

universal 3 stage 25 : Poland

Created: 5/1/2025 21:22

Updated: 5/1/2025 21:28

Exported: 5/1/2025 21:27

Bài A: Từ viết tắt (Acronym)

 **Giới hạn thời gian:** 2 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Aga rất thích các bí mật và trò chơi chữ. Ngày mai là **ngày đặt tên** (tương tự sinh nhật, gọi là *imieniny* trong tiếng Ba Lan) của cô ấy. Bạn cô, Ania, đang trang trí một tấm thiệp chúc mừng và muốn viết một câu mà **chữ cái đầu tiên của mỗi từ ghép lại tạo thành một từ đặc biệt**. Từ này được gọi là **từ viết tắt (acronym)**.

Ania đã nghĩ ra một số ví dụ sau:

- **WONDERFUL IMMENSE SUPERB HAPPINESS** tạo thành từ viết tắt **WISH**.
- **LIVE LAUGH LOVE** tạo thành từ viết tắt **LLL**.
- **AKRONIM NA IMIENINY AGI** tạo thành từ viết tắt **ANIA**.
- **BABY AND BABY YETI** tạo thành từ viết tắt **BABY**.

Ví dụ cuối cùng cho thấy rằng:

- ✓ Từ trong câu **có thể lặp lại**,
- ✓ Và từ viết tắt cũng **có thể xuất hiện trong câu**.

Giờ Ania muốn chọn ra một từ viết tắt phù hợp, và nhờ bạn giúp. Bạn được cung cấp một **từ điển gồm n từ khác nhau**. Nhiệm vụ là kiểm tra xem **có thể tạo một câu nào đó từ các từ trong từ điển**, sao cho **chữ cái đầu tiên của mỗi từ trong câu ghép lại thành một từ cũng có trong từ điển hay không?**

Input

- Dòng đầu tiên: số nguyên n ($1 \leq n \leq 200$) — số từ trong từ điển.
- n dòng tiếp theo: mỗi dòng chứa một từ viết hoa (A-Z), độ dài từ 2 đến 8 ký tự. Các từ là **khác nhau**.

Output

- Nếu **không thể** tạo câu như yêu cầu → in ra `-1``.
- Nếu **có thể** tạo câu → in ra 2 dòng:
 - Dòng 1: số từ trong câu.
 - Dòng 2: câu được tạo (các từ cách nhau bởi dấu cách).

👉 Nếu có nhiều đáp án đúng, in ra **bất kỳ** một đáp án.

Ví dụ

Input

```
5
ANIA
IMIENINY
AGI
AKRONIM
NA
```

Output

4

AKRONIM NA IMIENINY AGI

Input

1

XX

Output

2

XX XX

Tóm tắt đề bài

Cho một tập từ điển gồm các từ in hoa. Hãy kiểm tra xem có tồn tại một câu (gồm các từ có thể lặp lại, mỗi từ từ từ từ điển) sao cho **chữ cái đầu tiên của mỗi từ trong câu ghép lại thành một từ cũng nằm trong từ điển**.

Nếu có → in ra câu.

Nếu không → in ``-1``.

Bài B: Hành lý (Baggage)

 **Giới hạn thời gian:** 3 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Bartek là một người thích du lịch và mỗi năm đều khám phá các quốc gia và châu lục mới. Anh ta tiết kiệm chi phí bằng cách:

- Ở nhờ khách sạn do các cuộc thi lập trình địa phương tài trợ.
- Nhờ bạn bè giữ giúp hành lý (hai chiếc vali).

Có **n thành phố** (đánh số từ 1 đến n) và **m chuyến bay một chiều**. Mỗi chuyến bay được mô tả bởi 4 số:

- a : thành phố xuất phát
- b : thành phố đến
- c : chi phí bay
- d : số lượng vali tối đa có thể mang theo trên chuyến bay ($0 \leq d \leq 2$)

Bartek có thể dùng một chuyến bay **nhiều lần**. Chi phí **không phụ thuộc vào số vali mang theo**.

Vali **không thể tự di chuyển** mà **phải đi cùng người hoặc được để lại tại thành phố**.

Tại **mỗi thành phố**, bạn của Bartek **có thể giữ giúp bất kỳ số lượng vali nào, trong bất kỳ khoảng thời gian nào và bao nhiêu lần cũng được**.

Nhiệm vụ

Bartek muốn di chuyển từ **mọi cặp thành phố** $x \rightarrow y$ (có tổng cộng n^2 cặp), xuất phát từ thành phố x với **2 vali**, và đến thành phố y cũng với **2 vali**.

Hãy tính chi phí **nhỏ nhất** để thực hiện chuyến đi đó. Nếu không thể, in ra **-1**.

