Pokemon GO

Từ khi bắt đầu chơi Pokemon GO, giáo sư X rất thích đi bộ. Ông đi khắp các địa điểm trong Hà Nội để bắt Pokemon. Giáo sư muốn đi bộ càng nhiều càng tốt, vì vậy ngày hôm nay các bạn hãy giúp giáo sư X chọn một lộ trình tốt nhất để đi bắt Pokemon nhé.

Bản đồ thành phố gồm N địa điểm và M con đường một chiều nối các địa điểm với nhau. Giáo sư X đang đứng ở địa điểm 1, có thể đi đến các địa điểm khác thông qua M con đường nêu trên. Giáo sư X đi qua mỗi con đường đều mất một lượng thời gian nhất định (lượng thời gian này có thể âm bởi vì giáo sư X khi cần có thể tăng tốc vượt qua tốc độ ánh sáng để đi ngược thời gian). Giáo sư X tự hỏi đường đi dài nhất đến $N - 1$ địa điểm còn lại tốn bao nhiêu thời gian, ông luôn luôn chọn cách đi lâu nhất có thể để bắt nhiều Pokemon nhất. Bạn hãy giúp giáo sư X tính đường đi dài nhất để giáo sư dễ dàng lên kế hoạch đi bộ nhé.

Tên bài: POKEMON

Input:

- Dòng đầu tiên chứa hai số nguyên N, M.
- M dòng tiếp theo, mỗi dòng chứa ba số nguyên u, v, w cho biết có đường đi từ địa điểm u đến địa điểm v và giáo sư X mất w thời gian để đi con đường này.

Output:

- In ra $N - 1$ dòng, dòng thứ i gồm một số nguyên cho biết thời gian dài nhất để đi đến thành phố $i + 1$. Nếu giáo sư X có cách đi vào một chu trình dương, in ra $INFINITY$. Nếu không có cách nào để đi đến nơi, in ra $IMPOSSIBLE$.

Giới hạn:

- $1 \le N \le 1000$.
- $0 \le M \le 5000$.
- $-2000 \le w \le 2000$.

Sample Input	Sample Output
5 4 1 2 -999 2 3 2 3 2 -1 1 4 -2	INFINITY INFINITY -2 IMPOSSIBLE
2 1 1 2 -100	-100

Sơn Tường M-TP

Nếu các bạn còn nhớ, trong một vòng nào đó của CSP Marathon, giáo sư X đã được tiết lộ là một Sky cứng, fan hâm mộ đặc biệt của Sơn Tùng M-TP. Tuy nhiên, bài này không nói về chàng ca sĩ "We don't thuộc về nhau" mà là về sơn tường.

Một ngày đẹp trời, giáo sư X muốn đi sơn lại bức tường của mình do bị Sky vẽ bậy. Bức tường của giáo sư X có thể biểu diễn bằng một hình chữ nhật $M \times T$. Giáo sư X có P thùng sơn, mỗi thùng chứa một loại sơn có màu khác nhau. Giáo sư đánh số P màu sơn từ 1 đến P. Ông sẽ thực hiện Q lần sơn, mỗi lần sơn chọn ra một hình chữ nhật con có ô góc trái trên tọa độ (xL, yL) và ô góc phải dưới tọa độ (xH, yH), và sơn toàn bộ hình chữ nhật đó thành màu c. Tất cả các màu trên hình chữ nhật con đó khi trước sẽ bị sơn đè lên thành màu mới.

Bạn hãy vẽ lại bức tường của giáo sư X sau khi đã thực hiện Q lần sơn nhé.

Tên bài: STMTP

Input:

- Dòng đầu tiên gồm bốn số nguyên M, T, P,.
- Q dòng tiếp theo, mỗi dòng gồm năm số nguyên xL, yL, xH, yH, c mô tả một lần sơn.

Output:

- In ra M dòng, mỗi dòng gồm T số nguyên cách nhau bởi dấu cách, mô tả trạng thái màu của bức tường sau khi sơn. Mỗi số nguyên in ra cho biết màu của một ô tương ứng, nếu ô đó không được sơn thì in ra 0.

Giới hạn:

- $1 \le M, T \le 1000$.
- $1 \le P \le 100$.
- $1 \le Q \le 10000$.
- 40% số điểm có $1 \le M, T, P, \le 100$.

Sample Input	Sample Output
4 5 3 3 1 1 2 3 1 2 2 3 3 2 4 4 4 5 3	1 1 1 0 0 1 2 2 0 0 0 2 2 0 0 0 0 0 3 3

Quicksort Killer

Giáo sư X là một fan cuồng của Pascal, vì vậy nên ông rất tức giận khi ngày càng nhiều học sinh của mình chuyển qua code C++. Giáo sư X có lí do của mình khi nổi cáu, bởi vì các học sinh của ông quá lệ thuộc vào các công cụ của C++ mà quên mất cách code các thuật toán cơ bản như Quicksort, Heap, Splay Tree, Tattoo Tree, ...

