ZeroJudge c291 小群體

ZeroJudge c291 小群體 https://zerojudge.tw/ShowProblem?problemid=e283

106/03/04 APCS #2 小群體 https://apcs.csie.ntnu.edu.tw/wp-content/uploads/2018/12/1060304APCSImplementation.pdf

丁培毅

110/08

問題描述

- 一群人在一起時經常會形成一個一個的小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**									

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**				**6**					

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**				**6**		**8**			

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**				**6**		**8**		**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**				**6**	**0**	**8**		**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**				**6**	**0**	**8**		**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**			**6**	**0**	**8**		**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**			**6**	**0**	**8**	**1**	**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**			**6**	**0**	**8**	**1**	**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**		**9**	**6**	**0**	**8**	**1**	**5**	

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**		**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**		**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。

問題描述

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

問題描述

範例一輸入
10
4 7 2 9 6 0 8 1 5 3

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

- 輸入每個人的好友編號

問題描述

範例一輸入
10
4 7 2 9 6 0 8 1 5 3

範例一輸出
4

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

- **輸**入每個人的好友編號，**計算**出共有幾個小群體。

問題描述

範例一輸入
10
4 7 2 9 6 0 8 1 5 3

範例一輸出
4

範例二輸入
3
0 2 1

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

- **輸**入每個人的好友編號，**計算**出共有幾個小群體。

問題描述

範例一輸入
10
4 7 2 9 6 0 8 1 5 3

範例一輸出
4

範例二輸入
3
0 2 1

範例二輸出
2

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

- **輸**入每個人的好友編號，**計算**出共有幾個小群體。

問題描述

範例一輸入
10
4 7 2 9 6 0 8 1 5 3

範例一輸出
4

範例二輸入
3
0 2 1

範例二輸出
2

- 一群人在一起時經常會形成一個一個的小群體。假設有 **n** 個人，編號由 **0** 到 **n-1**，每個人都寫下他最好朋友的編號(如果他自己沒有其他好友，最好朋友就是他自己)，例如 n=10，好友編號如下：在本題中，每個人的好友編號絕對不會重複，也就是說 0 到 n-1 每個數字都恰好出現一次。這種好友的關係會形成一些小群體。

	0	1	2	3	4	5	6	7	8	9
好友編號	**4**	**7**	**2**	**9**	**6**	**0**	**8**	**1**	**5**	**3**

- 0 的好友是 4，4 的好友是 6，6 的好友是 8，8 的好友是 5，5 的好友是 0，所以 0、4、 6、8、和 5 就形成了一個小群體；另外，1 的好友是 7 而且 7 的好友是 1，所以 1 和 7 形成另一個小群體；同理，3 和 9 是一個小群體；而 2 的好友是自己，因此他自己是一個小群體。在這個例子裡總共有 4 個小群體：{0,4,6,8,5}、{1,7}、{3,9}、{2}。

- **輸**入每個人的好友編號，**計算**出共有幾個小群體。
- 時間限制: 1 sec

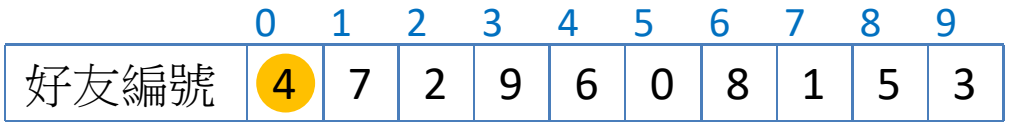
提示與分析

- 範例

	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

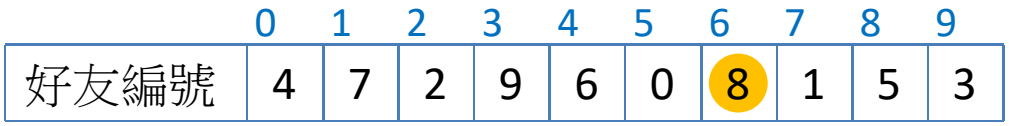
提示與分析

- 範例

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

提示與分析

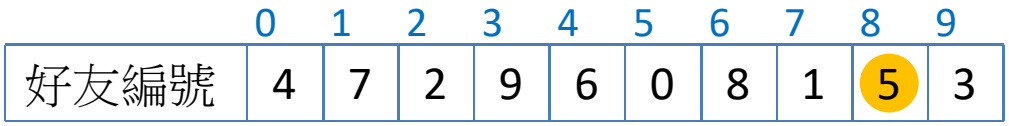
- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

