EXPLORATION AND ANALYSIS OF RED-BLACK TREES AND MAX FLOW ALGORITHMS

JORDAN DEHMEL, KATE ECKHART, LOGAN HUMBERT, DARRIN MILLER

Contents

1. Abstract

We explore various properties and real-world applications of red-black trees and max-flow optimization algorithms. We explore several algorithms for the later, and provide experimental comparative analysis of their performances.

2. Introduction

3. Red-Black Trees

3.1. **Theoretical Analysis.**

3.2. **Practical Implementation Details.**

3.3. **Real-World Application.**

Date: Fall 2023.

3.4. **Theoretical Questions. Prove that the height of a Red-Black tree with n nodes is guaranteed to be $O(logn)$ in the worst case scenario. Provide a rigorous mathematical proof.**

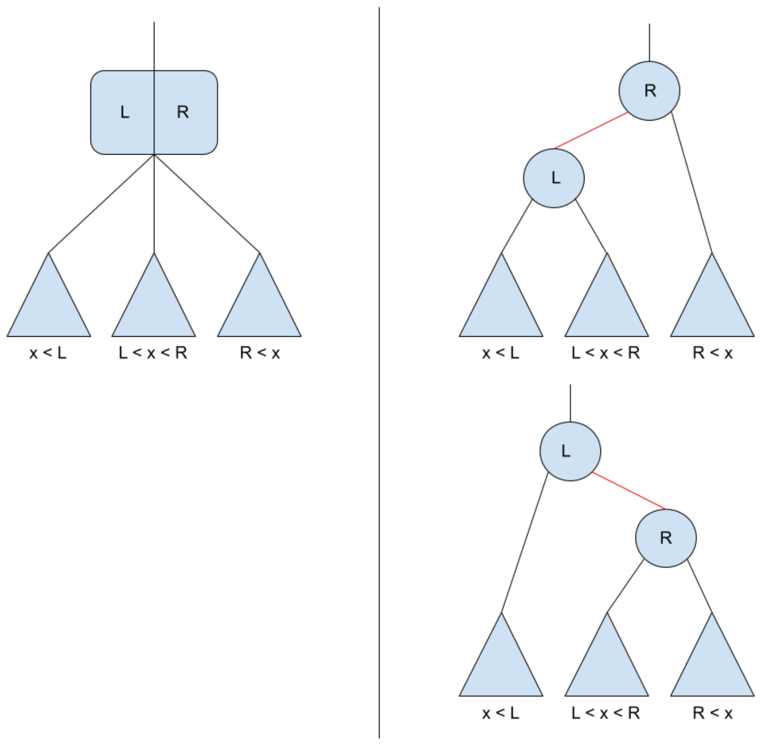
In this proof, we will first prove that the height of a 2-3-4 tree is limited by $O(logn)$. Then, we will prove that any valid red-black tree can be converted directly into a 2-3-4 tree. Finally, we will note that, so long as height is measured only in black links, tree height is maintained by this conversion.

A 2-3-4 tree, by its nature, only grows by pushing the root "upwards". The only time at which the height of such a tree increases is when a 4-node at the root splits, sending a node upwards to become the new root. In this case, the height of the tree uniformly increases by one for all leaf nodes. This means that the height of the tree is precisely equal for all leaf nodes no matter what.

Now we will examine the equivalency between red-black trees and 2-3-4 trees. We will show that each node in a 2-3-4 tree corresponds to exactly one black link in a red-black tree, and that the only additions needed are red links.

First, we will consider a 2-node. This is a node with two output links. This is equivalent to the standard node in a binary tree- no modifications are needed to modify it into red-black tree form.

