

题目链接: <https://leetcode-cn.com/problems/couples-holding-hands/>



n对情侣，手拉手成环，至少需要n-1次交换

证明：n = 1对情侣手拉手成环，ab，最少0次交换

n = 2对情侣手拉手成环，ac，bd，最少1次交换

n = 3对情侣手拉手成环，ac，be，df，最少2次交换

n = k(k ≥ 3)对情侣手拉手成环，最少要k-1次交换

n = k+1对情侣手拉手成环，凑成一对，变成k对成环，需要k-1+1=k次交换

n对情侣，手拉手成环，通过每次凑成一对的方式可以获得最少的交换次数

证明：每次凑成一对，最后剩两对时，一次交换可以凑两对，最少的交换次数就是n-1次

n对情侣中，有m个环，则至少n-m次交换。

证明：n对情侣，

m=1，最少n-1次交换

m=2，最少n-2次交换

$m=k$, 最少 $n-k$ 次交换

$m=k+1$, 每个环内部交换, 为 $n-k-1$ 次交换, 若环之间进行交换, 将 $k+1$ 个环, 分成1个包含 x 对情侣的环和包含 $n-x$ 对情侣的 k 个环, 1个环和 k 个环中的任意环进行交换都不会凑成情侣, 且只会将两个交换的环构成更大的环, 那么原来的 $k+1$ 个环就变成了 k 个环, 最少需要 $n-k$ 次交换, 总计 $n-k+1$ 次交换。所以只能每个环内部进行交换, 可以获得最少的交换次数。

因此我们可以得到以下两点:

- n 对情侣, m 个环, 最少的交换次数为 $n-m$ 次
- 通过每次凑成一对的策略, 可以以最少的交换次数把所有情侣凑成对

于是问题变成了如何求得环个数, 即连通个数的问题。或者每次凑成一对的交换总个数问题。

解题思路一:

广度优先搜索。

时间复杂度: $O(N)$ 。 N 是情侣对数。

空间复杂度: $O(N)$ 。

```
1 class Solution:
2     def minSwapsCouples(self, row: List[int]) -> int:
3         l = len(row)
4         graph = [i for i in range(l)]
5         flag = [False for _ in range(l)]
6
7         for i in range(0, l, 2):
8             graph[row[i]] = row[i+1]
9             graph[row[i+1]] = row[i]
10
11         count = 0
12
13         for i in range(0, l, 2):
14             if not flag[i]:
15                 q = collections.deque()
16                 q.append(i)
17                 while len(q) != 0:
18                     top = q[0]
19                     flag[top] = True
20                     q.popleft()
21                     if not flag[graph[top]]:
22                         q.append(graph[top])
23                     if not flag[top^1]:
24                         q.append(top^1)
25                 count += 1
26
27         return l // 2 - count
```

解题思路二:

并查集

时间复杂度: $O(N \log N)$ 。 N 是情侣对数。

空间复杂度: $O(N)$ 。

```
1 class unionFind:
```

```

2     def __init__(self, n):
3         self.root = [i for i in range(n)]
4         self.count = n
5     def union(self, x, y):
6         if self.find(x) != self.find(y):
7             self.count -= 1
8             self.root[self.find(x)] = self.find(y)
9     def find(self, x):
10        while self.root[x] != x:
11            x = self.root[x]
12        return x
13    def getCount(self):
14        return self.count
15
16    class Solution:
17        def minSwapsCouples(self, row: List[int]) -> int:
18            couples = len(row) // 2
19            uf = unionFind(couples)
20            for i in range(couples):
21                uf.union(row[i*2] // 2, row[i*2+1] // 2)
22            return couples - uf.getCount()

```

解题思路三：

元素交换。由广度优先搜索建图的思想，我们可以知道，如果每次通过凑成一对的方式进行交换，其实就是在环内操作，从左到右不断凑成对，假设第 i 个环的对数为 X_i ，则每次凑成一对，交换 X_i-1 次就能将环内的所有对凑好。累加所有的环，则总的交换次数为 $n-m$ 次， n 是总的对数， m 是环的个数。我们通过数学归纳法证明了 $n-m$ 是最小的交换次数，所以，模拟元素交换就可以得到最优解。

时间复杂度： $O(N)$ 。 N 是情侣对数。

空间复杂度： $O(N)$ 。

```

1    class Solution:
2        def minSwapsCouples(self, row: List[int]) -> int:
3            l = len(row)
4            pos = [i for i in range(l)]
5            for i in range(l):
6                pos[row[i]] = i
7            count = 0
8            for i in range(0, l, 2):
9                if row[i]^1 != row[i+1]:
10                    pos[row[i+1]] = pos[row[i]^1]
11                    row[pos[row[i]^1]], row[i+1] = row[i+1], row[i]^1
12                    count += 1
13            return count

```