universal cup 3 stage 32 : Dhaka

Dịch đề bài:

Bài A. Sắp xếp và &

Bạn được cho một hoán vị độ dài N. Nhiệm vụ là sắp xếp hoán vị này theo thứ tự tăng dần.

Trong một lần thao tác, bạn có thể hoán đổi phần tử thứ i và phần tử thứ j ($1 \leq i, j \leq N$), chi phí của thao tác này là $i \& j$ (phép **AND** bitwise giữa i và j).

Tổng chi phí để sắp xếp hoán vị là tổng chi phí của tất cả các thao tác hoán đổi bạn thực hiện. Bạn **không được thực hiện quá $3N$ thao tác**.

Hãy in ra chi phí tối thiểu để sắp xếp hoán vị và liệt kê một dãy các thao tác hoán đổi giúp sắp xếp hoán vị với chi phí tối thiểu đó.

Input:

- Dòng đầu tiên chứa số nguyên T ($1 \leq T \leq 10^4$): số lượng test.
- Mỗi test gồm:
  - Một số nguyên N ($1 \leq N \leq 10^4$): độ dài hoán vị.
  - Một dòng chứa N số nguyên $a_1, a_2, ..., a_N$ — là một hoán vị của các số từ 1 đến N.

Ràng buộc: Tổng tất cả N trong các test không vượt quá 10^5.

Output:

- Với mỗi test:
  - Dòng đầu tiên in chi phí tối thiểu để sắp xếp.
  - Dòng thứ hai in số lượng thao tác hoán đổi.
  - Mỗi dòng tiếp theo in hai số nguyên i và j: các chỉ số được hoán đổi.

Ví dụ:

```
Input:
1
4
3 1 4 2

Output:
0
3
1 4
4 3
1 2
```

Tóm tắt:

- **Cho:** Một hoán vị độ dài N.
- **Mục tiêu:** Sắp xếp hoán vị theo thứ tự tăng dần.
- **Quy tắc hoán đổi:** Được phép hoán đổi hai vị trí bất kỳ, chi phí hoán đổi là **i & j**.

- **Yêu cầu:** Tìm cách sắp xếp sao cho **tổng chi phí hoán đổi nhỏ nhất**, không quá **3N thao tác**.
- **Kết quả cần in:**
  - Tổng chi phí tối thiểu.
  - Số thao tác hoán đổi.
  - Danh sách các thao tác hoán đổi.

Dịch đề bài:

Bài B. Gona Guni

Bạn được cho một **cây** gồm N đỉnh. Bạn có thể chọn một tập hợp không rỗng các đỉnh S từ cây này. Từ tập S, bạn tạo ra **đồ thị con liên thông nhỏ nhất** G sao cho G chứa tất cả các đỉnh trong S.

Ta định nghĩa **cost(S)** là **kích thước của tập đỉnh phủ nhỏ nhất** của đồ thị G. (**Tập đỉnh phủ nhỏ nhất** là tập các đỉnh sao cho mỗi cạnh trong đồ thị đều có ít nhất một đầu mút thuộc tập này, và kích thước tập là nhỏ nhất có thể).

Lưu ý: Nếu đồ thị chỉ có một đỉnh, thì **cost(S) = 0**.

Nhiệm vụ của bạn là **tính tổng** $\text{cost}(S)^M$ cho tất cả các tập S có thể chọn, và lấy kết quả **modulo 998244353**.

Input:

- Dòng đầu tiên là số nguyên T ($1 \leq T \leq 3000$): số lượng test.
- Mỗi test gồm:

- Một dòng gồm hai số nguyên N ($1 \leq N \leq 3 \cdot 10^5$) và M ($0 \leq M \leq 200$): số đỉnh của cây và số mũ.
- Sau đó là $N-1$ dòng, mỗi dòng gồm hai số nguyên U và V ($1 \leq U, V \leq N$): biểu diễn cạnh nối giữa đỉnh U và V.

Ràng buộc: Tổng tất cả các giá trị N trong các test không vượt quá $3 \cdot 10^5$.

Output:

- Với mỗi test, in ra một dòng chứa kết quả tính tổng $\text{cost}(S)^M$ với tất cả tập S, lấy **modulo 998244353**.

Ví dụ:

```
Input:
2
3 1
1 2
1 3
20 200
1 2
1 3
2 4
1 5
5 6
1 7
6 8
6 9
3 10
4 11
6 12
11 13
```

```
4 14
13 15
15 16
6 17
13 18
15 19
13 20

Output:
20
xxx
```

Tóm tắt:

- **Cho:** Một cây N đỉnh và một số mũ M.
- **Với mỗi tập S không rỗng** các đỉnh trong cây:
  - Xây dựng đồ thị con liên thông nhỏ nhất bao phủ S.
  - Tính **cost(S)** là kích thước tập đỉnh phủ nhỏ nhất của đồ thị con đó.
  - Tính **cost(S)^M**.
- **Nhiệm vụ:** Tính tổng tất cả **cost(S)^M** với mọi tập S, lấy **modulo 998244353**.
- **Kết quả cần in:** Tổng trên với mỗi test.

