Cộng số modulo p

Created: 4/28/2025 06:42
Updated: 4/28/2025 06:48
Exported: 4/28/2025 06:46

Đề bài A. (A + B) mod P

AGI (trí tuệ nhân tạo tổng quát) đã rất phát triển nhưng vẫn chưa thể cộng hai số nguyên modulo một số nguyên tố! Trong bài này, bạn sẽ huấn luyện một mô hình perceptron một tầng với **n** neuron để thực hiện phép cộng hai số nguyên modulo một số nguyên tố **p**.

Kiến trúc mô hình:

- Mô hình nhận hai số nguyên **a** và **b** ($0 \le a, b < p$).
- Biến chúng thành dạng **one-hot** kích thước $1 \times p$ (dòng toàn số 0, chỉ có đúng một số 1 tại cột tương ứng giá trị).
- Áp dụng phép biến đổi qua hai ma trận:
  - **Winput** kích thước $p \times n$ (p hàng, n cột).
  - **Woutput** kích thước $p \times n$ (p hàng, n cột).

Quy trình tính toán:

1. Lấy dòng thứ **a** và dòng thứ **b** trong **Winput**, cộng từng phần tử.
2. Áp dụng hàm **ReLU** (biến âm thành 0, giữ nguyên dương).
3. Gọi dòng kết quả là **M**.
4. Tính điểm số (score) cho mỗi kết quả **i** từ 0 đến p-1 bằng tích vô hướng giữa **M** và dòng thứ **i** của **Woutput**.
5. Kết quả là chỉ số **i** có điểm số lớn nhất, và nó phải bằng **(a + b) mod p**.

Yêu cầu:

- Với mỗi cặp (a, b), mô hình phải tính đúng **(a + b) mod p**.
- Xuất ra:
  1. Số neuron **n** (1 ≤ n ≤ 25).
  2. Ma trận **Winput** (p dòng, mỗi dòng **n** số thực, trị tuyệt đối ≤ 1000).
  3. Ma trận **Woutput** (p dòng, mỗi dòng **n** số thực, trị tuyệt đối ≤ 1000).

Tóm tắt:

- **Nhiệm vụ:** Thiết kế mô hình perceptron 1 tầng với **n** neuron, sử dụng hai ma trận **Winput** và **Woutput** để tính **(a + b) mod p** với mọi **0 ≤ a, b < p**.
- **Đầu vào:** Số nguyên tố **p**.
- **Đầu ra:** Số neuron **n**, ma trận **Winput** và **Woutput**.

Mục tiêu là đảm bảo mô hình chọn đúng **(a + b) mod p** với điểm số cao nhất.

Đề bài B. Người vợ tuyệt nhất

Sofia là một người vợ tuyệt vời, cô luôn nghĩ ra những ý tưởng mới để dành thời gian với chồng.

Mỗi **ý tưởng i** được mô tả bằng một đoạn **[li..ri]**, nghĩa là Sofia muốn tham gia một sự kiện bắt đầu vào ngày **li** và kết thúc vào ngày **ri**. Khi tham gia một sự kiện, cô ấy sẽ tham gia trọn vẹn từ ngày **li** đến ngày **ri** (bao gồm cả hai ngày này).

Quy định:

- Mỗi ngày, Sofia **chỉ tham gia một sự kiện**.
- Nếu một sự kiện kết thúc vào ngày **d** và sự kiện khác bắt đầu cũng vào ngày **d**, thì Sofia **chỉ được chọn một trong hai**.

Sau mỗi ý tưởng mới xuất hiện, Sofia muốn biết **số lượng sự kiện lớn nhất mà cô có thể tham gia** từ danh sách các ý tưởng hiện có.

Input:

- Dòng đầu tiên: Số nguyên **n** ($1 \le n \le 3 \cdot 10^5$) — số lượng ý tưởng.
- **n** dòng tiếp theo: Mỗi dòng chứa hai số nguyên **li** và **ri** ($1 \le li \le ri \le 6 \cdot 10^5$) — mô tả sự kiện thứ i.

Output:

- Với mỗi **i** từ 1 đến n, in ra một số nguyên — **số lượng sự kiện tối đa Sofia có thể tham gia** xét trên các sự kiện từ 1 đến i.