Input

- Dòng đầu: hai số nguyên n, m — số thành phố và số chuyến bay ($1 \leq n \leq 400$; $0 \leq m \leq 500000$)
 - Tiếp theo m dòng, mỗi dòng gồm 4 số nguyên:
 - a_i, b_i : chuyến bay từ thành phố a_i đến b_i
 - c_i : chi phí chuyến bay
 - d_i : số vali tối đa có thể mang theo (0, 1 hoặc 2)
-

Output

- In ra n dòng, mỗi dòng gồm n số nguyên.
 - Dòng thứ i , cột thứ j : chi phí **ít nhất** để đi từ thành phố i đến j mang **đầy đủ 2 vali**.
 - Nếu **không thể đi**, in `-1`.
-

Ví dụ

Input

```
5 7
1 2 500 2
2 3 100 1
3 5 20 2
5 4 5 1
4 2 1 0
3 4 40 2
5 4 77 1
```

Output

```
0 500 726 751 746
-1 0 226 251 246
-1 -1 0 40 20
-1 -1 -1 0 -1
-1 -1 -1 131 0
```

Giải thích 1 dòng ví dụ:

Từ thành phố **1 đến 3**:

- 1 → 2 (mang theo cả 2 vali)
- Gửi 1 vali ở thành phố 2
- 2 → 3 (mang 1 vali)
- Gửi vali ở thành phố 3
- 3 → 5 → 4 → 2 (trên đường nhặt lại vali đã gửi)
- 2 → 3 (nhặt vali còn lại)

Tổng chi phí:

$$500 + 100 + 20 + 5 + 1 + 100 = 726$$

👉 Tóm tắt đề bài

Bạn được cho một đồ thị có hướng gồm các chuyến bay một chiều, có giới hạn số vali mang theo.

Bartek bắt đầu từ thành phố x với **2 vali**, phải đến thành phố y cũng với **2 vali**.

Anh ta có thể gửi vali lại tại các thành phố, và quay lại lấy sau.

Tính **chi phí nhỏ nhất** cho mỗi cặp x, y .

Nếu không thể đi → in `-1`.

Bài C: Những chú bò (Cows)

 **Giới hạn thời gian:** 4 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Một **đồng cỏ dài** được chia thành **n đoạn**, đánh số liên tiếp từ 1 đến n .

Chiều cao cỏ ban đầu ở đoạn thứ i là h_i .

Có đúng **một con bò** đứng ở **mỗi đoạn**.

Giữa các đoạn có hàng rào, **bò không thể di chuyển**, nhưng **có thể vươn cổ sang đoạn liền kề** để ăn cỏ.

(Cụ thể, đoạn i và đoạn $i + 1$ là liền kề với nhau)

Cách ăn cỏ của bò (mỗi phút):

- Nếu đoạn của bò **vẫn còn cỏ** ($h_i > 0$) \rightarrow bò **luôn ăn ở đoạn của mình**.
- Nếu đoạn đó **không còn cỏ** \rightarrow bò chọn **một trong hai đoạn liền kề** (nếu còn cỏ) để ăn.
- Nếu đoạn mình và các đoạn liền kề đều **hết cỏ** \rightarrow bò **không làm gì**.

Nếu có **x con bò cùng ăn một đoạn**, thì đoạn đó **bị giảm đi x đơn vị cỏ trong 1 phút** (nhưng không âm, tối thiểu là 0):

$$h_i := \max(0, h_i - x)$$

Các con bò **phối hợp** với nhau một cách tối ưu để **ăn hết cỏ trên đồng cỏ trong thời gian ngắn nhất**.

Nhiệm vụ

Tính số phút **ít nhất** để tất cả cỏ trên đồng cỏ **bị ăn hết**.

Input

- Dòng đầu: số nguyên n ($1 \leq n \leq 200\,000$) — số đoạn của đồng cỏ.
 - Dòng thứ hai: n số nguyên h_1, h_2, \dots, h_n — chiều cao cỏ ban đầu ở mỗi đoạn ($0 \leq h_i \leq 10^9$)
Đảm bảo ít nhất **một đoạn có cỏ** ($h_i > 0$)
-

Output

- In ra một số nguyên: **số phút tối thiểu để ăn hết tất cả cỏ**.
-

Ví dụ

Input

```
5
5 4 0 4 6
```


Output

4

Giải thích

[5, 4, 0, 4, 6] - bò ở đoạn 3 ăn cỏ ở đoạn 4. Các bò khác ăn đoạn của mình.
[4, 3, 0, 2, 5]
[3, 2, 0, 0, 4] - bò 3 chuyển sang ăn đoạn 2; bò 4 ăn đoạn 5
[2, 0, 0, 0, 2]
[0, 0, 0, 0, 0]
→ Tổng cộng 4 phút.

Input

3
1 4 6

Output

5

👉 Tóm tắt đề bài

Cho một dãy gồm n đoạn cỏ, mỗi đoạn có chiều cao ban đầu. Mỗi đoạn có một con bò đứng sẵn. Mỗi phút, bò chọn ăn cỏ theo quy tắc:

- Ưu tiên ăn cỏ tại chỗ mình đứng.
- Nếu hết cỏ thì vươn cổ sang đoạn kế bên (nếu còn cỏ).
- Nhiều bò ăn cùng một đoạn → đoạn đó giảm chiều cao tương ứng.

