universal cup 3 stage 29 : Metropolis

Created: 4/26/2025 22:17
Updated: 4/26/2025 22:22
Exported: 4/26/2025 22:21

Đề bài

Một chiến binh tên Alan nghi ngờ rằng có những khu vực nguy hiểm (ẩn) trong các công trình kiến trúc dạng mê cung. Các bức tường trong mê cung được biểu diễn bằng các đa giác 2D và có thể chặn tầm nhìn cũng như di chuyển. Một điểm được coi là *ẩn* nếu nó **có thể tiếp cận** từ bên ngoài nhưng **không thể quan sát** từ bên ngoài.

Cho một bản đồ với các bức tường (mỗi bức tường là một đa giác), hãy tính diện tích tổng của các vùng *có thể tiếp cận nhưng bị ẩn* (vùng nguy hiểm).

- Một điểm **có thể tiếp cận** nếu không nằm bên trong bất kỳ bức tường nào.
- Một điểm **bị ẩn** nếu **không có đường thẳng nào từ điểm đó ra ngoài mà không cắt qua tường** (theo mọi hướng).

Input:

- Dòng đầu tiên chứa số nguyên n $(1 \leq n \leq 30)$: số lượng bức tường (đa giác).
- Mỗi dòng tiếp theo mô tả một đa giác:
  - Số nguyên k_i $(3 \leq k_i \leq 90)$: số đỉnh của đa giác.
  - Sau đó là $2k_i$ số nguyên: tọa độ các đỉnh $(x_1, y_1), (x_2, y_2), ..., (x_{k_i}, y_{k_i})$, theo thứ tự chiều kim đồng hồ hoặc ngược.

Giới hạn:

- Tổng số đỉnh của tất cả các đa giác không vượt quá 90.
- Không có hai đa giác nào giao nhau hoặc đỉnh trùng nhau.

- Tọa độ: $-10^4 \leq x_j, y_j \leq 10^4$.

Output:

- Một số thực: diện tích vùng *có thể tiếp cận nhưng bị ẩn*, với sai số tuyệt đối hoặc tương đối không vượt quá 10^{-6}.

Ví dụ:

```
Input:
2
8 0 0 7 0 7 7 6 7 6 1 1 1 1 7 0 7
4 2 7 5 7 5 6 2 6

Output:
2.25
```

```
Input:
2
4 2 4 5 8 9 4 5 7
4 4 5 5 1 6 5 5 4

Output:
1.4
```

Tóm tắt:

- **Bài toán:** Tính diện tích vùng ẩn (không nhìn thấy từ bên ngoài nhưng có thể đi vào) trong bản đồ có nhiều bức tường (đa giác).

- **Input:** Số lượng đa giác và tọa độ các đỉnh.
- **Output:** Diện tích vùng ẩn với độ chính xác cao.

Đề bài

Aqua đang học về lý thuyết đồ thị, cụ thể là thuật toán **Duyệt theo chiều sâu (DFS)**. Aqua viết thuật toán DFS như sau:

```
function dfs(u)
    đánh dấu u đã thăm
    in ra u
    for v là đỉnh kề với u
        nếu v chưa thăm
            dfs(v)
```

```
function run_dfs()
    for mỗi đỉnh v trong đồ thị
        nếu v chưa thăm
            dfs(v)
```

Tuy nhiên, Aqua nhận ra rằng **thứ tự duyệt các đỉnh không cố định**, vì:

- Dòng 4: thứ tự xét các đỉnh kề u không cố định.
- Dòng 12: thứ tự xét các đỉnh toàn cục cũng không cố định.

Aqua đặt ra một bài toán:

Cho một **đồ thị vô hướng** với n đỉnh và m cạnh, cùng với một **hoán vị** $p_1, p_2, ..., p_n$ của các đỉnh. Hãy xác định **số cạnh ít nhất cần thêm vào đồ thị** sao cho, nếu chạy thuật toán DFS của Aqua thì **thứ tự in ra các đỉnh sẽ chính xác là hoán vị p**.

