排列遊戲 (permgame)

Alice 與 Bob 是兒時玩伴，他們喜歡玩益智遊戲， 他們今天在玩圖論上的一種新遊戲。

這個遊戲給定一個無向的連通圖(connected graph)，共有 m 的頂點(vertex)與 e 個邊(edge)，頂點的編號從數字 0 到 $(m-1)$，邊的編號則從數字 0 到 $(e-1)$；第 i 個邊連接頂點 $u[i]$ 與 $v[i]$。

這個遊戲也給定一個長度為 n 的排列(permutation)$p[0], p[1], \ldots, p[n-1]$，其中 $m \leq n$。這裡的排列可視為一個陣列(array)，陣列中元素的值為 0 到 $(n-1)$ 的某個數字，且每個數字恰好出現一次。排列 p 的積分(score)係指滿足 $p[i] = i$ 的註標(index)的總個數。

這個遊戲可以持續最多 10^{100} 回合，每回合的過程如下：

1. 如果 Alice 決定遊戲停止，那麼遊戲就結束。
2. 否則，Alice 將選出排列 p 的 m 個相異註標(令其為 $t[0], t[1], \ldots, t[m-1]$，其中 $0 \leq t[i] < n$)，注意，此處並未要求 $t[0] < t[1] < \ldots < t[m-1]$。
3. 接下來，Bob 將根據 Alice 的選擇，挑出圖中的一個邊(令編號為 j，其中 $0 \leq j < e$)，並且交換(swap)排列 p 的兩個元素 $p[t[u[j]]]$ 與 $p[t[v[j]]]$。

Alice 希望排列的積分愈高愈好，Bob 則希望積分愈低愈好。

你的任務是協助 Alice 對抗 Bob，其中 Bob 是由評分程式來操作。 所謂的最佳積分，是指當玩家們皆以最佳策略來操作，遊戲結束時可得到的最大積分。

你需要計算出最佳積分是多少。同時，你還需要與 Bob 對弈，使得在某一回合，遊戲的積分至少會達到(或超過)前述的最大積分(此時遊戲結束)。

注意， Alice 的對弈策略必須能適用在 Bob 任意的對弈策略，縱使當 Bob 進行非最佳的策略時。

實作細節 (Implementation details)

你需要實作出下列函式(procedure)：

```
int Alice(int m, int e, std::vector<int> u, std::vector<int> v,
          int n, std::vector<int> p)
```

- m：圖的頂點數。
- e：圖的邊數。
- u 與 v：各為長度為 e 的陣列，表示圖的邊。
- n：排列的長度。

- p：長度為 n 的陣列，用來表示此排列。
- 此程序僅被呼叫一次。
- 此程序須回傳一整數(integer)，亦即當Alice與Bob皆以最佳策略對奕時，可得到的最大積分。

你的函式可呼叫下列函式：

```
int Bob(std::vector<int> t)
```

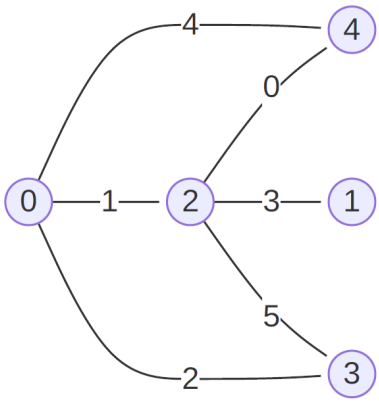
- t：長度為 m 的陣列，表示相異註標，其中 $0 \le t[i] < n$，且對於$i \ne j$ 則 $t[i] \ne t[j]$。
- 此函式回傳單一整數 j，其中 $0 \le j < e$。
- 此函式可被呼叫多次。

範例 (Example)

考慮下列呼叫實例：

```
Alice(5, 6, [4, 0, 3, 1, 4, 2], [2, 2, 0, 2, 0, 3],
      10, [8, 2, 7, 6, 1, 5, 0, 9, 3, 4])
```

同時給定下列之圖：



且 p 一開始是 $[8, 2, 7, 6, 1, 5, 0, 9, 3, 4]$。

給定上述條件後，可證明最佳策略下的最大積分為 1。

假設 Alice 操作下列 4 回合：

Bob 的參數 t	**Bob** 的回傳值	p 中**Bob**欲交換的註標	交換後的 p
$[3, 1, 5, 2, 0]$	5	$5, 2$	$[8, 2, 5, 6, 1, 7, 0, 9, 3, 4]$
$[9, 3, 7, 2, 1]$	0	$1, 7$	$[8, 9, 5, 6, 1, 7, 0, 2, 3, 4]$
$[5, 6, 7, 8, 9]$	1	$5, 7$	$[8, 9, 5, 6, 1, 2, 0, 7, 3, 4]$
$[7, 5, 2, 3, 6]$	3	$5, 2$	$[8, 9, 2, 6, 1, 5, 0, 7, 3, 4]$

注意，在上述對弈過程中，雙方並非使用最佳策略，此處僅為了展示與說明。同時注意，假如一開始所給定的排列的積分已經是 1，那麼 Alice 可以立刻中止遊戲。

當 Alice 完成上述操作，此時積分為 3(由於 $p[2]=2$、$p[5]=5$、$p[7]=7$ 之故)。

最終，函式 `Alice()` 將回傳數值 1，也就是雙方皆以最佳策略對奕後之最佳積分。

注意，在上述範例中，雖然實際的最終積分為 3，但如果函式 **`Alice()`** 回傳數值 3 (而不是1)，你將會得到 0 分。

限制 (Constraints)

- $2 \le m \le 400$
- $m-1 \le e \le 400$
- $0 \le u[i], v[i] < m$
- $m \le n \le 400$
- $0 \le p[i] < n$
- 輸入為連通圖，並且沒有自環(self-loops，自己連到自己的邊)與多重邊(multiple edges，兩點間有許多邊)。
- p 為一排列，亦即對於任意 $i \ne j$， 則 $p[i] \ne p[j]$。

子任務 (Subtasks)

1. (6 分) $m=2$
2. (6 分) $e>m$
3. (10 分) $e=m-1$
4. (24 分) $e=m=3$
5. (24 分) $e=m=4$
6. (30 分) $e=m$

對於每項子任務，你可得到部分給分。就每項子任務而言，令 r 為此處所有測資中比值 $\frac{k}{n}$ 的最大值，其中 k 為該測資的總回和數(亦即呼叫`Bob()`的次數)。 那麼，你在該子任務實際的得分則為原始配分乘上下列係數：

條件	係數
$12 \le r$	0
$3 < r < 12$	$1-\log_{10}(r-2)$
$r \le 3$	1

換言之，若你總是能在 $3n$ 回合內(含)解出，那麼你將獲得該項子任務配分的滿分。同樣的，若你需要用到超過 $12n$ 回合，那麼該子任務將獲得 0 分(系統會顯示 `Output isn't correct`)。

範例評分程式 (Sample Grader)

範例評分程式的輸入格式如下：

- line 1: $m\ e$
- line $2+i$ (其中 $0 \le i \le e-1$)： $u[i]\ v[i]$
- line $2+e$: n
- line $3+e$: $p[0]\ p[1]\ \ldots\ p[n-1]$

範例評分程式的輸出格式如下：

- line 1 : 遊戲結束時的排列 p
- line 2: 函式 `Alice()` 的回傳值
- line 3: 遊戲結束時的積分
- line 4: 總回合數