Tính **số phút ít nhất** để **toàn bộ** **cỗ biến mất**.

Bài D: Phân số nhỏ dần (Diminishing Fractions)

 **Giới hạn thời gian:** 2 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Daria đã lập trình một chương trình tính toán phân số. Chương trình có thể tính toán biểu thức **gồm phép cộng và trừ giữa nhiều phân số** (tối đa n phân số), mỗi phân số có **tử số và mẫu số là số nguyên dương không vượt quá n** .

Daria muốn kiểm tra hiệu năng chương trình trong hai trường hợp:

- Khi kết quả là một **số rất lớn** → dễ kiểm tra bằng cách nhập $n/1 + n/1 + \dots + n/1$.
- Nhưng còn trường hợp **tổng là một số dương rất nhỏ** thì sao?

Nhiệm vụ

Cho trước t bộ test, mỗi test có một số nguyên n . Với mỗi n , bạn cần **tìm một biểu thức cộng/trừ giữa các phân số có tử và mẫu $\leq n$** sao cho:

- Giá trị biểu thức là **số dương nhỏ nhất có thể**.
- Biểu thức gồm từ **1 đến n** phân số, định dạng a/b ($1 \leq a, b \leq n$).



Quy tắc biểu thức

- Các phân số nối với nhau bằng dấu `+` hoặc `-`.
 - Dấu `-` **có thể** đứng trước phân số đầu tiên, nhưng **không được dùng dấu `+` đầu dòng**.
 - Không chứa dấu cách** hoặc ký tự dư thừa nào.
 - Nếu có nhiều biểu thức đúng → in ra **bất kỳ** một biểu thức.
-



Input

- Dòng đầu tiên: số nguyên t ($1 \leq t \leq 1000$) — số test.
 - t dòng tiếp theo: mỗi dòng là một số nguyên n ($1 \leq n \leq 50\,000$).
Tổng tất cả các $n \leq 4\,000\,000$.
-



Output

- Với mỗi test, in ra một dòng chứa biểu thức có giá trị **số dương nhỏ nhất có thể**, tạo từ các phân số hợp lệ.
-



Ví dụ

Input

```
2
3
```



Output

```
1/2-1/3
-3/6+1/4+2/5-5/6+6/4-4/5
```

Giải thích ví dụ

- **Test 1: $n = 3$**

Biểu thức $\frac{1}{2} - \frac{1}{3} = \frac{1}{6}$ là số dương nhỏ nhất có thể tạo được.

Các cách khác cũng cho kết quả giống hoặc tương đương như $-\frac{1}{3} + \frac{1}{2}$ hoặc $\frac{2}{3} - \frac{1}{2}$.

- **Test 2: $n = 6$**

Biểu thức $-\frac{3}{6} + \frac{1}{4} + \frac{2}{5} - \frac{5}{6} + \frac{6}{4} - \frac{4}{5} = \frac{1}{60}$ là một ví dụ hợp lệ cho giá trị nhỏ nhất dương.

Tóm tắt đề bài

Với mỗi n , hãy tìm một biểu thức gồm cộng/trừ các phân số dạng $\frac{a}{b}$ ($1 \leq a, b \leq n$) sao cho:

- Kết quả là **số dương nhỏ nhất có thể**.
- Biểu thức hợp lệ, không chứa khoảng trắng, từ 1 đến n phân số.

Nếu có nhiều đáp án → in ra **bất kỳ** một cái đúng.

Bài E: Xoay dãy số (Express Rotations)

 **Giới hạn thời gian:** 3 giây

 **Giới hạn bộ nhớ:** 1024 MB








Đề bài (dịch tiếng Việt)

Bạn được cung cấp một dãy số gồm n phần tử:

$$a_1, a_2, \dots, a_n$$

Nhiệm vụ của bạn là **xóa toàn bộ các phần tử khỏi dãy, theo thứ tự từ lớn đến nhỏ** (các phần tử bằng nhau có thể xóa theo bất kỳ thứ tự nào).

Bạn được phép thực hiện **3 thao tác**:

1.  **Di chuyển phần tử đầu tiên ra cuối dãy**
 Chi phí: **bằng giá trị của phần tử đó**
 2.  **Di chuyển phần tử cuối lên đầu dãy**
 Chi phí: **bằng giá trị của phần tử đó**
 3.  **Xóa phần tử đầu tiên**
 Điều kiện: **nó phải là phần tử lớn nhất còn lại trong dãy**
 Chi phí: **0 (miễn phí)**
-

Mục tiêu

Tính **tổng chi phí nhỏ nhất** để **xóa hết toàn bộ dãy số** bằng các thao tác trên.