Dịch đề bài:

Bài C. Truyền Gói Tin

Bạn được cho một **cây mạng** gồm các router và các kênh truyền. Mỗi đỉnh là một **router**, mỗi cạnh là một **kênh truyền** kết nối hai router. Các kênh này dùng để

truyền các **gói tin** giữa các router liền kề theo giao thức **TCP/IP**. Mỗi cạnh có nhãn t_i biểu thị **thời gian truyền** một gói tin giữa hai router u_i và v_i.

Mỗi router có đủ bộ nhớ để lưu trữ bất kỳ số lượng gói tin nào trong thời gian không giới hạn và có thể **truyền nhiều gói tin cùng lúc nhưng qua các kênh khác nhau**. Điều này có nghĩa là **một kênh chỉ truyền được một gói tin tại một thời điểm**.

Nhiệm vụ của bạn là đồng thời truyền **hai gói tin**:

- Một gói từ router s_1 đến router d_1.
- Một gói từ router s_2 đến router d_2.

Hãy tìm **chiến lược truyền tối ưu** để **giảm thiểu thời gian tối đa** trong việc truyền cả hai gói tin này (tức là **tối thiểu hóa thời gian của gói tin đến chậm nhất**).

Input:

- Dòng đầu là số nguyên T ($1 \leq T \leq 1000$): số lượng test.
- Mỗi test gồm:
  - Hai số nguyên N và Q ($1 \leq N, Q \leq 10^5$): số router và số truy vấn.
  - $N-1$ dòng tiếp theo, mỗi dòng gồm ba số nguyên u_i, v_i, t_i — cạnh nối router u_i và v_i, mất t_i thời gian để truyền.
  - Q dòng tiếp theo, mỗi dòng gồm bốn số nguyên s_1, d_1, s_2, d_2 — nguồn và đích của hai gói tin.

Ràng buộc:
Tổng tất cả các N trong các test không vượt quá $500,000$, tổng tất cả các Q không vượt quá $500,000$.

Output:

- Với mỗi truy vấn, in ra thời gian tối thiểu cần thiết để hoàn tất việc truyền cả hai gói tin (gói tin đến chậm nhất).

Ví dụ:

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

Output:
7
15
24
1
3
```

Tóm tắt:

- **Cho:** Một cây mạng N router với các kênh truyền có trọng số (thời gian truyền).
- **Nhiệm vụ:** Đồng thời truyền hai gói tin từ:
  - $s_1 \to d_1$
  - $s_2 \to d_2$
- **Yêu cầu:** Tìm cách truyền sao cho **giảm thiểu thời gian tối đa** cần để cả hai gói tin đến đích (tức là thời gian của gói chậm nhất).
- **Kết quả cần in:** Với mỗi truy vấn, in ra **thời gian tối thiểu** để hoàn thành việc truyền cả hai gói tin.

Dịch đề bài:

Bài D. Qwiksort

Bạn được cho một mảng gồm các số nguyên từ **1 đến** $2n$, mỗi số xuất hiện đúng **một lần**. Mục tiêu của bạn là **sắp xếp mảng này theo thứ tự tăng dần** bằng cách sử dụng một thao tác đặc biệt gọi là **"Qwiksort"**.

- Trong **mỗi lần Qwiksort**, bạn có thể chọn **một đoạn liên tiếp** gồm đúng **n phần tử** trong mảng và **sắp xếp chúng tăng dần tại chỗ**.

Ví dụ:

- Nếu $n = 3$ và mảng là $[3, 2, 4, 1, 6, 5]$,
- Bạn chọn đoạn từ vị trí **2 đến 4** ($[2, 4, 1]$),
- Sau khi Qwiksort, mảng trở thành: $[3, 1, 2, 4, 6, 5]$.

Bạn có thể áp dụng Qwiksort **nhiều lần** (tối đa **10 lần** cho mỗi mảng).
Yêu cầu: In ra các thao tác Qwiksort được sử dụng để sắp xếp mảng.

- **Bạn không cần tối ưu số lần thao tác**.
- Đề bài đảm bảo rằng **có thể sắp xếp mảng với tối đa 10 thao tác**.

Input:

- Dòng đầu là số nguyên T ($1 \leq T \leq 40,000$): số lượng test.
- Mỗi test gồm:
  - Một số nguyên n ($2 \leq n \leq 1,000$).
  - Một dòng gồm $2n$ số nguyên — các phần tử của mảng cần sắp xếp.

Ràng buộc: Tổng tất cả các n trong các test **không vượt quá** $2 \cdot 10^5$.

Output:

- Với mỗi test:
  - In số thao tác Qwiksort k (số lần áp dụng Qwiksort).
  - Tiếp theo k dòng, mỗi dòng gồm hai số l và r — vị trí bắt đầu và kết thúc (1-based) của đoạn được Qwiksort (đảm bảo $r - l + 1 = n$).

Ví dụ:

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

Output:
7
1 5
3 7
5 9
6 10
1 5
3 7
1 5
6
1 2
2 3
3 4
1 2
2 3
1 2
```

Tóm tắt:

- **Cho:** Mảng $2n$ phần tử, gồm các số từ **1 đến** $2n$, mỗi số xuất hiện một lần.
- **Mục tiêu:** Sắp xếp mảng tăng dần bằng thao tác **Qwiksort**.
- **Qwiksort:** Chọn đoạn liên tiếp dài **n** phần tử, sắp xếp tại chỗ đoạn đó.
- **Giới hạn:** Tối đa **10 lần Qwiksort** cho mỗi test.
- **Kết quả cần in:**

- Số lần Qwiksort k.
- Danh sách các đoạn được Qwiksort (chỉ cần sắp xếp được, không yêu cầu tối ưu số lần).

