3925$

Output:

YES

1

3 76

→ sửa số thứ 3 (86 → 76) giúp biểu thức đúng.

Ví dụ 2:

Input:

$97 + 14 * 31 * 76 + 99 * 73 = 40930$

Output:

NO

Chấm điểm theo từng Subtask

Subtask	Điểm	Giới hạn bổ sung	Yêu cầu các subtask trước	Ghi chú
1	7	<code>`n ≤ 45`</code>	-	
2	13	<code>`n ≤ 100`</code>	1	
3	15	Tất cả số trong biểu thức bên trái < 10	-	
4	12	Không có phép nhân <code>`*`</code>	-	
5	13	Không có phép cộng <code>`+`</code>	-	
6	15	Tất cả số trong biểu thức bên trái < 10^5	3	

Subtask	Điểm	Giới hạn bổ sung	Yêu cầu các subtask trước	Ghi chú
7	25	Không giới hạn - mỗi test chấm riêng (25 test riêng)	1-6	potest

 **Ghi chú:** Trong Subtask 3 và 6, chỉ các số bên trái dấu `=` bị giới hạn.

Bạn muốn mình trình bày ý tưởng hoặc code cho bài này chứ?