Couples Holding Hands

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`.

Solution 1

Since the solution that simply searches and swaps is accepted, why the difficulty of this problem is hard?

When I see "hard", I expect that something like DFS or DP is required. Could be a missing test case?

Update 1: I posted below an inductive proof that the simple search and swap solution gives the correct result.

Also, OJ accepted the O (n * n) solution, may be the TLE limit is too high?

Update 2: Here is the O(n) solution I came up with : C++ O(n) unordered_multimap.

Update 3: An elegant development of the solution above, but the intuition is hard to get: [Monster Style] C++ O(n) unordered_map.

written by votrubac original link here

Solution 2

```
class Solution {

    public int minSwapsCouples(int[] row) {
        int n = row.length;
        int[] pos = new int[n];
        for (int i = 0; i < n; i++) {
            pos[row[i]] = i;
        }
        int count = 0;
        for (int i = 0; i < n; i += 2) {
            int j = row[i] % 2 == 0 ? row[i] + 1 : row[i] - 1;
            if (row[i + 1] != j) {
                swap(row, pos, i + 1, pos[j]);
                count++;
            }
        }
        return count;
    }

    void swap(int[] row, int[] pos, int x, int y) {
        int temp = row[x];
        pos[temp] = y;
        pos[row[y]] = x;
        row[x] = row[y];
        row[y] = temp;
    }
}
```

written by ibmtp380 original link here

Solution 3

```
    int minSwapsCouples(vector<int>& row) {
        for (int& i: row)
            i /= 2;
        unsigned len = 0;
        for (auto it = row.begin(); it != row.end(); it += 2)
            if (*it != *(it+1)) {
                auto toswap = find(it+2, row.end(), *it);
                iter_swap(it+1, toswap);
                ++len;
            }
        return len;
    }
```

written by fuchedzhy original link here