TREAPS - IMPLEMENTATION AND APPLICATION

November 7, 2022

HARSHDEEP SINGH (2021MCB1044) ,
ADITI (2021MCB1227)

Instructor:
Dr. ANIL SHUKLA

Teaching Assistant:
VINAY

Summary: In this project, We have created a fully function-able dictionary with the help of Treap data structure.This dictionary has feature to insert word with their meanings, delete the existing word, update meaning of a word, view whole dictionary at once, search any word.

---

1. Introduction

Treaps also known as Cartesian tree is a data structure which combines binary tree and binary heap (hence the name: tree + heap = Treap). More specifically treap is a data structure in which each node stores pairs(x,y) in a node such that X follows the property of binary tree and Y follows property of heaps. X here refers to key value that is basically entered by the user and Y is priority value that is generated randomly using rand() function.That's why sometime we also call treaps as randomised binary search tree.

2. Operations

Our dictionary provide the following operations.

- **Insertion -:** Adds a new node to the tree. User will give data(both word and meaning) and the priority will be assigned randomly. Time - O(log N)
- **Search -:** Looks a node with specified key value X. If found, it will print the meaning on dashboard. Time - O(log N)
- **Delete -:** Looks a node with specified value X and remove it from tree. Time - O(log N).
- **View -:** It will print the whole tree.
- **Update -:** Looks a node with specified value X and update it's meaning.

3. Figures and Algorithms

3.1. Figures

Here firstly we will show you some pics related to treaps. How the structure of node looks like, How the tree looks like after insertion of nodes, How the rotations take place to maintain the properties. After that we will show you the algorithm for the same.

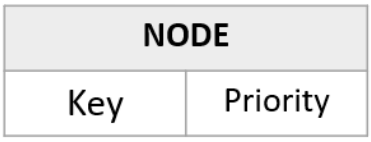


Figure 1: Structure of node in treap.

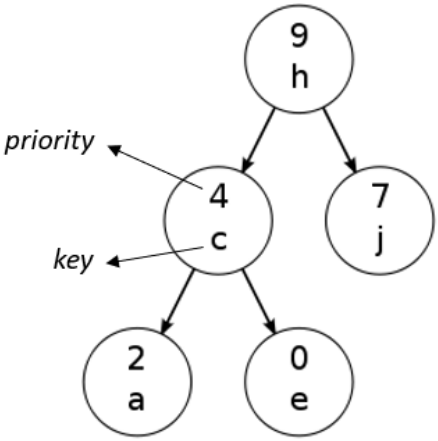


Figure 2: Representation of tree.

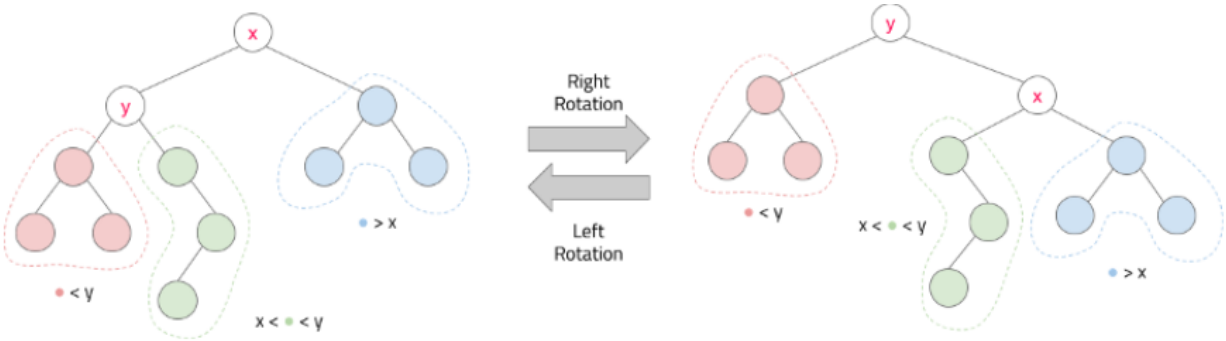


Figure 3: Rotations to maintain max heap property.

3.2. Algorithms

Here we will write algorithm for the same.

Algorithm 1 INSERTION

1: Insert a node firstly as bst and then do rotations to satisfy heap properties.
2: We have assumed that we have already define structure of node.
3: TreapNode * insert(root,key)
4: if(key <= key.root)
5: insert in left subtree.
6: if(root.left.priority > root.priority)
7: rightrotate(root).
8: else
9: insert in right subtree.
10: if(root.right.priority > root.priority)
11: leftrotate(root).
12: return root;

Algorithm 2 SEARCHING

1: Searching is same as bst.
2: TreapNode * search(root,key)
3: if(root == NULL || root.key==key)
4: return root
5: if(root.key<key)
6: return search(root.right,key).
7: else
8: return (root.left,key);

Algorithm 3 DELETION

1: If a node is a leaf, just delete it.
2: If node has one child NULL and other as NON - NULL, replace node with non empty child.
3: If node has both children as non-NULL, find max of left and right children.
4: If priority of right child is greater, perform left rotation at node
5: If priority of left child is greater, perform right rotation at node.

Algorithm 4 VIEW

1: Same as inorder traversal in bst
2: Traverse the left subtree i.e call inorder(left.subtree).
3: Visit the root.
4: Traverse the right subtree i.e call inorder(right.subtree).

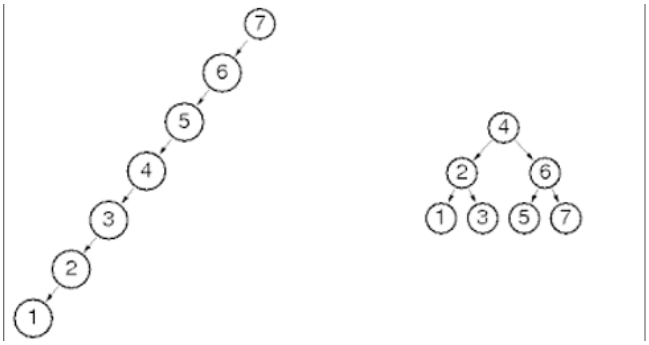
4. Advantages of treaps.

Why we dont use bst here?

- Sorted insertion increase space complexity.
- Depth become linear instead of logarithmic.
- Can be highly unbalanced.

Why better in treaps then?

- All operation (search,insert,delete) in O(logN).
- Easy to maintain and use.



5. Conclusions

As from above explanations, we can see that treaps has much more advantages over other tree. It's much more faster to insert and search data in treaps.

6. Bibliography

Here in completing this project, we took help from my mentor,a few website such as

- geeks for geeks [1]
- cp-algorithm[2]
- research paper[3].

Acknowledgements

We would like to thank our professor to give us this oppurtunity to make this project and our mentor who help us to do it.

References

[1] Treap (a randomized binary search tree), Mar 2017.

[2] Treap (cartesian tree), Jun 2022.

[3] Guy E Blelloch and Margaret Reid-Miller. Fast set operations using treaps. In *Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures*, pages 16–26, 1998.