Dịch đề bài:

Bài E. Du hành trên lưới

Bạn được cho một **lưới** $N \times M$. Nhiệm vụ của bạn là di chuyển từ **ô (1, 1)** đến **ô (N, M)**.

- Bạn có thể di chuyển đến **8 ô lân cận** (bao gồm ngang, dọc, chéo).

Trên lưới có **mìn** ở một số ô. Khi một mìn **nổ**, nó sẽ tiêu diệt:

- Bất kỳ ai đứng **trên ô chứa mìn** đó.
- Hoặc đứng trên **4 ô lân cận theo hàng và cột** (trên, dưới, trái, phải).

Lưu ý:

- Bước vào **5 ô** này (mìn + 4 ô xung quanh) sẽ **kích nổ** mìn ngay lập tức.

Bạn có thể **chế tạo và sử dụng các thiết bị gỡ mìn (diffuser)** để vô hiệu hóa mìn.

- Có thể chế tạo **không giới hạn số diffuser**.
- **Mỗi diffuser có thể sử dụng nhiều lần**.
- **Chi phí chế tạo diffuser:** X.
- **Chi phí tái sử dụng diffuser:** Y.

Cách sử dụng diffuser:

1. **Chế tạo diffuser:**
    - Chỉ có thể **chế tạo khi đứng ở ô an toàn** (không có mìn và không bị đe dọa bởi mìn lân cận).
    - Có thể **đặt diffuser vào 1 trong 8 ô lân cận** có chứa mìn để **vô hiệu hóa mìn**.
2. **Tái sử dụng diffuser:**
    - Di chuyển đến ô chứa diffuser để **nhặt diffuser** (miễn phí).
    - Khi **mang diffuser**, bạn không thể chế tạo hoặc nhặt diffuser khác.
    - Trong khi **mang diffuser**, di chuyển qua mỗi ô sẽ mất thêm **chi phí riêng** của ô đó (được cho trước với mỗi ô).
    - Khi đến gần mìn (ô lân cận), bạn có thể **đặt diffuser** để vô hiệu hóa mìn, mất thêm chi phí **tái sử dụng** Y.

Nhiệm vụ:

- Tìm cách di chuyển từ **(1, 1)** đến **(N, M)** với **chi phí thấp nhất** (bao gồm cả chi phí di chuyển và gỡ mìn).
- **Đảm bảo:** Không có mìn nào đe dọa ô **(1, 1)** và **không có ô nào bị nhiều mìn đe dọa cùng lúc**.

Input:

- Dòng đầu tiên chứa số nguyên T (số test).

- Mỗi test gồm:
  - Bốn số nguyên N, M, X, Y: kích thước lưới, chi phí chế tạo diffuser và chi phí tái sử dụng diffuser.
  - N dòng tiếp theo: mô tả lưới ('.' là ô an toàn, '#' là ô có mìn).
  - N dòng tiếp theo: ma trận chi phí V_{ij} (chi phí di chuyển qua ô (i, j) khi mang diffuser).

Output:

- Với mỗi test, in ra **chi phí thấp nhất** để đến ô **(N, M)**.

Ví dụ:

```
Input:
2
3 4 10 5
....
.#..
...#
0 0 0 0
0 1 2 3
0 0 0 0
3 4 10 10
..#.
....
#..#
0 0 1 0
0 0 0 0
1 0 0 0

Output:
16
20
```

Tóm tắt:

- **Cho:** Lưới $N \times M$, một số ô có **mìn**, có thể chế tạo hoặc tái sử dụng **diffuser** để vô hiệu hóa mìn.
- **Di chuyển:** 8 hướng, nhưng cần tránh các ô có mìn và khu vực bị đe dọa (mìn + 4 ô lân cận).
- **Chi phí:**
  - **Chế tạo diffuser:** X.
  - **Tái sử dụng diffuser:** Y + **chi phí di chuyển qua ô** khi mang diffuser.
- **Mục tiêu:** Tìm **chi phí thấp nhất** để đi từ ô **(1,1)** đến ô **(N,M)** an toàn.

Dịch đề bài:

Bài F. Lại Một Tập Giao Thoa Khác

Pebae có một mảng số nguyên $a_1, a_2, ..., a_n$.
Nhiệm vụ của bạn là:

1. **Tìm giá trị lớn nhất** của biểu thức:

   $$\gcd(a_i \oplus a_j, a_i \& a_j)$$

   trong tất cả các cặp chỉ số $1 \leq i, j \leq n$.

   - $\oplus$: Phép **XOR** bitwise giữa hai số.
   - $\&$: Phép **AND** bitwise giữa hai số.

- $\gcd(x, y)$: **Ước chung lớn nhất** của hai số x và y. Với $gcd(x, 0) = x$ khi $x \geq 0$.