Ngoài ra, bạn cần liệt kê các cạnh đã thêm để Aqua kiểm tra.

Đảm bảo luôn tồn tại đáp án.

Input:

- Dòng đầu tiên chứa hai số nguyên n, m $(1 \leq n \leq 3 \times 10^5, 0 \leq m \leq 5 \times 10^5)$: số đỉnh và số cạnh.
- m dòng tiếp theo, mỗi dòng gồm hai số nguyên u_i, v_i $(1 \leq u_i, v_i \leq n, u_i \neq v_i)$: cạnh nối giữa đỉnh u_i và v_i.
- Dòng cuối cùng chứa hoán vị $p_1, p_2, ..., p_n$ của các đỉnh từ 1 đến n.

Output:

- Dòng đầu tiên: số cạnh k cần thêm.
- k dòng tiếp theo, mỗi dòng gồm hai số nguyên a_i, b_i: cạnh được thêm nối giữa hai đỉnh a_i và b_i.

Nếu có nhiều đáp án, in ra bất kỳ đáp án nào.

Ví dụ:

```
Input:
6 6
1 3
1 4
2 3
3 4
```

```
3 6
5 6
1 2 3 4 5 6

Output:
2
1 2
4 5
```

```
Input:
8 8
2 8
3 8
5 6
1 6
6 3
8 7
2 3
4 3
1 8 7 5 4 2 3 6

Output:
4
1 8
7 5
5 4
4 2
```

Tóm tắt:

- **Bài toán:** Thêm ít nhất bao nhiêu cạnh vào đồ thị sao cho chạy DFS in ra đúng thứ tự cho trước (hoán vị p).
- **Input:** Đồ thị với n đỉnh, m cạnh, và một hoán vị p.
- **Output:** Số cạnh cần thêm và danh sách các cạnh đó.

Đề bài

Alice và Bob là hai chiến lược gia xuất sắc, họ tình cờ phát hiện ra một chiếc bảng đen huyền bí trong một ngôi đền cổ. Trên bảng đen có ghi một dãy số nguyên liên tiếp từ l đến r. Bên cạnh bảng đen là một con số đặc biệt x, ban đầu được đặt bằng l.

Luật chơi:

- Hai người chơi lần lượt xóa **một bội số của** x trên bảng đen.
- Alice luôn là người đi trước.
- Sau mỗi lượt xóa, giá trị của x **tăng thêm 1**.
- Nếu người chơi không thể xóa được bội số của x (vì không còn bội số nào phù hợp trên bảng đen), người đó **thua cuộc**.

Cả hai chơi tối ưu. Hãy xác định **ai sẽ thắng**.

Input:

- Dòng đầu tiên: số lượng test T $(1 \leq T \leq 10^5)$.
- Mỗi test gồm một dòng: hai số nguyên l, r $(1 \leq l \leq r \leq 10^9)$, là đoạn số ban đầu trên bảng đen.

Output:

- Với mỗi test, in ra **"Alice"** nếu Alice thắng, hoặc **"Bob"** nếu Bob thắng.

Ví dụ:

```
Input:
3
2 4
4 4
6 7

Output:
Alice
Alice
Bob
```

Tóm tắt:

- **Bài toán:** Hai người chơi xóa bội số của x từ dãy l đến r, x tăng sau mỗi lượt. Ai không xóa được thì thua. Xác định người thắng khi cả hai chơi tối ưu.
- **Input:** Nhiều test, mỗi test cho đoạn l, r.
- **Output:** Người thắng cuộc ("Alice" hoặc "Bob") cho mỗi test.

Đề bài

David Liu có n chiếc hộp xếp thành một hàng, hộp thứ i được đánh số từ 1 đến n. Ban đầu tất cả các hộp đều **trống**.

Frank có một **xâu nhị phân** s độ dài n:

- Nếu $s_i = 1$, Frank sẽ đặt 10^{18} **viên đá** vào hộp thứ i.
- Nếu $s_i = 0$, Frank sẽ **không đặt viên đá nào** vào hộp thứ i.

