Aufgabe4:

1. Kategorien von Symboltabellen

Es gibt zwei Hauptkategorien von Symboltabellen:

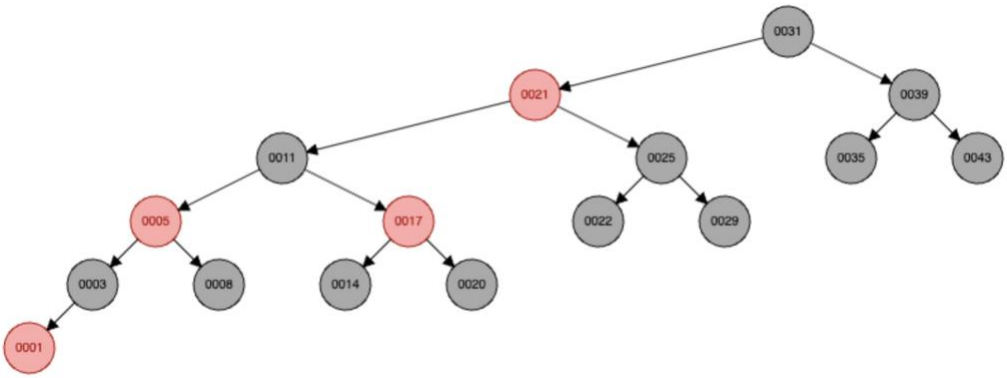
1. **Geordnete Symboltabellen (Ordered Symbol Tables)**:
   - Diese Symboltabellen speichern die Schlüssel in einer bestimmten Reihenfolge und unterstützen Reihenfolge-basierte Operationen wie `min()`, `max()`, `floor()`, `ceiling()`, `rank()`, `select()` und Bereichsanfragen.
   - Beispiele: Rot-Schwarz-Bäume (RB-BST), AVL-Bäume.£
2. **Ungeordnete Symboltabellen (Unordered Symbol Tables)**:
   - Diese Symboltabellen speichern die Schlüssel ohne Berücksichtigung der Reihenfolge und unterstützen grundlegende Operationen wie `put()` und `get()`.
   - Beispiele: Hash-Tabellen.

2. Laufzeitverhalten für den schlechtesten Fall beim BinarySearchST

Der BinarySearchST (Binary Search Symbol Table) speichert die Schlüssel in einem sortierten Array und ermöglicht die Binärsuche. Die Laufzeiten für verschiedene Operationen im schlechtesten Fall sind wie folgt:

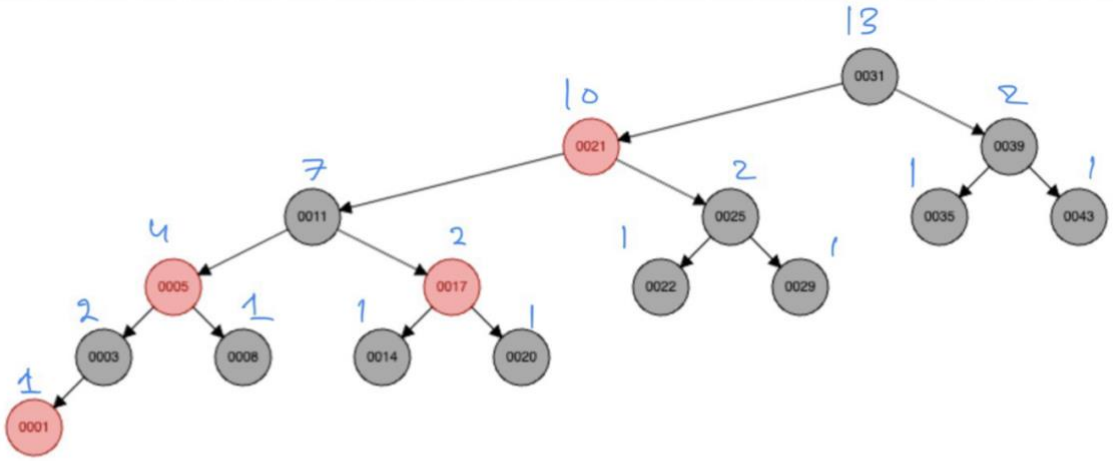
- **Suche (`get()`)**: Da die Binärsuche verwendet wird, beträgt die Laufzeit logN
- **Einfügen (`put()`)**: Im schlechtesten Fall muss ein Schlüssel an die richtige Position eingefügt werden, was eine Verschiebung aller nachfolgenden Elemente erfordert. Dies hat eine Laufzeit von N.

3.RED BLACK-TREE



4.Der Insertion Algorithmus erzeugt einen halbwegs balancierten Baum.
=> longest path kann maximal 2x länger sein als der shortest path.

5.



keys(1,24)

= ganz linker Pfad 81 ..... 1,(0) = null

=> (12,11,5,3,1)

=> (12,11,5,3,1,8) 81 5,8,0

=> (12,11,5,3,1,8,17,14,20)

=> (" " " , 22)

rank(15)

15 < 81 => 15 < 21 => 15 > 11

=> 4+1

15 < 17 => 15 > 14

=> 1+0+0

=> 4+1+1+0+0 = 6

select(24)

10 < 24 => 10+1

(24-10-1) = 18 => 13 < 24

=> 10+1+2 => (13-13) = 0

=> null