Ví dụ:

Input	Output
5	
1 3	1
3 5	1
1 2	2
5 6	2
4 4	3

Tóm tắt:

- **Nhiệm vụ:** Sau mỗi lần thêm một đoạn **[li..ri]**, tính số lượng đoạn rời nhau tối đa (không giao nhau, không tiếp giáp nhau) từ **1** đến **i**.
- **Giống:** Bài toán **chọn số lượng đoạn rời nhau tối đa (interval scheduling)**, nhưng phải tính dần theo từng bước thêm đoạn.

Giải pháp tối ưu sẽ cần **sắp xếp các đoạn theo thời gian kết thúc** và sử dụng **cây segment hoặc cấu trúc dữ liệu phù hợp** để cập nhật kết quả nhanh chóng sau mỗi lần thêm đoạn mới.

Đề bài C. Cardinality

Tương tác (interactive problem)

Có **n** tập hợp được đánh số từ **1** đến **n**, trong đó tập hợp thứ **i** ban đầu chỉ chứa đúng số nguyên **i**.

Bạn cần xử lý **q** truy vấn. Mỗi truy vấn thứ **i** (đánh số từ 1) là một cặp số nguyên **(Xi, Yi)**, yêu cầu bạn **tạo ra tập hợp số n + i**, bằng **hợp** của hai tập hợp **Xi** và **Yi**.

Nhiệm vụ:

- Sau khi tạo tập hợp mới, bạn **phải báo cáo số lượng phần tử khác nhau** trong tập hợp đó (không cần chính xác tuyệt đối).
- Nếu kết quả đúng là **A** và bạn in ra số **B**, kết quả vẫn được **chấp nhận** nếu **0.5·A ≤ B ≤ 2.0·A**.

Lưu ý về tương tác:

- Để tối ưu hiệu suất chấm, **các truy vấn được chia thành các nhóm 50 truy vấn**.
- Cụ thể:
  - Đọc 50 truy vấn đầu tiên, in ra 50 kết quả, rồi **flush output**.
  - Tiếp tục với 50 truy vấn tiếp theo, lặp lại quá trình trên.
  - **Nhóm cuối** có thể ít hơn 50 truy vấn.

Interactor (bộ kiểm tra) không thay đổi theo thời gian, tức là các truy vấn đã được cố định từ trước.

Input:

- Dòng đầu tiên: hai số nguyên **n** và **q** ($1 \le n \le 50\,000$, $1 \le q \le 500\,000$) — số tập hợp ban đầu và số lượng truy vấn.
- **q** dòng tiếp theo: mỗi dòng chứa hai số nguyên **Xi, Yi** ($1 \le Xi, Yi < n + i$) — chỉ số của hai tập hợp cần hợp nhất.

Output:

- Với mỗi truy vấn, in ra **một số nguyên duy nhất — ước lượng số lượng phần tử khác nhau** trong tập hợp vừa tạo.
- Sau mỗi **50 truy vấn**, **flush output**.

Ví dụ:

Input	Output
4 5	
1 2	2
2 3	2
5 6	3
6 7	3
4 7	4

Giải thích ví dụ:

- Tập hợp ban đầu:
  - 1: {1}, 2: {2}, 3: {3}, 4: {4}
- Sau truy vấn 1: {1} ∪ {2} = {1, 2} → kích thước 2.
- Sau truy vấn 2: {2} ∪ {3} = {2, 3} → kích thước 2.
- Sau truy vấn 3: {5} ∪ {6} = {1, 2} ∪ {2, 3} = {1, 2, 3} → kích thước 3.
- Sau truy vấn 4: {6} ∪ {7} = {2, 3} ∪ {1, 2, 3} = {1, 2, 3} → kích thước 3.
- Sau truy vấn 5: {4} ∪ {7} = {4} ∪ {1, 2, 3} = {1, 2, 3, 4} → kích thước 4.

Tóm tắt:

- **Nhiệm vụ:** Sau mỗi lần hợp hai tập, in ra kích thước (ước lượng) tập hợp mới.

- **Tương tác:** Chia thành từng nhóm 50 truy vấn, flush sau mỗi nhóm.
- **Đầu vào:** n, q, và danh sách các truy vấn (Xi, Yi).
- **Đầu ra:** Ước lượng kích thước tập hợp sau mỗi truy vấn, đảm bảo trong khoảng [0.5·A, 2·A].

