

Homework 1  
Due Date: June 9, 2009

Problem 1: (16 points)

- Show step by step how the heapsort algorithm sorts the following array: 10, 5, 3, 6, 9, 15, 20, 12, 4, 8.
- Show step by step how the quicksort algorithm sorts the same array. Indicate at each step what the partitioning element is.
- Show step by step how the mergesort algorithm sorts the same array.
- Show step by step how the insertion sort algorithm sorts the same array.

Problem 2: (15 points)

The non-recursive way of sorting an array using mergesort is to scan the array merging consecutive pairs of size one, then merging pairs of size 2, then merging pairs of size four, etc. Write the algorithm which carries out this process.

Problem 3: (15 points)

Let  $T$  be a complete binary tree of  $n$  nodes.

- Show that the height  $h$  of  $T$  is  $h = \lceil \log(n + 1) \rceil - 1$ .
- Show that either the left subtree of  $T$  is a full tree or the right subtree of  $T$  is a full tree.
- Let  $n_l$  be the number of nodes in the left subtree of  $T$ . Compute  $n_l$  in terms of  $n$ . (Note that the value of  $n_l$  is one of two values depending on whether the left or the right subtree of  $T$  is full).
- If  $T$  is also a binary search tree, and the data item in the root is the  $k^{\text{th}}$  smallest item in the tree. What is the value of  $k$ ?

Problem 4: (15 points)

Suppose you have an unsorted array of  $n$  numbers. Give a divide-&-conquer algorithm to construct a complete binary search tree containing the data items of the array. Each node in the tree is represented by a record of 3 fields: data field, left-pointer and right-pointer. Give the time complexity of your algorithm.

Problem 5: (16 points)

- Give a recursive algorithm that prints out in a sorted order the data stored in a binary search tree. Give the time complexity of your tree.
- Give a divide-&-conquer algorithm that takes as input a sorted array and constructs a binary search tree of the data in the input array. Give the time complexity of the algorithm.

c) Conclude from (a) and (b) an algorithm that merges two binary search trees into one binary search tree. Your algorithm should take  $O(n + m)$ , where  $n$  and  $m$  are the number of nodes in the first tree and second tree, respectively.

d) If  $m$  is a small constant and  $n$  is very large, is there a faster algorithm for merging the two trees? Explain.

Problem 6: (10 points)

Consider this instance of the knapsack problem:

The weights are: 13, 10, 12, 5, 20, 9, 5, 3

the prices are : 5, 3, 9, 4, 8, 3, 2, 7

and the capacity  $M=30$ .

Apply the greedy method algorithm to find the solution (that is, how much of every item we are going to get). Show the process of deriving the solution step by step.

Problem 7: (13 points)

Let  $G = (V, E)$  be the following weighted graph:  $V = \{1, 2, 3, 4, 5, 6, 7\}$  and  $E = \{(1, 2); 3\}, \{(1, 4); 5\}, \{(2, 4); 4\}, \{(1, 3); 1\}, \{(3, 5); 2\}, \{(4, 5); 3\}, \{(2, 6); 3\}, \{(4, 6); 2\}, \{(5, 7); 2\}, \{(6, 7); 3\}$ , where  $\{(i, j); a\}$  means that  $(i, j)$  is an edge of weight  $a$ . Using the greedy single-source-shortest-path algorithm, find the distances between node 1 and all the other nodes. Show the values of the DIST array at every step.