L5: Binärsuchbaum

Bekommen in L6

Abgabe in L7

Implementiere in C++ den gegebenen **Container (ADT)** mithilfe der gegebenen **Repräsentierung** und mit einem **Binärsuchbaum (BST - BinarySearchTree)** als Datenstruktur. Für die Implementierung dürft ihr keine Containers oder Datenstrukturen aus STL (oder aus andere Bibliotheken) benutzen.

1. **ADT Matrix** – repräsentiert als schwachbesetzte Matrix (sparse) mit Tupeln der Form (Zeile, Spalte, Wert) (Wert ≠ 0), wobei die Tupel in lexikographischen Reihenfolge nach (*Zeile*, *Spalte*) in einem BST mit einer verketteten Repräsentierung mit dynamischer Allokation gespeichert werden.
2. **ADT Matrix** – repräsentiert als schwachbesetzte Matrix (sparse) mit Tupeln der Form (Zeile, Spalte, Wert) (Wert ≠ 0), wobei die Tupel in lexikographischen Reihenfolge nach (*Zeile*, *Spalte*) in einem BST mit einer verketteten Repräsentierung auf Arrays gespeichert werden.
3. **ADT SortedBag** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation, wo die Elemente gespeichert sind. Falls ein Element mehrmals vorkommt, dann wird es mehrmals gespeichert.
4. **ADT SortedBag** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays, wo die Elemente gespeichert sind. Falls ein Element mehrmals vorkommt, dann wird es mehrmals gespeichert.
5. **ADT SortedBag** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation, wo Paare der Form (*eindeutiges Element*, *Frequenz*) gespeichert werden.
6. **ADT SortedBag** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays, wo Paare der Form (*eindeutiges Element*, *Frequenz*) gespeichert werden.
7. **ADT SortedSet** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation
8. **ADT SortedSet** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays
9. **ADT SortedMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation
10. **ADT SortedMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays
11. **ADT SortedMultiMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation wo Paaren der Form (*key*, *value*)

gespeichert werden. Wenn ein Schlüssel mehrere entsprechende Werte hat, dann kommt der Schlüssel in mehreren Paaren vor.

12. **ADT SortedMultiMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays wo Paaren der Form (*key*, *value*) gespeichert werden. Wenn ein Schlüssel mehrere entsprechende Werte hat, dann kommt der Schlüssel in mehreren Paaren vor.
13. **ADT SortedMultiMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation wo eindeutige Schlüssel zusammen mit einem dynamischen Array von Werten gespeichert werden.
14. **ADT SortedMultiMap** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays wo eindeutige Schlüssel zusammen mit einem dynamischen Array von Werten gespeichert werden.
15. **ADT PriorityQueue** – repräsentiert mithilfe eines BST von Paaren (Element, Priorität) mit einer verketteten Repräsentierung mit dynamischer Allokation
16. **ADT PriorityQueue** – repräsentiert mithilfe eines BST von Paaren (Element, Priorität) mit einer verketteten Repräsentierung auf Arrays

---

----

17. **ADT SortedList** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung mit dynamischer Allokation, wobei für jeden Knoten man auch die Anzahl der Elemente, die sich links des Knoten befinden speichert (siehe Vorlesung)
18. **ADT SortedList** – repräsentiert mithilfe eines BST mit einer verketteten Repräsentierung auf Arrays, wobei für jeden Knoten man auch die Anzahl der Elemente, die sich links des Knoten befinden speichert (siehe Vorlesung)