(a, b)-tree experiment

Test program

The test program evaluates the implementation of (2, 3)-tree and (2, 4)-tree.
It performs three types of tests:

1. Insert test
2. Min test
3. Random test

Performance

Performance is measured in terms of the average number of structural changes.
Structural change is either a node split (in insert) or merging of two nodes (in delete).

- #SPLIT the number of performed split operations
- #INSERT the number of performed insert operations (number of keys)
- ASC the average number of structural changes
  ASC = #SPLIT / #INSERT

- #NODES the number of nodes
- HEIGHT the height of the tree

Plots

Each plot shows the dependence of the average number of structural changes ASC on the set size #KEYS.

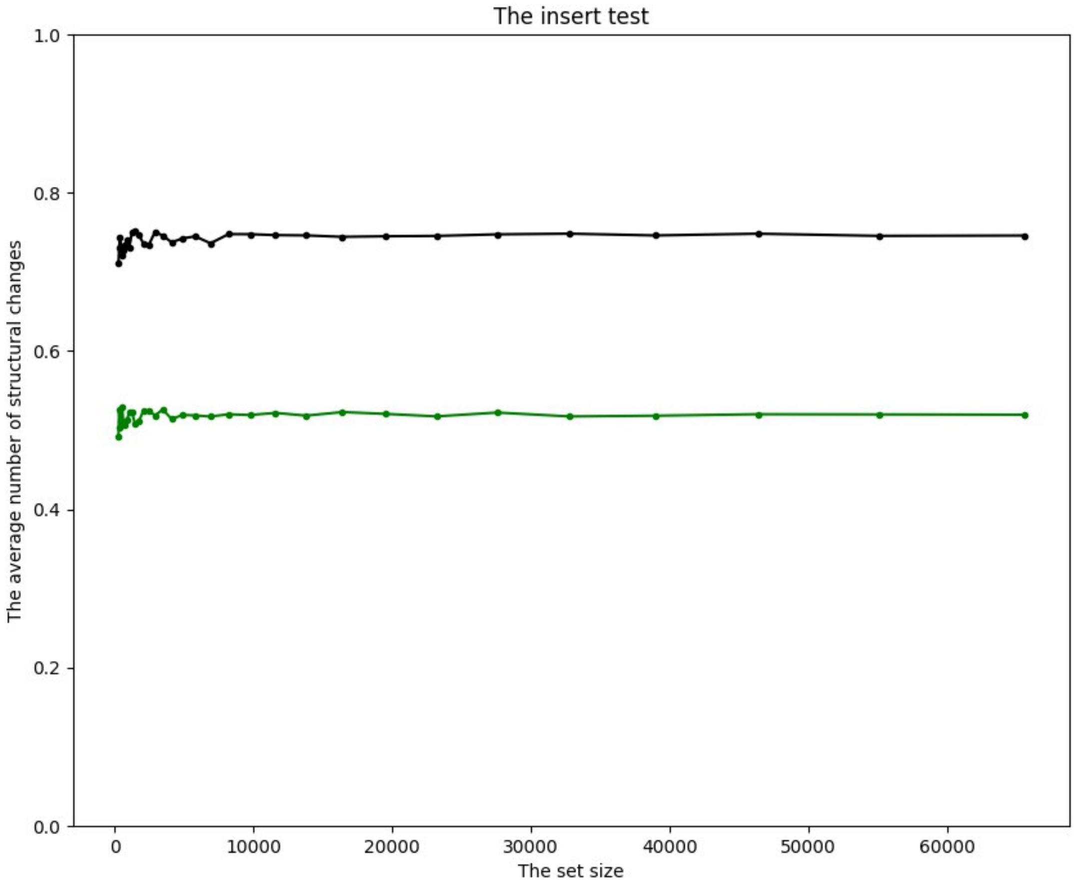
- (2, 3)-tree the black curve
- (2, 4)-tree the green curve

Theoretical facts

- The number of splits is bounded by the number of nodes.
- The height of (a, b)-tree is bounded by $(1 + \log_a ((\#INSERT + 1) / 2))$.
-

1. Insert test

Insert n elements in random order.



The number of structural changes is bounded by 1:

- Worst case:

  (2, 3)-tree: insert sequence in ascending or descending order

  (2, 4)-tree: insert sequence in ascending order

  - There is only one branch, to that the keys are inserted.
    Let's say it has internal nodes $n_1, n_{2, \ldots,} n_h$, where n_1 is the lowest internal node and n_h is root.
  - For each such node holds:
    - node was created with one key and it has one key after each split
    - after each insertion without a split, the next insertion is performed with a split, because the node can't contain more than two keys

- First insertion creates n_1 with one key.
- Second insertion adds key to n_1.
- Next insertion splits n_1 and creates n_2 (as new root) with one key.

- After creating n_i, n_i is the root and all nodes have exactly one key.
  We need to perform 2^i insertions { $insert_1$, $insert_2$,..., $insert_{2^i}$} until n_{i+1} is created and added as new root.

  - For j: $1 <= j <= 2^i$ each $insert_j$ starts in n_1.
  - For j: $1 <= j <= 2^{i-1}$ each $insert_{2j}$ is performed with splitting of n_1.
    After this, some key is inserted to n_2.
  - For j: $1 <= j <= 2^{i-2}$ each $insert_{2^2 j}$ is performed with splitting of n_1 and n_2.
    After this, some key is inserted to n_3.
  - Last insertion, $insert_{2^i}$, is performed with splitting of n_1,...,n_i.
    After this, n_{i+1} is created and added as new root.

  We perform exactly 2^{i-1} splits on node n_i during these 2^i insertions.
  Sum of the number of splitting over n_1,...,n_i is:

  - $2^{i-1} + 2^{i-2} + 2^{i-3} + ... + 2^0 = 2^i - 1$.

- ASC <= 1:

  - #SPLIT is bounded by #NODES.

  - upper bound for the worst case (sorted sequence of keys was inserted) is 1:

    - #NODES <= #KEYS.
    - ASC = #SPLIT / #INSERT <= #NODES / #INSERT <= 1.

    - there is only one branch, to that the keys were inserted, and only nodes on this branch can contain two keys

#INSERT $\le$ #NODES + HEIGHT

$\le$ #NODES + 1 + log2((#KEYS + 1) / 2)

#SPLIT / #INSERT >=

○

Min test:

Insert n elements sequentially and then n times repeat: remove the minimal element in the tree and then insert it back.

Random test:

Insert n elements sequentially and then n times repeat: remove random element from the tree and then insert random element into the tree.

Removed element is always present in the tree and inserted element is always not present in the tree.