Red-Black Tree

Pokatilo Pavel

Peter the Great St. Petersburg Polytechnical University
gradergage@gmail.com

December 1, 2016

Abstract

The paper is concerned with programming algorithms and considering of their mathematical base, generalized information about Red-Black trees. Recommended for users for familiarization with Red-Black binary trees including history, rules of implementing and areas of use.

I. Introduction

Complex data structures are a common way to store data, which is widely used in programming. This paper considers many parameters of data structures such as speed of search, insert and deleting. According to that, developing of data structures is wide and well-studied problem.

The effective algorithms are developed and used nowadays and most of them are based on binary trees. We studied balanced Trees, which provide guaranteed whorst-case running time proportional to log(N) for all operations. These algorithms are based on modifying standart trees to make the length to every node proportional to log(N).

Different implementations have different features used in special cases. Examples of these algorithms are: AVL tree, 2-3-4 tree, B tree, and etc. One of the most interesting subject is Red-Black tree, which combines an unusual organization and good operation performing complexity.

Self-balanced Red-Black tree was invented by Rudolf Bayer in 1972. Algorithm is based on node colors, which can be used to control the balance. This tree implementation guarantees log(N) proportional tree length and high speed of search, insert and delete operations.

II. Methodology

Red-black trees using common structure of binary tree operations with searchable keys and associated values. To describe the main aspects should be used the following terms: insert, remove, and search. Inserting new element into structure is unreachable without searching necessary place for this element. Removing also uses search via key to find elements to remove. Considering this, we can decide, that search is one of the main operation of this data structure, so tree must be effectively balanced to reach minimum complexity and maximum speed of this operations.

III. Exposition

Often programmers meets a problem of choosing the most appropriate data structure. Complexity and speed are main parameters of data structures. Red-black tree uses unusual algorithms to reach one of the best run times of binary trees. Existing some rules of this algorithm building:

1. Nodes only two colors: red or black
2. Root of the tree is always black
3. All leafs are black
4. Descendants of red node are always black

5. Every way from current node to every leaf contains equal count of red and black nodes

This is main rules of red-black trees algorithms, which allow us to programm balanced binary tree, but the different implementations can affect the perfomance of structure.

In common case insertion uses up to two rotations, whose complexity is O(1), deleting uses up to three rotations, which might be best result of rebalancing after basic operations.

Tree height does not exceed O(log(N)) in best cases, where N is count of internal nodes. According to this we measured search operation complexity, which is O(log(N)) too. This is caused by count of iterations of search, whose cant be more than tree height.

Search and remove also provide O(log(N)) complexity. Using simple calculations we summorized search and maximum rotations complexity, as result we have: O(1)+O(1)+O(log(N))=O(log(N)).

IV. Results

This exposition gives mathematical information about algorithm, but it is not correctly represents all real cases and complete running time could be different.

Compared to AVL tree, we considered, that maximum height of red-black tree usually more, than same in AVL tree up in 1.388 times, which affects on speed. But the AVL tree removing operation could use up to height count rotations.

Above this, we estimated, that red-black tree have O(log(N)) complexity for all operations.

V. Discuss

Different implementations give different results, which used in specified cases. Red black tree can be used to work with large amount of data and be more efficient than AVL trees, but lose in short data stack.

Existing some implementations, which stores color bit in the pointer binary adress to reduce memory consumption. Many of high level languages collection based on red-black trees. That all reached by unique algorithm organization and effectivity.

There are wrong opinion, that trees have difficulty to use instead of arrays. Some containers like maps in C ++ are built on Red-Black trees, but they are user-friendly and simple in use, if they used right.

VI. Conclusion

Red-black trees are algorithms with high performance which must be known by every programmer. Wide range of applications uses those algorithms to perform different tasks.

Concluding the exposition, we recommend to measure complexity of your work and use most effective algorithms or those combinations to reach the best result

References

[1] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms. - 3rd. - MIT Press, 2009

[2] Wirth N. Algorithms and Data Structures, 2004