**Title and Introduction**

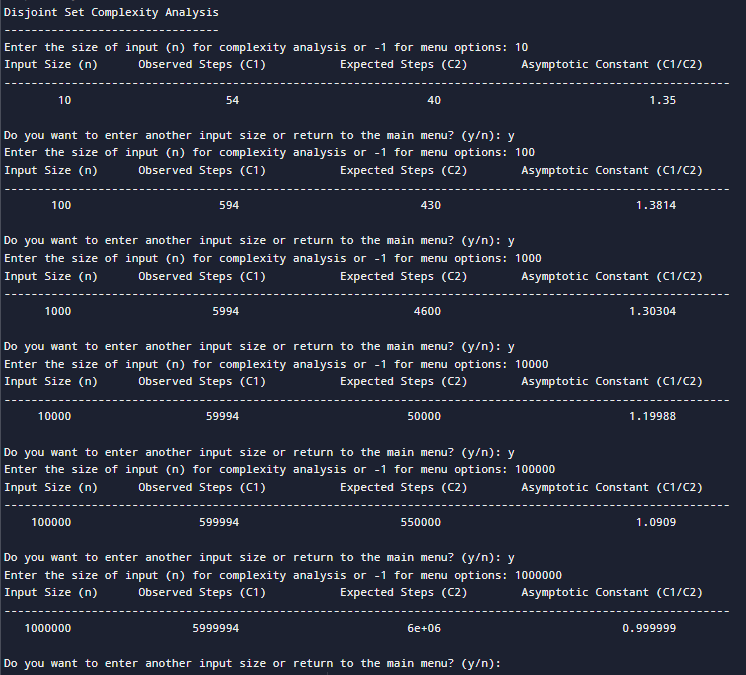
* **Title**: Implementation and Complexity Analysis of Disjoint Set Algorithm

**Introduction**

* **Purpose of the Disjoint Set Algorithm**: The Disjoint Set, also known as Union-Find, is a data structure that keeps track of a partition of a set into disjoint (non-overlapping) subsets. It is particularly useful in scenarios where we need to manage, and merge sets efficiently.
* **Applications**: Applications such as network connectivity, image processing, and Kruskal’s algorithm for finding the Minimum Spanning Tree (MST) in a graph.
* **Operations**:
  + **Find**: Determines which subset a particular element is in. This can be optimized with Path Compression.
  + **Union**: Merges two subsets into a single subset. This can be optimized with Union by Rank or Union by Size.
* **Optimizations**: Importance of Path Compression and Union by Rank/Size in improving the efficiency of the algorithm. These optimizations ensure that the operations run in nearly constant time, making the Disjoint Set very efficient for large datasets.

**Table of Number of Operations for Various Input Sizes (Question 4 (a))**

|  |  |  |  |
| --- | --- | --- | --- |
| **Size of Input n** | **Observed Number of Steps C1** | **Expected Number of Steps, f(n) C2** | **Asymptotic Constant of Proportionality C1/C2** |
| **10** | **54** | **40** | **1.35** |
| **100** | **594** | **430** | **1.3814** |
| **1000** | **5994** | **4600** | **1.30304** |
| **10000** | **59994** | **50000** | **1.19988** |
| **100000** | **599994** | **550000** | **1.0909** |
| **1000000** | **5999994** | **600000006** | **0.999999** |

****

**Conclusion (Question 4 (b))**

**b) State in one sentence if the order of growth of observed number of steps is same as expected from the complexity analysis of the algorithm with more of less stable and well-bounded constant of proportionality.**

* **Observation of Growth Rate**: The observed number of steps for the Disjoint Set operations grows in line with the expected near-constant time complexity, demonstrating the efficiency of path compression and union by rank.