提示與分析

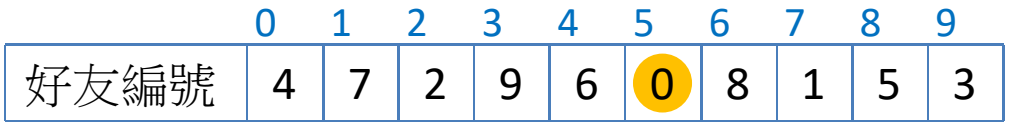
- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

提示與分析

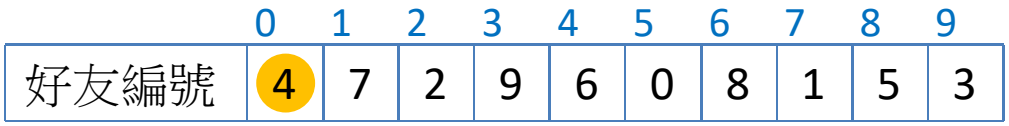
- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

提示與分析

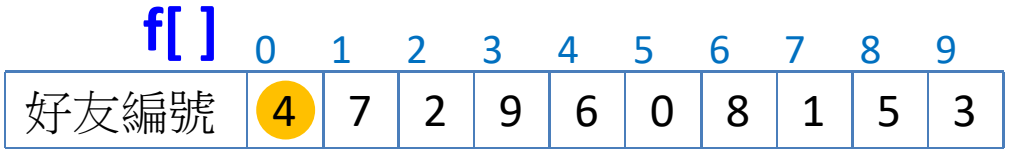
- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

提示與分析

- 範例

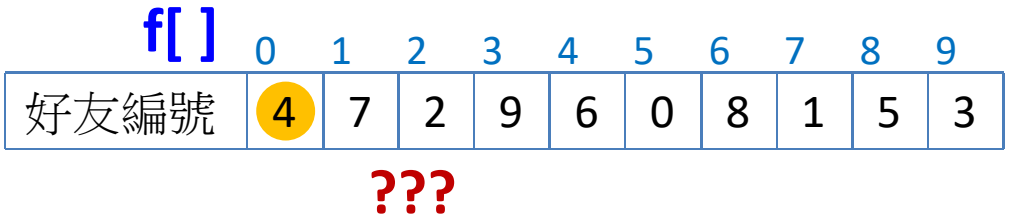


- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

提示與分析

- 範例



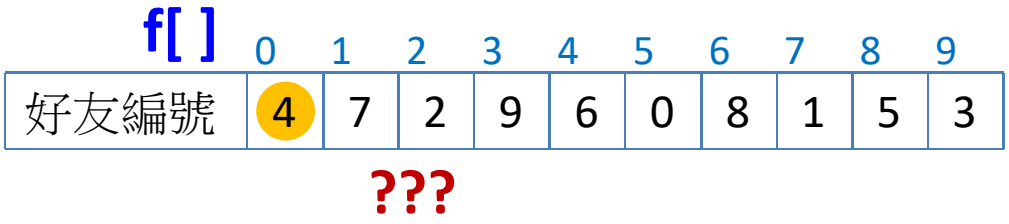
- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3

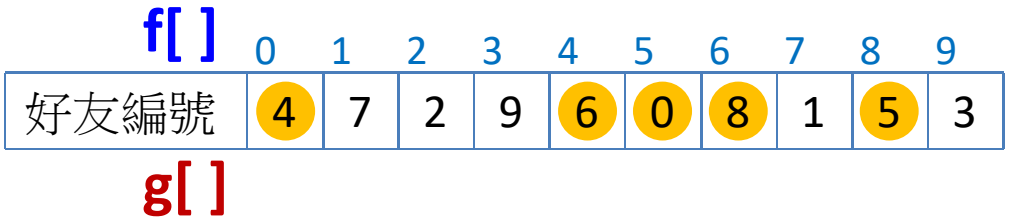
- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

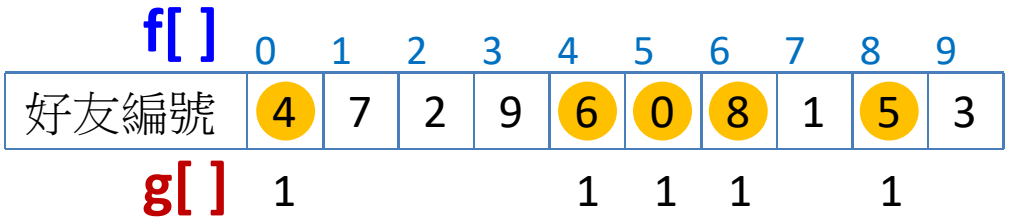
```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

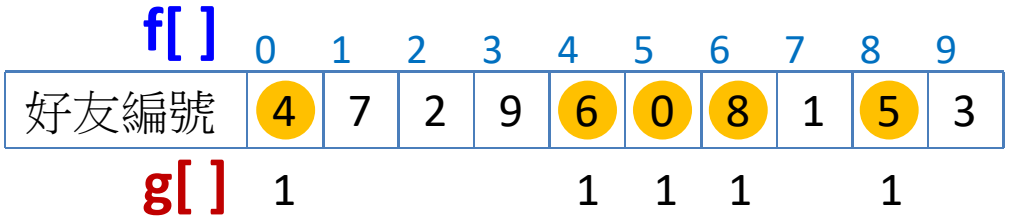
```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

