Tyler Acuna
Tacuna@csu.fullerton.edu

Algorithm 1: Connecting Pairs of Persons

Pseudocode:

Problem: Determine the minimum number of swaps required to ensure that all couples are seated together in a row.

Input: An array/vector of integers representing individuals in seats, where consecutive integers represent couples.

Output: The minimum number of swaps needed to arrange the couples side by side.

Constraints: The number of individuals is always even, with unique, consecutive IDs for each person in a 2n-sized array/vector.

```
Function FindPartnerPosition(row, partner)
for each seat in row
   if the person in the seat is the partner
      return the seat number

Function ConnectCouples(row)
swaps = 0
for i from 0 to length of row - 1, step by 2
   person1 = row[i]
   partner = find the partner of person1

   if row[i + 1] is not the partner
         partnerPosition = FindPartnerPosition(row, partner)
         swap the person in seat i + 1 with the partner
         increase swap count by 1
return swaps
```

Tyler Acuna
Tacuna@csu.fullerton.edu

Proving Efficiency Using Step Counts

```
Function FindPartnerPosition(row, partner)
for each seat in row                                  // n
   if the person in the seat is the partner           // 1 comparison per iteration
      return the seat number                          // 1

Function ConnectCouples(row)
swaps = 0                                             // 1
for i from 0 to length of row - 1, step by 2          // n/2
   person1 = row[i]                                   // 1
   partner = find the partner of person1              // 1

   if row[i + 1] is not the partner                   // 1
         swap(row[i + 1], row[partnerPosition])       // n
         swap the person in seat i + 1 with the partner  // 1
         increase swap count by 1                     // 1
return swaps                                          // 1
```

<u>FindPartnerPosition Function</u>
O(n + 1 + 1) = **O(n) for worst case time complexity.**

<u>Function ConnectCouples(row)</u>
First we'll handle the loop. The number of iterations will be n/2. Taking the rest of the steps inside the loop: O(1 + 1 + 1 + n + 1 + 1) = O(n + 5).

Since the number of iterations will be iterations is n/2, we'll multiply that with (n + 5).

T(n) = n/2 * (n + 5) = (n^2)/2 + 5n/2.

We'll add the 1 for initializing swaps and the 1 for returning swaps.

T(n) = (n^2)/2 + 5n/2 + 2

This will make this function **O(n^2) for worst case time complexity.**

<u>Overall:</u>
T(n) = O(n) + O(n^2)

O(n^2) would be the overall worst case time complexity.