獅身人面像的謎語 Sphinx's Riddle

偉大的獅身人面像有一個謎語要問你。 給定了一個具有 N 個頂點的圖， 這些頂點從 0 到 $N-1$ 編號。 圖中有 M 條邊，從 0 到 $M-1$ 編號。 每條邊連接一對不同的頂點並且是雙向的。 具體來說，對於每個 j 從 0 到 $M-1$（包括邊界），邊 j 連接兩個頂點 $X[j]$ 和 $Y[j]$ 。 任意一對頂點之間最多只有一條邊。 如果兩個頂點有一條邊連接，則它們被稱為**相鄰**。

一個頂點序列 $v_0, v_1, \ldots, v_k$（對於 $k \geq 0$）如果每兩個相鄰的頂點 v_l 和 v_{l+1}（對於每個 l 滿足 $0 \leq l < k$）是相鄰的，則被稱為**路徑**。 我們說一條路徑 $v_0, v_1, \ldots, v_k$ **連接**頂點 v_0 和 v_k。 在給定的圖中，每對頂點都通過某條路徑相連。

有 $N+1$ 種顏色，從 0 到 N 編號。 顏色 N 是特殊的，被稱為**獅身人面像的顏色**。 每個頂點都被分配一種顏色。 具體來說，頂點 i（$0 \leq i < N$）的顏色是 $C[i]$。 可能有多個頂點具有相同的顏色，也可能有一些顏色沒有分配給任何頂點。 沒有頂點的顏色是獅身人面像的顏色，即 $0 \leq C[i] < N$（$0 \leq i < N$）。

一個頂點序列 $v_0, v_1, \ldots, v_k$（對於 $k \geq 0$）如果所有頂點的顏色都相同，即對於每個 l 滿足 $0 \leq l < k$，$C[v_l] = C[v_{l+1}]$，則被稱為**單色的**。 此外，我們說頂點 p 和 q（$0 \leq p < N$，$0 \leq q < N$）在同一**單色連通分量**中，當且僅當它們通過一條單色路徑相連。

你知道頂點和邊，但你不知道每個頂點的顏色。 你想通過執行**重新著色實驗**來找出頂點的顏色。

在一個重新著色實驗中，你可以重新著色任意多個頂點。 具體來說，要執行一個重新著色實驗，你首先選擇一個大小為 N 的數組 E，對於每個 i（$0 \leq i < N$），$E[i]$ 在 -1 到 N **包括**之間。 然後，每個頂點 i 的顏色變為 $S[i]$，其中 $S[i]$ 的值是：

- 如果 $E[i] = -1$，則為 $C[i]$，即 i 的原始顏色，或者
- 否則為 $E[i]$。

請注意，這意味著你可以在重新著色中使用獅身人面像的顏色。

最後，偉大的獅身人面像宣布在將每個頂點 i 的顏色設置為 $S[i]$（$0 \leq i < N$）後，圖中的單色連通分量的數量。 新的著色僅應用於這個特定的重新著色實驗，因此**在實驗結束後，所有頂點的顏色都恢復為原始的顏色**。

你的任務是通過進行最多 2 750 次重新著色實驗來識別圖中頂點的顏色。 如果你正確確定每對相鄰頂點是否具有相同的顏色，你也可以獲得部分分數。

實現細節

你應該實現以下過程。

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

- N：圖中的頂點數量。
- X、Y：長度為 M 的數組，描述了邊。
- 這個過程應該返回一個長度為 N 的數組 G，表示圖中頂點的顏色。
- 對於每個測試案例，這個過程只會被調用一次。

上述過程可以調用以下過程進行重新著色實驗：

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

- E：長度為 N 的數組，指定頂點應該如何重新著色。
- 這個過程返回根據 E 重新著色後的單色連通分量的數量。
- 這個過程最多可以被調用 $2\,750$ 次。

評分機制**不是自適應的**，也就是說，在調用 `find_colours` 之前，頂點的顏色是固定的。

限制條件

- $2 \leq N \leq 250$
- $N - 1 \leq M \leq \frac{N \cdot (N-1)}{2}$
- 對於每個 j，滿足 $0 \leq j < M$，$0 \leq X[j] < Y[j] < N$。
- 對於每對 j 和 k，滿足 $0 \leq j < k < M$，$X[j] \neq X[k]$ 或 $Y[j] \neq Y[k]$。
- 每對頂點都由某條路徑連接。
- 對於每個 i，滿足 $0 \leq i < N$，$0 \leq C[i] < N$。