Sau đó, David Liu muốn **xóa hết tất cả viên đá** trong các hộp. Tuy nhiên, anh ta chỉ có thể thực hiện **một thao tác duy nhất nhiều lần (hoặc không làm gì)**:

- **Chọn một hộp thứ i $(1 \leq i \leq n-2)$ mà trong đó còn ít nhất 1 viên đá**, sau đó:
  - **Lấy ra 1 viên đá** từ hộp i.
  - **Hoán đổi toàn bộ nội dung** của hộp $i+1$ và hộp $i+2$.

David Liu hỏi bạn: **Liệu có thể xóa sạch tất cả viên đá trong các hộp không?**

Input:

- Dòng đầu tiên chứa số nguyên T $(1 \leq T \leq 10^6)$: số lượng test.
- Mỗi test gồm 2 dòng:
  - Dòng đầu tiên: số nguyên n $(3 \leq n \leq 10^6)$: số lượng hộp.
  - Dòng thứ hai: xâu nhị phân s độ dài n.

Giới hạn:
Tổng độ dài tất cả các xâu s không vượt quá 10^6.

Output:

- Với mỗi test, in ra **"Yes"** nếu có thể xóa hết đá, hoặc **"No"** nếu không thể.

Ví dụ:

```
Input:
3
3
101
4
1010
5
00000

Output:
No
Yes
Yes
```

Tóm tắt:

- **Bài toán:** Kiểm tra xem có thể xóa sạch đá trong dãy hộp không, chỉ với thao tác: lấy 1 viên đá từ hộp i (1 ≤ i ≤ n-2) và hoán đổi hộp $i+1$ và $i+2$.
- **Input:** T test, mỗi test gồm số hộp n và xâu nhị phân s.
- **Output:** "Yes" hoặc "No" cho mỗi test.

Đề bài

Bạn đang tham quan một **vườn bách thảo** gồm n khu vực và m con đường nối giữa các khu vực. Một **lộ trình** trong vườn bách thảo được định nghĩa là một **chu trình đơn** thỏa mãn:

- Có ít nhất **3 khu vực**.
- Không có khu vực nào lặp lại (ngoại trừ điểm đầu trùng điểm cuối).
- Mỗi cặp khu vực liên tiếp được nối bởi một con đường.

- Khu vực cuối cùng nối lại với khu vực đầu tiên.

Bạn muốn kiểm tra rằng: **tất cả các lộ trình trong vườn đều có độ dài bằng nhau**.

- Nếu **không có lộ trình nào tồn tại**, coi như **tất cả các lộ trình đều có độ dài bằng nhau**.

Input:

- Dòng đầu tiên: số lượng test T $(1 \leq T \leq 5 \times 10^5)$.
- Mỗi test gồm:
  - Dòng đầu tiên: hai số nguyên n, m $(1 \leq n \leq 5 \times 10^5, 0 \leq m \leq 5 \times 10^5)$: số khu vực và số con đường.
  - m dòng tiếp theo, mỗi dòng gồm hai số nguyên u_i, v_i: con đường nối khu vực u_i và v_i.

Giới hạn:

- Tổng số khu vực và số con đường của tất cả các test không vượt quá 5×10^5.
- Đồ thị liên thông (có thể đi từ bất kỳ khu vực nào đến bất kỳ khu vực nào).

Output:

- Với mỗi test, in ra **"Yes"** nếu tất cả các lộ trình có độ dài bằng nhau, hoặc **"No"** nếu ngược lại.

Ví dụ:

```
Input:
1
5 6
1 2
2 3
3 1
1 4
4 5
5 1

Output:
Yes
```

```
Input:
2
2 1
1 2
5 6
1 2
2 3
3 1
2 4
3 5
4 5

Output:
Yes
No
```

Tóm tắt:

- **Bài toán:** Kiểm tra trong đồ thị vô hướng có liên thông, tất cả các **chu trình đơn** có cùng độ dài hay không.

- **Input:** Nhiều test, mỗi test gồm số đỉnh n, số cạnh m, và danh sách các cạnh.
- **Output:** "Yes" hoặc "No" cho mỗi test.

