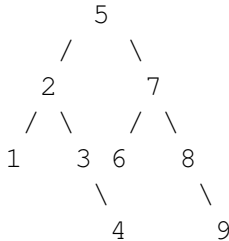


## Solutions to Exam 2011 Assignments

### Exercise 2

1.



2. To build the tree from the whole array, call `BUILDTREE( $A, 1, n$ )`.

`BUILDTREE( $A, l, r$ )`

```
1  if  $r < l$  then return NIL
2   $q \leftarrow \lfloor (l + r) / 2 \rfloor$ 
3   $root \leftarrow newNode()$ 
4   $root.key \leftarrow A[q]$ 
5   $root.left \leftarrow BUILDTREE(A, l, q - 1)$ 
6   $root.right \leftarrow BUILDTREE(A, q + 1, r)$ 
7  return  $root$ 
```

The running time of the algorithm is  $O(n)$ , which can be easily seen as a solution to the recurrence  $T(n) = 2T(n/2) + O(1)$ .

3. We assume that the algorithm gets a non-nil tree. It returns a triple  $(l, r, n)$ . If  $n = -1$  then the tree is not a completely balanced binary search tree, otherwise  $[l, r]$  is the range of values in the tree and  $n$  is the number of nodes in the tree.

`CHECKTREE( $t$ )`

```
1  if  $t.left = \text{NIL}$  then
2       $ln \leftarrow 0$ 
3       $l \leftarrow t.key$ 
4  else
5       $(l, lr, ln) \leftarrow \text{CHECKTREE}(t.left)$ 
6      if  $lr > t.key$  then  $ln \leftarrow -1$   $\triangleright$  binary search tree property broken
7  if  $t.right = \text{NIL}$  then
8       $rn \leftarrow 0$ 
9       $r \leftarrow t.key$ 
10 else
11      $(rl, r, rn) \leftarrow \text{CHECKTREE}(t.right)$ 
12     if  $rl < t.key$  then  $rn \leftarrow -1$   $\triangleright$  binary search tree property broken
13 if  $ln = -1$  or  $rn = -1$  or  $|ln - rn| > 1$  then  $n \leftarrow -1$ 
14 else  $n \leftarrow ln + rn + 1$ 
15 return  $(l, r, n)$ 
```

The running time of the algorithm is  $O(n)$  as it traverses the whole tree performing  $O(1)$  work on every node of the tree.

### Exercise 3

1. A weighted, directed graph is a suitable model. An edge corresponds to a single travel direction of a road. Weights are average travel times. An intersection is modeled as two vertices *in* and *out* connected with an edge (*in*, *out*) weighted with an average time to traverse the intersection. The vertex *in* has only ingoing edges connected to it and the vertex *out* has only outgoing edges connected to it.

2. Run BFS from  $s$ . Then, check all vertices and output those that have  $d = \infty$ . The worst-case complexity is the same as that of the BFS, which is  $O(V + E)$ .

3. Let  $G^T$  be a transpose of  $G = (V, E)$ , the road network graph.  $G^T = (V, E^T)$  is a graph where edge directions are reversed:  $E^T = \{(u, v) : (v, u) \in E\}$ . Given an adjacency-list representation of  $G$ , it is trivial to compute  $G^T$  in  $O(V + E)$  time. To solve the assignment, we run Dijkstra's from  $s$  in  $G$  and run Dijkstra's again from  $s$  in  $G^T$ . Then, we check all vertices and output those vertices  $v$ , that have  $|v_G.d - v_{G^T}.d| \leq d$ . Here  $v_G$  and  $v_{G^T}$  is vertex  $v$  in graphs  $G$  and  $G^T$ .

The running time of the algorithm is  $O(E \lg V)$ , if a min-heap is used in Dijkstra's. Less efficient algorithm would run  $V$  Dijkstra's: from each vertex backwards towards  $s$ .