765. Couples Holding Hands

- Difficulty:Hard
- Total Accepted:811
- Total Submissions:1.9K
- Contributor:awice
-
- Subscribe to see which companies asked this question.

Related Topics Graph

N couples sit in 2N seats arranged in a row and want to hold hands. We want to know the minimum number of swaps so that every couple is sitting side by side. A *swap* consists of choosing **any** two people, then they stand up and switch seats.

The people and seats are represented by an integer from `0` to `2N-1`, the couples are numbered in order, the first couple being `(0, 1)`, the second couple being `(2, 3)`, and so on with the last couple being `(2N-2, 2N-1)`.

The couples' initial seating is given by `row[i]` being the value of the person who is initially sitting in the i-th seat.

Example 1:

```
Input: row = [0, 2, 1, 3]
Output: 1
Explanation: We only need to swap the second (row[1]) and third (row[2]) person.
```

Example 2:

```
Input: row = [3, 2, 0, 1]
Output: 0
Explanation: All couples are already seated side by side.
```

Note:

1. `len(row)` is even and in the range of `[4, 60]`.
2. `row` is guaranteed to be a permutation of `0...len(row)-1`.

Seen this question in a real interview before?

A naive approach is to iterate all the pairs and swap if the couple in pair has a distance larger than 1, swap the right partner

```
int minSwapsCouples(vector<int>& row) {
    int swaps = 0;
    unordered_map<int,int> places;
    for(int i=0;i<(int)row.size();++i)
    {
        places[row[i]]=i;
    }
    for(int i=0;i<(int)row.size();i++)
    {
        int x = row[i]; int y;
        if(x%2==0) y=x+1;
        else y=x-1;
        int j = places[y];
        //swap
        if(abs(i-j)>1)
        {
            swap(row[i+1],row[j]);
            places[row[i+1]]=i+1;
            places[row[j]]=j;
            swaps++;
        }
    }
    return swaps;
}
```

python

```
class Solution:
    def minSwapsCouples(self, row):
        """
        :type row: List[int]
        :rtype: int
        """
        nb_swap = 0
        place = {x:i for (i,x) in enumerate(row)}
        for i in range(len(row)):
            x = row[i]

            # find y partner of x :
            if x % 2 == 0:
                y = x + 1
            else:
                y = x - 1
            j = place[y]

            # if need a swap
            if abs(i-j) > 1:
                row[i+1], row[j] = row[j], row[i+1]
                place[row[i+1]] = i+1
                place[row[j]] = j
                nb_swap += 1

        return nb_swap
```