Requirement Document

Group 2 – AI/DS-B – 2023-27

Overview

Project 1 – van Emde Boas Tree

Van Emde Boas tree (or vEB tree) is a tree data structure which implements an associative array with m-bit integer keys.

VEB stores the keys in a bounded universe and performs operations (mentioned in the *Objectives & Functions* section) in double logarithmic time complexity with respect to universe size.

Project 2 – Treap

A **treap** is a data structure which combines binary tree and binary heap, also known as **randomized binary search tree**.

There are two quantities - Keys and Priorities, and without priorities, it is just a regular Binary Search Tree.

Stakeholders

Developers

vEB Tree

- Saran Shankar R
- Raghav Sridharan

Treap

- Nikhilesh H
- Prem Danasekaran
- Nighil Natarajan

Testers

vEB Tree

- Sathya Narayanan
- Sanjeev Krishna S

Treap

- Nithilan M

Objectives & Functions

vEB Tree: To be able to insert and delete in the tree and get the successor of given node.

Treap: To be able to insert, search, delete a given node while synchronously maintaining the properties of treap and displaying the treap in both inorder and level-ordered methods.

"The essential part is to achieve the best possible time and space complexities for each of the functions in the data structures."

vEB Tree

1. Insert → **O(log log u)**
2. Delete → **O(log log u**)
3. Successor → **O(log log u)**

Backend Functions

1. Root → O(1)
2. High → O(1)
3. Low → O(1)
4. Index → O(1)

[u → size of universe, due to complexity of the data structure, it's recommended to approach the target objective step-by-step, and also find potential ways to improve memory usage **]**

Treap

1. Insert → **O(log n)**
2. Search → **O(log n)**
3. Delete → **O(log n)**
4. Inorder Display → **O(n)**
5. Level-order Display → **O(n)**

Backend Functions

1. Left Rotation → **O(1)**
2. Right Rotation → **O(1)**

[n → number of nodes **]**

Target Working Model

vEB Tree

The user should be able to insert, delete and find the successor of the current value within the target time.

[VEB trees are implemented in areas where the three functions mentioned above are supposed to be performed at a very fast rate, hence the time complexity should be given utmost preference **]**

Treap

The user should be able to give values and priorities of choice, and the system must be capable enough to maintain the treap's balance, making operations such as insertion, search, delete very efficient.

Requirements & Constraints

vEB Tree

In simple terms, vEB tree works on getting the successor, so integer or float value should be utilized.

Possible Constraints

A predecessor function can be implemented, but since it's very much similar in working to the successor function, it is better we focus on other functions.

Treap

1. Priorities will be undergoing arithmetic comparisons, hence it must be of integer or float data type
2. Keys are basically values, and there is no specific data type to use.

Possible constraints

Operations like

- Union → merge two treaps and remove duplicates
- Intersection → merge two treaps, remove elements without duplicates, finally remove duplicates of elements that remain

Prescribed Resources

- Lecture 1
- Stanford Archive
- Lecture 2
- Treap Article

Intimation of Approval

From Prajesh Raam H S (Business Analyst) & Ramana K S (Project Manager)