Insert and Delete for BST

SML Graph Theory (Spring 2024)

Vilem Fiala

June 2024

1 Introduction

In this report I will introduce the operations insert and delete for Binary Search Tree with personally drawn examples. I will show different problems regarding deleting from BTS and how to safely delete the elements, while keeping the correct order and structure of BTS.

2 Binary Search Tree

First I would like to remind us definition of Binary Search Tree from class.

Definition 3.16 (Binary Search Tree (BST)). Binary search tree (BST), also called ordered or sorted bin trees is a binary tree T = (V,E) together with a (weight) function $\omega : V \to S$, with $(S, \leq)$ a totally ordered set, such that for any $x \in V$:

- $\omega(y) \leq \omega(x)$ for any $y \in L(x)$,
- $\omega(x) \leq \omega(y)$ for any $y \in R(x)$,

where $L(x)$ and $R(x)$ are the left and and the right subtree defined below x. The values at the vertices, namely $\{\omega(x) \in S \mid x \in V\}$, are called the keys.

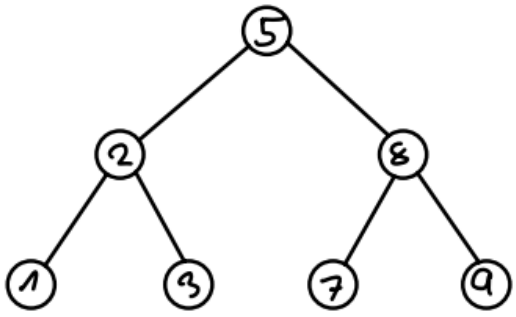
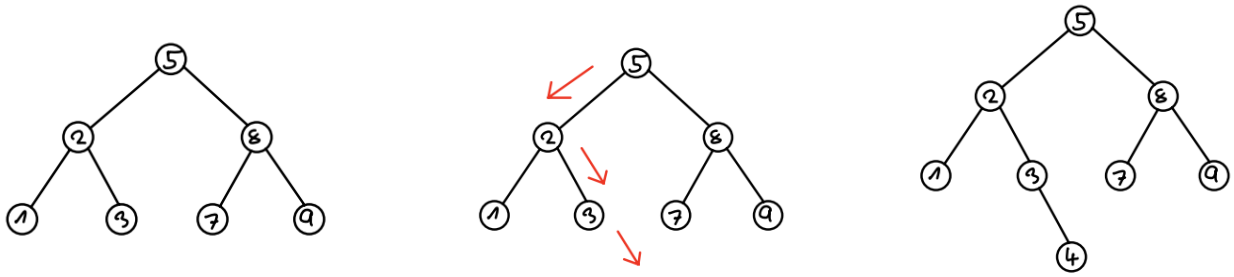


Figure 1: Example of Binary Search Tree

3 Insert to Binary Search Tree

For inserting a new vertex into Binary search tree, we simply follow the rules and go through the graph and finding empty spot that aligns with the inserted vertexe's key.

In this example, I will be adding into my existing BST vertex with key 4, which is not in the tree. First we will compare it with first vertex with key 5. As 4 is less than 5 we go into the left subtree. After going through the tree, we find an empty spot in right subtree for vertex with key 3.



(a) Initial BST (b) Action of comparing keys (c) BST with added vertex

Figure 2: Inserting vertex with key 4 into BST

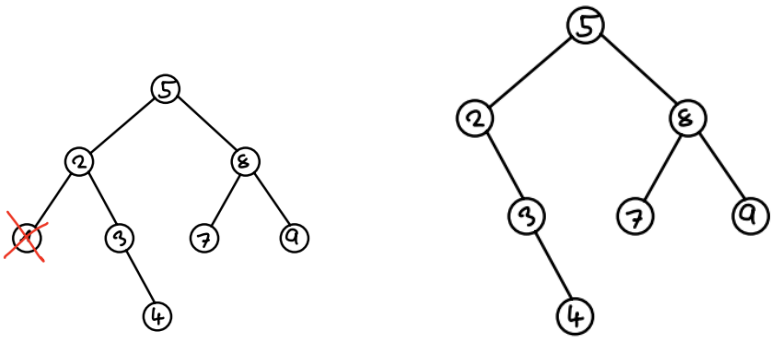
4 Delete from Binary Search Tree

Now I will examine how to correctly delete a vertex from the Binary search tree. Deleting vertex is much more complicated than inserting. As the vertex in question can possible have ascendants and we have to solve the problem without ruining the correct order in the BST.

