Määrittelydokumentti

I choose 4 trees from most popular binary trees:

1. **Red-black tree** – colored self-balancing binary search tree. This tree is not perfectly balanced, but has height of O(log n)

A node is either red or black

The root is black

All leaves are black

If a node is red, then both its children are black

Every path from a given node to its descendant leaf nodes contains the same number of black nodes.

Performance:

Searching, insertion and removal in O(log *n*)

2. **AA-tree** – simple balanced search tree (fast add / find / delete)

It is used for storing and retrieving ordered data efficiently.

Easier to implement than AVL and Red-Black trees.

Some Red-Black rotations are not needed

Slower than AVL & RB.

Performance:

The performance of an AA tree is equivalent to the performance of a red-black tree. While an AA tree makes more rotations than a red-black tree, the simpler algorithms tend to be faster, and all of this balances out to result in similar performance. A red-black tree is more consistent in its performance than an AA tree, but an AA tree tends to be flatter, which results in slightly faster search times

3. **Rope** – balanced binary tree that preserves the order of elements .

Each node holds a short string. Each node has a weight value equal to length of its string

Rope provides fast access by index / add / edit / delete operations

Allows fast string edit operations on very long strings

This structure is efficient for very large strings

E.g. length > 10 000 000

For small strings ropes are slower.

List<T> and StringBuilder performs better for 100 000 chars

Performance of rope:

Faster insert / delete operations at random position – O(log(n))

Slower access by index position – O(log(n))

Arrays provide O(1) access by index

4. **Splay tree** - self-adjusting binary search tree with the additional property that recently accessed elements are quick to access again.

For many sequences of non-random operations, splay trees perform better than other search trees, even when the specific pattern of the sequence is unknown.

All normal operations on a binary search tree are combined with one basic operation, called splaying. Splaying the tree for a certain element rearranges the tree so that the element is placed at the root of the tree.

Performance:

Insertion, look-up and removal in O(log n) amortized time.

The main task:

To compare the 4 different structures and conclude what are their advantages and disadvantages. Base on these results to show suitable applications for every structure.

Plan for work:

1. More deeper theoretical understanding of all 4 tree structures
2. I still have to choose the algorithms, which will realize all 4 trees.
3. I plan to compare the basic tree operations for all 4 structures, as well as to exploit the main properties of all trees and find suitable implementations to show them.

I still have not found the algorithms, which I will use, and therefore at this stage I do not know space requirement for different tree structures.

Sources:

From Wikipedia:

1. Red-black tree: https://en.wikipedia.org/wiki/Red%E2%80%93black_tree
2. AA – tree: https://en.wikipedia.org/wiki/AA_tree
3. Rope: https://en.wikipedia.org/wiki/Rope_(data_structure)
4. Splay tree: https://en.wikipedia.org/wiki/Splay_tree