Đề bài

Bạn có biết **Bogosort**? Đây là một thuật toán sắp xếp ngẫu nhiên rất thú vị:

- Trộn ngẫu nhiên toàn bộ mảng.
- Kiểm tra xem mảng đã được sắp xếp chưa.

Thời gian chạy trung bình của **Bogosort** là $O(n \cdot n!)$, nhưng nếu mảng đã sắp xếp sẵn thì là $O(n)$.

Để cải thiện Bogosort, ta có một phiên bản **Fast Bogosort**, sắp xếp một hoán vị từ 1 đến n:

1. Nếu mảng đã được sắp xếp, dừng lại.
2. Nếu chưa, chia mảng thành nhiều đoạn nhỏ nhất có thể, sao cho mỗi đoạn là một **hoán vị liên tiếp**:
    - Mỗi đoạn $[l_i, r_i]$ chứa đúng các số từ l_i đến r_i.
    - Các đoạn **không giao nhau** và **phủ hết toàn bộ mảng**.
3. Với mỗi đoạn $[l, r]$ có $l < r$, gọi hàm **shuffle(l, r)** để trộn ngẫu nhiên đoạn đó.

Nhiệm vụ:

- Cho một hoán vị $a_1, a_2, ..., a_n$, hãy tính **số lần kỳ vọng** gọi hàm **shuffle(l, r)** trong Fast Bogosort.

- Do kết quả là **số hữu tỉ**, hãy biểu diễn nó dưới dạng phân số tối giản $\frac{P}{Q}$.
- In ra số nguyên duy nhất R sao cho:

$$R \cdot Q \equiv P \pmod{998244353}, \quad 0 \leq R < 998244353$$

Input:

- Dòng đầu tiên: số nguyên n $(1 \leq n \leq 10^5)$.
- Dòng thứ hai: hoán vị $a_1, a_2, ..., a_n$.

Output:

- Một số nguyên R như mô tả.

Ví dụ:

```
Input:
5
2 1 5 3 4

Output:
332748123
```

```
Input:
10
4 2 3 1 6 5 9 7 10 8
```

```
Output:
453747445
```

Tóm tắt:

- **Bài toán:** Tính kỳ vọng số lần gọi **shuffle(l, r)** của thuật toán **Fast Bogosort** trên một hoán vị từ 1 đến n.
- **Input:** Số nguyên n và hoán vị a.
- **Output:** Số nguyên R theo modulo 998244353.

Đề bài

Có n **đường thẳng đỏ** và n **đường thẳng xanh** trong mặt phẳng 2D:

- Đường thẳng đỏ thứ i: $y = a_i x + b_i$.
- Đường thẳng xanh thứ i: $x = c_i$.

Giá trị ghép cặp giữa một đường đỏ và một đường xanh là **tọa độ y** tại giao điểm của hai đường này.

Bạn cần **ghép mỗi đường đỏ với đúng một đường xanh** (không bỏ sót), thu được n giá trị. Nhiệm vụ của bạn là tìm **giá trị trung vị lớn nhất có thể** của n giá trị đó.

- **Trung vị** của dãy độ dài n là phần tử đứng ở vị trí thứ $\lceil \frac{n}{2} \rceil$ sau khi sắp xếp giảm dần.

Input:

- Dòng đầu tiên: số lượng test T $(1 \leq T \leq 10^5)$.
- Với mỗi test:
  - Dòng đầu tiên: số nguyên n $(1 \leq n \leq 10^5)$.
  - Dòng thứ hai: n số nguyên $a_1, a_2, ..., a_n$ $(-10^9 \leq a_i \leq 10^9)$.
  - Dòng thứ ba: n số nguyên $b_1, b_2, ..., b_n$ $(-10^{18} \leq b_i \leq 10^{18})$.
  - Dòng thứ tư: n số nguyên $c_1, c_2, ..., c_n$ $(-10^9 \leq c_i \leq 10^9)$.

Giới hạn:

- Tổng tất cả n của các test không vượt quá 10^5.

Output:

- Với mỗi test, in ra **giá trị trung vị lớn nhất có thể**.

