Homework 5
Fundamental Algorithms, Fall 2011, Professor Yap

Due: Thu Oct 27, in class.

INSTRUCTIONS:

- Please write clearly, and economically. Use notations we introduce in lecture notes and in lectures.

1. (8 Points, Checking BST Property) In the previous homework, you designed a recursive subroutine CheckBST(u) to check if the binary tree Tu rooted at u represents a BST. We now explore a different solution. Design a recursive subroutine bCheckBST(u, min, max) that returns a Boolean value; the value is true iff Tu is a BST with keys lying in the range [min, max]. REMARK: we also define bCheckBST(u) to be bCheckBST(u, −∞, +∞).

2. (4 Points, (a, b)-Search Tree) Justify the following statements about (a, b)-search trees:
   (a) If we only have insertions into an (a, b)-tree, then the keys in an internal node are just copies of keys of items found in the leaves.
   (b) It is possible to maintain the property in part (a) even if there are both insertions and deletions.

   ![Figure 1: A (3,4)-search tree on 14 items](image)

3. (4 Points, (a, b)-Search Tree) Consider the tree shown in Figure 1. Although we previously viewed it as a (3, 4)-tree, we now want to view it as a (2, 4)-tree. For insertion/deletion we further treat it as a (2, 4, 1)-tree.
   (a) Insert an item (whose key is) 14 into this tree. Draw intermediate results.
   (b) Delete the item (whose key is) 4 from this tree. Draw intermediate results.

4. (6 Points, (a, b)-Search Tree) Our insertion and deletion algorithms tries to share (i.e., donate or borrow) children from siblings only. Suppose we now relax this condition to allow sharing among “first cousins” (i.e., nodes that share a common grandparent). Modify our insert/delete algorithms so that we try to share with direct siblings or cousins before doing the generalized split/merge.

5. (8 Points, Graph Reversal) The following is a basic operation for many algorithms: given a digraph G represented by adjacency lists, compute the reverse digraph Grev in time O(m + n). Recall (Lecture 1, Appendix) that u−v is an edge of G iff v−u is an edge of Grev. Show that your algorithm has the stated running time.

6. (8 Points, BFS Classification of Digraph Edges) (a) Give a classification of the edges of a digraph G relative to the operations of running the BFS algorithm on (G; s0). You must prove why your classification is complete. (b) Now turn your answer in part(a) into a “computational classification”. I.e., devise an algorithm to classify every edge of G according to (a). Recall that you must use shell programming.

7. (12 Points, Graph Center, Radius, and Diameter) Let G = (V, E) be a connected bigraph. For any vertex v ∈ V define

    \[ \text{radius}(v, G) := \max_{u \in V} \text{distance}(u, v) \]

where distance(u, v) is the length of the shortest (link-distance) path from u to v. The center of G is the vertex v0 such that radius(v0, G) is minimized. We call radius(v0, G) the radius of G and denote it by radius(G).
Define the *diameter* \( \text{diameter}(G) \) of \( G \) to be the maximum value of distance \((u, v)\) where \( u, v \in V \).

(a) Prove that \( 2 \cdot \text{radius}(G) \geq \text{diameter}(G) \geq \text{radius}(G) \).

(b) Show that for every natural number \( n \), there are graphs \( G_n \) and \( H_n \) such that \( n = \text{radius}(G_n) = \text{diameter}(G_n) \) and \( \text{diameter}(H_n) = n \) and \( \text{radius}(H_n) = \lceil n/2 \rceil \). This shows that the inequalities in (a) are the best possible.

(c) Using DFS, give an efficient algorithm to compute the diameter of a undirected tree (i.e., connected acyclic undirected graph). Please use shell programming. Prove the correctness of your algorithm. What is the complexity of your algorithm? HINT: write down a recursive formula for the diameter of a tree in terms of the diameter and height of its subtrees.

(d) Same as (c), but compute the radius instead of diameter.

(e,f) Same as (c) and (d) but using BFS instead of DFS.