人面獅身像的謎題 (Sphinx's Riddle)

偉大的人面獅身像出了一道謎題給你。給定一個有 N 個節點的圖，節點編號由 0 到 $N-1$。圖有 M 條邊，編號由 0 至 $M-1$。每個邊連接一組不同節點而且是雙向的。確切來說，對每一個介於 0 到 $M-1$ (包含) 的 j，邊 j 連接節點 $X[j]$ 和 $Y[j]$。任何一組節點最多有一個邊連接。如果兩個節點有邊相連，則稱為相鄰 (adjacent)。

一個節點的序列 $v_0, v_1, \ldots, v_k$ $(k \geq 0)$ 稱為一個路徑 (path)，若其中每兩個相鄰的節點 v_l 與 v_{l+1} (對每個 l 滿足 $0 \leq l < k$) 是相鄰的。我們稱一個路徑 $v_0, v_1, \ldots, v_k$ 連接(connects)節點 v_0 與 v_k。在這給定的圖中，任一組節點是被某個路徑連接。

有 $N+1$ 個由 0 到 N 編號的顏色。顏色 N 是特別的，且稱為人面獅身像的顏色(Sphinx's colour)。每一個節點被分配一個顏色。確切來說，節點 i $(0 \leq i < N)$ 的顏色是 $C[i]$。多個節點可能會有同樣的顏色，且有可能某些顏色沒被分配給任何節點。沒有節點有人面獅身像的顏色，也就是 $0 \leq C[i] < N$ $(0 \leq i < N)$。

一個路徑 $v_0, v_1, \ldots, v_k$ $(k \geq 0)$ 被稱為單色 (monochromatic)，如果其中所有節點有同樣的顏色，也就是 $C[v_l] = C[v_{l+1}]$ (對每個 l 滿足 $0 \leq l < k$)。此外，我們稱節點 p 與 q $(0 \leq p < N, 0 \leq q < N)$ 屬於相同的單色組件 (monochromatic component)，若且唯若他們由一個單色路徑連接。

你知道節點和邊，但不知道每一個節點的顏色。你想要透過重新著色實驗（recolouring experiments）來找出節點的顏色。

在重新著色實驗中，你可以任意重新著色許多節點。確切來說，為執行重新著色實驗，首先你選擇一個大小為 N 的陣列 E，其中對每一個 i $(0 \leq i < N)$，$E[i]$ 介於 -1 與 N 之間(**包含**inclusive）。然後，每一個節點 i 的顏色變成 $S[i]$，其中 $S[i]$的值為：

- 若 $E[i] = -1$ 則為 $C[i]$，也就是 i 原始的顏色，
- 否則為 $E[i]$。

請注意這代表在你的重新著色中，你可使用人面獅身像的顏色。

最後，在設定每一個節點 i 的顏色為 $S[i]$ $(0 \leq i < N)$ 之後，偉大的人面獅身像會宣布在圖中單色組件的數量。新的著色只會套用在這個特定的重新著色實驗中，因此所有節點的顏色在這個實驗結束後，會還原到原來的顏色。

你的任務是利用執行最多 $2\,750$ 次重新著色實驗，以辨識出圖中節點的顏色。如果你對每一組相鄰節點有正確地判斷他們是否有相同顏色，將可得到部分分數。

實作細節 (Implementation Details)

你應該實作以下程序：

```
std::vector<int> find_colours(int N,
    std::vector<int> X, std::vector<int> Y)
```

- N：圖中節點的數量。
- X，Y：長度為 M 的陣列，用來描述邊。
- 這個程序應該回傳長度為 N 的陣列 G， 表示圖中節點的顏色。
- 對於每筆測資，這個程序恰好被呼叫一次。

以上程序可以呼叫以下程序來執行重新著色的實驗：

```
int perform_experiment(std::vector<int> E)
```

- E：長度為 N 的陣列，指定節點應該如何重新著色。
- 這個程序回傳根據 E 重新著色後的單色組件數量。
- 這個程序最多可以被呼叫 $2\,750$ 次。

評分程式是**非漸進式**(**not adaptive**)的，意即節點的顏色在呼叫 `find_colours` 前就已經固定了。

限制 (Constraints)

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N(N-1)}{2}$
- 對於 $0 \leq j < M$ 的 j， $0 \leq X[j] < Y[j] < N$。
- 對於 $0 \leq j < k < M$ 的 j 和 k， $X[j] \neq X[k]$ 或 $Y[j] \neq Y[k]$。
- 每對節點都被某條路徑連接。
- 對於 $0 \leq i < N$ 的 i， $0 \leq C[i] < N$。

子任務 (Subtasks)