Đề bài D. 3D

Có **n** điểm $\mathbf{a_1, a_2, ..., a_n}$ được tạo ngẫu nhiên bên trong một **hình lập phương kích thước 1**.

Bạn được cung cấp một **ma trận d**, trong đó:

- $\mathbf{d_{ij} = d_{ji} = dist(a_i, a_j) + rand(-0.1..0.1)}$, với:
  - **dist(p, q)** là **khoảng cách Euclid 3D** giữa hai điểm p và q:

    $$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$$
  - **rand(-0.1..0.1)** là một số ngẫu nhiên trong khoảng [-0.1, 0.1], **được chọn độc lập** cho từng cặp điểm.
- $\mathbf{d_{ii} = 0}$ (khoảng cách từ một điểm đến chính nó).

Yêu cầu:

- Dựa trên ma trận khoảng cách **d**, hãy xây dựng **n điểm mới** $\mathbf{b_1, b_2, ..., b_n}$ trong không gian 3D sao cho:

$$\forall i, j \quad |dist(b_i, b_j) - d_{ij}| \leq 0.1$$

Input:

- Dòng đầu tiên: số nguyên **n** ($1 \le n \le 10$) — số lượng điểm.
- **n** dòng tiếp theo: mô tả **ma trận d**. Dòng thứ **i** chứa **n** số thực d_{ij} (có 6 chữ số sau dấu phẩy).

Output:

- In **n dòng**, mỗi dòng chứa **3 số thực x_i y_i z_i** (tọa độ của điểm b_i), với:

$$-10.0 \le x_i, y_i, z_i \le 10.0$$

Ví dụ:

Input	Output
4	
0.000000 0.758400 0.557479 0.379026	0.210269 0.581333 0.000000
0.758400 0.000000 0.516608 0.446312	0.090086 0.000000 0.458722
0.557479 0.516608 0.000000 0.554364	0.000000 0.498388 0.501723
0.379026 0.446312 0.554364 0.000000	0.204618 0.204262 0.075724

Tóm tắt:

- **Nhiệm vụ:** Dựng lại **n điểm** trong không gian 3D dựa trên **ma trận khoảng cách bị nhiễu** (d sai số ±0.1).

- **Đầu vào:** số điểm **n** và **ma trận d** (khoảng cách giữa các điểm).
- **Đầu ra:** tọa độ **(x, y, z)** của các điểm mới sao cho khoảng cách giữa chúng xấp xỉ ma trận **d** trong sai số 0.1.

Đề bài E. Equal Strings

Tương tác (interactive problem)

Có **n − 1** chuỗi nhị phân ngẫu nhiên độ dài 50 được sinh ra. Sau đó, **một chuỗi được nhân đôi** (tạo thành một bản sao), và tất cả **n** chuỗi được xáo trộn lại.

Nhiệm vụ của bạn:
Tìm **chỉ số** của hai chuỗi **giống hệt nhau** (cùng nội dung).

Truy vấn:

Bạn có thể thực hiện **tối đa 25 000 truy vấn**.

- Mỗi truy vấn là một cặp **(i, j)**, bạn hỏi khoảng cách Hamming giữa hai chuỗi s_i và s_j.
- **Khoảng cách Hamming** là số vị trí khác nhau giữa hai chuỗi ($0 \le d \le 50$).
- Nếu nhận được kết quả **d = 0** (hai chuỗi giống hệt nhau), chương trình của bạn **phải kết thúc ngay lập tức**.

Quy tắc tương tác:

1. Ban đầu, interactor in ra **n** ($2 \le n \le 1000$) — số chuỗi.
2. Sau đó, bạn gửi truy vấn **(i, j)** ($1 \le i, j \le n, i \ne j$), **flush output** sau mỗi truy vấn.

