Synthesis Assignment 1
Erdun E
September 20, 2024
Dr. Alan Jamieson

CS 5800: Algorithm
Synthesis Assignment 1

1 Asymptotic Analysis

1.1 Big O

1.11 Explanation

Big-O notation is a mathematical concept to describe the efficiency of algorithms in computer science, specially to describe the time-complexity and space-complexity. Big-O notation allows people to discuss in the abstract that how the performance of an algorithm changes when the size grows. For example:

Time-complexity: it refers to the relationship between the execution time of algorithms and input size n.

- O(1): constant time. Whatever the input size, the execution time is always the same.
- O(n): linear time. When the input size increases, the execution time increases linearly as well.
- O(n^2): quadratic time. When the input size increases, the execution time increases quadratically.
- O(logn): logarithmic time. When the input size increases, the execution time increases logarithmically.

Space-complexity: it refers to the relationship between the required memory space and the size of the inputs by an algorithm during execution. The pattern is similar to the time complexity.

Big O is an upper bound notation that describes the maximum resource consumption of an algorithm in the worst case. This is particularly important for measuring the performance of an algorithm under the most unfavorable conditions.

1.12 Exercises

Analyze the time complexity of the Merge Sort algorithm for the following array:

[3,1,4,1,5,9,2,6]

1.13 Solution

Merge sort is a divide-and-conquer sorting algorithm. It works by recursively splitting the array into two halves, sorting each half, and then merging the sorted halves together. The specific step is as follows:

Divide:

- Split the array into two halves that [3,1,4,1] and [5,9,2,6].
- Further split each subarray until it only has one element, such as [3],[1],[4],[1],[5],[9],[2],[6].

Conquer:

- Merge adjacent two subarrays and sort them like [1,3],[1,4],[5,9],[2,6].
- Keep merging adjacent two sorted subarrays, [1,1,3,4],[2,5,6,9].

Combine:

- Merge the two halves, [1,1,2,3,4,5,6,9].

Time Complexity: is O(nlogn)

Divide Step:

- The array will be splited into two subarrays recursively, and each split takes O(1). Due to the array is divided logn time, so the total time is O(nogn).

Merge Step:

- Each step merges n elements of two subarrays, the merging process for each recursion level is linear, so it's O(n).

Total Time Complexity:

- Given there are O(logn) levels need to be splited, and each level takes O(n) to merge, so the total time complexity of Merge Sort is O(nlogn).

Space Complexity: is O(n)

- Merge Sort need additional array to store the auxiliary arrays for merging, so for an array of size n, the additional space requirement is O(n).