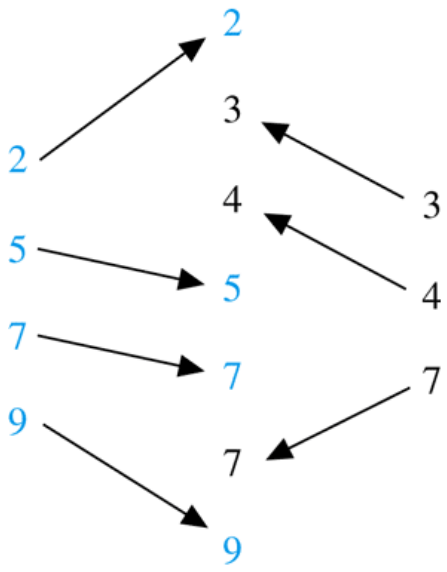
COMP 2121

DISCRETE

MATHEMATICS

Lecture 17



Optimal Weighted Trees and Sorted Lists

Another application of Huffman's optimal binary tree algorithm arises in regard to the merging of sorted lists.

To merge together two sorted lists L_1 and L_2 into a single sorted list L we perform the following algorithm:

Merge Algorithm

Step 1: Set L equal to the empty list.

Step 2: Compare the first elements in L_1, L_2. Remove the smaller of the two from the list it is in and place it at the end of L.

Step 3: If either of L_1, L_2 is empty then

the other list is concatenated to the end of L

else go to step 2.

Example 1. Apply the Merge Algorithm to the two lists of numbers sorted in increasing order:

2, 5, 7, 9 and 3, 4, 6.

Example 2. Give an example of two lists L_1, L_2, each of which is sorted in ascending order and contains five elements, and where

a) five comparisons are needed

b) nine comparisons are needed

to merge L_1, L_2 by the Merge algorithm.

Theorem. Let L_1, L_2 be two sorted lists of ascending numbers, where L_1 contains n_1 elements and L_2 contains n_2 elements. Then L_1 and L_2 can be merged into one ascending list L using at most $n_1 + n_2 - 1$ comparisons.

Example 3. Suppose we have three sorted lists L_1, L_2 and L_3 containing 150, 320 and 80 numbers, respectively. By merging only two lists at the time, how can we merge these three lists into one sorted list so that we minimize the number of needed comparisons?

Example 4. We wish to merge five lists with 20, 30, 40, 60, and 80 numbers (each sorted in ascending order) into a single list. What is the optimal merge pattern? What is the maximum number of comparisons that the optimal merge pattern uses?