Minimum and Maximum Ratio of Number of Red Internal Nodes to Black Internal Nodes in Red Black Tree

Pawan Jindal[1], Amit Kumar[2], Shishir Kumar[3]
1. Deptt. of Computer Science & Engineering
Jaypee University of Engineering & Technology, Guna , M.P. India

2, 3 Deptt. of Computer Science & Engineering
Jaypee University of Engineering & Technology, Guna , M.P. India

ABSTRACT
Data structure is being used for designing of databases, software system etc. Different data structures are being used for different kinds of applications and some data structures are particular used for specific tasks. B-trees are particularly used for implementation of databases. Hash tables are widely used for implementations of compilers. Tree is non linear data structure which is better than array, linked list because the time complexity for different operations like searching, insertion, deletion etc. is less in the case of tree. The time complexity for dynamic operations like searching, insertion, deletion, updating a node etc. is directly proportional to the height of red black tree. As the height(h) of red black tree is directly proportional to the O(lg(n)) where n represents the total number of nodes in a particular red black tree. So the time complexity for different dynamic operations like searching, insertion, deletion, updating a node etc. is directly proportional to the O(lg(n)). In this paper, Minimum and maximum ratio of number of red internal nodes to black internal nodes in Red Black tree is determined and explained with the help of diagram.

General Terms
Algorithms, Theory.

Keywords
Tree, Red black tree, time complexity.

1. INTRODUCTION
Algorithm [7,8,9] is defined as any computational procedure which takes the input as a single value or set of values and produces the output as a single value or set of values. Tree is non linear data structure [4,5]. There are large numbers of trees like binary tree, complete binary tree, full binary tree, balanced binary tree, Binary search tree, perfectly balanced binary tree, AVL tree, B tree, red black tree, splay tree, B^+ tree etc. Every red black tree is binary search tree but binary search tree is not red black tree. Red black tree is approximately balanced it means that the length of the longest path from a particular node y to a descendant leaf is at most twice of that of shortest path from that particular node y to any descendant leaf node. Every red black tree must satisfy the following property.

1. Tree must be binary search tree.
2. Every node of red black tree has a particular color i.e. either red color or black color.
3. Root node of red black tree must be black.
4. All leaf nodes of red black tree must be black in color.
5. Red node must have exactly two black children.
6. The total number of black nodes from root node to any of the leaf node (excluding the root node but including the leaf node) is known as black height of red black tree. The black height of red black tree must be same from root node to any of the leaf node.

2. OPERATIONS ON RED BLACK TREE
Each and every operation for red black tree has a particular algorithm [1,2,5,6] for it. Some important operations in red black tree are given below.

2.1 Insertion
The operation in which a particular node is inserted in red black tree is known as insertion [2]. After inserting one or more nodes in red black tree, the resultant tree must be red black tree. There may be need of one or more rotation to maintain all the properties of red black tree. If n is the total number of nodes in red black tree, then the time complexity [3,4] for insertion is O(lg(n)).

2.2 Deletion
The operation in which a particular node is deleted from red black tree is known as deletion[2]. After deleting one or more nodes in red black tree, the resultant tree must be red black tree. There may be need of one or more rotation to maintain all the properties of red black tree. Maximum number of rotation in the case of deletion in red black tree can be three. If n is the total number of nodes in red black tree, then the time complexity for deletion is O(lg(n)).

2.3 Searching
The process in which a particular node is being searched from red black tree is known as searching. If the node which we want to search is present in red black tree, then the search will be successful otherwise search will not be successful. The time complexity for searching a particular node in red black tree is O(lg(n)).

2.4 Updation
The process in which the key value of a particular node is being changed from previous one to the new key value is known as updation. If n is the total number of nodes in red black tree, then the time complexity for updation is O(lg(n)).

3 PROPOSED WORK
Suppose "r" represents the total number of red nodes in a particular red black tree.
"b" represents the total number of black nodes in red black tree.

Suppose k=r/b. Where k is a constant for a particular red black tree.

3.1 Maximum value of k

The value of k will be maximum if the value of r is maximum and the value of b is minimum.
It means the number of red nodes should be maximum & the number of black nodes should be minimum in a particular red black tree with the constraint that all properties of red black tree should be maintained. For this particular case, we will not consider that particular red black tree in which only root node exist because root node has always black color so there will not be any red color nodes. Consider the case of red black tree which has black height one as shown below.

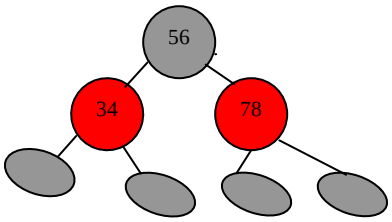

Figure 1. Red black tree with black height one which has maximum number of red nodes.

In the above red black tree, the root node has the key value 56 and the left child of root node has the key value 34 and the right child of root node has key value 78. Also the total number of black nodes=1 and the total number of red nodes are 2. It means that r=2 and b=1.
So the value of k will be 2.
Suppose red black tree with black height 2 as shown below.