Input

- Dòng đầu: số nguyên n ($1 \leq n \leq 500\,000$) — số phần tử trong dãy.
 - Dòng thứ hai: n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^6$) — các phần tử của dãy.
-



Output

- In ra **một số nguyên**: tổng chi phí **ít nhất** để xóa toàn bộ dãy.
-



Ví dụ

Input

```
6
6 10 6 5 4 5
```

Output

```
16
```



Giải thích

Dãy ban đầu:

```
`[6, 10, 6, 5, 4, 5]`
```

- Di chuyển `6` ra cuối \rightarrow `[10, 6, 5, 4, 5, 6]` \rightarrow **cost = 6**
- Xóa `10` (lớn nhất) \rightarrow `[6, 5, 4, 5, 6]`



- Xóa `6` → `[5, 4, 5, 6]`
- Di chuyển `6` (cuối) lên đầu → `[6, 5, 4, 5]` → **cost = 6**
- Xóa `6` → `[5, 4, 5]`
- Xóa `5` → `[4, 5]`
- Di chuyển `4` ra cuối → `[5, 4]` → **cost = 4**
- Xóa `5` → `[4]`
- Xóa `4` → `[]`

✅ Tổng chi phí: **6 + 6 + 4 = 16**

👉 Tóm tắt đề bài

Cho dãy số, bạn phải **xóa từng phần tử theo thứ tự giảm dần**, chỉ được xóa phần tử đầu tiên nếu nó là lớn nhất trong dãy.

Bạn được phép **di chuyển phần tử đầu hoặc cuối**, với chi phí bằng giá trị phần tử.

👉 Mục tiêu: **Tối thiểu hóa tổng chi phí** để xóa hết dãy.

📖 Bài F: Những chú ếch (Frogs)

🕒 **Giới hạn thời gian:** 2 giây

💾 **Giới hạn bộ nhớ:** 1024 MB

📖 Đề bài (dịch tiếng Việt)

Có một cái ao dài và hẹp. Một **mẹ ếch** đã đẻ ra **n quả trứng**, với quả trứng thứ i nằm ở vị trí a_i .

Từ mỗi quả trứng sẽ nở ra một con ếch, **theo thứ tự từ 1 đến n** . Bạn cũng được cung cấp thêm một số nguyên k là độ dài mỗi cú nhảy.

Tại một thời điểm nào đó, **p con ếch đầu tiên** (tức là ếch số 1 đến ếch số p) sẽ bắt đầu chơi một trò chơi gọi là **đuổi bắt vòng tròn (frog-tag)**.

Trong trò chơi này:

- Ếch i sẽ **đuổi theo ếch $i + 1$** .
- Ếch thứ p sẽ **đuổi theo ếch thứ 1** (nối vòng tròn).
- Mỗi **giây, tất cả các con ếch cùng nhảy một lần**, hoặc sang trái hoặc sang phải một đoạn k , **theo hướng tới con ếch mà nó đang đuổi theo**.
 - Nếu hai con ếch đứng cùng vị trí \rightarrow ếch đuổi sẽ **nhảy sang phải**.

Các con ếch từ $p + 1$ đến n **không tham gia** trò chơi này.

Nhiệm vụ

Với mỗi giá trị p từ 2 đến n , hãy kiểm tra nếu các con ếch 1 đến p **chơi trò chơi frog-tag vô hạn**,

có tồn tại con ếch nào di chuyển ra khỏi vị trí ban đầu của nó một khoảng ≥ 9999999 hay không?

- Nếu **có** \rightarrow in `1`
- Nếu **không** \rightarrow in `0`

👉 In ra **xâu nhị phân** dài $n - 1$ ký tự, tương ứng với các giá trị $p = 2$ đến n .

Input

- Dòng đầu: hai số nguyên n, k ($2 \leq n \leq 500\,000$; $1 \leq k \leq 10^9$)
- Dòng hai: n số nguyên a_1, a_2, \dots, a_n ($-10^9 \leq a_i \leq 10^9$) — vị trí ban đầu của các quả trứng



Output

- In ra một xâu nhị phân dài $n - 1$, ký tự thứ $p - 1$ là:
 - ``1`` nếu tồn tại ếch nào đi xa khỏi vị trí ban đầu của nó ≥ 9999999 khi chơi frog-tag với p ếch đầu tiên
 - ``0`` nếu không xảy ra điều đó



Ví dụ

Input

```
6 2
4 3 -3 5 100 100
```

Output

```
01011
```



Giải thích

- Với $p = 2$: các ếch ở vị trí 4 và 3 sẽ đuổi nhau qua lại trong khoảng nhỏ \rightarrow in ``0``



- Với $p = 3$: các ếch ở vị trí 4, 3, -3 \rightarrow sẽ có chuyển động rộng dần \rightarrow in `1`
 - Và tiếp tục như vậy...
-

Tóm tắt đề bài

Một nhóm p ếch đầu tiên (với $p = 2 \dots n$) chơi trò chơi đuổi bắt vòng tròn, nhảy mỗi giây về hướng ếch mà mình đuổi.

Bạn cần kiểm tra với mỗi p , có con ếch nào **đi xa khỏi vị trí ban đầu ít nhất 9999999** hay không.

✅ In ra xâu nhị phân thể hiện kết quả đó cho từng p .