4.1 Vertex with no ascendants - leaf

This is the easiest option out of the three, as it doesn't require any special redoing of our tree. Just simply deleting the vertex, which is leaf, solves the operation.

We will be deleting vertex with key 1, which doesn't have any ascendants, it is a leaf. We can just remove this vertex and nothing else.



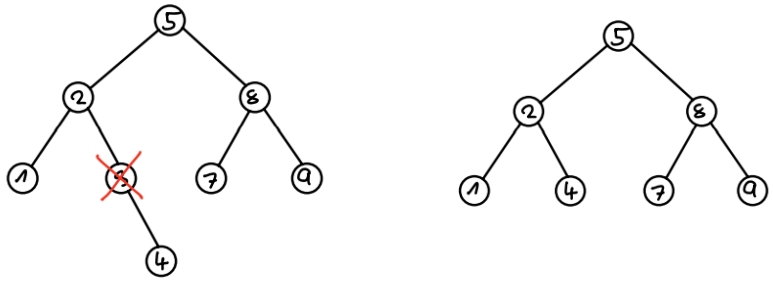
(a) Vertex to be deleted (b) Tree without the deleted vertex

Figure 3: Deleting vertex with no ascendants - leaf

4.2 Vertex with one ascendant

In this situation, it still has an easy solution. We should look out for the ascendant, so we don't forget to reconnect the edge between the vertex's ancestor and ascendant.

We will be removing vertex with key 3. We have to remember vertex with key 4, which will replace the position where the vertex with key 3 was before.



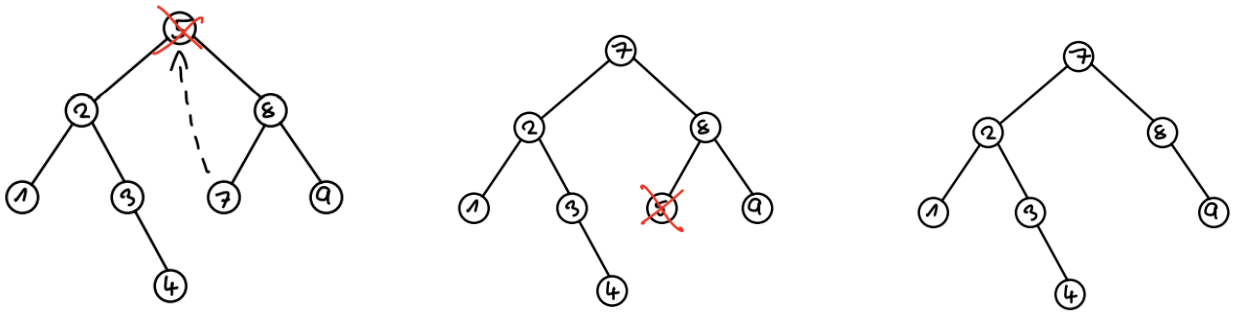
(a) Vertex to be deleted (b) Tree without the deleted vertex

Figure 4: Deleting vertex with one ascendant

4.3 Vertex with two ascendants

In the last situation, we have to look for an alternative vertex, that we can use instead of the one to be deleted, which could help us keep the structure while still removing the vertex. If we imagine the tree just as an ordered list, we can notice that the vertex will have on both sides vertices with one smaller key and one bigger. We can choose either for the purpose of removing vertex, but I chose the one with bigger key. We switch the positions of these vertices, now disrupting the order of the vertices. But then we delete the vertex with the key we wanted to delete. Now the delete problem is the same as the one, where the vertex has only one or no ascendants.

Here we will be deleting vertex with key 5, the root. We have to look for a replacement, which in our example will be the vertex with key 7, as it is the smallest bigger key compared to our vertex to be deleted. Than we just proceed like when we were suppose to remove a leaf.



(a) Vertex to be removed, potential swap (b) Swapping vertices, deleting the vertex (c) BST with deleted vertex

Figure 5: Deleting vertex with two ascendants

Swapping the first vertex with the next bigger key ensures that this vertex will have only one ascendant, with bigger key. Vertices with smaller keys is the vertex we want to delete and other vertices in the left subtree of the vertex we want to delete. That means we will keep the correct order after we proceed with the deleting.