Project report

Algorithm 1

Group 1

1. Name

1) Chanran Kim (shining04@csu.fullerton.edu)
2) Vinh Nguyen (vinhgod123@csu.fullerton.edu)
3) Brian Alvarez (briandalvarez@csu.fullerton.edu)
4) Christopher Contreras (CContreras71@csu.fullerton.edu)

2. Pseudocode

Problem : To find the minimum swaps needed to seat `n` couples together in a row of `2n` seats, where couples are defined by consecutive IDs like `(0, 1)`, `(2, 3)`, etc.

Inputs:

- An array row with a length of $2n$, where n is the number of couples.
- row[i] represents the ID of the person sitting in the ith seat.
- Couples are represented as (2k, 2k+1), where k is the index of each couple.

Outputs:

- The minimum number of swaps required to make every couple sit next to each other (integer value).

Constraints and Assumptions:

- The array row has a length of $2n$ (even number), where n is the number of couples.
- IDs range from 0 to $2n$-1 and each ID appears exactly once in the array.
- Couples are defined as consecutive pairs: (0, 1), (2, 3), and so on.
- A swap involves exchanging two people in the array.

Pseudocode :

```
function minSwapsCouples(row):
```

```
    n = length of row / 2
    couple_map = {}
    swaps = 0


        couple_map[row[i]] = i


    for i from 0 to length of row step 2:
        first_person = row[i]
        second_person = row[i + 1]
        couple_id = first_person ^ 1

        if second_person != couple_id:
            swaps += 1
            swap_pos = couple_map[couple_id]
            row[i + 1], row[swap_pos] = row[swap_pos], row[i + 1]



            couple_map[second_person] = swap_pos
            couple_map[couple_id] = i + 1

    return swaps

function minSwapsCouples(row):
    n = length of row / 2
    couple_map = {}
    swaps = 0

    for i from 0 to length of row:
        couple_map[row[i]] = i

    for i from 0 to length of row step 2:
        first_person = row[i]
        second_person = row[i + 1]
        couple_id = first_person ^ 1

        if second_person != couple_id:
            swaps += 1
            swap_pos = couple_map[couple_id]

            row[i + 1], row[swap_pos] = row[swap_pos], row[i + 1]
```

```
            couple_map[second_person] = swap_pos
            couple_map[couple_id] = i + 1

        return swaps
```

3. Proving efficiency of the pseudocode

` Time complexity

1. O(n) - this loop goes through the row of length n where n is the number of people.
2. O(n) - this loop goes through row in 2 steps, meaning n/2, but since the operation loop is the constant time, it becomes O(n/2) simplifies to O(n)
3. O(n) + O(n) = O(n) - combining both steps

Space Complexity

1. O(2n) = O(n) - couple_map stores each person and their index, using space proportional to the number of people in the row
2. O(n)

Time complexity: O(n)
Space Complexity: O(n)