Bài G: Trò chơi MPO

 **Giới hạn thời gian:** 2 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Trong trò chơi trên bàn cờ **MPO**, bạn ghi điểm bằng cách **xây dựng công trình** trên các ô của một bảng vuông kích thước $n \times n$.

Bạn được cung cấp **trạng thái ban đầu của bàn cờ**, mỗi ô được mô tả bằng **1 trong 7 ký tự** sau:

- **Chữ hoa:** `M`, `P`, `O` tương ứng với **đã xây Metropolis, Park, Ocean**.
- **Chữ thường:** `m`, `p`, `o` là các ô trống có thể xây **Metropolis, Park, Ocean** tương ứng.
- Dấu chấm `.` là ô trống **không thể xây gì cả**.

Cách tính điểm

- Mỗi **công trình đã xây** được **+1 điểm**
- Mỗi **cặp công trình liền kề nhau theo cạnh hoặc góc** được:
 - +1 điểm nếu là **Metropolis - Park**
 - +1 điểm nếu là **Ocean - Ocean**

👉 Mỗi **nước đi** là việc **xây dựng một công trình hợp lệ** (đổi chữ thường → chữ hoa tương ứng).

Luật cấm

Bạn **KHÔNG** được thực hiện nước đi nếu nước đi đó **chỉ đem lại đúng 1 điểm** (chỉ cộng điểm từ việc xây công trình, mà không tạo ra điểm nào từ cặp kề).

Chiến lược chơi

- Tại mỗi lượt, bạn **chọn một nước đi đem lại nhiều điểm nhất**
 - Nếu có nhiều nước đi cùng điểm, chọn bất kỳ (kết quả không thay đổi)
 - **Dừng lại** khi **không còn nước đi nào hợp lệ** (không có nước đi ≥ 2 điểm)
-

Yêu cầu

- Tính và in ra:
 - **Điểm số ban đầu**
 - **Điểm số sau khi chơi xong**
 - **Bàn cờ cuối cùng sau khi kết thúc trò chơi**
-

Input

- Dòng đầu tiên: số nguyên n ($2 \leq n \leq 10$) — kích thước bàn cờ
 - n dòng tiếp theo: mỗi dòng là chuỗi n ký tự trong tập ``.mpoMPO``
-

Output

- Dòng đầu tiên: hai số nguyên — **điểm số ban đầu, điểm số sau khi kết thúc**
 - n dòng tiếp theo: trạng thái bàn cờ cuối cùng
-

Ví dụ

Input

```
4
.pm.
Mom.
Oom.
p.p
```

Output

4 13
.PM.
MOM.
00m.
p . .p



Giải thích ví dụ

Trạng thái ban đầu:

. p m .
M o m .
O O m .
p . . p

- Có 3 công trình: `M`, `O`, `O` → +3 điểm
- Có 1 cặp liên kề Ocean-Ocean → +1 điểm ➡ Tổng: **4 điểm**

Lượt 1:

- Xây thêm `O` tại dòng 2 cột 2 (liên kề 2 ocean) → +3 điểm

Lượt 2:

- Xây `P` tại dòng 1 cột 2 (liên kề metropolis) → +2 điểm

Lượt 3, 4:

- Xây `M` tại hai vị trí liên kề park → mỗi lượt +2 điểm

Tổng điểm cuối:



- 7 công trình → +7 điểm
- 3 cặp metropolis-park → +3 điểm
- 3 cặp ocean-ocean → +3 điểm

➡ **Tổng: 13 điểm**

Tóm tắt đề bài

Cho một bàn cờ MPO. Tính:

- Tổng điểm ban đầu (theo công trình và cặp hợp lệ)
- Tự động thực hiện các nước đi tối ưu (điểm cao nhất ≥ 2)
- Dừng lại khi không còn nước đi hợp lệ
- In ra điểm cuối cùng và bàn cờ sau cùng.

Bài H: Nhảy Cao (High Jump)

 **Giới hạn thời gian:** 2 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Henry đang tham gia một cuộc thi **nhảy cao**.

Trước mỗi lần nhảy, anh có thể chọn **độ cao mà nhảy là một số nguyên từ 1 đến n** .

Là huấn luyện viên lâu năm của Henry, bạn biết rõ khả năng của anh:

- Với mỗi độ cao h , bạn biết xác suất thành công là p_h .

- $p_1 > p_2 > \dots > p_n$, nghĩa là càng nhảy cao thì xác suất thành công càng giảm.

Luật thi đấu:

- Nếu **Henry thất bại** một lần → **dừng thi**, điểm là độ cao cao nhất đã vượt qua trước đó.
 - Nếu **nhảy qua độ cao** n → cuộc thi kết thúc **tự động** với điểm số là n .
-

Nhiệm vụ

Giúp Henry chọn ra **chiến lược chọn độ cao nhảy** để **tối đa hóa kỳ vọng điểm số cuối cùng**.

Tính và in ra **giá trị kỳ vọng lớn nhất có thể đạt được**.

