**SET OPERATIONS**

**Semester Project**

Alex Waweru

Oluwa Oyeluhunnu

Derek Bansah

Joshua Zormelo

Discrete Structures and Theories

Ashesi University College

**Project Summary**

The project’s focus is on the implementation of the set abstract data type, and the main operations of the unordered set data structure; union, difference, compliment, intersection and exclusive\_or. To implement the set abstract data type the authors use the python list, which is an array-based dynamic data structure. The authors explore two implementations, one that converts the subsets into a binary sets of 1’s and 0’s depending on the presence or absence of a particular item in the subset. The binary sets are the used in the set operations. The second implementation uses the subsets in the operations without converting them into binary sets first. The authors perform a complexity analysis to determine the relative effectiveness of each implementation with regards to its utilization of computing resources. Computing resources in this context refers to execution time and storage space. The execution time is evaluated using the Big-O Notation while storage space is evaluated using the number of python objects (or instances of python objects) created all through the implementation. It is important to also note that the project is implemented in python 3.5.4, and it might not be compatible with python 2 or other previous versions of python. This is because python is not backward compatible.

The goal of the project is to cement theoretical knowledge acquired in the Discrete Structures and Theories course. It also provides scholars an opportunity to further their knowledge on sets through research and practical work.

Keywords: set, set operations, python, binary sets, Big-O Notation

**Introduction**

A set is a collection of objects that is viewed as an object in its own right. Take for example, the picture on the left. It is a set of of polygons, and it is considered an object in itself.

Sets are the basic building blocks for the types of objects considered in discrete structures and theories. Sets are important for counting and they are also found in programming languages.

**The Project Scope**

The project involves implementing a superset and set abstract data type. The set abstract data type should have five main operations; union, intersection, difference, compliment and exclusive-or. The final submission should be such that when given a superset U of n elements and A and B, two subsets of U, it should return the following:

1. The compliment of A
2. The union of the two subsets, A and B
3. The intersection of the two subsets, A and B
4. The difference of the two subsets, A and B
5. The exclusive\_or of the two subsets

**The First Implementation:** Use of binary sets

The set is implemented using a python list (equivalent to arrayList in java). The list is initialized to be the size of the superset, and it initially contains 0’s in all positions. When adding an item into the subset, a value 1 is stored in the position where the value occurs in the superset.

For example if you have a superset U = {1,2,3,4,5}, a subset A that contains only item 3 would be represented as A = {0,0,1,0,0}.

When performing a union between two subsets, the it loops through the two subsets and whenever any of the subsets has item 1 at a position the value in that position in the superset would be added to the new union set.

Other operations also make use of the binary sets.

**The Second Implementation:** No binary sets

The set is also implemented using a python list. The list is initially empty and whenever an item is added, the item to be added is appended into the list.

For example if you have a superset U = {1,2,3,4,5}, a subset A that contains only item 3 would be represented as A = {3}

When performing a union between two sets, it adds together the two subsets and then removes all duplicated elements.

Other operations also followed such direct logic.

**Complexity Analysis**

The following represents the Big-O Notations for the operations from the different implementations:

|  |  |  |
| --- | --- | --- |
|  | First Implementation | Second Implementation |
| Union | O(n) | O(n) |
| Difference | O(n) | O(n) |
| Intersection | O(n) | O(n) |
| Compliment | O(n) | O(n) |
| Exclusive-or | O(n) | O(n) |

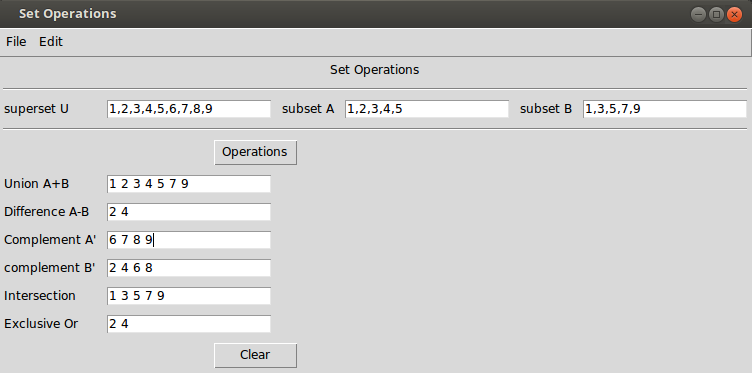
The methods in both implementations have a O(n) runtime, however, the first implementation requires conversion from the binary sets to real sets. The method that converts the binary sets to real sets has a run time of O(n). This makes the each operation have a runtime of 2n. This renders the first implementation more redundant relative to the second implementation.

Space-wise, since the binary sets need to be converted into real sets, extra space is needed to allow for this conversion. This makes the first implementation even worse off in terms of efficiency.

In summary, the second implementation is more efficient in terms of both storage space and execution time.

**Conclusion**

The team went ahead to develop a user interface as shown below. The interface serves the purposes of public demonstration and interactive learning of the set operation.

All in all, the project provided an amazing opportunity for the team members to enrich their knowledge and understanding on sets, set operations, and algorithms analysis.