Thế là một lần, giáo sư X lôi cổ tất cả các học sinh nghiện C++ của mình ra và yêu cầu code lại Quicksort. Sau khi các bạn code xong, giáo sư X đọc code của từng người và tức giận đến mức đấm nát màn hình, bởi vì tất cả đều code sai hết. Chẳng hạn, Huy Anh code như sau:

```
#include <bits/stdc++.h>
using namespace std;
int N, a[100001];

void Quicksort(int l, int h) {
    if (l >= h)
        return;
    int pos = l + rand() % (h - l + 1);
    int pivot = a[pos];
    a[pos] = a[l];
    int i = l, j = h;
    while (i < j) {
        while (i < j && a[j] > pivot)
            j--;
        if (i < j)
            a[i] = a[j];
        while (i < j && a[i] < pivot)
            i++;
        if (i < j)
            a[j] = a[i];
    }
    a[i] = pivot;
    Quicksort(l, i - 1);
    Quicksort(i + 1, h);
}

int main()
{
    cin >> N;
    for (int i = 1; i <= N; i++)
        cin >> a[i];
    Quicksort(1, N);
    for (int i = 1; i <= N; i++)
        cout << a[i] << " ";
    return 0;
}
```

Các bạn có biết code của Huy Anh sai ở đâu không? Thực ra chẳng sai cái gì cả, giáo sư X toàn làm màu.

Thực ra, tuy hàm Quicksort này không sai, nhưng Huy Anh đã mắc một lỗi chết người: thiếu lệnh $sran(time(NULL))$; ($Randomize$; trong Pascal). Vì thiếu lệnh này nên hàm $ran()$ của Huy Anh hoàn toàn có thể đoán được. Chính vì vậy, giáo sư X hoàn toàn có thể sinh test để giết thuật toán Quicksort này bằng cách nắm bắt hàm $ran()$ và cho nó TLE. Hàm $ran()$ là một hàm lấy số ngẫu nhiên giả (pseudorandom), sử dụng bộ bốn số nguyên A, B, X, M để tạo số ngẫu nhiên. X được gọi là hạt giống ngẫu nhiên (random seed), ban đầu nó được gán cho một giá trị nào đó. Sau đó, cứ mỗi lần hàm $ran()$ được gọi, X được gán lại bằng $(AX + B)\ mod\ M$, và trả ra giá trị X mới sau khi gán. Cụ thể:

```
int rand() {
    X = (A * X + B) % M;
    return X;
}
```

Lệnh $sran(time(NULL))$ sẽ gán X bằng số đo thời gian trong đồng hồ máy tính, vì vậy mỗi lúc hàm $rand()$ sẽ thay đổi một kiểu, không thể đoán được. Còn nếu không, X mặc định là bằng 0. A, B, M thì tùy vào bộ dịch, tuy nhiên giáo sư X khẳng định là với một bộ dịch bất kì, chỉ cần biết A, B, M là ông sẽ sinh được test chết.

Giáo sư X thêm vào code của Huy Anh một biến toàn cục *cnt* để đo thời gian. Biến này ban đầu bằng 0, và cứ mỗi thao tác tăng *i*, giảm *j* biến được tăng thêm 1. Cụ thể:

```
#include <bits/stdc++.h>
using namespace std;
int N, a[100001], cnt = 0;

void Quicksort(int l, int h) {
    if (l >= h)
        return;
    int pos = l + rand() % (h - l + 1);
    int pivot = a[pos];
    a[pos] = a[l];
    int i = l, j = h;
    while (i < j) {
        while (i < j && a[j] > pivot) {
            j--;
            cnt++;
        }
        if (i < j)
            a[i] = a[j];
        while (i < j && a[i] < pivot) {
            i++;
            cnt++;
        }
        if (i < j)
            a[j] = a[i];
    }
    a[i] = pivot;
    Quicksort(l, i - 1);
    Quicksort(i + 1, h);
}
```

```
int main()
{
    cin >> N;
    for (int i = 1; i <= N; i++)
        cin >> a[i];
    Quicksort(1, N);
    for (int i = 1; i <= N; i++)
        cout << a[i] << " ";
    cout << "\n" << cnt;
    return 0;
}
```

Nhiệm vụ của giáo sư X là phải sinh được một test sao cho biến *cnt* này càng to càng tốt. Huy Anh cũng khá cứng đầu, cãi là: "Thầy chỉ sinh được test chết với $N = 69696$ thôi, N khác chắc chắn là không sinh được." Vì vậy, để cho Huy Anh tâm phục khẩu phục, giáo sư X muốn sinh test với tất cả các độ dài dãy số N và tất cả các bộ tạo ngẫu nhiên giả của trình dịch.

Tên bài: SORTKILL

Input:

- Một dòng duy nhất chứa bốn số nguyên N, A, B, M.

Output:

- In ra một dòng duy nhất chứa N số nguyên cách nhau bởi dấu cách, mô tả dãy số test của bạn đưa ra. Các số nguyên phải nằm trong khoảng $[0, 10^9]$.

Giới hạn:

- $1 \le N \le 10^5$.
- $1 \le A, B, M \le 10^9$.
- 20% số điểm có $1 \le N \le 10$.
- 20% số điểm khác có $1 \le N \le 100$.
- 20% số điểm khác có $1 \le N \le 1000$.
- 20% số điểm khác có $1 \le N \le 10000$.

Chấm điểm:

- Nếu đáp án của bạn sai định dạng (không in ra đúng N số nguyên, in ra số nguyên nằm ngoài khoảng quy định, ...), bạn sẽ được 0 điểm cho test tương ứng.
- Nếu đáp án của bạn làm cho biến *cnt* đạt giá trị vượt quá 2×10^7, bạn sẽ được toàn bộ số điểm của test tương ứng.
- Nếu đáp án của bạn làm cho biến *cnt* đạt giá trị là O, còn đáp án chuẩn làm cho biến *cnt* đạt giá trị là A, bạn sẽ được $\min(1, \frac{O}{A})$ điểm cho test tương ứng.

Sample Input	Sample Output
2 1 2 3	2 1