Pseudo Code:

1) Set count to 0.
2) Create a map with index as key and num[index] as value
3) for every element in nums - 1.
   a) get what should be the neigbor at element i.
   b) get what the neighbor element of element i is i.e. what is element i+1.
   c) if (actual neigbor != expected neighbor) {
      1) Find what index the expected neighbor is in.
      2) Swap element i+1 with the element that is in the expected index.
      3) update the map.
      4) Increment count.
   }
4) return count.

```
void printVec(std::vector<int>& nums) {
    for (const auto& num : nums) {
        std::cout << num << " "; // O(1) per iteration
    }                            // O(n) for the entire loop
    std::cout << "\n\n";         // O(1)
}

// helper function for the expected neighbor
int checkNeighbor(int val) {
    if (val % 2 == 0) {
        return val + 1; // O(1)
    } else {
        return val - 1; // O(1)
    }
}

int swapPair(std::vector<int>& nums) {
    int count = 0;                      // O(1)
    std::unordered_map<int, int> table;    // O(1) to initialize

    for (std::size_t i = 0; i < nums.size(); ++i) {
        table[nums[i]] = i;             // O(1) average per insert -> O(n) total
    }

    for (std::size_t i = 0; i < nums.size(); i = i + 2) {
```

$$O(1) + O(1) + [O(n) \cdot O(1)] +$$

$$O(\tfrac{n}{2}) \cdot [O(1) + O(1) + O(1) + O(1) + O(1) +$$

$$O(1) + O(1) + O(1)] + O(1)$$

$$O(2) + O(n) + O(\tfrac{n}{2}) \cdot O(8) + O(1)$$

```
    for (std::size_t i = 0; i < nums.size(); i = i + 2) {
        int neighborValue = checkNeighbor(nums[i]); // O(1)
        int nextVal = nums[i + 1];                  // O(1)
        if (nextVal != neighborValue) {             // O(1)
            int actualNeighborIndex = table[neighborValue]; // O(1) average lookup
            std::swap(nums[i+1], nums[actualNeighborIndex]); // O(1)
            table[neighborValue] = i + 1;           // O(1)
            table[nextVal] = actualNeighborIndex;   // O(1)
            count++;                                // O(1)
        }
    }
    return count; // O(1)
}
```

$O(3) + O(n) \neq O(n)$

let $f(n)$ = Step count function

$f(n) = 4n + n + 3$

prove $f(n) \leq c \cdot g(n)$ for $n \geq n_0$

let $g(n) = n$

$4n + n + 3 \leq c \cdot n$

$c \geq 4 + 1 + \frac{3}{n}$

$c \geq 5 + \frac{3}{n}$

for $n_0 = 1$

$c \geq 5 + \frac{3}{1}$

$c \geq 8$

Show that $f(n) \leq c \cdot g(n)$ for

$n_0 = 1$ & $c = 8$

$4(1) + (1) + 3 \leq 8(1)$

$8 \leq 8$ ✓

So $f(n) \in O(n)$