Difference between AVL trees and splay trees

I am studying about various trees, and came across AVL trees and splay trees. I want to know

1. What is difference between AVL trees and splay trees?
2. On what basis do we select these tress?
3. What are positive's and negative's of these trees?
4. What are the performances of these trees in terms of big O notation?

algorithm data-structures binary-search-tree avl-tree splay-tree

edited Feb 4 '12 at 22:12
templatetypedef
167k 37 412 656

asked Sep 19 '11 at 6:12
venkysmarty
2,966 9 39 97

3 Answers

1. Both splay trees and AVL trees are binary search trees with excellent performance guarantees, but they differ in how they achieve those guarantee that performance. In an AVL tree, the shape of the tree is constrained at all times such that the tree shape is balanced, meaning that the height of the tree never exceeds O(log n). This shape is maintained on insertions and deletions, and does not change during lookups. Splay trees, on the other hand, maintain efficient by reshaping the tree in response to lookups on it. That way, frequently-accessed elements move up toward the top of the tree and have better lookup times. The shape of splay trees is not constrained, and varies based on what lookups are performed.

2. There is no hard-and-fast rule about this. However, one key difference between the structures is that AVL trees guarantee fast lookup (O(log n)) on each operation, while splay trees can only guarantee that any sequence of n operations takes at most O(n log n) time. This means that if you need real-time lookups, the AVL tree is likely to be better. However, splay trees tend to be much faster on average, so if you want to minimize the total runtime of tree lookups, the splay tree is likely to be better. Additionally, splay trees support some operations such as splitting and merging very efficiently, while the corresponding AVL tree operations are more involved and less efficient. Splay trees are more memory-efficient than AVL trees, because they do not need to store balance information in the nodes. However, AVL trees are more useful in multithreaded environments with lots of lookups, because lookups in an AVL tree can be done in parallel while they can't in splay trees. Because splay trees reshape themselves based on lookups, if you only need to access a small subset of the elements of the tree, or if you access some elements much more than others, the splay tree will outperform the AVL tree. Finally, splay trees tend to be easier to implement than AVL trees, since the rotation logic is much easier.

3. See (2)

4. AVL tree insertion, deletion, and lookups take O(log n) time each. Splay trees have these same guarantees, but the guarantee is only in an amortized sense. Any long sequence of operations will take at most O(n log n) time, but individual operations might take as much as O(n) time.

Hope this helps!

answered Feb 4 '12 at 22:06

templatetypedef
167k 37 412 656

1 >> Splay trees are more memory-efficient than AVL trees, because they do not need to store balance information in the nodes, But how much memory is required per node for AVL-trees? – 4esn0k Feb 25 '13 at 4:18

@4esn0k- You need to store one of three different balance factors (-1, 0, or +1). Typically, though, there's no overhead, since the two bits required to store this can be packed into the left and right pointers. – templatetypedef Feb 25 '13 at 7:08

1) What is difference between AVL trees and splay trees?

They are similar in structure and the operations we call on them. The difference is that in splay trees, after each operation, we try to keep the tree almost perfectly balanced so that future operations take less time.

2) On what basis do we select these tress?

Splay trees are always better than binary search trees when, your application deals with a lot of data in the tree but, will need access to a subset of the data very frequently than others. In this case the data you access frequently will come near the root as a result of the splay. Also, any node can then be accessed with less time than before.

As a general rule for selecting these trees, if you need "Average" log(n) time over a period of tree operations then use splay tree. Binary tree cannot guarantee this.

3) What are positive's and negative's of these trees?

Positives for both is that you get around log(n) in both these data structures theoretically.

As mentioned splay trees have average log(n) over a number of operations. This means that, maybe you got n time complexity for an operation atleast once in that set. But this will be compensated when accessing the frequent items.

The negative of the binary search tree is that, you need to be lucky to have log(n) always. If the keys are not random, then the tree will reduce to a list like form with only one side.

4) What are the performances of these trees in terms of big O notation?

Splay tree Log(n) on Average for a group of tree operations. Binary tree Log(n) only if your keys are going in random.

The results on the runtime are obvious heresplay tree runtime profiling You can see the runtime difference in searching with and without splaying.

answered Jan 18 '12 at 3:40
Harisankar Krishna Swam
41 3

http://en.wikipedia.org/wiki/AVL_tree

vs.

http://en.wikipedia.org/wiki/Splay_tree

Will give you big-o etc.

From a one-over I see that AVL is more consistent and good overall, However, Splay seems to be more useful in instances where you might be using elements soon/immediately after their addition to the tree.

Sorry i don't know more.

answered Sep 19 '11 at 6:19
oorosco
139 10

Additionally one more thing, i just noticed, the AVL will spread equally in height and in width, however the splay tree will grow taller faster than wider in several situations probably due to the ability to "splay" – oorosco Sep 19 '11 at 6:20

0 You added references and observations but obviously don't know the whole story. Even just looking at the wikipedia page, you should have been able to mention memory usage and that the splay tree has good amortized times. – quasiverse Sep 19 '11 at 6:27

It's pointless to recite an entire wiki page to the man, all his questions can be answered by reading the page. – oorosco Sep 19 '11 at 6:30