Kurskod	D0012E
Tentamensdatum	2008-01-14
Skrivtid	4 timmar
Lärare	Jingsen Chen (492044)

Tillåtna hjälpmedel: räknare, Beta, språk ordbok

1. Sort the following numbers step by step:

   (a) Sort 24, 13, 17, 34, 15, 27, 15, 29, 22, 26 using *mergesort*; **(5p)**

   (b) Sort 24, 13, 17, 34, 15 using *insertionsort*. **(5p)**

2. By repeatedly inserting the input elements 22, 13, 17, 32, 19, 9 in turn:

   (a) Construct a hash table of size 10 using linear probing; **(5p)**

   (b) Construct a binary search tree. **(5p)**

3. Prove (by using the definitions) or disprove (by giving counterexamples) the following assertions.

   (a) Let $f(n)$ and $g(n)$ be positive integer functions. If $f(n) = O(g(n))$, then $g(n) = \Omega(f(n))$. **(5p)**

   (b) Let the worst-case time complexity of two algorithms A and B be $\Theta(t(n))$ and $O(t(n))$, respectively. Then, algorithm A is asymptotically faster than algorithm B. **(5p)**

4. Given three pegs A, B, and C with n disks, $d_1, d_2, \cdots, d_n$, of different sizes. Initially, all the disks are placed on peg A in order of size (that is, a larger disk is always below a smaller disk). The goal is to move all the disks to peg C using peg B as an auxiliary, if necessary. Assume that one can move only one single disk at the top of a peg to another peg at a time, and it is forbidden to place a larger disk on the top of a smaller disk. The following algorithm solves this problem:

   - Move recursively the top $n-1$ disks from peg A to peg B, using peg C as auxiliary;
   - Move the remaining disk from peg A to peg C;
   - Move recursively the $n-1$ disks from peg B to peg C, using peg A as auxiliary.

$n \log n$

   (a) Compute the number of disk moves used by this algorithm for n disks (using Θ-notation). **(6p)**

   (b) Assume that the size of the disk d_i is i and $2i$ time units are needed to move the disk d_i from one peg to another peg for $i = 1, 2, \cdots, n$. Compute the total number of time units consumed by this algorithm for n disks (using Θ-notation). **(4p)**

5. Given a graph with the adjacency matrix:

$$\begin{array}{c|ccccc} & a & b & c & d & e \\ \hline a & - & -2 & 6 & - & - \\ b & -2 & - & - & 7 & 4 \\ c & 6 & - & - & - & 2 \\ d & - & 7 & - & - & 3 \\ e & - & 4 & 2 & 3 & - \end{array}$$

   (a) Draw this graph. **(2p)**

   (b) Consider the problem of computing a minimum spanning tree of this graph. Edges (a, b), (c, e), and (d, e) have already been put into the minimum spanning tree. Determine whether these edges were selected by Prim ´s algorithm or by Kruskal ´s algorithm. Justify your answers. No credit will be given without justifications. **(4p)**

(c) Show how to perform a depth-first search and a breadth-first search on the above graph starting at the node b, respectively. **(5p)**

(d) Explain why Bellman-Ford´s single-source shortest path algorithm does not work on this graph. No credit will be given without explanations. **(4p)**

6. Let **S** be an array of n elements. Each element in **S** is colored with either red, blue, or black. The task is to rearrange the array so that all the red elements precede all the blue ones and all the blue elements precede all the black ones. Your algorithm should be in-place and run in $O(n)$ time in the worst case. **(10p)**

7. Let $A = \{a_1, a_2, ..., a_n\}$ be distinct numbers. Design an algorithm to compute the sum $\sum_{i=1}^{\lfloor \log_2 n \rfloor} b_{2^i}$, where b_j the j^{th} smallest element in A. You may assume that n is a power of 2 and your algorithm should run in time $O(n)$ in the worst case. **(10p**

8. An *anagram* of a word W is another word made up of the same letters as W. For example, stop, tops, and post are anagrams of each other. Given a set of words, design an efficient algorithm to make a list for each word of all its anagrams that appear in the set. Let n denote the sum of lengths of the words in the set. If we count only the number of letter-letter comparisons used, your algorithm should run in $O(n)$ time and space in the worst case. **(10p)**

9. Show how to determine in $O(n^2 \log n)$ time whether any three points in a given set of n points in the plane are colinear (that is, they lie on the same line). **(15p)**