Ví dụ:

```
Input:
3
5
0 5 -2 1 2
9 -4 0 10 5
-4 -1 4 -2 4
10
-6 3 1 0 6 -2 -4 3 0 10
22 65 11 1 -34 -1 -39 -28 25 24
10 9 1 -2 -5 8 -7 -10 -7 -7
1
101
48763
```

```
651

Output:
9
25
48763
```

Tóm tắt:

- **Bài toán:** Ghép từng đường đỏ với từng đường xanh, tính giá trị giao điểm (tọa độ y), sao cho **trung vị lớn nhất có thể**.
- **Input:** T test, mỗi test có số đường n, hệ số a, b của các đường đỏ và hoành độ c của các đường xanh.
- **Output:** Trung vị lớn nhất có thể của các giá trị giao điểm.

Đề bài

Kotoha nhận được một trò chơi xếp hình hình **lục giác** từ bạn bè:

- Bàn cờ gồm **n hàng** lục giác đều:
  - Hàng thứ i có đúng i ô lục giác.
  - Mỗi ô lục giác ở hàng i (với $i > 1$) tiếp giáp với hai ô lục giác ở hàng trên (nếu có).
- Kotoha có nhiều mảnh ghép hình chữ **V**:
  - Mỗi mảnh chiếm đúng **3 ô lục giác**.
  - Có **6 cách xoay** mảnh ghép (xoay theo bội số của 60 độ).

Nhiệm vụ:

- Đặt càng nhiều mảnh ghép chữ V càng tốt, thỏa mãn:
  - Mỗi mảnh nằm hoàn toàn trong bàn cờ và chiếm đúng 3 ô lục giác.
  - **Không có mảnh nào đè lên nhau**.
  - **Không có hai mảnh liền kề nào có cùng màu** (hai mảnh liền kề nếu **có cạnh chung**).

Màu sắc:

- Mỗi mảnh có một màu trong **26 chữ cái in hoa A-Z**.

Yêu cầu xuất kết quả:

- Xuất ra dạng lưới thể hiện trạng thái bàn cờ:
  - **Dòng thứ i** có đúng $n + i - 1$ ký tự.
  - Ký tự ở cột thứ j trong dòng thứ i là:
    - `' '` (dấu cách) nếu $j \leq n - i$.
    - `' '` (dấu cách) nếu $j - (n - i)$ là **số chẵn**.
    - `'.'` nếu ô lục giác **không bị phủ**.
    - Một chữ cái (A-Z) nếu ô lục giác bị một mảnh phủ, thể hiện màu của mảnh.

Ràng buộc:

- $1 \leq n \leq 1024$.

Ví dụ:

```
Input:
2

Output:
```

```
 .
. .
```

```
Input:
3

Output:
 W
W .
. W .
```

```
Input:
4

Output:
   .
  . R
 . B R
B B R .
```

Tóm tắt:

- **Bài toán:** Xếp tối đa mảnh ghép chữ V vào lưới hình tam giác lục giác đều kích thước n, sao cho:
  - Không đè lên nhau.
  - Không hai mảnh liền kề nào trùng màu.
- **Input:** Số nguyên n.
- **Output:** Bàn cờ sau khi xếp, đúng định dạng mô tả.

Đề bài

Justin – một pháp sư tài ba – sở hữu **n hiện vật phép thuật**, mỗi hiện vật có sức mạnh là một số nguyên không âm.

Anh ta phát hiện ra một phép thuật gọi là **“k-hợp nhất” (k-fusion)**:

- Chọn **chính xác k hiện vật** để hợp nhất thành một hiện vật mới.
- Sức mạnh của hiện vật mới là **tích** của k hiện vật được hợp nhất.
- Các hiện vật gốc sẽ **biến mất** sau khi hợp nhất.

Justin muốn sử dụng phép **k-hợp nhất** (có thể thực hiện nhiều lần hoặc không) để **tối đa hóa sức mạnh lớn nhất còn lại** sau tất cả các hợp nhất.

