系所組別： 資訊管理研究所乙組

考試科目： 資料結構

考試日期：0225，節次：3

1. Let B be the transpose of square matrix A with n rows. Matrix B can be obtained by the following three ways. One is to store every element by using a two-dimensional array and move the element in row j and column i of A to be the element in row i and column j of B. The second approach is to consider A as a sparse matrix and store every nonzero element in A as a 3-tuple (row, column, value). Then the elements in the same column of A are searched and moved to be the elements in the same row of matrix B. In the third approach, every nonzero elements in A is also stored as a 3-tuple (row, column, value), and the number of elements in a row of B and the starting position of that row are determined by A and recorded. Then the position in B for the element in row i and column j of A can be calculated by the recorded values.

   (1) (10%) Which approach will have the smallest storage requirement? Justify your answer.

   (2) (15%) Which approach will have the smallest computing time? Justify your answer.

2. (10%) Give an expression to calculate the number of nodes in a full k-ary tree, and prove it.

3. Let the node structure of the generalized list for polynomials have three fields: Coefficient, Exponent, and Link.

   (1) (5%) Use this data structure to represent polynomial $P(x, y, z) = 3x^6y^2z^2 + 2x^8y^4z + 6x^2y^2z^2 + xy^2z^2+5x^3$.

   (2) (3%) Point out a potential problem of this data structure.

   (3) (7%) Provide another data structure that can overcome the potential problem given in (2), and use this new data structure to represent the polynomial P(x, y, z) given in (1).

4. TRUE OR FALSE (2% each; in total, 26%)

(1) Any binary decision tree that sort n distinct elements must have 2^{n-1} leaves.

(2) An advantage of the division method for creating hash function $h(k)$, where a key k is a natural number and is mapped into one of m slots, is that the value of m is not critical.

(3) When sorting n elements, any algorithm which sorts by comparisons only must have a worst case computing time of $O(n \log_2 n)$.

(4) In a height-h binary search tree T whose keys are distinct, k successive calls to TREE-SUCCESSOR take $O(k+h)$ time.

(5) Every n-node arbitrary binary search tree has at most $n-1$ possible right rotations, and it can be transformed into any other arbitrary n-node binary search tree using $O(n)$ rotations.

(6) The cost of properly updating the f fields in order-statistic tress after per rotation, in many cases, is $O(\log n)$.

(7) There is no way in general to tell if a greedy algorithm will solve a particular optimization problem even when the properties of the problem have presented in front of us.

(8) Huffman's greedy algorithm can be proved to be wrong when there exists an optimal prefix code for C in which the ternary codewords (i.e., using the symbols 0, 1 and 2) for x and y have different length and differ in the last bit.

(9) In an amortized analysis, the time required to perform a sequence of data-structure operation can be averaged to guarantee the cost of each single operation in the best case is small.

(10) For each minimum spanning tree T of the same input graph G, there is a way to sort the edges of G in Kruskal's algorithm so that the algorithm returns T.

(11) Dijkstra's algorithm is like Kruskal's algorithm for computing minimum spanning trees.

(12) Many database systems use B-trees, or variants of B-trees, to store information.

(13) Fibonacci heaps are loosely based on binomial heaps.

5. SHORT ANSWER (6% each; in total, 24%)

(1) Please list the red-black properties for a *binary search tree* so that it is a red-black tree.

(2) How the *greedy programming* can be different from dynamic programming in a classical optimization problem? Please give at least one example to explain such difference.

(3) How and why an arbitrary schedule can always be put into *early-first form* or *canonical form*?

(4) How are *B-trees* different or similar to *red-black trees*?