Input

- Dòng đầu: số nguyên n ($1 \leq n \leq 500\,000$) — số độ cao khả thi.
 - Dòng thứ hai: n số thực p_1, p_2, \dots, p_n ($0 < p_i < 1$), mỗi số có **tối đa 9 chữ số sau dấu chấm**.
-

Output

- Một số thực — giá trị kỳ vọng lớn nhất có thể.
- **Độ chính xác tuyệt đối hoặc tương đối $\leq 10^{-6}$** . Bạn có thể in đến 20 chữ số sau dấu phẩy.

Ví dụ

Input

```
5
0.9 0.85 0.6 0.456000 0.000000017
```

Output

```
2.475200006589
```



Giải thích ví dụ

Chiến lược tối ưu là:

1. Nhảy ở độ cao **2**: xác suất thành công là 0.85 → nếu trượt (0.15) thì điểm = 0
2. Nếu qua được, nhảy độ cao **4**: xác suất thành công 0.456 → nếu trượt thì điểm = 2
3. Nếu tiếp tục qua, thử **độ cao 5** (xác suất cực nhỏ) → nếu trượt thì điểm = 4

Kỳ vọng:

```
= 0 * 0.15
+ 2 * 0.85 * (1 - 0.456)
+ 4 * 0.85 * 0.456 * (1 - 0.000000017)
+ 5 * 0.85 * 0.456 * 0.000000017
= 2.4752000065892
```


👉 Tóm tắt đề bài

Bạn được cung cấp dãy xác suất $p_1 > p_2 > \dots > p_n$, với p_h là xác suất Henry **vượt qua** độ cao h .

Hãy xác định chuỗi các độ cao nhảy tối ưu để **kỳ vọng điểm số cuối cùng là lớn nhất**.

Điểm số là độ cao cao nhất đã vượt qua **trước khi thất bại**, hoặc 0 nếu trượt ngay lần đầu.

Nếu vượt qua được độ cao n , điểm là n và cuộc thi kết thúc.

✅ **Tính giá trị kỳ vọng tối đa có thể đạt được.**

📖 Bài 1: Đội không cân bằng (Imbalanced Teams)

🕒 **Giới hạn thời gian:** 4 giây

💾 **Giới hạn bộ nhớ:** 1024 MB

📖 Đề bài (dịch tiếng Việt)

Có n người chơi tham gia tập luyện bóng chuyền thường xuyên.

Huấn luyện viên Igor biết rõ **trình độ** của từng người: người chơi thứ i có trình độ là a_i .

.

Trong mỗi buổi tập, huấn luyện viên sẽ tổ chức **nhiều trận đấu**. Mỗi trận gồm:

- **Hai đội không trùng nhau**, mỗi đội có k người.
- **Tổng trình độ của một đội** là tổng trình độ các thành viên trong đội.

👉 Igor nhận thấy rằng:

Độ chênh lệch trình độ giữa hai đội càng lớn, trận đấu kết thúc càng nhanh → chơi được nhiều trận hơn.

Nhiệm vụ

Giúp Igor lên kế hoạch cho buổi tập bằng cách:

1. **Tìm độ chênh lệch trình độ lớn nhất có thể** giữa hai đội k người.
2. **Đếm số trận** (hai đội không trùng người) có độ chênh lệch đó.

👉 Do số trận có thể rất lớn, in kết quả **modulo** $10^9 + 7$.

Input

- Dòng đầu tiên: hai số nguyên n, k ($2 \leq n \leq 2000$; $1 \leq k \leq \lfloor n/2 \rfloor$)
 - Dòng thứ hai: n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^6$)
-

Output

- Hai số nguyên:
 - **Độ chênh lệch trình độ lớn nhất**
 - **Số trận có độ chênh lệch đó** (modulo $10^9 + 7$)
-

Ví dụ

Input

```
6 2
2 5 7 2 5 2
```

Output

8 6

Giải thích:

- Huấn luyện viên chọn các cặp đội 2 người.
- Đội có tổng nhỏ nhất: $\{2, 2\} = 4$
- Đội có tổng lớn nhất: $\{5, 7\} = 12$
- Chênh lệch = 8
- Có **6 trận** khác nhau đạt được độ chênh lệch 8.

Input

5 2
1 1 1 1 1

Output

0 15

Giải thích:

- Tất cả người đều trình độ 1 \rightarrow mọi đội đều có tổng trình độ = 2.
- Chênh lệch luôn = 0.
- Có **15 trận** không trùng người chơi.

👉 Tóm tắt đề bài

Bạn có n người chơi với trình độ đã biết. Với mỗi trận:

- Chọn 2 đội k người **không trùng nhau**
- Tính chênh lệch trình độ giữa hai đội

✅ Hãy tìm:

- **Độ chênh lệch tối đa**
- **Số trận đạt được chênh lệch đó, modulo $10^9 + 7$.**

📖 Bài J: Chỉ là các số 0 (Just Zeros)

🕒 **Giới hạn thời gian:** 1.5 giây

💾 **Giới hạn bộ nhớ:** 1024 MB

📖 Đề bài (dịch tiếng Việt)

Julia có một **bảng nhị phân** gồm $h \times w$ ô, được chia thành h hàng và w cột.