Tuy nhiên, do sức mạnh có thể rất lớn, hãy tính **phần dư** của sức mạnh lớn nhất sau khi chia cho **998244353**.

Input:

- Dòng đầu tiên: số lượng test T $(1 \leq T \leq 1000)$.
- Với mỗi test:
  - Dòng đầu tiên: hai số nguyên n, k $(2 \leq n \leq 2 \times 10^5, 2 \leq k \leq n)$.
  - Dòng thứ hai: n số nguyên $p_1, p_2, ..., p_n$ $(0 \leq p_i \leq 10^9)$: sức mạnh của các hiện vật.

Giới hạn:

- Tổng tất cả n của các test không vượt quá 2×10^5.

Output:

- Với mỗi test, in ra phần dư của **sức mạnh lớn nhất cuối cùng** sau tất cả các phép hợp nhất.

Ví dụ:

```
Input:
3
8 3
44 5 2018 8 8 2024 8 28
5 4
4 5 5 1 0
5 2
0 0 0 0 0

Output:
923923948
100
0
```

Tóm tắt:

- **Bài toán:** Tối đa hóa giá trị lớn nhất còn lại sau một loạt các phép **k-hợp nhất** (lấy tích k số), rồi in ra **phần dư** của giá trị đó khi chia cho **998244353**.
- **Input:** Nhiều test, mỗi test có n, k, và dãy sức mạnh.
- **Output:** Phần dư của sức mạnh lớn nhất cuối cùng sau hợp nhất.

Đề bài

Bạn đang phát triển một trình duyệt web, nơi các **thành phần giao diện** được mô phỏng bằng các **hình chữ nhật** trên mặt phẳng tọa độ.

- Các hình chữ nhật được **lồng nhau đúng quy tắc**:
  - Hoặc **chứa nhau hoàn toàn**.
  - Hoặc **không giao nhau**.

Trình duyệt **render** (vẽ) các thành phần theo **lớp (depth)**:

- Trong mỗi lượt, **các hình ngoài cùng (không bị hình nào chứa)** sẽ được render và bỏ qua ở các lượt sau.
- **Render depth** của một hình là:
  - **0** nếu không bị hình nào chứa.
  - **1 + độ sâu lớn nhất của hình chứa nó** nếu bị hình khác chứa.

Animation:

- Mỗi hình chữ nhật có một **công tắc animation** (bật/tắt).
- Một hình được coi là **có animation** nếu **nó hoặc bất kỳ hình nào nằm bên trong nó** có animation đang bật.

Nhiệm vụ:

- Xử lý các sự kiện:
  1. **Toggle animation**: Đảo trạng thái animation của một hình.
  2. **Truy vấn**: Đếm số hình **có animation** ở một render depth cụ thể.

Input:

- Dòng đầu: hai số nguyên n, q $(1 \leq n, q \leq 5 \times 10^5)$: số hình chữ nhật và số sự kiện.
- n dòng tiếp theo: mỗi dòng 4 số nguyên x_1, y_1, x_2, y_2 mô tả hình chữ nhật:
  - Góc dưới trái: (x_1, y_1).
  - Góc trên phải: (x_2, y_2).
  - Các hình không giao nhau, kể cả tại biên.
- q dòng tiếp theo: mỗi dòng là một sự kiện:
  - `^ i`: Đảo trạng thái animation của hình thứ i.
  - `? k`: Truy vấn số lượng hình **có animation** tại **depth k**.

Output:

- Với mỗi truy vấn `? k`, in ra số hình có animation tại depth k.

Ví dụ:

```
Input:
8 7
1 1 15 11
2 2 7 4
2 5 7 10
3 6 4 9
5 6 6 7
8 2 14 10
9 5 13 9
```

```
11 6 12 7
^ 4
^ 5
^ 8
? 0
? 1
? 2
? 3

Output:
1
2
3
1
```

Tóm tắt:

- **Bài toán:** Xử lý **toggle animation** và đếm số hình có animation tại các **mức depth** khác nhau, với **hình chữ nhật lồng nhau**.
- **Input:** Số hình chữ nhật, các hình, và danh sách sự kiện (toggle/truy vấn).
- **Output:** Trả về kết quả cho các truy vấn `` `? k` ``.

