Algoritmi e Strutture Dati 15/16 *Durata: 3 ore*

Prova Scritta del 3 Febbraio 2016

1. In matematica una matrice sparsa è una matrice i cui valori sono quasi tutti uguali a zero. Rappresentare una matrice sparsa con un array bidimensionale corrisponderebbe ad un grosso spreco di memoria. Si preferisce definire una matrice sparsa come un insieme di triple $< r, c, v >$ dove ogni combinazione $r - c$ nell'insieme è unica (r sta per riga, c per colonna e v per valore).

   Si vuole progettare una struttura dati per matrici sparse. Completare la specifica di *matrice sparsa* (ms), fornendo la specifica semantica per mezzo di pre e post condizioni (specifica costruttiva o modello astratto), rispetto alla seguente specifica sintattica:

   domini: sm, intero, valore

   operatori:

   (a) crea() → ms
   // *crea una nuova matrice sparsa*

   (b) aggiungi(ms, intero, intero, valore) → ms
   // *aggiunge alla matrice un valore non nullo v in posizione riga r, il primo intero, e colonna c, il secondo intero. La matrice include la nuova tripla* $< r, c, v >$

   (c) rimuovi(ms, intero, intero) → ms
   // *rimuove dalla matrice l'elemento in riga r, il primo intero, e colonna c, il secondo intero*

   (d) leggi(ms, intero, intero) → valore
   // *restituisce il valore in riga r, il primo intero, e colonna c, il secondo intero*

   (e) trasposta(ms) → ms
   // *calcola la trasposta di una matrice*

   (f) somma(ms, ms) → ms
   // *effettua la somma di due matrici* [7pt]

2. Fornire in C++ una possibile realizzazione della struttura dati matrice sparsa al punto 1), riportando solo la definizione di classe: variabili di classe e definizione dei metodi. Motivare la scelta di altre strutture dati nel caso se ne faccia uso. [4pt]

3. Fornire la specifica sintattica e semantica degli operatori *insSottoAlbero* e *insPrimoSottoAlbero* per la struttura dati *Alberi n-ari* [3pt]

4. Spiegare la realizzazione di alberi n-ari mediante *vettore dei padri*, *liste di figli* e con *cursori*, fornendo vantaggi e svantaggi di ognuna [5pt]

5. Fornire in pseudocodice l'algoritmo di *ricerca in profondità* (DFS) per alberi n-ari [3pt]

6. Descrivere come uno stack e una coda possano essere rappresentati mediante vettore [3pt]

7. Spiegare la tecnica algoritmica del *backtracking* [4pt]

8. Motivare le differenze fra una tecnica greedy e una di backtracking quando applicate ad un *problema di ottimizzazione* [4pt]