子任務	配分	額外限制
1	3	$N = 2$
2	7	$N \leq 50$
3	33	圖為一條路徑： $M = N - 1$ 且節點 j 與節點 $j + 1$ 相鄰 $(0 \leq j < M)$。
4	21	圖是完全圖： $M = \frac{N \cdot (N-1)}{2}$ 而且任意兩個節點皆相鄰。
5	36	無額外限制。

在每個子任務中，根據您的程式是否正確地決定每對相鄰節點的顏色是否相同可以獲得部分分數。

更精確地說，如果所有的測資中，由 `find_colours` 回傳的陣列 G 的內容與陣列 C 完全相同的話(意即，對 $0 \le i < N$ 的 i， $G[i] = C[i]$)，將可獲得該子任務的所有分數。

否則， 若所有測資皆滿足以下條件，您可獲得該子任務 50% 的分數：

- 對於 $0 \le i < N$ 的 i， $0 \le G[i] < N$;
- 對於$0 \le j < M$ 的 j，
  - $G[X[j]] = G[Y[j]]$ 若且唯若 $C[X[j]] = C[Y[j]]$。

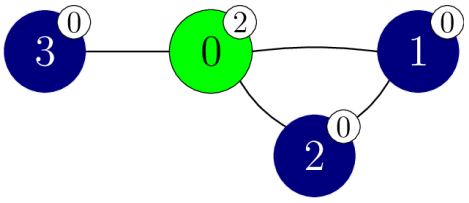
範例 (Example)

考慮以下呼叫：

```
find_colours(4, [0, 1, 0, 0], [1, 2, 2, 3])
```

以本例來說，假設節點的(隱藏)顏色是由 $C = [2, 0, 0, 0]$ 指定。

此情境如下圖。 顏色由附加到每個節點上的白底數字標籤所標示。



此程序會以下面方式呼叫 `perform_experiment`：

```
perform_experiment([-1, -1, -1, -1])
```

在這個呼叫中，沒有節點被重新著色，因為所有節點皆維持原始顏色。

考量節點 1 與節點 2。 它們的顏色皆是 0，且路徑 $1, 2$ 為單色路徑。 因此節點 1 與節點 2 屬於同一個單色組件。

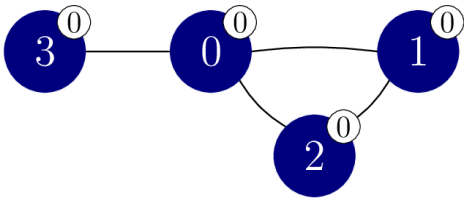
考量節點 1 與節點 3。 即使它們的顏色都是 0，因為它們之間沒有單色路徑連接，就分屬不同的單色組件。

整體來說，這樣有 3 個單色組件，由節點 $\{0\}$、$\{1, 2\}$ 和 $\{3\}$ 組成。 因此，這個呼叫會回傳 3。

現在程序以下面方式呼叫 `perform_experiment`：

```
perform_experiment([0, -1, -1, -1])
```

在這個呼叫中，只有節點 0 被重新著色為 0，著色結果如下圖所示：

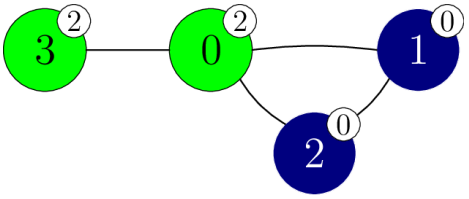


因為所有節點都屬於同一個單色組件，這個呼叫會回傳 1。 我們可以推論節點 1、2、3 的顏色都是 0。

這個程序可能會再以下面方式呼叫 `perform_experiment`：

```
perform_experiment([-1, -1, -1, 2])
```

在這個呼叫中，節點 3 被重新著色為 2，著色結果如下圖所示：



因為有 2 個單色組件，分別為 $\{0, 3\}$ 以及 $\{1, 2\}$，這個呼叫會回傳 2。 我們可以推論節點 0 的顏色為 2。

然後 `find_colours` 程序會回傳陣列 $[2, 0, 0, 0]$。 因為 $C = [2, 0, 0, 0]$，可以獲得全部分數。

注意，在有多個回傳值的情況之下，例如 $[1, 2, 2, 2]$ 或 $[1, 2, 2, 3]$，只能拿到 50% 的分數。

範例評分程式 (Sample Grader)

輸入格式 (Input format):

```
N  M
C[0]  C[1] ... C[N-1]
X[0]  Y[0]
X[1]  Y[1]
...
X[M-1]  Y[M-1]
```

輸出格式 (Output format):

```
L  Q
G[0]  G[1] ... G[L-1]
```

L 是由 `find_colours` 回傳的陣列 G 的長度，且 Q 是呼叫 `perform_experiment` 的次數。