3. Interactor sẽ trả về khoảng cách Hamming **d**.
4. Khi nhận được **d = 0**, chương trình dừng.

Ví dụ:

Input (Interactor)	Output (Your program)
4	1 2
	21
	2 3
	23
	1 4
	21
	2 4
	0

- Các chuỗi:
  1. 10110000010010010101011100011000000011001011000101
  2. 10101101000111001111100000010010000100101011100101
  3. 10011110110010011111100111011000010100011101111011
  4. 10101101000111001111100000010010000100101011100101 (bản sao của chuỗi 2)

Tóm tắt:

- **Nhiệm vụ:** Tìm hai chuỗi **giống hệt nhau** (có khoảng cách Hamming = 0) qua truy vấn Hamming giữa các cặp chỉ số.
- **Giới hạn:** Tối đa **25 000 truy vấn**, phải **dừng ngay** khi tìm được cặp giống nhau.
- **Đầu vào:** số lượng chuỗi **n**.
- **Đầu ra:** các truy vấn **(i, j)**, nhận lại khoảng cách **d** từ interactor, dừng khi **d = 0**.

Đề bài F. Fast Tree Queries

Bạn được cho một **cây** gồm **n đỉnh**. Ban đầu, tại đỉnh **i** có ghi số nguyên **i**.

Bạn cần xử lý **q truy vấn** với hai loại sau:

1. **+ a v x**:
   Cộng thêm **x** vào tất cả các giá trị trên **đường đi đơn giản** từ đỉnh **a** đến đỉnh **v**.
2. **? a v**:
   Tính **xor** tất cả các giá trị trên **đường đi đơn giản** từ đỉnh **a** đến đỉnh **v**.

Input:

- Dòng đầu: hai số nguyên **n** và **q** ($1 \le n, q \le 10^5$) — số đỉnh và số truy vấn.
- **n − 1** dòng tiếp theo: mỗi dòng chứa hai số nguyên $\mathbf{u_i}$, $\mathbf{v_i}$ ($1 \le u_i, v_i \le n$) — mô tả cạnh của cây.
- **q** dòng tiếp theo: mỗi dòng là một truy vấn:
  - **+ a v x** ($1 \le a, v \le n$, $1 \le x \le 10^4$)
  - **? a v** ($1 \le a, v \le n$)

Output:

- Với mỗi truy vấn **? a v**, in ra một dòng chứa kết quả **xor** của các giá trị trên đường đi từ **a** đến **v**.

Ví dụ:

Input	Output
5 6	
1 2	5
1 3	1
3 4	6
3 5	2
? 2 5	
+ 1 4 1	
? 2 5	
+ 4 5 2	
? 4 5	
? 1 1	

Giải thích:

- Cấu trúc cây:

```
  1
 / \
2   3
   / \
  4   5
```

- Ban đầu: giá trị tại mỗi đỉnh là chính số hiệu của nó.

1. `? 2 5`: Đường đi từ 2 → 1 → 3 → 5, xor(2,1,3,5) = 5.
2. `+ 1 4 1`: Cộng 1 vào các đỉnh trên đường 1 → 3 → 4.
    - Giá trị mới:
        - 1 → 2
        - 3 → 4
        - 4 → 5
3. `? 2 5`: Đường đi từ 2 → 1 → 3 → 5, xor(2,2,4,5) = 1.
4. `+ 4 5 2`: Cộng 2 vào các đỉnh trên đường 4 → 3 → 5.
    - Giá trị mới:
        - 4 → 7
        - 3 → 6
        - 5 → 7
5. `? 4 5`: Đường đi từ 4 → 3 → 5, xor(7,6,7) = 6.
6. `? 1 1`: Chỉ có đỉnh 1, giá trị là 2.

Tóm tắt:

- **Nhiệm vụ:**
  - Hỗ trợ **cộng giá trị** trên đường đi giữa hai đỉnh.
  - Tính **xor** các giá trị trên đường đi giữa hai đỉnh.
- **Đầu vào:** Cấu trúc cây và các truy vấn.
- **Đầu ra:** Kết quả của các truy vấn **? a v**.

Đề bài G. Geo Sharding

Bạn được cho một **lưới n × n**. Nhiệm vụ là **tô màu** mỗi ô trong lưới bằng một trong **C màu** (với **C = 10 + ⌊n² / 100⌋**).