子任務

子任務	分數	額外約束條件
1	3	$N = 2$
2	7	$N \leq 50$
3	33	圖形為路徑：$M = N - 1$ 且頂點 j 和 $j + 1$ 相鄰（$0 \leq j < M$）。
4	21	圖形為完全圖：$M = \frac{N \cdot (N-1)}{2}$ 且任意兩個頂點相鄰。
5	36	無額外約束。

在每個子任務中，如果您的程序對於每對相鄰頂點正確確定它們是否具有相同的顏色，則可以獲得部分分數。

更確切地說，如果在所有測試案例中，`find_colours` 返回的陣列 G 完全與陣列 C 相同（即對於所有 i 滿足 $0 \leq i < N$，$G[i] = C[i]$），則您將獲得子任務的全部分數。否則，如果在所有測試案例中以下條件成立，則您將獲得子任務分數的 50%:

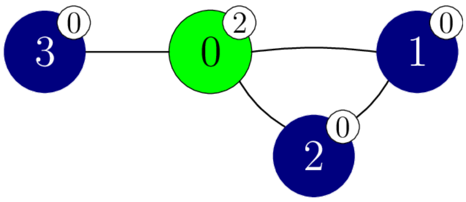
- 對於每個 i 滿足 $0 \leq i < N$，$0 \leq G[i] < N$;
- 對於每個 j 滿足 $0 \leq j < M$:
  - 如果且僅如果 $C[X[j]] = C[Y[j]]$，則 $G[X[j]] = G[Y[j]]$。

範例

考慮以下調用。

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

對於此範例，假設頂點的（隱藏）顏色由 $C = [2, 0, 0, 0]$ 給出。此情況如下圖所示。顏色還通過附加到每個頂點的白色標籤上的數字進行表示。



該過程可能如下調用 `perform_experiment`。

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

在此調用中，沒有頂點被重新著色，因為所有頂點保持其原始顏色。

考慮頂點 1 和頂點 2。它們都具有顏色 0，並且路徑 $1, 2$ 是單色路徑。因此，頂點 1 和 2 屬於同一單色分量。

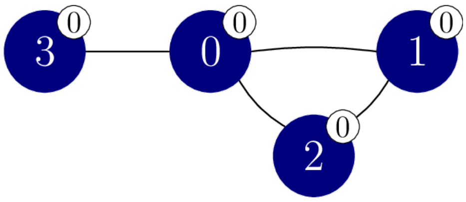
考慮頂點 1 和頂點 3。儘管它們都具有顏色 0，但它們屬於不同的單色分量，因為它們之間沒有單色路徑連接。

總的來說，有 3 個單色分量，分別是頂點 $\{0\}$、$\{1, 2\}$ 和 $\{3\}$。因此，此調用返回 3。

現在，該過程可能如下調用 `perform_experiment`。

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

在此調用中，僅將頂點 0 重新著色為顏色 0，這導致如下圖所示的著色。

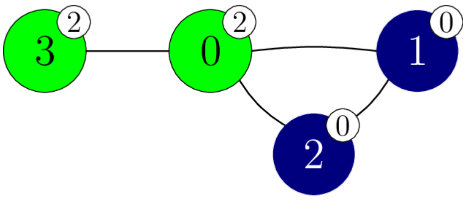


此調用返回 1，因為所有頂點都屬於同一單色分量。現在我們可以推斷頂點 1、2 和 3 的顏色為 0。

該過程隨後可能如下調用 perform_experiment。

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

在此調用中，將頂點 3 重新著色為顏色 2，這導致如下圖所示的著色。



此調用返回 2，因為有 2 個單色分量，分別是頂點 $\{0, 3\}$ 和 $\{1, 2\}$。我們可以推斷頂點 0 的顏色為 2。

然後，過程 find_colours 返回陣列 $[2, 0, 0, 0]$。由於 $C = [2, 0, 0, 0]$，因此給出完整分數。

請注意，還存在多個返回值，例如 $[1, 2, 2, 2]$ 或 $[1, 2, 2, 3]$，將給出分數的 50%。

範例評測程式

輸入格式:

```
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:

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

這裡，L 是由 find_colours 返回的數組 G 的長度， 而 Q 是 perform_experiment 的呼叫次數。