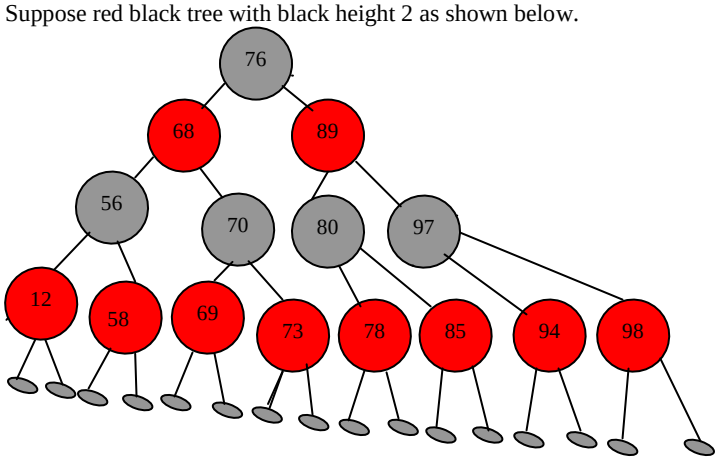

Figure 2. Red black tree with black height two which has maximum number of red nodes.
In above red black tree. The total number of red nodes=r=10.
The total number of black nodes=b=5.
So k=r/b=10/5=2.
In order to get the maximum value of k, the value of r must be maximum with the constraint that the all properties of red black tree must be maintained. From the above discussion, it is very clear that the maximum value of k is two.

3.2 Minimum Value of k

The value of k will be minimum if the value of r is minimum and the value of b is maximum. As r represents the total number of red nodes in red black tree. So the minimum value of r will be zero because the minimum number of red color nodes in red black tree will be zero because the number of red color nodes in red black tree can never be negative.
If the black height[2] of red black tree is one and there is only root node in red black tree as shown below:

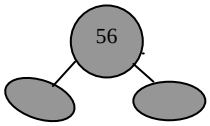

Figure 3. Red black tree with black height one which has minimum number of red nodes.
In the above red black tree, the root node has key value 56. There is no any red color nodes in that particular red black tree. So b=1, r=0. So the value of k=r/b=0/1=0.
If the black height of red black tree is two and there is no any red node in red black tree as shown below:

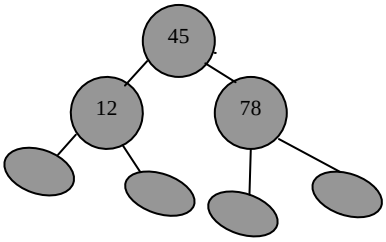

Figure 4. Red black tree with black height two which has minimum number of red nodes.

In the above red black tree, the root node has key value 45.
The left child of root node has key value 12 and the right child of the root node has the key value 78. There is no any red color nodes in that particular red black tree. So
Total number of black nodes in red black tree= b=3.
Total number of red nodes in red black tree r=0.
So the value of k=r/b=0/3=0.
So the minimum value of the ratio of number of red internal nodes to black internal nodes in Red Black tree is zero.

If the black height of red black tree is three and there is no any red node in red black tree. It means that the total number of red nodes in red black tree will be equal to zero. In red black tree of figure 5, the total number of black nodes are 15. It means the value of b=15 but the value of r=0. So the value of k will be equal to 0.

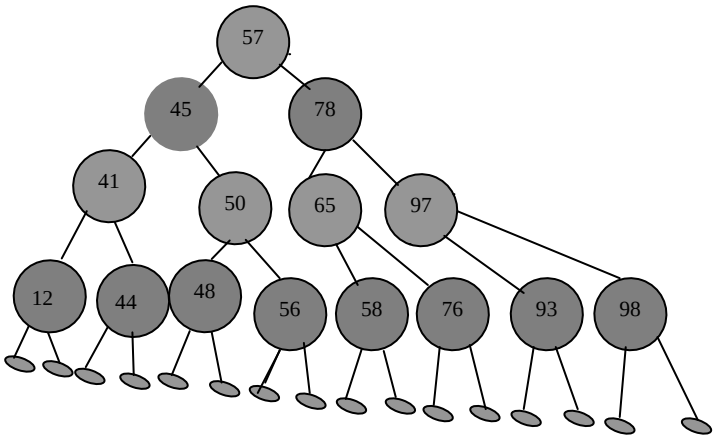

Figure 5. Red black tree with black height three which has minimum number of red nodes.

4 CONCLUSIONS

The time complexity for dynamic operations like searching, insertion, deletion, updating a node etc. is directly proportional to the height of red black tree. As the height(h) of red black tree is directly proportional to the O(lg(n)) where n represents the total number of nodes in a particular red black tree.

So the time complexity for different dynamic operations like searching, insertion, deletion, updating a node etc. is directly proportional to the O(lg(n)). In this paper, Minimum and maximum ratio of number of red internal nodes to black internal nodes in Red Black tree is determined and explained with the help of diagram.

REFERENCES

[1] E. Horowitz and S. Sahni, "Fundamental of Computer Algorithms", Computer Science Press, 1982.

[2] T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", MIT Press, 1991

[3] A.V. Aho, J.E. Hopcroft and J.D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[4] A.V. Aho, J.E. Hopcroft and J.D. Ullman, "Data Structures and Algorithms", Addison-Wesley,1984.

[5] N. Wirth, "Algorithms and Data Structures" Prentice-Hall, 1986

[6] D. I-Iarel, "Algorithmics: The spirit of computing", Addison-Wesley 1987.

[7] R. Baeza-Yates, "Teaching Algorithms", IV Iberoamerican Congress on Computer Science Education, Canela, Brazil, July 1995.

[8] G. Tel, "Introduction to Distributed Algorithms"Cambridge Univ. Press 1995.

[9] G- Brassard, and P. Bratley, "Algorithmics: Theory and Practice", Prentice-Hall, 1988.

[10] E.M. Reingold, J. Nievergelt and N. Deo, "Combinatorial Algorithms: Theory and Practice",Prentice-Hall, 1977.