Yêu cầu tô màu:

1. **Mỗi màu chỉ được sử dụng tối đa 150 lần** trong toàn bộ lưới.
2. Với **mỗi ô (r, c)**, xét tất cả các ô **(r_i, c_i)** thỏa mãn:

$$(r - r_i)^2 + (c - c_i)^2 \leq 100$$

(tức là các ô nằm trong **bán kính Euclid 10** so với ô (r, c)), **tập hợp các màu** được sử dụng trong vùng này **không vượt quá 8 màu khác nhau**.

Input:

- Một số nguyên **n** (1 ≤ n ≤ 1000) — kích thước của lưới.

Output:

- In ra **n dòng**, mỗi dòng chứa **n số nguyên** từ 1 đến **C** — biểu diễn màu của mỗi ô **(r, c)**.

Ví dụ:

Input	Output
3	1 2 3
	4 5 6
	7 8 8

Tóm tắt:

- **Nhiệm vụ:** Tô màu lưới **n × n** sao cho:
  - Mỗi màu dùng **tối đa 150 lần**.
  - Trong **bán kính 10** quanh bất kỳ ô nào, **không quá 8 màu khác nhau**.
- **Đầu vào:** Kích thước lưới **n**.
- **Đầu ra:** Bảng màu (các số từ **1** đến **C**) của lưới.

Đề bài H. Have You Seen This Subarray?

Bạn có một **mảng a** ban đầu với:

$$a[i] = i \quad (1 \le i \le n)$$

Có **m thao tác hoán đổi** được thực hiện trên mảng này. Mỗi thao tác hoán đổi chọn ngẫu nhiên hai chỉ số **i** và **j** (**i < j**) và **đổi chỗ a[i] và a[j]**.

Bạn cần trả lời **q truy vấn**. Mỗi truy vấn là một **mảng con b = [b_1, b_2, ..., b_k]**, yêu cầu tìm **thời điểm sớm nhất (số thao tác hoán đổi)** mà **b xuất hiện liên tiếp** trong mảng **a**.

- Nếu mảng **a** ban đầu đã chứa **b**, in ra **0**.
- Đảm bảo rằng **tại một thời điểm nào đó**, **b** sẽ xuất hiện liên tiếp trong **a**.

Input:

- Dòng đầu: ba số nguyên **n, m, q** ($1 \le n, m, q \le 10^5$).
- **m dòng tiếp theo:** mỗi dòng chứa hai số nguyên **i, j** ($1 \le i < j \le n$) — thao tác **hoán đổi a[i] và a[j]**.
- **q dòng tiếp theo:** mỗi dòng mô tả một truy vấn:
  - Một số nguyên **k** ($1 \le k \le n$), tiếp theo là **k số nguyên b_1, b_2, ..., b_k** — mảng con cần kiểm tra.
- Tổng độ dài tất cả các mảng **b** không vượt quá **10^5**.

Output:

- Với mỗi truy vấn, in ra **thời điểm sớm nhất (số thao tác hoán đổi)** mà mảng **b** xuất hiện liên tiếp trong **a**.

Ví dụ:

Input	Output
6 3 5	
1 5	1
3 4	3
1 6	0
2 4 1	
3 3 1 5	
3 3 4 5	
4 5 2 4 3	
2 6 2	
Giải thích:	
- Ban đầu: 1 2 3 4 5 6	
- Sau hoán đổi 1 5: 5 2 3 4 1 6	
- Sau hoán đổi 3 4: 5 2 4 3 1 6	
- Sau hoán đổi 1 6: 6 2 4 3 1 5	

- Truy vấn 1: [4, 1] xuất hiện lần đầu ở bước 1.
- Truy vấn 2: [3, 1, 5] xuất hiện lần đầu ở bước 3.
- Truy vấn 3: [3, 4, 5] **chưa từng xuất hiện**, nhưng bài bảo đảm có nên trong ví dụ **giả định có xuất hiện** ở bước 0.

- Truy vấn 4: [5, 2, 4, 3] xuất hiện ở bước 2.
- Truy vấn 5: [6, 2] xuất hiện ở bước 3.

Tóm tắt:

- **Nhiệm vụ:** Tìm thời điểm sớm nhất khi **mảng con b** xuất hiện liên tiếp trong **a** sau **m thao tác hoán đổi**.
- **Đầu vào:** Kích thước **n**, số thao tác **m**, số truy vấn **q**, danh sách hoán đổi và danh sách các truy vấn.
- **Đầu ra:** Với mỗi truy vấn, in ra **thời điểm** (số thao tác) mà **b** xuất hiện liên tiếp trong **a**.