2. **Đếm số cặp** (i, j) đạt được giá trị lớn nhất này.

Input:

- Dòng đầu tiên chứa số nguyên t ($1 \leq t \leq 10^5$): số lượng test.
- Mỗi test gồm:
  - Một số nguyên n ($1 \leq n \leq 2^{19}$): độ dài mảng.
  - Một dòng chứa n số nguyên $a_1, a_2, ..., a_n$ ($0 \leq a_i < 16 \cdot n$).

Ràng buộc: Tổng tất cả các n trong các test không vượt quá 2^{20}.

Output:

- Với mỗi test, in ra hai số nguyên:
  - Giá trị lớn nhất của $\gcd(a_i \oplus a_j, a_i \& a_j)$.
  - Số cặp (i, j) đạt giá trị này.

Ví dụ:

```
Input:
4
4
1 9 1 9
```

```
1
0
3
0 0 1
7
12 2 3 0 110 1 69

Output:
9 4
0 1
1 5
111 2
```

Tóm tắt:

- **Cho:** Mảng số nguyên a.
- **Yêu cầu:**
  1. Tìm **giá trị lớn nhất** của $\gcd(a_i \oplus a_j, a_i \& a_j)$ với mọi cặp (i, j).
  2. Đếm số **cặp** (i, j) đạt được giá trị này.
- **Kết quả cần in:**
  - Giá trị lớn nhất của $\gcd$.
  - Số cặp (i, j) đạt giá trị này.

Dịch đề bài:

Bài G. Đường chân trời tuyệt đẹp

Thành phố bạn sống có rất nhiều tòa nhà cao tầng, mỗi tòa nhà có chiều cao **không trùng nhau**. Một ngày, khi leo lên núi nhìn xuống thành phố, bạn nhận ra hai điều kỳ

lạ:

1. Tất cả các tòa nhà **nằm thẳng hàng**.
2. Các tòa nhà được chia thành **nhiều nhóm** tạo thành **dạng kim tự tháp**:
    - Mỗi nhóm gồm $2k + 1$ tòa nhà (với k là số nguyên dương).
    - Trong nhóm, k tòa nhà đầu tiên **tăng dần độ cao**, tiếp theo là tòa cao nhất, sau đó k tòa nhà **giảm dần độ cao**.

Giữa các nhóm có **khoảng trống nhỏ** (nhưng **không có khoảng trống** giữa các tòa nhà trong cùng một nhóm).

Vấn đề:

Khi bạn đến thành phố khác, các tòa nhà **cũng nằm thẳng hàng**, **không có khoảng trống** giữa các tòa nhà, nhưng **không được chia nhóm theo kiểu kim tự tháp**. Bạn có **2 cỗ máy** để chỉnh sửa:

1. **Group Maker 3000**:
    - Chia các tòa nhà thành **nhiều nhóm** (mỗi nhóm phải có kích thước **lẻ và lớn hơn 2**).
    - **Miễn phí** sử dụng.
2. **Building Swapper 2000**:
    - **Hoán đổi hai tòa nhà liền kề** trong cùng một nhóm.
    - Mỗi lần hoán đổi **tốn 1 đơn vị nhiên liệu**.

Quy trình:

1. **Dùng Group Maker 3000** để chia các tòa nhà thành các nhóm thỏa mãn yêu cầu kích thước (nhưng **chưa cần đúng dạng kim tự tháp**).
2. **Dùng Building Swapper 2000** để **sắp xếp từng nhóm** thành dạng **kim tự tháp** (tăng dần, đỉnh cao nhất, giảm dần).

Nhiệm vụ:

- Tính **tổng nhiên liệu tối thiểu** cần dùng để hoán đổi các tòa nhà và tạo thành các **nhóm kim tự tháp** hoàn chỉnh.

Input:

- Dòng đầu là số nguyên T — số lượng test.
- Mỗi test gồm:
  - Một số nguyên N ($3 \leq N \leq 1000, N \neq 4$): số lượng tòa nhà.
  - Một dòng gồm N số nguyên — chiều cao các tòa nhà (không trùng nhau).

Ràng buộc: Tổng tất cả các N không vượt quá **10,000**.

Output:

- Với mỗi test, in ra **tổng nhiên liệu tối thiểu** cần thiết.

Ví dụ:

```
Input:
4
3
1 3 2
6
1 2 4 8 16 32
5
1 2 3 4 5
7
6 4 2 1 3 5 7

Output:
0
2
3
9
```

Tóm tắt:

- **Cho:** Dãy tòa nhà nằm thẳng hàng với chiều cao khác nhau.
- **Yêu cầu:**
  1. **Chia tòa nhà thành các nhóm** có kích thước **lẻ và lớn hơn 2**.
  2. **Sắp xếp từng nhóm** thành dạng **kim tự tháp** (tăng dần, đỉnh cao nhất, giảm dần).
- **Mục tiêu:** Tính **nhiên liệu tối thiểu** để hoàn thành việc hoán đổi (mỗi lần hoán đổi hai tòa nhà liền kề tốn 1 đơn vị nhiên liệu).

Dịch đề bài:

Bài H. Các đỉnh có thể đến được không?

Bạn được cho một **đồ thị có hướng, không chu trình** gồm N đỉnh (đánh số từ 1 đến N) và M cạnh có hướng.

