

Algorithmus Split(T,x)

Idee: Führe eine binäre Suche nach x durch und merke jeweils die Übergangsknoten von kleiner nach größer/gleich x bzw. umgekehrt. Dies sind die „Trennlinien“ die T in T1 und T2 teilen. Setze damit T1 und T2 zusammen.

Datenstrukturen: Baum besteht aus Knoten der Form:

```
Record Knoten:
    left,right : Zeiger auf Knoten
    Schluessel : Integer
```

Desweiteren benötigen wir zwei Queues TA und TB, deren Elemente folgendermaßen strukturiert sind:

```
Record TqueueElem:
    Wurzel : Zeiger auf Knoten
    Einfueg : Zeiger auf Knoten
```

Alle weiteren Hilfsvariablen sind Zeiger auf Knoten

Zeiger := T

if Zeiger.Schluessel < x:

```
    while Zeiger != NIL:
        KleinerTemp.Wurzel := Zeiger
        KleinerTemp.Einfueg := Zeiger

        while Zeiger.Schluessel < x and Zeiger != NIL:
            KleinerTemp.Einfueg := Zeiger
            Zeiger := Zeiger.right

        TA.enqueue(KleinerTemp)
        GroesserTemp.Wurzel := Zeiger
        GroesserTemp.Einfueg := Zeiger
        Zeiger := Zeiger.left

        While Zeiger.Schluessel >= x and Zeiger != NIL:
            GroesserTemp.Einfueg :=Zeiger
            Zeiger := Zeiger.left
        TB.enqueue(GroesserTemp)
```

if Zeiger.Schluessel >= x:

```
    while Zeiger != NIL:
        GroesserTemp.Wurzel := Zeiger
        GroesserTemp.Einfueg := Zeiger

        while Zeiger.Schluessel >= x and Zeiger != NIL:
            GroesserTemp.Einfueg := Zeiger
            Zeiger := Zeiger.left

        TB.enqueue(GroesserTemp)
        KleinerTemp.Wurzel := Zeiger
        KleinerTemp.Einfueg := Zeiger
        Zeiger := Zeiger.right

        While Zeiger.Schluessel < x and Zeiger != NIL:
            KleinerTemp.Einfueg :=Zeiger
            Zeiger := Zeiger.right
        TA.enqueue(KleinerTemp)
```

TA.dequeue(k)

T1 := k.Wurzel

While Elemente in TA:

```
    TA.dequeue(m)
    m.Einfueg.right := NIL
    k.Einfueg.right := m.Wurzel
    k := m
```

TB.dequeue(k)

T2 := k.Wurzel

While Elemente in TB:

```
    TB.dequeue(m)
    m.Einfueg.left := NIL
    k.Einfueg.left := m.Wurzel
    k := m
```

Resultat: T1 enthält Zeiger auf Wurzel des Baumes mit Elementen kleiner x und T2 enthält Zeiger auf Wurzel des Baumes mit Elementen größer/gleich x. Das Aufbauen der Queues TA und TB geht in $O(|Tiefe(T)|)$, da in jedem Schleifendurchlauf der Zeiger um eins an Tiefe zunimmt, also maximal $Tiefe(T)$ Schritte stattfinden. Ebenso findet der Aufbau von T1 und T2 in $O(|Tiefe(T)|)$ statt, da zusammen maximal $Tiefe(T)$ Elemente für T1 und T2 in den beiden Queues sind. Das ergibt zusammen also $O(|Tiefe(T)|) + O(|Tiefe(T)|) = O(|Tiefe(T)|)$.