Đề bài I. Interactive Casino

Tương tác (interactive problem)

Bạn có **1000$** và sẽ chơi một trò chơi kéo dài **T vòng** (**T = 1000**).

Luật chơi:

- Ở mỗi vòng:
  - Ban giám khảo (jury) chọn **ngẫu nhiên một số nguyên b** trong khoảng **[1..m]**, với **m** là số tiền hiện tại của bạn.
  - Mỗi số trong khoảng đó có **xác suất bằng nhau**.
- Bạn có thể:

    - **PLAY** (chơi vòng này).
    - **SKIP** (bỏ qua vòng này).
- Nếu bạn chọn **PLAY**, kết quả sẽ được chọn ngẫu nhiên với xác suất **50-50**:
    - Bạn **trả b đô la**.
    - Bạn **nhận 2b đô la**.

Điều kiện thắng/thua:

- **Thắng (WIN):** Nếu số tiền của bạn **vượt quá 10 000$**.
- **Thua (LOSE):**
    - Nếu tiền của bạn về **0$**.
    - Hoặc **hết T vòng** mà bạn chưa đạt 10 000$.

Quy tắc tương tác:

1. Dòng đầu tiên: số nguyên **T** (số vòng chơi).
    - **T = 5** chỉ dùng cho ví dụ mẫu (sample).
2. Mỗi vòng:
    - In ra: `` `ROUND m b` ``
        - **m**: số tiền hiện tại.
        - **b**: số tiền cược được chọn ngẫu nhiên.
    - Bạn phản hồi:

- `PLAY` hoặc `SKIP`.
- **Flush output** sau mỗi phản hồi.

3. Nếu bạn thắng, interactor in ra: **WIN**.
   Nếu bạn thua, in ra: **LOSE**.

Ví dụ:

Input	Output
5	PLAY
ROUND 1000 43	SKIP
ROUND 957 433	SKIP
ROUND 957 525	PLAY
ROUND 957 125	SKIP
ROUND 832 685	
WIN	

Tóm tắt:

- **Nhiệm vụ:** Tối ưu hóa chiến lược **PLAY** hoặc **SKIP** để **tăng tiền vượt 10 000$** hoặc ít nhất **không về 0$** sau **T vòng**.
- **Đầu vào:** T, và trong mỗi vòng là số tiền hiện tại **m** và số tiền cược **b**.
- **Đầu ra:** Với mỗi vòng, in ra **PLAY** hoặc **SKIP** để quyết định hành động của bạn.

Đề bài J. Jigsaw Puzzle

Có một tờ giấy hình vuông kích thước **1 × 1**. Thao tác sau được thực hiện tối đa **20 lần** trên tờ giấy này:

- Chọn **2 điểm ngẫu nhiên** bên trong hình vuông.
- Vẽ một **đường thẳng** đi qua hai điểm đó.

Sau khi hoàn thành, tờ giấy được **cắt dọc theo các đường thẳng** thành **n mảnh**.

Tiếp theo:

- Các mảnh **bị xoay ngẫu nhiên** (không bị lật ngược), **dịch chuyển vị trí**, và được đưa cho bạn.

Yêu cầu:

- **Xác định vị trí ban đầu** của các mảnh ghép, tức là đưa chúng **về lại trong hình vuông 1 × 1** đúng như trước khi bị dịch chuyển.
- Các mảnh **không được giao nhau** (diện tích giao nhau giữa bất kỳ hai mảnh nào $\leq 10^{-6}$).

Input:

- Dòng đầu tiên: số nguyên **n** ($2 \leq n$) — số mảnh ghép.
- Tiếp theo là mô tả của **n mảnh**:
  - Dòng đầu mỗi mảnh: số nguyên **m** ($3 \leq m$) — số đỉnh của mảnh.

- **m dòng tiếp theo:** mỗi dòng chứa **hai số thực x_i y_i** ($0.0 \le x_i, y_i \le 2.0$) — tọa độ các đỉnh, theo thứ tự **ngược chiều kim đồng hồ**.