Nhiệm vụ của bạn là **trả lời Q truy vấn** về việc **một đỉnh có thể đến được một đỉnh khác hay không**.

Với mỗi truy vấn gồm hai đỉnh U và V:

- **Nếu V có thể đến được từ U** theo các cạnh có sẵn, in ra **0** (không cần xây thêm cạnh).
- **Nếu V không thể đến được từ U**, bạn được **phép xây thêm một cạnh tạm thời** giữa **bất kỳ hai đỉnh** trong đồ thị để làm cho V có thể đến được từ U.
    - Sau khi trả lời xong, **cạnh tạm thời bị xóa bỏ**.

Chi phí xây cạnh giữa hai đỉnh X và Y là $|X - Y|$ (hiệu tuyệt đối giữa nhãn hai đỉnh).

Bạn cần **tính chi phí nhỏ nhất** để làm cho V có thể đến được từ U (nếu cần xây cạnh).

Input:

- Dòng đầu là số nguyên T ($1 \leq T \leq 100$): số lượng test.
- Mỗi test gồm:
    - Hai số nguyên N, M ($1 \leq N \leq 10^4$, $1 \leq M \leq 8 \cdot 10^4$): số đỉnh và số cạnh.
    - M dòng tiếp theo, mỗi dòng gồm hai số nguyên U, V — cạnh có hướng từ U đến V.

- Một số nguyên Q ($1 \leq Q \leq 10^6$): số lượng truy vấn.
- Q dòng tiếp theo, mỗi dòng gồm hai số nguyên U, V — truy vấn về khả năng đi từ đỉnh U đến đỉnh V.

Ràng buộc:
Tổng các giá trị N, M, Q trên tất cả các test **không vượt quá** 10^5, 10^5, 10^6.

Output:

- Với mỗi truy vấn, in ra một số nguyên:
  - **0** nếu V đã đến được từ U.
  - **Chi phí tối thiểu** để làm cho V có thể đến được từ U nếu không (xây thêm một cạnh).

Ví dụ:

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

Output:
1
1
```

Tóm tắt:

- **Cho:** Đồ thị có hướng, không chu trình (DAG), với N đỉnh, M cạnh, và Q truy vấn.
- **Mỗi truy vấn:** Kiểm tra đỉnh V có thể đến được từ đỉnh U không.
  - Nếu **được**, in **0**.
  - Nếu **không**, **xây thêm 1 cạnh tạm thời** giữa **bất kỳ hai đỉnh** để tạo đường đi, in ra **chi phí tối thiểu** (theo $|X - Y|$).
- **Mục tiêu:** Trả lời tất cả truy vấn với chi phí tối thiểu khi cần xây cạnh.

Dịch đề bài:

Bài I. Đội ngũ không hạnh phúc

Bạn quản lý một đội gồm N người (đánh số từ **1 đến** N). Trong đội, **mỗi người đều có quan điểm** về hiệu suất làm việc của những người khác là **mạnh hơn** hay **yếu hơn** mình.

Cuối năm, bạn cần **xếp hạng** mọi người để chia thưởng. Một **xếp hạng P** là một **hoán vị** của các số từ **1 đến** N:

- P_1: Người đứng hạng cao nhất.
- P_N: Người đứng hạng thấp nhất.

Khi xếp hạng xong, **mọi người đều thấy được** thứ hạng của nhau.

Chỉ số không hài lòng (unhappiness score):

- Với mỗi người, **chỉ số không hài lòng** là số lượng người **xếp hạng cao hơn họ**, nhưng người đó **coi là yếu hơn** mình.

Tổng chỉ số không hài lòng của đội:

- **Tính chỉ số không hài lòng của từng người**.
- Sắp xếp dãy chỉ số này **tăng dần**.
- **Lấy tổng của K chỉ số lớn nhất**.

Ví dụ:

- Chỉ số không hài lòng của đội: $[0, 1, 2, 2, 3]$.
- Với $K = 2$: Tổng = $3 + 2 = 5$.
- Với $K = 3$: Tổng = $3 + 2 + 2 = 7$.

Nhiệm vụ:

- Với một xếp hạng **ngẫu nhiên** (chọn hoán vị ngẫu nhiên), tính **kỳ vọng** (expected value) của **tổng chỉ số không hài lòng** (lấy K chỉ số lớn nhất).
- Kết quả phải tính dưới dạng **phân số tối giản** $\frac{P}{Q}$.
- Cuối cùng, in ra:

$$(P \cdot Q^{-1}) \mod 998244353$$

(Q^{-1} là **nghịch đảo modular** của Q).

Input:

- Dòng đầu: số nguyên T ($1 \leq T \leq 50$): số lượng test.
- Mỗi test gồm:
  - Hai số nguyên N, K ($1 \leq K \leq N \leq 16$).
  - N dòng tiếp theo: mỗi dòng N ký tự ('S', 'W', 'X'):
    - **'S'**: Người i nghĩ người j **mạnh hơn**.
    - **'W'**: Người i nghĩ người j **yếu hơn**.
    - **'X'**: $i = j$.

Ràng buộc:

- Chỉ có **tối đa 3 test** có $N > 12$.

Output:

- Với mỗi test, in ra **kỳ vọng tổng chỉ số không hài lòng** (dưới dạng $(P \cdot Q^{-1}) \mod 998244353$).

Ví dụ:

