Balanced B+ Trees

Algorithms and Data Structures Report

Degree in Bioinformatics, UPF

Dante Aviñó - 106390

Contents

1. Introduction

The goal of this report is to study and learn from authoritative sources topics related to the Algorithms and Data Structures course.

In order to understand what B+ Trees are we first need to talk about some of the history related to this data structure. Why it is created? What problem solves? What was before?

So it's first necessary a bit about to talk about Binary search trees (BST), balanced trees, B-trees and the differences

Explain why B-trees are used when the search structure is on disk

The difference is that in B+-trees only leaf nodes contain the actual key values. The nonleaf nodes of the B+-trees contain router values.

Routers are entities of the same type as the key values, but they are not the keys stored in the search structure. They are only used to guide the search in the tree. In classical B-trees, the key values are stored in both leaf and non-leaf nodes of the tree.

2. B+ tree

2.1. Structure

A B+ tree T consists of nodes. One of the nodes is a special node T.root.

If a node x is a non-leaf node, it has the following fields:

- x.n the number of router values currently stored in node x
- The router values stored in node x in increasing order

  x.router1 < x.router2 < ... < x.routerx.n
- x.leaf , a boolean field whose value is FALSE meaning that x is a non-leaf node •
- x.n+1 pointers x.c1, x.c2, x.c3,...,x.cx.n+1 to the children of x.

If the node x is a leaf node, it has the following fields

- x.n number of key values currently stored in x. 1
- The key values stored in node x in increasing order

  x.key1 < x.key2 < ... < x.keyx.n
- x.leaf , a boolean field whose value is TRUE meaning that x is a leaf node.

If we consider a non-leaf node x and pointers in it x.c1, x.c2, x.c3,...,x.cx.n+1
the following condition is true:
$k1 \leq x.router1 < k2 \leq x.router2 < k3 \ldots < kx.n \leq x.routerx.n < kx.n+1$

where ki is any key or router value in the subtree pointed by the pointer x.ci.

An example of the non-leaf node containing 5 router values:

2.2. B+ Properties

The B+ tree has to satisfy the following balance conditions:

• Every path from the root node to a leaf node has an equal length, i.e. every leaf node has the same depth which is the height of the tree.

• Every node of the B+-tree except the root node is at least half-filled. The latter condition can be formulated more exactly: denote by s(x) the number of children of node x if x is a non-leaf node and the number of keys stored in x if x is a leaf node. Let $t \geq 2$ be a fixed integer constant. For each node x of the B+-tree, the following is true:

• $1 \leq s(x) \leq 2t$, if x is the only node in the tree.

• $2 \leq s(x) \leq 2t$, if x is the root node and the tree contains also other nodes in addition to the root node.

• $t \leq s(x) \leq 2t$, otherwise.

Because the B+-tree satisfies the given balance conditions, we can prove that the hight h of the B+ tree

$$h \leq \log_t n$$

where n is the number of the keys stored in the tree.

2.3. B+ Operations

Searching

```
BTreeSearch(T,k)
x = T.root
while not x.leaf
    i = 1
    while i ≤ x.n and k > x.routeri
        i = i+1
    x = x.ci
    DiskRead(x)
i = 1
while i ≤ x.n and k > x.keyi
    i = i+1
if i ≤ x.n and k = x.keyi
    return ( x, i )
else return NIL
```

Time Complexity

Insertion

Deletion