Computing Structures – Fall 2023
Project 4
Due: Dec 1st 2023 at 11:59 PM

Objective:
The goal of this project is to handle multidimensional data through a tree structure. Once the data is stored in the tree structure, the data should be able to be queried for different dimensions using point queries and ranged queries.

Description:
The project is based on a multidimensional attribute tree, which stores multidimensional data. The multidimensional data is represented by different columns in the input file. Different rows represent different data points. As an example, consider the following data:

5 10 11
5 10 12
5 10 13
5 15 17
5 80 10
5 80 11
9 10 6
9 15 10
9 15 11
.....
…..
…..
45 30 11

This data is three dimensional, since there are three columns. There are multiple data points, which is defined by the number of rows. A multidimensional attribute tree for the given data will be:

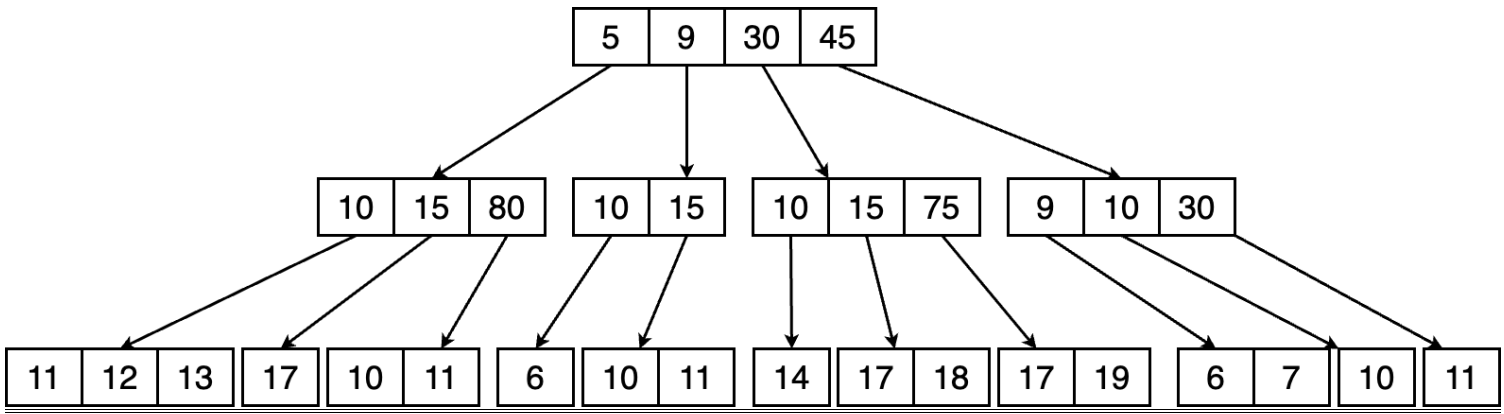

Fig. 1: An example of a multidimensional attribute tree.

The tree will have as many levels as there are dimensions in the data. Since the data is of 3 dimensions, the tree has 3 levels.
The queries will be of the following two formats:

Point queries: A point query defines the exact value of each dimension. The query will contain '-1' for don't care conditions. For example:
5 10 -1
The given query requires traversing from the value 5 in the root node to the value 10 in the child node. The next value in the query is '-1', so all routes from 10 in second dimension will be taken. Thus, the solution to the given query will be:
5 10 11
5 10 12
5 10 13

Range queries: A range query work like point queries. However, instead of point values for each dimension, a range is defined. For example:
5 : 10 10 : 15 -1

This query defines ranges in dimension 1 and 2. For dimension 3, -1 means don't care condition. The solution to the given query will be:

5 10 11
5 10 12
5 10 13
5 15 17
9 10 6
9 15 10
9 15 11

Input explanation:

The input file will be in the following format:

3
18

5 10 11
5 10 12
5 10 13
….
….
…..
45 30 11

2
P 5 10 -1

R 5 : 10 10 : 15 -1

The first row defines the dimensions of the data
The second row defines the number of rows of data in the file.
The subsequent lines define the data.
The line after date ends denotes the number of queries in the input file.
'P' stands for point query.
'Q' stands for range query.

In the interest of making this project simpler, all don't care values will be preceded by exact values. Thus, you won't get queries like:

P -1 10 -1

This is done so you don't have to implement backtracking.

Output:

You will have to determine the solution to the given queries and print them on screen in the format provided previously. An output file will be provided to you shortly.

Class Definitions:

The following class has been created for you in the boilerplate code:

```
class myTree {
    public:
    std::vector<myTree*> children;
    std::set<int> values;
    // Constructor
    myTree(std::set<int> val) : values(val){}

};
```

A set is used to store the values in each node as a set automatically stores values in a sorted order. Also, duplicate values are removed. Thus, the output to queries will be in order for each dimension. The same is denoted in the figure provided. All values in each node are sorted and unique.

Submission:

Submission will be through *Gradescope*. Your program will be autograded. You may submit how many ever times to check if your program passes the test cases provided. One test case will be released at the beginning and later other test cases will be released while grading. For the autograder to work, the program you upload must be named as ***projec4.cpp***.

Your final submission must contain your source file named project4.cpp. You access Gradescope using the tab on the left in our course page on canvas.

Constraints:

1. None of the projects is a group project. Consulting on programming projects with anyone is strictly forbidden and plagiarism charges will be imposed on students who do not follow this.
2. Please review academic integrity policies in the modules.

How to ask for help:

1. Please attend the help sessions. You can even stay there while you work.
2. Please always have a lookout for announcements on canvas. This class has video lectures that were recorded in the past, so the current information on the course will be mainly through announcements.
3. Email the TA with your ***precise*** questions. Please attach a snapshot of the error and your source file to the email if the question is regarding error(s) in compilation or runtime.