```
Input:
3
3 2
XSW
SXW
WSX
4 3
XSWS
```

```
WXSW
SSXS
SWWX
1 1
X

Output:
499122178
499122179
0
```

Tóm tắt:

- **Cho:** Đội N người, mỗi người có quan điểm ai mạnh/yếu hơn mình.
- **Yêu cầu:**
  - Với **một xếp hạng ngẫu nhiên**:
    - Tính **kỳ vọng tổng chỉ số không hài lòng** (lấy K người **không hài lòng nhất**).
- **Kết quả:** In ra **kỳ vọng** dưới dạng **modulo 998244353**.

Dịch đề bài:

Bài J. Hand Cricket

Alice và Bob chơi một trò chơi trên một **mảng số nguyên** A có độ dài N.

- **Alice** chọn **một chỉ số bí mật** i.
- **Bob** chọn **một chỉ số bí mật** j.

- Nếu $i \neq j$, Alice nhận được A_i **điểm**.
- Nếu $i = j$, Alice nhận **0 điểm**.

Mục tiêu:

- Alice muốn **tối đa hóa số điểm**.
- Bob muốn **tối thiểu hóa số điểm**.

Chiến lược:

- Mỗi người chọn chỉ số theo **xác suất phân phối** (probability distribution):
  - Với dãy con $[L, R]$, mỗi người có phân phối xác suất:
    - $P_L, P_{L+1}, ..., P_R$ sao cho:

$$P_L + P_{L+1} + ... + P_R = 1, \quad 0 \leq P_i \leq 1$$

- **Chiến lược của mỗi người phải tối ưu**, không bị khai thác ngay cả khi bị lộ.

Thay đổi của Alice:

Trước khi chơi, **Alice có thể thực hiện tối đa K lần tăng giá trị**:

- **Mỗi lần tăng:** Chọn một phần tử bất kỳ trong đoạn $[L, R]$, tăng giá trị đó thêm **1**.
- Có thể tăng **nhiều lần trên cùng một phần tử**.
- **Việc tăng giá trị chỉ áp dụng cho truy vấn hiện tại**, **không ảnh hưởng đến các truy vấn sau**.

Nhiệm vụ:

Với mỗi truy vấn $[L, R, K]$:

- **Tính số điểm kỳ vọng (expected value)** mà Alice nhận được khi cả hai chơi tối ưu.
- Kết quả được biểu diễn dưới dạng phân số **tối giản** $\frac{X}{Y}$.
- Cuối cùng, in ra:

$$(X \cdot Y^{-1}) \mod 998244353$$

(Y^{-1} là **nghịch đảo modular** của Y).

Input:

- Dòng đầu tiên: số nguyên N ($1 \leq N \leq 2 \cdot 10^5$).
- Dòng thứ hai: N số nguyên $A_1, A_2, ..., A_N$ ($1 \leq A_i \leq 10^8$).
- Dòng thứ ba: số nguyên Q ($1 \leq Q \leq 10^5$): số lượng truy vấn.
- Tiếp theo Q dòng: mỗi dòng gồm ba số nguyên L_i, R_i, K_i ($1 \leq L_i \leq R_i \leq N$, $0 \leq K_i \leq 10^8$).

Output:

- Với mỗi truy vấn, in ra kết quả **kỳ vọng số điểm của Alice** dưới dạng:

$$(X \cdot Y^{-1}) \mod 998244353$$

Ví dụ:

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

Output:
2
0
598946613
```

Tóm tắt:

- **Cho:** Mảng A và các truy vấn $[L, R, K]$.
- **Mỗi truy vấn:**
  - Alice và Bob chọn chỉ số trong đoạn $[L, R]$ theo phân phối xác suất tối ưu.
  - **Alice có thể tăng tối đa K lần giá trị các phần tử** trước khi chơi.
- **Yêu cầu:** Tính **kỳ vọng số điểm Alice nhận được**, in ra kết quả dưới dạng:

$$(X \cdot Y^{-1}) \mod 998244353$$

Dịch đề bài:

Bài K. Cuộc Xâm Lăng Hòn Đảo

Có hai hòn đảo nằm cạnh nhau trên biển **Little Sea**:

- **Big Island** (Đảo Lớn)
- **Tiny Island** (Đảo Nhỏ)

Nhà vua của **Big Island** muốn xâm chiếm **Tiny Island** để **thỏa mãn lòng tự tôn** của mình.
Tuy nhiên, **gió mạnh** thổi trên biển, làm cho **tàu thuyền nhẹ và chậm**. Do đó, việc đến **Tiny Island** có thể **mất nhiều thời gian** hoặc thậm chí là **không thể**.

Thêm vào đó, **thời tiết biến đổi thất thường** khiến **hướng gió khó dự đoán**. Vì vậy, nhà vua muốn lên kế hoạch trước và tìm **thời gian tối thiểu** để đến **Tiny Island** từ **Big Island** với các điều kiện gió khác nhau.

