Give a sequence of ten keys (using the letters A through K) for which, when the keys are inserted to a Binary Search Tree using the method of "Insertion at Root", the maximum amount of comparisons is required for the tree to be built. Give this number of comparisons.

Name: Anastasia Marinakou

AM: 1115202400120

Introduction

Binary Search Trees (BSTs) are a very useful structure for searching algorithms, due to their properties:

- every node has at most two children: the **left** and the **right** child
- when examining a node v , the **left subtree** will always be **less than v** , while the **right subtree** will always be **greater** than it.

So, for a BST to be most efficient, it needs to be as balanced as possible (as little differences in depth between subtrees of the same level as possible).

An un-balanced BST will also affect the insertion process of a new key, if it's done with the method of "insertion at root", where a new key is initially inserted as a leaf node. Through comparisons with other nodes and rotations, it is brought to the top of the tree, thus becoming the new root.

Insertion at Root

Let's say that we have keys A, B,..., K. When inserting the keys in that order, we will initially have A as the sole key in the tree, which will look like this:

```
  A
/ \
```

After A, B will be inserted and will be compared with A, so that it can be rotated to root position, and we will have the following tree:

```
              1. Compare A with B → B > A → go right
   A             2. Bring B to root via rotations.              B
  / \       ---------------------------------------------->    / \
     B                                                        A
    / \                                                      / \
```

That was one comparison. In continuation, key Γ will be inserted and will be compared with both A and B, which are two comparisons.

```
              1. Compare Γ with B → Γ > B → go right
              2. Compare Γ with A → Γ > A → go right
   B          3. Then bring Γ to root through rotations.         Γ
  / \       ---------------------------------------------->     / \
 A   Γ                                                         B
/ \ / \                                                       / \
                                                            A
                                                           / \
```

When inserting Δ, it will be compared with A, B and Γ (three comparisons).

```
                1. Compare Δ with Γ → Δ > Γ → go right
                2. Compare with B → Δ > B → go right
     Γ          3. Compare with A → Δ > A → go right
    / \         4. Bring Δ to root.                          Δ
   B   Δ      ---------------------------------------->    / \
  / \ / \                                                 Γ
 A                                                       / \
/ \                                                     B
                                                       / \
                                                      A
                                                     / \
```

At this point, a **pattern** can be noticed, where, when **inserting a key `i`** , it will be **compared `i-1` times** in order to rotate a specific subtree, and **bring key `i` to the root**.

So, since we have **10 keys**, the total number of comparisons required is: *1 + 2 +...+ 9 = 45* This is the **maximum amount** of comparison we get with this method and this amount of keys due to the **imbalance of the BST** that was explained in the *Introduction*.