Project: Find All Unique Elements

1 Overview

This OJ Question 897 is used for testing the correctness of your codes.

In this individual project, you are required to explore different methods (at least three) to find out all unique elements in a given list of numbers. **It is not allowed to use any C++ STL containers (e.g., std::vector, std::set, std::map, and others)**.

2 Detailed Requirements

In this project, you are asked to implement and compare different methods of finding out all unique elements in a given list of numbers. In particular, you need to provide at least **three** (i.e., ≥ 3) different methods and compare their performance with time complexity analysis.

For example, to find all unique elements in the list $\{7, 4, 5, 9, 5, 8, 3, 3\}$, you can try using array based brute force approach or binary search tree to find out the targeted output $\{7, 4, 9, 8\}$.

In your submitted report, you need to explicitly explain:

- the algorithms and data structures of the methods you have tried;
- the time complexity of each method (mainly the worst-case time complexity);

Note that you are NOT required to provide formal time complexity analysis, but some intuitive or informal discussion.

Experimental Evaluations

In this project, you may design and generate the test data for each method so that you can evaluate and compare the differences in efficiency among different methods over different generated inputs. Figures that intuitively show how the input data affects the selection among different methods are appreciated but optional. Note that, the compiling command for compiling your programs (written in C++) should be clearly stated in the report. Some available compiling options are -std=c++11, -lm, -O2.

3 OJ Input and Output

Input
The input contains multiple test cases. For each test case, it contains a line of integers (the number of integers is less than 10^4 while each integer is $\in (-10^5, 10^5)$).

Output
Each line of the output is a list of all unique elements. The order of the output elements follows the order in the original input.

Sample Input	Sample Output
-1 1 -1 8	1 8
1 2 3 3 3 4 4 5	1 2 5
2 3 1 5 4 3 2 1	5 4