- **Big Island** và **Tiny Island** được mô tả dưới dạng hai **đa giác lồi** với lần lượt n và m đỉnh (**không giao nhau**).
- Nhà vua sẽ đưa ra q truy vấn.
- Trong mỗi truy vấn:
  - Một **vector gió** $\vec{w} = (w_x, w_y)$.
  - Tốc độ tối đa của tàu thuyền s.

Tàu thuyền có thể:

- Khởi hành từ **bất kỳ điểm nào** trên **Big Island**.
- Chọn **bất kỳ hướng di chuyển nào**, nhưng tốc độ riêng $\vec{v}$ phải thỏa mãn:

$$|\vec{v}| \leq s$$

- **Tốc độ thực tế của tàu** là:

$$\vec{v} + \vec{w}$$

Nhiệm vụ:

- Với mỗi truy vấn:
  - Xác định **tàu có thể đến được Tiny Island hay không**.
  - Nếu **không thể**, in ra **-1**.
  - Nếu **có thể**, in ra **thời gian tối thiểu** để đến **bất kỳ điểm nào** trên **Tiny Island**.

Input:

- Dòng đầu tiên: số nguyên T ($1 \leq T \leq 1111$): số lượng test.
- Với mỗi test:
  - Hai số nguyên n, m ($3 \leq n, m \leq 10^5$): số đỉnh của **Big Island** và **Tiny Island**.
  - Tiếp theo n dòng: tọa độ các đỉnh của **Big Island** (theo thứ tự **ngược chiều kim đồng hồ**).
  - Tiếp theo m dòng: tọa độ các đỉnh của **Tiny Island** (theo thứ tự **ngược chiều kim đồng hồ**).
  - Một số nguyên q ($1 \leq q \leq 2 \cdot 10^4$): số lượng truy vấn.
  - Tiếp theo q dòng: mỗi dòng gồm ba số nguyên w_x, w_y, s (vector gió và tốc độ tối đa của tàu).

Ràng buộc:

- Tổng tất cả các $n + m$ không vượt quá **200,000**.
- Tổng tất cả các q không vượt quá **20,000** trên tất cả các test.
- Đảm bảo rằng, nếu có thể đến được Tiny Island, thì thời gian tối thiểu luôn **≤ 10^9**.

Output:

- Với mỗi truy vấn:
  - In **-1** nếu không thể đến Tiny Island.
  - Nếu **có thể**, in **thời gian tối thiểu** (dưới dạng **số thực**).
  - Đáp án đúng nếu **sai số tuyệt đối hoặc tương đối** $\leq 10^{-6}$.

Ví dụ:

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

Output:
0.5857860308
-1
```

Tóm tắt:

- **Cho:** Hai đa giác lồi (Big Island và Tiny Island), mô tả hai hòn đảo.
- **Mỗi truy vấn:**
  - Một vector gió $\vec{w}$.
  - Tốc độ tối đa của tàu s.
- **Nhiệm vụ:**
  - Xác định **tàu có thể đến Tiny Island không**.
  - Nếu **có**, in **thời gian tối thiểu** để đến. Nếu **không**, in **-1**.

Dịch đề bài:

Bài L. Chú Bob và tổng XOR

Chú Bob, thị trưởng của **Bytelands**, là một người đam mê giải toán. Vì vậy, ông đã thuê hẳn một đội ngũ tạo ra các bài toán mới mỗi ngày cho mình giải.

Tuy nhiên, không có bài toán nào làm khó được Chú Bob — ông giải quyết chúng chỉ trong vài phút. Dần dần, ông cảm thấy **chán nản** vì chúng quá dễ.
Chú Bob đã **đe dọa** đội ngũ rằng nếu họ không nghĩ ra một bài toán thú vị hơn, họ sẽ **bị đuổi việc**.

Sau nhiều giờ suy nghĩ, đội ngũ đã tạo ra bài toán sau:

Cho hai mảng số nguyên A và B:

- Tìm số lượng **tập con không rỗng** của **các vị trí trong mảng** A sao cho:

- **Tổng XOR** của các giá trị tại những vị trí được chọn trong A **không chứa bất kỳ submask nào thuộc mảng** B.

Giải thích thêm:

- **Tổng XOR** của một tập con các vị trí $\{p_1, p_2, ..., p_k\}$ trong A được định nghĩa là:

  $$S = A[p_1] \oplus A[p_2] \oplus ... \oplus A[p_k]$$

  ($\oplus$ là phép **XOR bitwise**).

- **Submask** của một số n là **bất kỳ số nào** m sao cho **tất cả các bit 1 của** m đều là **bit 1 của** n.

- Nói cách khác:
  - Với **tổng XOR** S của tập con đã chọn:
  - **Không có submask nào của** S nằm trong mảng B.

Nhiệm vụ:

- **Đếm số tập con không rỗng** của **các vị trí trong** A thỏa mãn điều kiện trên.
- Do kết quả có thể rất lớn, **in ra kết quả modulo** $10^9 + 7$.

Input:

- Dòng đầu tiên chứa số nguyên T ($1 \leq T \leq 100$): số lượng test.
- Với mỗi test:

- Dòng đầu tiên gồm hai số nguyên N, K ($1 \leq N \leq 10^5$, $1 \leq K \leq 10$):
  - N: độ dài mảng A.
  - K: độ dài mảng B.
- Dòng tiếp theo chứa N số nguyên $a_1, a_2, ..., a_n$ ($0 \leq a_i \leq 2^{31} - 1$): mảng A.
- Dòng tiếp theo chứa K số nguyên $b_1, b_2, ..., b_k$ ($0 \leq b_i \leq 2^{31} - 1$): mảng B.

Ràng buộc:
Tổng tất cả các N trên mọi test **không vượt quá** 10^5.

Output:

- Với mỗi test, in ra kết quả **modulo** $10^9 + 7$ trên một dòng.

Ví dụ:

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

Output:
0
```

```
3
1
```

