

Homework 1
Due Date: May 28, 2009

Problem 1: (20 points)

A full binary tree is a binary tree where every internal node has two children and all the leaves are at the same level. The levels are labeled $0, 1, 2, \dots, h$, top to bottom, where h is the height of the tree.

- How many nodes are in level i ? Prove your answer by induction on i .
- Derive the number of nodes of a full binary tree of height h , in terms of h .
- Let $T(n)$ be a function of n where $T(1) = 1$ and $T(n) = T(n - 1) + n$ for all $n \geq 1$. Prove by induction on n that $T(n) = \frac{n(n+1)}{2}$.
- Give an example of a recursive algorithm whose time complexity satisfies the relation $T(n) = T(n - 1) + cn$.

Problem 2: (25 points)

Consider a binary tree T of n nodes where every node has 4 fields: LABEL, which is an integer between 1 and n representing the label of the node; *lchild*, which is a pointer to the left child; *rchild*, which is a pointer to the right child; and *parent*, which points to the parent. The values of the four fields of every node are assumed to have been computed and stored. Let $H[1..n]$; $C[1..n]$ and $D[1..n, 1..n]$ be three arrays.

- Give a recursive algorithm that takes T as input and computes the depth of every node. The algorithm should store the depth of node of LABEL i in $H[i]$.
- Give a recursive algorithm that takes the tree T as input, and computes the number of descendants of every node. The algorithm should store the number of descendants of node of LABEL i in $C[i]$.
- Define the distance between any two nodes i and j in the tree to be the number of edges of the shortest path between i and j . Assume the depth $H[i]$ of every node i has been computed. Let p be a pointer to a node of LABEL I in T . Give a recursive algorithm that takes T and p as input, and computes the distance between node p and every other node in the tree T . The algorithm should store the distance between node p and any node of LABEL j in $D[I,j]$.
- Give the time complexity of your algorithms of parts (a) and (b).

(For full credit, your algorithms should be most efficient.)

Problem 3: (11 points)

Insert 17, 19, 13, 10, 7, 14, 20, 15, 5 (as you read them) into an originally empty min-heap. Show how the heap looks like after each insertion. Show also step by step how one delete-min

is done on the heap.

Problem 4: (12 points)

Sort 17, 19, 13, 10, 7, 14, 20, 15, 5 using heapsort. Show how the heap looks like (in a tree form) after each delete-min. (You may use the heap you constructed in Problem 3).

Problem 5: (12 points)

Insert 17, 19, 13, 10, 7, 14, 20, 15, 5 (as you read them) into an originally empty binary search tree. Show how the tree looks like after each insertion. Perform delete(13) on this tree step by step, showing how the tree looks like after each step.

Problem 6: (20 points)

Consider elements 1, 2, . . . , 10. Perform the following sequence of Unions (U) and Finds (F) using the path compression algorithms, show how the forest looks like after each operation, and display the PARENT array alongside each snapshot of the forest: U(1,3) U(1,2) U(1,4) U(5,6) U(1,5) F(4) U(7,8) U(9,10) U(7,9) F(8) U(1,7) F(10) F(8) F(6).