Project 1: Algorithm and Analysis

A Report Submitted to:
Professor Dsouza
CPSC 335

Prepared By

Huy Bui
HUYBUI02@csu.fullerton.edu

California State University, Fullerton
February 27, 2025

Algorithm 1: Connecting Pairs of Persons

Psuedo-code:

1) Set $count$ to 0.

2) Create a map where $nums[index]$ is the key and $index$ is the value.

3) For every other element i in $nums - 1$:

a) Get $expected\_neighbor$ for position $i + 1$.

b) Get $actual\_neighbor$ at $nums[i + 1]$.

c) If $(actual\_neighbor \neq expected\_neighbor)$:

1. Find $target\_idx$ of $expected\_neighbor$ using map.
2. Swap $nums[i + 1]$ and $nums[target\_idx]$.
3. Update the map with new positions.
4. Increment $count$.

4) Return $count$.

Mathematical Analysis:

$$O(1) + O(1) + [O(n) \cdot O(1)]+$$
$$O\left(\frac{n}{2}\right) \cdot [O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1)] + O(1)$$

$$O(2) + O(n) + O\left(\frac{n}{2}\right) \cdot O(8) + O(1)$$
$$O(3) + O(n) + O(4n)$$

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 $n_0 = 1$**:**

$c \geq 5 + \frac{3}{1}$
$c \geq 8$

$f(n) \leq c \cdot g(n)$ **for** $n_0 = 1$ **and** $c = 8$**:**
$4(1) + 1(1) + 3 \leq 8(1)$
$8 \leq 8 \quad \checkmark$

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

Big O Efficiency Class:
O(N) where N will be the size of the input array.

Algorithm:

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


int swapPair(std::vector<int>& nums) {
   int count = 0;
   std::unordered_map<int, int> table; // create a hash map of value at index and
index
   for (std::size_t i = 0; i < nums.size(); ++i) {
       table[nums[i]] = i;              // populate the table
   }

   for (std::size_t i = 0; i < nums.size(); i = i + 2) { // iterate every left
partner in seats
       int neighborValue = checkNeighbor(nums[i]);        // check what
expected/desired neighbor's value is
       int nextVal = nums[i + 1];                         // check what the actual
neighbor's value is
```

```
        if (nextVal != neighborValue) {                        // if actual is not what the
expected value is
            int actualNeighborIndex = table[neighborValue];      // where the expected
neighbor is
            std::swap(nums[i+1], nums[actualNeighborIndex]);    // swap the expected
neighbor and the actual neighbor
            table[neighborValue] = i + 1;                        // update the value in
hash map
            table[nextVal] = actualNeighborIndex;
            count++;                                             // increment answer
        }
    }
    return count; // return the answer
}
```

Algorithm 2: Greedy Approach to Hamiltonian Problem

Psuedo-code:

1. **Set** Size to nums.size()
2. **Set** Starting_city to 0
3. **Set** fuel_left to 0
4. **For every city** i from 0 to Size $-$ 1 :
   a) Calculate Cost $=$ (fuel[i] $\times$ mpg) $-$ distance[i]
   b) Update fuel_left $=$ fuel_left $+$ Cost
   c) If (fuel_left $<$ 0){
      1. Set Starting_city to $i + 1$
      2. Reset fuel_left to 0
   }
5. **Return** Starting_city

Mathematical Analysis:

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

$O(5) + O(5n) + O(1)$

Let $f(n) =$ step count function $f(n) = 5n + 6$

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

Let $g(n) = n$

$5n + 6 \leq c \cdot n$

$c \geq 5 + \frac{6}{n}$

For $n_0 = 1$**:**

$c \geq 5 + \frac{6}{1}$

$c \geq 11$

Show that $f(n) \leq c \cdot g(n)$ **for** $n_0 = 1$ **and** $c = 11$**:**

$5(1) + 6 \leq 11(1)$

$11 \leq 11 \quad \checkmark$

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

Big O Efficiency Class

O(N) where N will be the size of the input array.

Algorithm:

```
int startingCity(std::vector<int>& distance, std::vector<int>& fuel, int mpg) {
   int size = distance.size();
   int starting_city = 0;
   int fuel_left = 0;
   int cost = 0;
   for (int i = 0; i < size; i++) {
       cost = fuel[i] * mpg - distance[i]; // how much it takes to get to the i-th
city
       fuel_left += cost;                  // how much fuel the car have after
travelling to the i-th city
       if (fuel_left <= 0) {               // if there is no more fuel, that means i
+ 1 cannot be reached from i
           starting_city = i + 1;          // so starting city cannot be i, so we
assume its i + 1 and move to next iteration
           fuel_left = 0;                  // reset fuel
       }
   }
   return starting_city;                   // return answer
}
```