Đề bài

Có hai nhóm hiệp sĩ, mỗi nhóm gồm **n** hiệp sĩ:

- Hiệp sĩ thứ i của **nhóm 1** có sức mạnh a_i.
- Hiệp sĩ thứ j của **nhóm 2** có sức mạnh b_j.

Quy tắc đấu:

- Mỗi trận đấu gồm **một hiệp sĩ từ nhóm 1** và **một hiệp sĩ từ nhóm 2**.

- **Mỗi hiệp sĩ chỉ tham gia một trận đấu**.

Độ điên loạn (lunaticus) của một trận đấu:

- $(a_i + b_j) \mod 998244353$.

Yêu cầu:

- Có **m cặp hiệp sĩ bị cấm đấu** (không được chọn).
- Với mỗi x từ **1 đến k**, hãy tính:
  - **Tổng độ điên loạn lớn nhất có thể** khi tổ chức đúng **x trận đấu**.
  - Nếu không thể tổ chức **x trận đấu**, in ra **-1**.

Input:

- Dòng đầu: 3 số nguyên n, m, k $(1 \leq n \leq 10^5, 0 \leq m \leq 3 \times 10^5, 1 \leq k \leq \min(n, 200))$.
- Dòng tiếp: n số nguyên $a_1, ..., a_n$ $(0 \leq a_i < 998244353)$.
- Dòng tiếp: n số nguyên $b_1, ..., b_n$ $(0 \leq b_j < 998244353)$.
- m dòng tiếp: mỗi dòng chứa cặp **cấm đấu** u_i, v_i $(1 \leq u_i, v_i \leq n)$.

Output:

- In ra k số, số thứ i là tổng độ điên loạn lớn nhất khi tổ chức đúng **i trận đấu**. Nếu không thể, in **-1**.

Ví dụ:

```
Input:
3 4 3
10 998244352 5
998244352 8 6
1 2
1 3
2 2
3 3

Output:
998244351 998244364 27
```

Tóm tắt:

- **Bài toán:** Chọn tối đa **k trận đấu** hợp lệ (không bị cấm), sao cho:
  - **Tổng độ điên loạn** của các trận là lớn nhất.
  - Với từng x từ **1 đến k**, in ra tổng tốt nhất có thể.
- **Input:** Hai dãy sức mạnh và danh sách cặp cấm.
- **Output:** Tổng độ điên loạn tối đa cho từng số lượng trận từ 1 đến k.

Đề bài

Một bàn ăn có **n người** ngồi quanh **một chiếc mâm xoay tròn (lazy Susan)**.

- Người được đánh số từ **0 đến n-1**, ngồi theo chiều kim đồng hồ.
- Khoảng cách giữa các người bằng nhau.

- Người kế bên người số **n-1** là người số **0**.

Có **n món ăn** đánh số từ **0 đến n-1** được đặt lần lượt trên mâm:

- Người phục vụ bắt đầu từ người số **x** và đặt món:
  - Món số **0** trước mặt người số **x**.
  - Món số **1** trước mặt người số **(x+1) mod n**.
  - ... cho đến món số **n-1**.

Xoay mâm:

- Xoay **1 giây**:
  - **Theo chiều kim đồng hồ:** món ở trước người **i** chuyển sang người **(i+1) mod n**.
  - **Ngược chiều kim đồng hồ:** món ở trước người **i** chuyển sang người **(i+n-1) mod n**.

Tầm với:

- Người số **i** có tầm với **r_i**.
- Họ có thể **ăn món ăn trước mặt người số j** nếu:
  - Có một số nguyên **k** sao cho **$-r_i \le k \le r_i$** và **$(i + k + n) \bmod n = j$**.

Yêu cầu:

- Có **m yêu cầu** dạng:
  - Người **p_i** muốn ăn món **d_i** trước **t_i giây** (tính từ sau khi phục vụ xong).