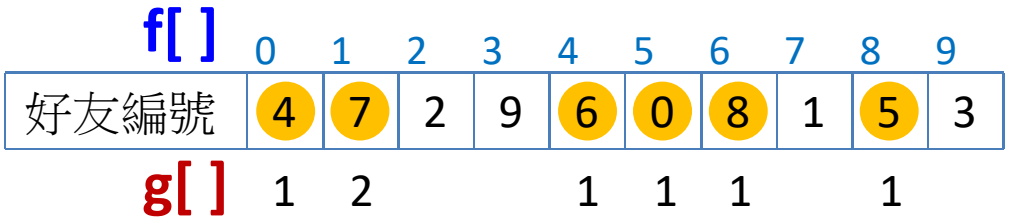
- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

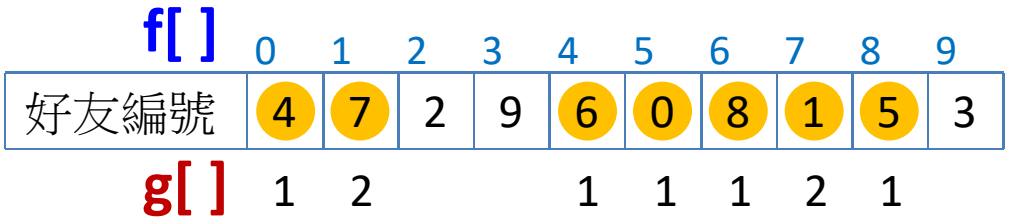
- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

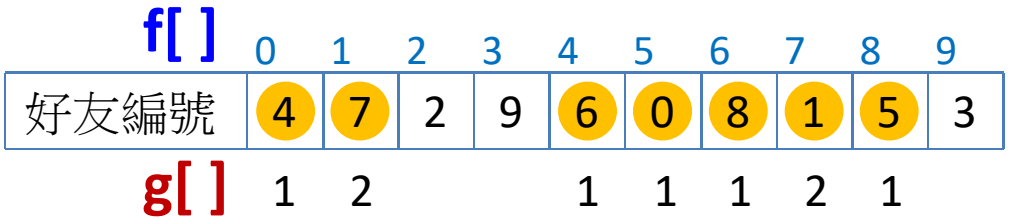
- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例



- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例

	f[ ] 0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3
g[ ]	1	2	3		1	1	1	2	1	

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例

	f[ ] 0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3
g[ ]	1	2	3		1	1	1	2	1	

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例

	f[ ] 0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3
g[ ]	1	2	3	4	1	1	1	2	1	

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3
g[ ]	1	2	3	4	1	1	1	2	1	4

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

提示與分析

- 範例

f[ ]	0	1	2	3	4	5	6	7	8	9
好友編號	4	7	2	9	6	0	8	1	5	3
g[ ]	1	2	3	4	1	1	1	2	1	4

- 如果從任何一人 x 開始，追蹤他的好友，好友的好友，…，這樣一直下去一定會形成一個圓圈回到 x，這就是一個小群體

```
1 x=0,y=f[x];
2 while (y!=x)
3     y=f[y];
```

- 當一個小群體追蹤完之後，需要由任何一個還未被追蹤過的人開始，執行前面的步驟找下一個小群體
  - 追蹤過程中需要把遇見的人標記起來，才知道哪些人已經追蹤過，才能夠找到還沒有被追縱過的人

```
1 int x=0,y,c=1,g[50000]={0};
2 g[x]=c, y=f[x];
3 while (y!=x)
4     g[y]=c, y=f[y];
5 for (x=0; x<n; x++)
6     if (g[x]==0) break;
```

重複上述步驟直到追蹤完所有人

實作

- 減少記憶體使用: 陣列 **f[ ]** 除了紀錄好友編號之外, 也可以用來記錄其群組編號

實作

- 減少記憶體使用: 陣列 **f[ ]** 除了紀錄好友編號之外, 也可以用來記錄其群組編號; 但是這樣子設計使得群組編號和好友編號完全無法分辨, 無法藉此找出哪些人還不在群組中

