```
algorithm InorderTreeWalk($x$)
    if $x \neq$ NIL then
        InorderTreeWalk(left[$x$])
        print(key[$x$])
        InorderTreeWalk(right[$x$])
    end if
end algorithm
```

```
algorithm PreorderTreeWalk($x$)
    if $x \neq$ NIL then
        print(key[$x$])
        PreorderTreeWalk(left[$x$])
        PreorderTreeWalk(right[$x$])
    end if
end algorithm
```

```
algorithm PostorderTreeWalk($x$)
    if $x \neq$ NIL then
        PostorderTreeWalk(left[$x$])
        PostorderTreeWalk(right[$x$])
        print(key[$x$])
    end if
end algorithm
```

```
algorithm TreeSearch($x, k$)
    if $x$ = NIL or $k = \text{key}[x]$ then
        return $x$
    else if $k < \text{key}[x]$ then
        return TreeSearch($\text{left}[x], k$)
    else
        return TreeSearch($\text{right}[x], k$)
    end if
end algorithm
```

```
algorithm IterativeTreeSearch($x$, $k$)
    while $x \neq$ NIL and $k \neq$ key[$x$] do
        if $k <$ key[$x$] then
            $x =$ left[$x$]
        else
            $x =$ right[$x$]
        end if
    end while
    return $x$
    ▷ alternative Rückgabe key[$x$] bei Suche nach Schlüssel
end algorithm
```

```
algorithm TreeMinimum($x$)
    while left[$x$] $\neq$ NIL do
        $x$ = left[$x$]
    end while
    return $x$
end algorithm
```

```
**algorithm** TreeMaximum($x$)
    **while** right[$x$] $\neq$ NIL **do**
        $x$ = right[$x$]
    **end while**
    **return** $x$
**end algorithm**
```

```
algorithm TreeSuccessor($x$)
    if right[$x$] $\neq$ NIL then
        return TreeMinimum(right[$x$])
    end if
    $y$ = parent[$x$]
    while $y \neq$ NIL and $x$ = right[$y$] do
        $x = y$
        $y$ = parent[$y$]
    end while
    return $y$
end algorithm
```

```
algorithm TreeInsert($T, z$)
    $y$ = NIL
    $x$ = root[$T$]
    while $x \neq$ NIL do
        $y = x$
        if key[$z$] < key[$x$] then
            $x$ = left[$x$]
        else
            $x$ = right[$x$]
        end if
    end while
    parent[$z$] = $y$
    if $y$ = NIL then  ▷ T war leer
        root[$T$] = $z$
    else if key[$z$] < key[$y$] then
        left[$y$] = $z$
    else
        right[$y$] = $z$
    end if
end algorithm
```

```
**algorithm** TreeDelete($T, z$)
    **if** left[$z$] = NIL **or** right[$z$] = $NIL$ **then**
        $y = z$
    **else**
        $y$ = TreeSuccessor($z$)
    **end if**
    **if** left[$y$] $\neq$ $NIL$ **then**
        $x$ = left[$y$]
    **else**
        $x$ = right[$y$]
    **end if**
    **if** $x \neq$ NIL **then**
        parent[$x$] = parent[$y$]
    **end if**
    **if** parent[$y$] = NIL **then**
        root[$T$] = $x$
    **else**
        **if** $y$ = left[parent[$y$]] **then**
            left[parent[$y$]] = $x$
        **else**
            right[parent[$y$] = $x$
        **end if**
    **end if**
    **if** $y \neq z$ **then**
        key[$z$] = key[$y$]
        Kopiere die Satellitendaten von $y$ in $z$.
    **end if**
    **return** $y$
**end algorithm**
```