Output:

- Với **mỗi mảnh**, in ra **tọa độ các đỉnh** (theo đúng thứ tự input), sao cho:
  - Tọa độ nằm trong khoảng **[0.0, 1.0]**.
  - Các mảnh **không giao nhau** trong hình vuông **1 × 1**.
- Nếu có **nhiều cách sắp xếp hợp lệ**, in ra **bất kỳ cách nào**.

Ví dụ:

Input	Output
4	0.277161636 0.000000000
4	0.473262431 0.793116645
0.440405375916 0.778474079786	0.000000000 0.728029248
0.000000000000 0.090337001520	0.000000000 0.000000000
0.469097990019 0.000000000000	...
0.702887505082 0.689470121906	...

Tóm tắt:

- **Nhiệm vụ:** Xác định lại vị trí các mảnh ghép (xoay, dịch chuyển ngược lại) để đặt chúng vào hình vuông **1 × 1**, **không giao nhau**.
- **Đầu vào:** Số lượng mảnh ghép và danh sách các đỉnh của từng mảnh (bị xoay và dịch chuyển).
- **Đầu ra:** Tọa độ các đỉnh của từng mảnh (trong khoảng [0.0, 1.0]).

Đề bài K. Knapsack

Bạn được cho $\mathbf{n = 10^4}$ số nguyên $\mathbf{a_1, a_2, ..., a_n}$. Mỗi số được sinh ngẫu nhiên từ khoảng $\mathbf{[1..10^{12}]}$.

Nhiệm vụ của bạn là:

- **Phân chia** mỗi số vào **một trong ba tập hợp A, B, hoặc C**, hoặc **loại bỏ** nó.
- Mục tiêu:
  - **Tổng các số trong A = tổng các số trong B = tổng các số trong C**.
  - **Mỗi tập hợp** phải chứa **ít nhất một số**.

Nếu có nhiều cách phân chia, bạn có thể in ra **bất kỳ cách nào**.

Bài toán đảm bảo rằng **luôn tồn tại đáp án hợp lệ** cho tất cả các test.

Input:

- Dòng đầu tiên: số nguyên **n** (**$\mathbf{n = 10^4}$** hoặc **n = 6** trong test mẫu).
- Dòng thứ hai: **n số nguyên $\mathbf{a_1, a_2, ..., a_n}$** ($\mathbf{1 \le a_i \le 10^{12}}$).

Output:

- Một dòng gồm **n ký tự**:
  - Mỗi ký tự tương ứng với a_i:
    - `"A"`: đưa vào tập hợp **A**.
    - `"B"`: đưa vào tập hợp **B**.
    - `"C"`: đưa vào tập hợp **C**.
    - `"."`: **loại bỏ** số đó.

Ví dụ:

Input	Output
6 4 3 8 1 5 4	ABC.BA

Giải thích:

- **A** chứa: 4, 5 (tổng 9).
- **B** chứa: 3, 4 (tổng 7).
- **C** chứa: 8, 1 (tổng 9).

(Đáp án chỉ cần **tổng ba tập hợp bằng nhau** và **mỗi tập hợp có ít nhất một phần tử**.)

Tóm tắt:

- **Nhiệm vụ:** Chia các phần tử vào ba tập hợp **A, B, C** (hoặc loại bỏ), sao cho **tổng của A = tổng của B = tổng của C** và **mỗi tập hợp không rỗng**.
- **Đầu vào:** Danh sách **n số nguyên**.
- **Đầu ra:** Một chuỗi ký tự thể hiện cách phân chia từng số.

Đề bài L. London Underground

Bạn được cho một **đồ thị** mô tả mạng lưới tàu điện ngầm London với:

- **426 ga tàu (đỉnh)**.
- **505 kết nối hai chiều (cạnh)** giữa các ga.

Bạn được cho một **tập con các ga ban đầu**. Nhiệm vụ của bạn là:

- **Đếm số cách thêm các ga khác** (có thể không thêm ga nào) vào tập hợp đó sao cho **không có hai ga nào được chọn trực tiếp kết nối với nhau**.

> *"Hai cách thêm các ga được coi là **khác nhau** nếu **ít nhất một ga** có mặt trong một tập hợp mà **không có mặt trong tập hợp kia**."*

Số cách có thể rất lớn, bạn cần in kết quả **modulo 998244353**.