Bạn cần kiểm tra với **mỗi vị trí bắt đầu x (0 ≤ x < n)**:

- Có thể **xoay mâm** sao cho **mọi yêu cầu được thỏa mãn** không.

Input:

- Số lượng test T $(1 \leq T \leq 2500)$.
- Mỗi test gồm:
  - Hai số nguyên n, m $(2 \leq n \leq 5000, 0 \leq m \leq \min(n^2, 10^5))$.
  - Một dòng: n số nguyên $r_0, ..., r_{n-1}$ $(0 \leq r_i \leq n)$: tầm với của từng người.
  - m dòng: mỗi dòng 3 số p_i, d_i, t_i: yêu cầu.

Giới hạn:

- Tổng n tất cả các test ≤ 5000.
- Tổng m tất cả các test ≤ 10^5.

Output:

- Với mỗi test, in ra **xâu nhị phân độ dài n**:
  - Ký tự thứ x: **1** nếu bắt đầu từ người **x** thỏa mãn tất cả yêu cầu.
  - Ngược lại, **0**.

Ví dụ:

```
Input:
4
5 6
0 0 1 2 1
3 2 2
0 0 2
4 4 4
1 1 3
4 3 5
0 3 2
8 10
2 2 2 0 3 0 0 1
4 3 1
1 0 1
0 7 6
4 4 1
1 7 5
0 4 2
5 3 4
4 6 1
7 7 3
0 2 4
4 4
0 0 0 0
1 0 2
2 0 2
0 0 2
3 0 2
13 0
1 1 4 5 1 4 1 9 1 9 8 1 0

Output:
10100
11111010
0000
1111111111111
```

Tóm tắt:

- **Bài toán:** Với **mỗi vị trí bắt đầu x**, kiểm tra xem có cách **xoay mâm** sao cho:
  - **Mọi người đều ăn được món mình muốn trước thời gian yêu cầu**.
- **Input:** Số người, tầm với, yêu cầu.
- **Output:** Xâu nhị phân thể hiện vị trí bắt đầu hợp lệ.

Đề bài

ICPC đang nghiên cứu loài **chim cánh cụt** và sử dụng xe tự hành để thu thập ảnh, từ đó xây dựng **mô hình 3D** cho chim cánh cụt. Tuy nhiên, mô hình máy học hiện tại thường **dự đoán sai hình dạng**, nên ICPC muốn **phạt** mô hình nếu đánh dấu những hình không giống **chim cánh cụt**.

Chim cánh cụt đứng thẳng có thể được mô hình hóa thành một **hình trụ tròn đứng**:

- Hình trụ này:
  - Có **trục thẳng đứng (vuông góc với mặt phẳng đáy)**.
  - **Mặt đáy nằm trên mặt phẳng x-y**.
  - **Không phải hình trụ suy biến** (không có thể tích = 0 hay vô hạn).

Nhiệm vụ:

- Cho n điểm trong không gian 3D.
- Kiểm tra xem có tồn tại **một hình trụ tròn đứng** sao cho **tất cả các điểm đều nằm trên mặt bên của hình trụ** không?
- Nếu **có**, in ra:

```
probably
```

- Nếu **không**, in ra:

```
not a penguin
```

Input:

- Dòng đầu tiên: số nguyên n $(1 \leq n \leq 10^5)$.
- n dòng tiếp theo: mỗi dòng chứa 3 số nguyên x, y, z $(-10^9 \leq x, y \leq 10^9, 0 \leq z \leq 10^9)$: tọa độ điểm.

Output:

- In ra **probably** hoặc **not a penguin**.

Ví dụ:

```
Input:
5
0 0 0
3 4 1
-3 -4 2
-3 4 3
3 -4 4
```

```
Output:
probably
```

```
Input:
4
0 6 2
3 4 1
-3 -4 1
-3 4 1

Output:
not a penguin
```

Tóm tắt:

- **Bài toán:** Kiểm tra xem các điểm 3D có thể nằm trên mặt bên của **một hình trụ tròn đứng duy nhất** không.
- **Input:** Danh sách tọa độ các điểm.
- **Output:** "probably" hoặc "not a penguin".