Algorithm 1 - Connecting Pairs of Persons

Carlos Lopez
Angel Penaloza
Ryan Monte
Boushra Bettir

carlosjlopezz27@csu.fullerton.edu
angelpenlza@csu.fullerton.edu
boushra.bettir04@csu.fullerton.edu
ryanmonte@csu.fullerton.edu

Pseudocode for Algorithm 1: Connecting Pairs of Persons

```
Function minSwaps(row) :
    position = {}
    count = 0

    For i from 0 to len(row) - 1:
        #created indices at each position index for each element at i
        Set position[row[i]] = i

    For n from 0 to (len(row) - // 2) - 1: # this is making it into pairs
        first = row[2*n] #first element of pair
        second = row[2*n+1] #second element of pair
        #check to see if the pairs are in correct order
        If (first is even & second is not first + 1) or (first is odd & second is not first - 1):
            If (first is even):
                target = position[first+1] #pos of next odd element
            Else:
                target = position[first-1] #pos of prev first element

            Swap second and row[target]

            position[second] = 2*n+1
            Position[row[target]] =  target

            count += 1 #since we swapped, we update count by 1

    Return count
```

Proof with Limits

def minSwaps (self, row):

- **Initialization**: position dictionary & row & count = $O(n)$

for i from 0 to len(row) - 1 : → $n-1$

- **Time complexity**: $O(n_{-1}) = O(n)$

for n from 0 to (len(row) // 2) - 1 : → $\frac{n}{2} - 1$

- **Inside loop**
  - checking order → $O(1)$
  - target position → $O(1)$
  - swap → $O(1)$
  - update dictionary → $O(1)$

Total = $O(\frac{n}{2}) + 3O(1) = O(n)$

$$T(n) = O(n) + O(n) + O(n) = O(n)$$

Proof:

$$\lim_{n \to \infty} \frac{T(n)}{n} = \lim_{n \to \infty} \frac{n \cdot c}{n} = c$$ which is a constant

This implies case 2 where $T(n)$ has the same order of growth as the efficiency class $O(n)$

Thus, minSwaps() $\in O(n)$

Time complexity ↑