實作

- 減少記憶體使用: 陣列 **f[ ]** 除了紀錄好友編號之外, 也可以用來記錄其群組編號; 但是這樣子設計使得群組編號和好友編號完全無法分辨, 無法藉此找出哪些人還不在群組中, 所以我們將**群組編號設為負數: -1, -2, …**, 找第一個小群組的程式如右

```
1 int x,y,t,c=1;
2 for (x=0; x<n; x++)
3     if (f[x]>=0) break;
4 y=f[x], f[x]=-c;
5 while (y!=x)
6     t=f[y], f[y]=-c, y=t;
```

實作

- 減少記憶體使用: 陣列 **f[ ]** 除了紀錄好友編號之外, 也可以用來記錄其群組編號; 但是這樣子設計使得群組編號和好友編號完全無法分辨, 無法藉此找出哪些人還不在群組中, 所以我們將**群組編號設為負數: -1, -2, …**, 找第一個小群組的程式如右

```
1 int x,y,t,c=1;
2 for (x=0; x<n; x++)
3     if (f[x]>=0) break;
4 y=f[x], f[x]=-c;
5 while (y!=x)
6     t=f[y], f[y]=-c, y=t;
```

- 另外運用**變數 p** 紀錄已確認為小群組成員的人數, 此人數可用來判斷迴圈是否繼續

實作

- 減少記憶體使用: 陣列 **f[ ]** 除了紀錄好友編號之外, 也可以用來記錄其群組編號; 但是這樣子設計使得群組編號和好友編號完全無法分辨, 無法藉此找出哪些人還不在群組中, 所以我們將**群組編號設為負數: -1, -2, …**, 找第一個小群組的程式如右

```
1 int x,y,t,c=1;
2 for (x=0; x<n; x++)
3     if (f[x]>=0) break;
4 y=f[x], f[x]=-c;
5 while (y!=x)
6     t=f[y], f[y]=-c, y=t;
```

- 另外運用**變數 p** 紀錄已確認為小群組成員的人數, 此人數可用來判斷迴圈是否繼續

```
1 int p=0,x,y,t,c;
2 for (c=1; p<n; c++) {
3     for (x=0; x<n; x++)
4         if (f[x]>=0) break;
5     y=f[x], f[x]=-c, p++;
6     while (y!=x)
7         t=f[y], f[y]=-c, p++, y=t;
8 }
```

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完
  - 每次由一個還沒有追蹤過的人開始追蹤時，不見得是一個新的群組，如果追蹤的過程中遇見任何一個已經標示在某一群的好友，就要使用那個群組的編號

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完
  - 每次由一個還沒有追蹤過的人開始追蹤時，不見得是一個新的群組，如果追蹤的過程中遇見任何一個已經標示在某一群的好友，就要使用那個群組的編號
- 另一個資料表示法接近的延伸就是有名的「約瑟夫問題」

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完
  - 每次由一個還沒有追蹤過的人開始追蹤時，不見得是一個新的群組，如果追蹤的過程中遇見任何一個已經標示在某一群的好友，就要使用那個群組的編號
- 另一個資料表示法接近的延伸就是有名的「約瑟夫問題」
  - 概念上有些接近但是執行時間要求嚴格時需要使用特別的資料結構 (樹狀樹組或是線段樹)

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完
  - 每次由一個還沒有追蹤過的人開始追蹤時，不見得是一個新的群組，如果追蹤的過程中遇見任何一個已經標示在某一群的好友，就要使用那個群組的編號
- 另一個資料表示法接近的延伸就是有名的「約瑟夫問題」
  - 概念上有些接近但是執行時間要求嚴格時需要使用特別的資料結構 (樹狀樹組或是線段樹)
  - APCS 2016/10 #3 定時 K 彈, ZeroJudge c296

延伸

- 這個題目最直接的延伸就是去除「好友編號不會重複」的限制
  - 在追蹤的時候一群中的人不見得可以一次追蹤完
  - 每次由一個還沒有追蹤過的人開始追蹤時，不見得是一個新的群組，如果追蹤的過程中遇見任何一個已經標示在某一群的好友，就要使用那個群組的編號
- 另一個資料表示法接近的延伸就是有名的「約瑟夫問題」
  - 概念上有些接近但是執行時間要求嚴格時需要使用特別的資料結構 (樹狀樹組或是線段樹)
  - APCS 2016/10 #3 定時 K 彈, ZeroJudge c296
  - https://zh.wikipedia.org/wiki/約瑟夫問題