Reem Alsharabi CS3012 Assignment 1

1 Greedy Algorithm for Interval Scheduling

1.1 Time Complexity Analysis

The time complexity of the interval scheduling problem can be analyzed by looking at the time complexity of the algorithm used to solve it.

- The O(n log n) time complexity comes from the fact that the algorithm sorts the intervals by their finish times, which takes O(n log n) time using a standard sorting algorithm such as mergesort or quicksort.
- After sorting the intervals, the algorithm then iterates over the sorted list once, which takes O(n) time.

Therefore, the overall time complexity of the algorithm is O(n log n).

1.2 Code and Output

[4]:
```
def interval_scheduling(intervals):
    intervals.sort(key=lambda x: x[1])  # sort intervals by their end time
    count = 0  # number of non-overlapping intervals
    end_time = -float('inf')
    for interval in intervals:
        if interval[0] >= end_time:
            count += 1
            end_time = interval[1]
    return count

with open('interval_scheduling.txt') as f:
    intervals = [tuple(map(int, line.strip().split(','))) for line in f]
print(interval_scheduling(intervals))
```

```
4
```