Input:

1. Dòng đầu tiên: số nguyên **m = 505** — số lượng kết nối.
2. **m dòng tiếp theo**: mỗi dòng chứa hai tên ga — mô tả một kết nối hai chiều giữa hai ga.

- Tên ga là chuỗi gồm **chữ cái**, **gạch dưới** hoặc **chữ số**.
- **Không có cạnh lặp** hoặc **tự nối chính nó**.

3. Một dòng tiếp: số nguyên **k** ($0 \le k \le 426$) — số ga trong **tập ban đầu**.
4. **k dòng tiếp theo**: tên các ga trong **tập ban đầu**.

Output:

- In ra số cách mở rộng tập hợp các ga (thêm các ga khác vào) sao cho **không có hai ga trực tiếp kết nối**, **modulo 998244353**.

Ví dụ:

Input (rút gọn)	Output
505	159589981
Baker_Street Regents_Park	
Charing_Cross Embankment	
...	
2	
Baker_Street	
Liverpool_Street	

Giải thích ý tưởng:

- Đây là bài toán **đếm số tập hợp độc lập mở rộng** từ một tập ban đầu trong đồ thị.
- Tập hợp độc lập (**Independent Set**) là tập hợp các đỉnh sao cho **không có hai đỉnh nào kề nhau**.

Tóm tắt:

- **Nhiệm vụ:** Đếm số cách thêm ga vào **tập ban đầu** sao cho **không có hai ga được kết nối trực tiếp**.
- **Đầu vào:** Đồ thị mô tả các ga và kết nối, cùng **tập con ban đầu**.
- **Đầu ra:** Số cách thêm ga hợp lệ **modulo 998244353**.

Đề bài M. Meta

Trong một kỳ thi lập trình, **3 thành viên** trong một đội chia sẻ **chỉ một máy tính**, nên tại mỗi thời điểm chỉ có **một người làm việc trên một bài**.

- Bạn được cho thời gian mỗi thành viên cần để **hoàn thành từng bài** (nếu họ có thể làm).
- Mỗi bài có thể được **giải bởi bất kỳ thành viên nào có khả năng**.
- **Thời gian tổng cộng là 300 phút**.

Nhiệm vụ của bạn là **tính số lượng bài tối đa** mà đội có thể giải được trong giới hạn thời gian đó.

Input:

- Dòng đầu tiên: số nguyên **n** ($1 \le n \le 14$) — số lượng bài trong kỳ thi.
- **n dòng tiếp theo**: mô tả từng bài:
  - **Tên bài** (tối đa 30 ký tự chữ và số).
  - Thời gian thực hiện bởi **thành viên 1**, **thành viên 2**, **thành viên 3** (t_1, t_2, t_3) — thời gian (phút).
  - Nếu một thành viên **không thể làm bài đó**, giá trị $t_i = -1$.

Output:

- In ra **số lượng bài tối đa** đội có thể giải trong **300 phút**.

Ví dụ:

Input	Output
13	10
AplusB -1 20 -1	
TheBestWife 80 90 60	
Cardinality 40 50 30	
3D 40 -1 70	
EqualStrings 25 15 20	
FastTreeQueries 120 -1 40	

Input	Output
GeoSharding 25 20 30	
HaveYouSeenThisSubarray 80 90 60	
InteractiveCasino 50 20 30	
JigsawPuzzle 40 50 80	
Knapsack -1 40 200	
LondonUnderground -1 200 40	
Meta 5 7 10	

Giải thích ý tưởng:

- Với **n ≤ 14**, bài toán cho phép **thử tất cả các tập hợp con các bài**.
- Với mỗi tập hợp con, thử tất cả các **phân công thành viên cho từng bài** (nhiều lựa chọn cho mỗi bài).
- Kiểm tra **tổng thời gian nhỏ hơn hoặc bằng 300 phút**, từ đó chọn **tập hợp có số bài tối đa**.

Tóm tắt:

- **Nhiệm vụ:** Tìm **số lượng bài tối đa** đội có thể giải được trong **300 phút**, với thời gian giải bài tùy thuộc vào từng thành viên.
- **Đầu vào:** Số lượng bài, thời gian giải bài của từng thành viên.
- **Đầu ra:** Số lượng bài tối đa có thể giải.