- Mỗi ô là một **bit**: 0 hoặc 1.
- Bạn có thể **lật bit** ($0 \rightarrow 1$ hoặc $1 \rightarrow 0$).

Có 3 loại thao tác được phép:

- `flip i j`: lật bit ở hàng i , cột j

- $\texttt{R } i$: lật **toàn bộ hàng** i
- $\texttt{K } j$: lật **toàn bộ cột** j

Mục tiêu: Julia muốn đưa toàn bộ lưới về **toàn bit 0**.

Độ khó của lưới là số **ít nhất thao tác cần thiết** để đạt mục tiêu.

Tình huống

Romek — một người thích phá rối — thực hiện q thao tác như trên để phá lưới của Julia.

Julia không thay đổi gì, nhưng sau **mỗi thao tác**, cô muốn biết **độ khó mới của lưới**.

Nhiệm vụ

Cho biết trạng thái ban đầu và chuỗi thao tác của Romek, in ra:

- Độ khó ban đầu
 - Độ khó sau mỗi thao tác
-

Input

- Dòng 1: ba số nguyên h, w, q ($1 \leq h \leq 8; 1 \leq w, q \leq 10^5$)
 - h dòng tiếp theo: mỗi dòng là chuỗi nhị phân dài w
 - q dòng tiếp theo: mỗi dòng là một thao tác: $\texttt{P } i \texttt{ } j$, $\texttt{R } i$, hoặc $\texttt{K } j$
-

Output

- In ra $q + 1$ dòng, mỗi dòng là **độ khó của lưới** tại thời điểm đó:
 - ban đầu
 - sau thao tác 1
 - sau thao tác 2
 - ...
 - sau thao tác q
-

Giải thích ví dụ

Input

```
3 4 6
1010
1101
0010
R 2
P 3 1
K 2
P 2 1
K 4
P 3 4
```

Output

```
3
2
3
4
3
```

3

4

Giải thích thuật toán (tóm tắt)

- Vì $h \leq 8$, bạn có thể **mã hóa trạng thái các hàng bằng bitmask**.
- Với mỗi **cột**, ánh xạ thành một số nhị phân độ dài h .
 - Ví dụ: cột có giá trị ``1 0 1`` \rightarrow ``101`` = 5
- Mỗi thao tác sẽ thay đổi nội dung bitmask của cột hoặc hàng.
- Sau mỗi lần thay đổi, bạn cần **tính toán lại độ khó**:
 - Độ khó = **số lượng thao tác tối thiểu** để biến các cột hiện tại thành 0s.
 - Có thể dùng **XOR, tiền xử lý**, và **tìm tổ hợp tối ưu** qua duyệt theo bitmask hàng.

Tóm tắt đề bài

Bạn có một bảng $h \times w$ gồm các bit. Có 3 thao tác:

- ``P i j``: lật ô (i, j)
- ``R i``: lật hàng i
- ``K j``: lật cột j

Sau mỗi thao tác (tổng q thao tác), bạn cần in ra **số thao tác tối thiểu** cần để biến toàn bộ bảng thành **0**.

✅ Sử dụng encoding theo bitmask do h nhỏ (≤ 8) để tối ưu hoá tính toán cho bảng lớn w , thao tác nhiều.

📖 Bài K: Ô vuông mẫu giáo (Kindergarten Square)

🕒 Giới hạn thời gian: 3 giây

💾 Giới hạn bộ nhớ: 1024 MB

📖 Đề bài (dịch tiếng Việt)

Cô giáo viết các số tự nhiên liên tiếp từ 1 đến $h \cdot w$ lên bảng, theo **dạng hình chữ nhật** $h \times w$:

- Mỗi hàng có w số, viết từ trái qua phải
- Hàng đầu tiên: $1 \rightarrow w$
- Hàng thứ hai: $w + 1 \rightarrow 2w$
- ...
- Hàng thứ i : $(i - 1) \cdot w + 1 \rightarrow i \cdot w$

Sau bài học, cô giáo **xoá hết**, chỉ để lại một **ô vuông con** 2×2 liền kề.

Bé **Kamilek** quên mất bảng lúc đầu có bao nhiêu hàng và cột. Bé chỉ nhớ 4 số trong ô vuông con đó.

