

Wie numbe

the following questions re

is the detailed processing during the lifetime of an AR? etc.).

ent sparse matrices as linked lists? Your answer should include the node

it deleted? WI

7.

(a) (8%) How to represent sparse matrices as lin

atrix and set up its linked representation. Note that it

structure and a pseudo code to read in a matrix and set up its linked repres

design introduced in the textbook and creative ideas are is not necessary to follow the design introduced in the textbook and encouraged.

of the sparse matrix read-in is an m x n matrix with k

**(6) (3%) Assume that the given argument of the sparse matrix read-in**

non-zero entries. Determine the time complexity of your pseudo code answered in (a). (©) (8%) Please explain how to perform "in place" chain inversion of a singly linked list.

8. (4%) Write the postfix forms of the following expressions:

(i) A-B\*D+E/F+A\* D+C (ii) (A - B) \* D + E / (F+A\* D) + C

9. (5%) In this question, you have to derive the worst case time complexity of the function padd

given as follows. It is used to add two polynomials A and B and the result is stored in D (Let m and n be the number of nonzero terms in polynomials A and B.) Assume that only one global array, terms, is used to store all polynomials and the index of the next free location in the array is given by avail. The index of the first term of A and B is given by startA and startB, respectively, while finishA and finishB give the index of the last term of A and B. Moreover, attach is a constant-time subprogram to place the terms of D into terms.

**#define MAX\_TERMS 100 typedef struct {**

**float coef; int expon;**

} polynomial; polynomial terms (MAX\_TERMS] ; int avail = 0; void padd (int starta, int finishA, int startB, int finishi,

int \*startD, int \*finishD)

float coefficient; \*startD = avail; while (startA <= finishA && startB <= finishB) switch (COMPARE (terms (startA] .expon,

terms [startB] .expon)) { case -1: /\* a expon < b expon \*/

attach (terms (startb].coef, terms (startB] .expon); startB++;

break; case 0: /\* equal exponents \*/

coefficient=terms (startA] .coef+terms (startb] .coef; if (coefficient)

attach(coefficient, terms [startA] . expon); startA++; startB++;

break; case 1: /\* a expon > b expon \*/

attach(terms [startA).coef, terms (startA] .expon);