Assignment 1: Let's Build Sets

COL106: Data Structures and Algorithms, Semester-I 2023–2024

Deadline: 8th Aug 2023 (11:59 PM)

Date Updated: 2nd Aug 2023

1 Introduction

In this assignment, you will build a library for Set data structure. This will involve using the concepts of OOPs that you have learnt in the class.

The goal of this assignment is not to optimise the code for runtime (though we do have some bonus marks for runtime!). Instead, our main focus is on how to implement Data Structures **correctly**.

1.1 Some Notes

For Assignment-1, you are expected to keep following things in mind:

1. Strictly adhere to the input output format. We may decide to auto-grade your assignment. No marks will be given if your code does not conform to the IO guidelines.

2. For evaluation, some test cases will be made available to you to help you debug your program but all the "evaluation test cases" will not be made available to you. You need to ensure that your program works correctly on all the inputs (not just on the test case inputs).

3. ~~We will provide the starter code shortly (on Piazza itself). You are **NOT** supposed to change the filename, function name or return type.~~ Follow the submission instructions as given at the end of this doc.

4. Do ***not*** use the set feature from C++ STL. Name your implemented class as **SET** (in all caps). Not following this instruction shall result in 0 marks.

5. From STL, you are ONLY allowed to use vector and the functions that it provides. Also, you may use sorting by implementing your own sort function (don't use STL here).

6. **IMPORTANT NOTE**: You need to submit an honour code on Gradescope. Remember that internet is your friend! While you are allowed to use Google, GFG, ChatGPT etc., if you choose to avail the help from Internet, your assignment will be graded out of 80 Marks (not out of 100) - and you'll not be eligible for bonus section. Not providing an honour code and taking material from the Internet will have overbearing costs (a Fail/D grade in the course if you are passing the course and referral to DISCO).

2 Sets

2.1 Theory

Set is a container that stores unique elements in a well-defined order. This ordering relation should be transitive and anti-symmetric. Typically, for integers, we store elements in ascending (or sometimes, descending) order.

2.2 Implementation (100 Marks)

Implement Set Data Structure from stratch. The functions to be supported are given in the table on next page. You need to store only unique integers in ascending order.

Input/Output specifications:

Each line in the input contains one or more numbers. The first number represents the command to be executed. The list of the commands to be implemented by you are given in Table 1. The next 1 or 2 (depending on command) numbers shall indicate the set number on which to perform the operation. Thus, you might need to maintain multiple set objects at a time.

~~In case the command doesn't need any data, the rest of line must be ignored.~~ Only adequate amount of data will be given.

2.3 Commands Explanation

Let's walk through each of the command one-by-one.

2.3.1 Insert

1 {set_num} {data}

This command will have 2 arguments. The set_num indicates which set needs to be operated on (if set with this number does not exist already, then create it). "data" indicates the data to be inserted into the set. Remember that after the insertion of data, the set properties must hold - specifically, the set should have unique elements in sorted order.

Code	Command name	Explanation	What is to be printed on output	Command Example
1	Insert	Insert the data into the set	Size of the set	1 1 5
2	Delete	Delete the specified element from the set	Size of the set	2 1 5
3	Belongs To	Check membership of element in the set	1 if element exists, 0 otherwise	3 1 6
4	Union	Take Union of both sets	Size of resultant set	4 1 2
5	Intersection	Take Intersection of both sets	Size of resultant set	5 1 2
6	Size	Number of elements in set	Number of elements in set	6 1
7	Difference	Refer this	Size of resultant set	7 1 2
8	Symmetric Difference	Refer this	Size of resultant set	8 1 2
9	Print	Print elements of set in sorted order	Elements of set	9 1

Table 1: Input output specification of your set library. There will be exactly one command per line. The first number on the line is the code to be used to interpret the command.

2.3.2 Delete

2 {set_num} {data}

Similar to insert. In this case, if set with this number does not exist, return -1. One thing to note here is that - if the data to be deleted does not exist in the set, then do nothing and just print the original size of the set.

2.3.3 Belongs To

3 {set_num} {data}

Check if the element "data" belongs to the set "set_num". If set does not exist, return -1. Return 1 if it belongs, else return 0.

2.3.4 Union

4 {set_num1} {set_num2}

Note that you need to store the result of union operation in set_num1 (This means that set_num2 will remain unchanged, unless set_num1 is same as set_num2). Ensure that the set properties hold even after union. If either of the sets doesn't exist - create them.

2.3.5 Intersection

5 {set_num1} {set_num2}

Similar instructions as Union.

2.3.6 Size

6 {set_num}

If set doesn't exist, create and return 0.

2.3.7 Difference

7 {set_num1} {set_num2}

If either of the sets doesn't exist, create it. Store the result in set_num1. (Again, don't modify set_num2)

2.3.8 Symmetric Difference

8 {set_num1} {set_num2}

Instructions similar to Difference.

2.3.9 Print

4 {set_num}

Given the set number, print all its elements as comma-separated values. Do *not* include any whitespaces (Just include a linebreak at the end). Should Look Like -

$$1, 2, 3, 5, 7$$

If the set does not exist, just print a linebreak.

3 Some Points to Note

1. Your program will be stress-tested with limited memory sizes and tested for correctness using inputs of different sizes.
2. Ensure that your set is always sorted, and there are no duplicate elements.
3. Your code should also include a main function which takes care of all the IO. The autograder shall only provide input, and read your output from stdout.

4. For maintaining different sets, use array/vector. The set numbers at any time shall be 0, 1, ... N where N $\leq 10^5$. The set numbers shall be consecutive.

5. After every output, make sure you print a linebreak.

4 Bonus (5 Marks)

We will be ranking each submission according to the runtime on large test cases. The top 5% submissions will receive bonus 5 Marks. We will only consider those submissions which pass *all* the test cases used for checking correctness.

Note that we only allow algorithmic optimizations and not data structure optimizations. So, don't use trees, etc. You are only expected to work with arrays/vectors.

5 Submission Instructions

1. Submit only one cpp file on Gradescope. Your file should be named "main.cpp". Strictly follow the naming convention (everything lowercase).

2. As mentioned above, include a main function in your code that takes care of all the IO. We will just be providing the values that need to be read by your code.