Tóm tắt:

- **Cho:** Hai mảng:
  - A (có N phần tử).
  - B (có K phần tử).
- **Tìm:** Số lượng **tập con không rỗng** của các vị trí trong A, sao cho:
  - **Tổng XOR** của các phần tử được chọn **không chứa submask nào thuộc** B.
- **In ra kết quả modulo** $10^9 + 7$.

Dịch đề bài:

Bài M. Lật Cây (Tree Flip)

Alu được cho một **cây có gốc** (rooted tree), mỗi đỉnh chứa một giá trị **0** hoặc **1**.
Alu có thể thực hiện **phép lật (flip)** ở một số đỉnh.

- Khi **lật một đỉnh**, giá trị của **chính đỉnh đó** và **tất cả các con trực tiếp** (children) của nó sẽ bị **đảo ngược** (flip):
  - **0 → 1**, **1 → 0**.
  - **Lưu ý:** Chỉ **các con trực tiếp** bị lật, **không phải tất cả các hậu duệ** (descendants).

Alu là người thông minh, nên anh ta sẽ thực hiện **số lần lật ít nhất** để **tất cả các đỉnh đều có giá trị bằng 0**.

Nhưng để **tăng phần thú vị**, có thêm **Begun**.
Begun có thể **cập nhật cây** theo hai cách:

- **Cập nhật 1:** Begun **lật giá trị của một đỉnh** (chỉ lật chính đỉnh đó, **không lật các con**).
- **Cập nhật 2:** Begun **chọn lại gốc cây** (đặt một đỉnh bất kỳ làm **gốc mới** của cây).

Nhiệm vụ:

- Sau mỗi lần **Begun cập nhật**, hãy tính **số lần lật tối thiểu** mà **Alu cần** để làm **mọi đỉnh** trong cây đều có **giá trị bằng 0**.
- **Lưu ý:** Alu thực hiện trên **một bản sao** của cây sau cập nhật, **không làm thay đổi cây của Begun**.

Input:

- Dòng đầu tiên: số nguyên T ($1 \leq T \leq 10^4$) — số lượng test.
- Mỗi test gồm:
  - Hai số nguyên n, q ($1 \leq n, q \leq 10^5$): số đỉnh và số lần cập nhật.
  - Một dòng gồm n số nguyên — giá trị ban đầu của các đỉnh (0 hoặc 1).
  - $n - 1$ dòng tiếp theo: mỗi dòng gồm hai số nguyên u, v — cạnh nối hai đỉnh.
  - q dòng tiếp theo: mỗi dòng gồm hai số nguyên $type, x$:
    - **type = 1: Lật giá trị đỉnh** x (chỉ lật chính đỉnh đó).

    - **type = 2: Chọn đỉnh x làm gốc cây mới**.

- **Giới hạn:**
    - Tổng tất cả các giá trị n trên mọi test **không vượt quá** 10^5.
    - Tổng tất cả các giá trị q trên mọi test **không vượt quá** 10^5.

Output:

- Với mỗi lần cập nhật, in ra **số lần lật tối thiểu** Alu cần để **tất cả các đỉnh đều có giá trị bằng 0**.

Ví dụ:

```
Input:
1
3 3
0 0 1
1 2
3 1
1 1
2 2
1 1

Output:
2
1
1
```

Giải thích ví dụ:

- Cây ban đầu:

```
1(0)
├── 2(0)
└── 3(1)
```

- **Cập nhật 1: Lật giá trị đỉnh 1** → Cây:

```
1(1)
├── 2(0)
└── 3(1)
```

- **Alu cần 2 lần lật:**
  - Lật **đỉnh 2** → 2(1).
  - Lật **đỉnh 1** → 1(0), 2(0), 3(0).
- **Cập nhật 2: Đặt đỉnh 2 làm gốc** → Cây:

```
2(0)
├── 1(1)
└── 3(1)
```

- **Alu chỉ cần 1 lần lật ở đỉnh 1** để tất cả về 0.
- **Cập nhật 3: Lật giá trị đỉnh 1** → Cây:

```
2(0)
├── 1(0)
└── 3(1)
```

- **Alu chỉ cần 1 lần lật ở đỉnh 3** để tất cả về 0.

Tóm tắt:

- **Cho:** Một cây gốc có các đỉnh mang giá trị 0 hoặc 1.
- **Có 2 loại cập nhật:**
  - **Lật giá trị đỉnh**.
  - **Đổi gốc cây**.
- **Yêu cầu:** Sau mỗi cập nhật, tính **số lần lật tối thiểu** để làm **tất cả các đỉnh về 0**.