🎯 Nhiệm vụ

Cho 4 số của ô vuông 2×2 , bạn cần:

- **Tìm ra một cặp số hợp lệ** h, w (số hàng và số cột ban đầu) sao cho **ô vuông 2×2** đó là một phần của bảng.
- Nếu không thể tồn tại bảng nào hợp lệ \rightarrow in ra ``-1``

Lặp lại cho t test case

Input

- Dòng đầu tiên: số nguyên t ($1 \leq t \leq 10$) — số test case
 - Tiếp theo $2 \cdot t$ dòng: mỗi test case gồm 2 dòng, mỗi dòng chứa 2 số nguyên trong khoảng $[1, 50000]$
-

Output

- Với mỗi test case:
 - Nếu tồn tại cặp h, w thỏa mãn \rightarrow in ra một cặp số $h w$ với $2 \leq h, w \leq 100000$
 - Nếu không tồn tại \rightarrow in ``-1``
-

Phân tích

Gọi các số của ô vuông con là:

```
a b
c d
```

=> Ta có dạng ô vuông 2×2 trích từ bảng tổng quát có cấu trúc như sau (các số viết liên tiếp):

x	$x+1$
$x+w$	$x+w+1$

→ So sánh với:

a	b
c	d

Để nhận diện, ta kiểm tra:

- $b = a + 1$
- $c = a + w$
- $d = a + w + 1$

→ Từ đó:

$$w = c - a$$

→ Sau đó kiểm tra lại xem:

$$b = a + 1 \text{ và } d = c + 1$$

Nếu đúng, ta có thể chọn **h bất kỳ ≥ 2** và in ra h, w .

Ví dụ

Input

```
4
6 7
10 11
2 3
4 5
8 5
5 13
1 2
5 6
```

Output

```
3 4
-1
-1
3 4
```

Tóm tắt đề bài

Cho một ô vuông 2×2 gồm 4 số nguyên dương.

Xác định xem ô này có thể là một phần của bảng $h \times w$ với các số viết từ 1 đến $h \cdot w$ hay không.

✅ Nếu có, in ra một cặp h, w hợp lệ

❌ Nếu không, in `-1``

Bài L: Lặp lại RPS (Looping RPS)

 **Giới hạn thời gian:** 4 giây

 **Giới hạn bộ nhớ:** 1024 MB

Đề bài (dịch tiếng Việt)

Trong bài này:

- 'K' = đá (rock — *kamień*),
- 'P' = giấy (paper — *papier*),
- 'N' = kéo (scissors — *nożyce*).

Trò chơi **Kéo - Búa - Bao** (RPS):

- Bao thắng đá ($\text{'P'} > \text{'K'}$)
- Đá thắng kéo ($\text{'K'} > \text{'N'}$)
- Kéo thắng bao ($\text{'N'} > \text{'P'}$)
- Nếu cùng ký hiệu \rightarrow hòa, chơi lại

Lặp vòng chiến lược

Vì các đấu thủ không muốn đoán chiến lược của nhau nên mỗi người chọn một **chuỗi ký hiệu** cố định và **lặp lại vô hạn**.

Ví dụ: 'KPP' \rightarrow chiến lược là: $\text{'K'} \rightarrow \text{'P'} \rightarrow \text{'P'} \rightarrow \text{'K'} \rightarrow \text{'P'} \rightarrow \text{'P'} \rightarrow \dots$

Nhiệm vụ

Cho bạn danh sách chiến lược của n đấu thủ.

Tìm số **bộ ba đấu thủ (không thứ tự)** thoả mãn:

- Với một hoán vị nào đó (A, B, C):
 - A **thắng** B

- B **thắng** C
- C **thắng** A

🕒 Một trận đấu giữa 2 người là **vô hạn**, đi theo từng cặp từng bước trong chiến lược, lặp lại nếu hòa mãi.

✅ Trận đấu **kết thúc** tại bước đầu tiên mà có **thắng/thua**, nếu mãi hòa thì **hòa**.

Input

- Dòng 1: số nguyên n ($3 \leq n \leq 10^5$) — số đấu thủ
 - n dòng tiếp theo: mỗi dòng là một chuỗi không rỗng chỉ gồm các ký tự `K`, `P`, `N`
(tổng độ dài tất cả chuỗi $\leq 10^6$)
-

Output

- Một số nguyên: số bộ ba đấu thủ thoả điều kiện trên
-

Ý tưởng giải nhanh

- **Không thể** so sánh từng cặp theo cách trực tiếp vì $O(n^3)$
- Dùng **hàm quy đổi chiến lược** của từng người thành:
 - Tập các kết quả mà họ sẽ **thắng** hoặc **thua** với đối phương

- Với mỗi **cặp chiến lược**, xác định ai sẽ thắng → từ đó xây dựng **đồ thị thắng-thua**
- Duyệt các bộ ba **đỉnh** trong đồ thị sao cho tồn tại chu trình **A → B → C → A**



Ví dụ

Input

```
6
P
PN
KK
N
PKK
PN
```

Output

```
6
```

Giải thích: Có 6 bộ ba (không theo thứ tự) mà thoả mãn: A thắng B, B thắng C, C thắng A.

Trong đó `"PN"` có thể xuất hiện 2 lần vì nó có thể đến từ 2 người khác nhau.

👉 Tóm tắt đề bài

Bạn được cung cấp các chuỗi chiến lược chơi kéo-búa-bao lặp vô hạn từ n đấu thủ.

✅ Hãy **đếm số bộ ba (không có thứ tự)** thoả mãn quan hệ:

A thắng B, B thắng C, C thắng A

(không cần xác định cụ thể ai là A, B, C — miễn là tồn tại chu trình thắng-thua).

