Source Code 1. Orthogonal range search, eindimensional.

```
def onedim_range_query(binary_tree, x_min, x_max):
    # Hilfsfunktion:
    def is_in_range(x):
        return x_min <= x.key and x.key < x_max
    # Liste der gefundenen Knoten:
    global report
    report = []
    # Hilfsfunktion: Gib alle Blätter von Teilbaum in Liste report
    def update_report(node,level,count):
        global report
        if node.is_leaf():
            report+= [node.key]

    # Finde den ersten Knoten, der innerhalb des Intervalls liegt:
    split_node = find_split_node(binary_tree, x_min, x_max)

    # Es kann natürlich sein, daß dieser Knoten ein Blatt ist:
    if split_node.is_leaf():
        # Wenn dieses Blatt im gesuchten Bereich liegt, muß
        # im Rückgabewert berücksichtigt werden:
        if is_in_range(split_node.key):
            report+=[split_node.key]
            return report

    # Implicit else:

    # Beginne Weg mit Kante zum linken Teilbaum:
    v = split_node.left
    while not v.is_leaf():
        if x_min <= v.key:
            v.right.bf_transversal(update_report)
            v = v.left
        else:
            v = v.right
    # v ist nun ein Blatt: Müssen wir es berücksichtigen?
    if is_in_range(v):
        report+=[v.key]

    # Beginne Weg mit Kante zum rechten Teilbaum:
    v = split_node.right
    while not v.is_leaf():
        if x_max > v.key:
            v.left.bf_transversal(update_report)
            v = v.right
        else:
            v = v.left
    # v ist nun ein Blatt: Müssen wir es berücksichtigen?
    if is_in_range(v):
        report+=[v.key]
```

```
    return sorted(report)
```