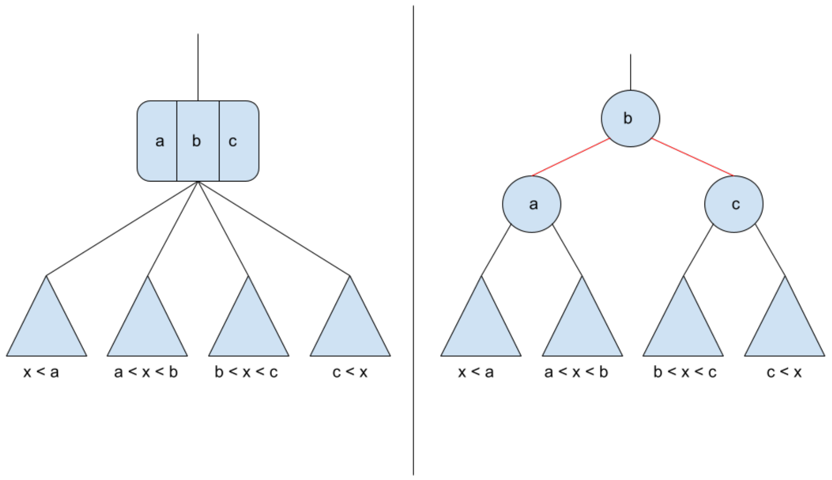
Next, we will consider a 3-node. This is a node with three output links. The leftmost represents the subtree wherein all nodes are less than the lesser item in the node. The rightmost similarly represents the subtree wherein all nodes are larger than the greater item, and the middle represents the subtree containing nodes who fit neither of these trees.



A 3-node and its possible red-black tree versions.

Since the height of a red-black tree is the number of black links the root must follow to get to a leaf, the two possible red-black subtrees above both have a height of 1: the same height as the 2-3-4 tree they came from.

The only remaining case is the 4-node. A 4-node usually only exists in a 2-3-4 tree for a moment before it is split apart. If we designate the 3 items within the node as a, b, and c, then we say that (from left to right) the child links represent the ranges $x < a$, $a < x < b$, $b < x < c$, and $c < x$ for any item x in the given child subtree. These cases, of course, can also be covered by an equivalent red-black tree, as shown below.



A 4-node and its red-black tree version.

Again, this red-black tree has the same height as its 2-3-4 tree equivalent: 1. Since we have accounted for all possible variations of red-black subtree herein, we can use the above rules to translate between red-black tree and 2-3-4 tree. Therefore, any statement we make about 2-3-4 trees holds for red-black trees.

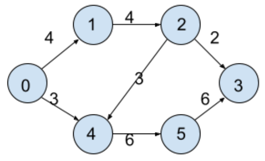
In the best-case scenario, a 2-3-4 tree (post 4-node splitting) will containing n nodes will have a height of $log_3(n)$, where every node is a 3-node. At worst case, it will have a height of $log_2(n)$, where every node is a 2-node. Since red-black and 2-3-4 trees are equivalent, we can thusly say that the worst-case height of a red-black tree of size n is limited by $log_2(n)$ black links.

Discuss how Red-Black trees are used in modern databases and file systems to maintain balanced structures. Explain the trade-offs and advantages of using Red-Black trees in these contexts.

4. Max Flow Algorithms

4.1. **Theoretical Analysis.**

4.2. **Practical Implementation Details.**



A benchmark graph for max-flow.

4.3. **Real-World Application.**

4.4. **Theoretical Questions. Describe the concept of augmenting paths and their role in the Ford-Fulkerson algorithm. Prove that the algorithm terminates and converges to the maximum flow in finite time, even for non-integer capacities.**

Explain the Min-Cut Max-Flow Theorem and its significance in the context of network flows. How can this theorem be used to find a maximum flow and minimum cut in a flow network?

5. Real-World Problem Solving

6. Comparative Analysis and Reporting

6.1. **Red-Black Trees.**

6.2. **Max Flow Algorithms.**

```
6 nodes:
FF ms:    0.057353
EK ms:    0.05099
EK is 12.4789% faster than FF.

100 nodes:
```

```
FF ms:     15.4379
EK ms:     5.45683
EK is 182.909% faster than FF.

1,000 nodes:
FF ms:     157.864
EK ms:     26.9243
EK is 486.325% faster than FF.

10,000 nodes:
FF ms:     1168.17
EK ms:     502.373
EK is 132.53% faster than FF.

100,000 nodes:
FF ms:     23405.1
EK ms:     4711.56
EK is 396.758% faster than FF.
```

7. Conclusion

8. References

- https://sedgewick.io/wp-content/themes/sedgewick/papers/2008LLRB.pdf
- https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap13b.pdf
- https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/
- https://brilliant.org/wiki/ford-fulkerson-algorithm/
- https://brilliant.org/wiki/edmonds-karp-algorithm/