Ryan Lu, Anthony Thornton, Carla Madriz
Group 9
Dr. Sampson Akwafuo
CPSC 335-04
02/25/2024

Emails: rlu132@csu.fullerton.edu, anthonythornton140@csu.fullerton.edu, cmadriz@csu.fullerton.edu

Project 1

Algorithm 1: Connecting Pairs of Persons

Pseudocode:

```
def min_swaps(list):
        total = 0
        for every other element in list:
                if element is even:
                        Pair_value = element + 1
                else:
                        Pair_value = element - 1
                if pair_value == succeeding_element:
                        continue
                pair_element = list.find(value)
                swap (succeeding_element, pair_element)
                total += 1
        return total
```

Proof by Induction for Efficiency Class:

Assuming the find function implemented is a linear search that has O(n) time complexity, we can see that the time complexity of the algorithm is $O(2 + 1 + \frac{1}{2}n(4 + n + 2) + 1)$ which simplifies into $O(4 + 2n + \frac{1}{2}n^2)$. Then using proof by induction:

1. We can see that this closely resembles $O(n^2)$
2. Find a 'c' such that $T(n) \le f(n) * c$, for large n, assume $T(n) = O(4 + 2n + \frac{1}{2}n^2)$, $f(n) = n^2$

   Hence $T(n) \le f(n) * c \Rightarrow 4 + 2n + \frac{1}{2}n^2 \le n^2 * c \Rightarrow \frac{4}{n^2} + \frac{2}{n} + \frac{1}{2} \le c$

   Since $n \ge 1$, we have $\frac{4}{1} + \frac{2}{1} + \frac{1}{2} = \frac{13}{2} = c$, so we assume $n_0 = 1$ and $c = \frac{13}{2}$

3. Solve for base case to check, when $n = 1$ we get, $T(1) = \frac{4}{1} + 2 + \frac{1}{2}(1) = \frac{13}{2}$ and $f(1) * c = 1 * \frac{13}{2}, \frac{13}{2} \leq \frac{13}{2}$. Therefore the base case holds.
4. If $n_0 < n$ and $T(n) \leq f(n) * c$, then $T(n + 1) \leq f(n + 1) * c$
   Plugging in $n + 1$ for n, we have $4 + 2(n + 1) + \frac{1}{2}(n + 1)^2 \leq \frac{13}{2}(n + 1)^2$
   Simplifying further $\frac{13}{2} + 3n + \frac{1}{2}n^2 \leq \frac{13}{2} + 13n + \frac{13}{2}n^2 \Rightarrow 0 \leq 10n + 6n^2$ for $n \geq 1$
5. Therefore by definition of O, $4 + 2n + \frac{1}{2}n^2 \in n^2$

Algorithm 2: Greedy Approach to Hamiltonian Problem

Pseudocode:

```
def preferred_city(array distances, array fuel, int mpg):
        n = length(distances)
        graph = {}
        for every element in (n - 1):
                graph[element] = next_element
        graph[last_element] = first_element

        for index in graph:
                mileage = 0
                visited = []
                while len(visited) < n:
                        mileage += fuel[index]
                        if mileage < distances[index]
                                break
                        mileage −= distances[index]
                        visited.append(index)
                        index = graph[index]
                if len(visited) == n:
                        return index
        return -1
```

Proof by Induction for Efficiency Class

The time complexity for the algorithm is $O(3 + 1 + n + 2 + 4n(6n) + 1)$ which can be simplified into $O(7 + n + 24n^2)$. Then by proof of induction:

1. The complexity resembles $O(n^2)$.

2. Assuming $T(n) \le f(n) * c$, $T(n) = 7 + n + 24n^2$ and $F(n) = n^2$, we have
   $7 + n + 24n^2 \le n^2c$
   Hence $\frac{7}{n^2} + \frac{1}{n} + 24 \le c$, and with $n_0 = 1$, we have $c = 32$
3. Solving for the base case we get $T(1) = 7 + 1 + 24(1)^2$ and $f(1) * c = 1^2 * 32$
   Together we get $32 \le 32$ which is true.
4. If $n_0 < n$ and $T(n) \le f(n) * c$, then $T(n + 1) \le f(n + 1) * c$
   Plugging in $n + 1$ for n, we have $7 + (n + 1) + 24(n + 1)^2 \le 32(n + 1)^2$
   Simplifying we have $24n^2 + 25n + 32 \le 32n^2 + 32n + 32 \Rightarrow 0 \le 8n^2 + 7n$ for $n \ge 1$
5. Therefore by definition of O, $7 + n + 24n^2 \in n^2$