Algoritmi e Strutture Dati

Appello del 24 Giugno 2008

Modulo Teoria

Attenzione: Scrivere nome, cognome e numero di matricola sul foglio protocollo

Esercizio 1 [Punti 20]

Si descriva un algoritmo di ordinamento che segua la struttura di MergeSort ma che operi su una partizione dell'array in tre parti anzicche' in due.

1. Scrivere lo pseudocodice della nuova procedura MergeSort3.
2. Descrivere a parole la nuova procedura Fusione3.
3. Impostare e risolvere l'equazione di ricorrenza associata.

Esercizio 2 [Punti 10]

Dato il grafo orientato $G = (V, E)$ di 8 vertici e 11 archi

$$E = \{(a, b), (a, f), (b, c), (c, d), (c, g), (e, a), (f, b), (f, e), (f, g), (g, h), (h, d)\}$$

rappresentato con liste di adiacenza ordinate alfabeticamente,

1. indicare l'ordine in cui i vertici sono scoperti dalle visite BFS e DFS del grafo partendo dal vertice a;
2. disegnare gli alberi BFS e DFS ottenuti con le visite.

Esercizio 3 [Punti 2]

Discutere qual é il caso ottimo di QuickSort e determinarne la complessitá.

Modulo Laboratorio

Attenzione: Scrivere nome, cognome e numero di matricola sul foglio del testo e sul foglio protocollo

Esercizio 1 [Punti 15]

Si dia un'implementazione in codice java dell'algoritmo di Boyer-Moore, assumendo di avere il seguente metodo (di cui omettiamo i dettagli implementativi). Tale metodo prende in ingresso una stringa, e restituisce un array di 128 interi, indicizzato dal codice ASCII dei caratteri. Al codice di ogni carattere presente nella stringa corrisponde l'indice di ultima occorrenza del carattere nella stringa, a tutti gli altri corrisponde il valore -1.

```
public static int[] buildLastFunction (String pattern);
```

Esercizio 2 [Punti 15]

1. In cosa la programmazione dinamica si differenzia da quella ricorsiva?
2. Si commenti con javadoc il seguente codice, che implementa un algoritmo per il calcolo della massima sottosequenza comune a due date stringhe.

```
  public static String maxSSC(String x, String y) {

int m = x.length();

int n = y.length();

// la matrice c contiene le lunghezze dei risultati parziali dei

// sottoproblemi: c[m][n] rappresenta la lunghezza della massima

// sottosequenza comune alle stringhe x[0..m-1] e y[0..n-1].

    int c[][] = maxLunghezza(x,y);

    int l = c[m][n];

    StringBuffer z = new StringBuffer(l);

    z.setLength(l);
```

```
  l--;
  while (m > 0 && n > 0) {
    if ( x.charAt(m - 1) == y.charAt(n - 1) ) {
      z.setCharAt(l--, x.charAt(--m));
      n--;
    } else
    if (c[m - 1][n] >= c[m][n - 1])
      m--;
    else
      n--;
  }
  return z.toString();
}
```