Comparing the costs of rotation-based tree balancing algorithms

CS583 - Analysis of Algorithms

Michael Reed

Abstract

This project implements three algorithms for completely balancing a binary search tree via rotations, as described in Luccio et al.'s *Complete Balancing via Rotation*. These algorithms take an arbitrary binary tree S and an almost-complete binary tree T, both containing the same set of keys, and performs rotations on S until S equals T. The authors claim in Theorem's 1 through 3 that these algorithms transforms S to T in an exact or upper-bounded number of rotations. This project empirically verifies these theorems.

Assumptions

I made the following assumptions:

- In Theorem 3, I interpreted the g term as being the number of maximally equivalent subtrees which did not equal each other, implying that size 1 subtrees would not contribute to g. The authors seem to hint at this definition but do not make it clear.
- I interpreted the **remove** operations done in A_L and A_R as just the **move** operation done in reverse, as mentioned in the proof of correctness section of the paper (top of page 5, first paragraph, first column). This allowed me to use a simpler, novel algorithm in place of A_L and A_R, although it uses $\Theta(n)$ additional space whereas A_L and A_R use $\Theta(1)$ additional space. This should not be an issue since my algorithm performs the same number of rotations as A_L and A_R.

Findings

Theorem 1 correctly predicted the number of rotations performed by A_1 in all experiments, barring the fact that its prediction was off by exactly one.

Theorem 2 did not predict the exact number of rotations performed by A_2, although it was always within 2% of the actual number of rotations. I found that when I removed the $c_S(root_T)$ term, the expected number of rotations was closer to the actual number more consistently. I never figured out why.

Theorem 3 was correct as an upper-bound for the number of rotations performed by A_3, although it was usually much higher than the actual number of rotations. I suspect part of the issue is that g is defined as the number of maximal equivalent subtrees, whereas it should instead be defined as the number of maximal equivalent subtrees of size > 1. Equivalent subtrees of size 1 are already identical subtrees, so they incur no additional rotations.

Just as the authors claimed, A_2 consistently performed better than A_1, but A_3 often performed on par or slightly worse than A_2. This may be caused by my trees not having many equivalent subtrees.

For each pairing of algorithm and value of n (n = 1000, 1100, and 1200), we perform 5 experiments. Here is the output of one such run:

```
A1 with n=1000
rotations actual = 1964, expected = 1964
rotations actual = 1967, expected = 1967
rotations actual = 1963, expected = 1963
rotations actual = 1962, expected = 1962
rotations actual = 1969, expected = 1969

A2 with n=1000
rotations actual = 1173, expected = 1173 +- 1
rotations actual = 1072, expected = 1072 +- 1
rotations actual = 1115, expected = 1111 +- 1
rotations actual = 1155, expected = 1149 +- 1
rotations actual = 1114, expected = 1114 +- 1

A3 with n=1000
rotations actual = 1123, upper bound = 1826
rotations actual = 1136, upper bound = 1835
rotations actual = 1126, upper bound = 1827
rotations actual = 1074, upper bound = 1811
rotations actual = 1159, upper bound = 1844

A1 with n=1100
rotations actual = 2163, expected = 2163
rotations actual = 2167, expected = 2167
rotations actual = 2170, expected = 2170
rotations actual = 2156, expected = 2156
rotations actual = 2162, expected = 2162

A2 with n=1100
rotations actual = 1223, expected = 1211 +- 1
rotations actual = 1228, expected = 1226 +- 1
rotations actual = 1269, expected = 1257 +- 1
rotations actual = 1229, expected = 1227 +- 1
rotations actual = 1248, expected = 1246 +- 1

A3 with n=1100
rotations actual = 1323, upper bound = 2044
rotations actual = 1270, upper bound = 2027
rotations actual = 1265, upper bound = 2014
rotations actual = 1240, upper bound = 2005
rotations actual = 1216, upper bound = 2015

A1 with n=1200
rotations actual = 2363, expected = 2363
rotations actual = 2362, expected = 2362
rotations actual = 2360, expected = 2360
rotations actual = 2365, expected = 2365
rotations actual = 2361, expected = 2361

A2 with n=1200
rotations actual = 1337, expected = 1331 +- 1
rotations actual = 1326, expected = 1318 +- 1
rotations actual = 1374, expected = 1374 +- 1
rotations actual = 1326, expected = 1324 +- 1
rotations actual = 1355, expected = 1343 +- 1

A3 with n=1200
rotations actual = 1350, upper bound = 2193
rotations actual = 1367, upper bound = 2200
rotations actual = 1375, upper bound = 2222
rotations actual = 1343, upper bound = 2192
rotations actual = 1429, upper bound = 2228
```

Reproduction

To reproduce my results you will need OpenJDK-15.0.1 and Maven 3.6.2 installed. Run the following commands from a terminal to run the code:

```
mvn compile
mvn test
```