



## Esercitazione su temi d'esame

31 Maggio 2019

### 1. Memoization (15 Luglio 2014)

La procedura `llcs3` determina la *lunghezza della sottosequenza comune più lunga* (LLCS) di tre stringhe:

```
(define llcs3
  (lambda (t u v)
    (cond ((or (string=? t "") (string=? u "") (string=? v ""))
           0)
          ((char=? (string-ref t 0) (string-ref u 0) (string-ref v 0))
           (+ 1 (llcs3 (substring t 1) (substring u 1) (substring v 1))))
          (else
           (max (llcs3 (substring t 1) u v)
                (llcs3 t (substring u 1) v)
                (llcs3 t u (substring v 1)))))))
```

Trasforma la procedura `llcs3` in un programma *Java* che applica opportunamente la tecnica *top-down* di *memoization*.

### 2. Programmazione in Java (15 Luglio 2014)

Come è noto dall'algebra lineare, una matrice quadrata  $Q$  si dice *simmetrica* se coincide con la propria trasposta, ovvero se  $Q_{ij} = Q_{ji}$  per tutte le coppie di indici della matrice. Scrivi un metodo statico in *Java* per verificare se l'argomento è una matrice simmetrica — assumendo che tale argomento rappresenti una matrice *quadrata*.

Per quanto possibile, struttura il codice in modo tale da ridurre al minimo il numero di confronti effettuati dal programma nei casi in cui la matrice sia effettivamente simmetrica.

### 3. Oggetti in Java (9 Settembre 2015)

Il modello della scacchiera realizzato dalla classe `Board` per affrontare il rompicapo delle  $n$  regine deve essere integrato introducendo due nuovi metodi: `isFreeRow(int)`, che dato un indice di riga  $i$  compreso fra 1 e la dimensione  $n$  della scacchiera, restituirà *true* se e solo se nella riga  $i$  non c'è alcuna regina; `addQueen(String)`, che svolge la stessa funzione di `addQueen(int, int)` ma ricevendo come argomento la codifica della posizione tramite una stringa di due caratteri, una lettera per la colonna e una cifra per la riga secondo le convenzioni consuete. Inoltre, `addQueen` e `removeQueen` non devono modificare lo stato della scacchiera se l'operazione è inconsistente perché due regine verrebbero a trovarsi sulla stessa casella oppure perché nella posizione data non c'è una regina da rimuovere. Per esempio, il metodo statico `listOfCompletions`, definito sotto a destra, stamperà tutte le soluzioni del rompicapo, se ve ne sono, compatibili con una disposizione iniziale di regine che non si minacciano reciprocamente.

In base a quanto specificato sopra, modifica opportunamente la classe `Board`.

```
Board( int n ) //costruttore

void addQueen( int i, int j )
void removeQueen( int i, int j )

int size()
int queensOn()
boolean underAttack( int i, int j )
String arrangement()

boolean isFreeRow( int i )
void addQueen( String pos )

// esempio: b.addQueen( "b6" )

public static void listOfCompletions( Board b ) {
  int n = b.size(); int q = b.queensOn();
  if ( q == n ) {
    System.out.println( b.arrangement() );
  } else {
    int i = 1;
    while ( !b.isFreeRow(i) ) {
      i = i + 1; // ricerca di una riga libera
    }
    for ( int j=1; j<=n; j=j+1 ) {
      if ( ! b.underAttack(i,j) ) {
        b.addQueen( i, j );
        listOfCompletions( b );
        b.removeQueen( i, j );
      }
    }
  }
}
```

#### 4. Ricorsione e iterazione (4 Luglio 2016)

Dato un *albero di Huffman*, il metodo statico `shortestCodeLength` determina la lunghezza del più corto fra i codici di Huffman associati ai caratteri che compaiono in un documento di testo. Più specificamente, la visita dell'albero, finalizzata alla determinazione di tale lunghezza, è realizzata attraverso uno schema iterativo. Completa la definizione del metodo `shortestCodeLength` riportata nel riquadro.

```
public static int shortestCodeLength( Node root ) {
    int sc = ..... ;
    Stack<Node> stack = new Stack<Node>();
    Stack<Integer> depth = new Stack<Integer>();
    stack.push( root );
    depth.push( 0 );
    do {
        Node n = ..... ;
        int d = ..... ;
        if ( n.isLeaf() ) {
            sc = Math.min( sc, d );
        } else if ( d+1 < ..... ) {
            .....
            .....
            .....
        }
    } while ( ..... );
    return sc;
}
```

#### 5. Programmazione in Java (5 Settembre 2016)

Dato un *array* di numeri (`double`) con almeno due elementi, il metodo statico `closestPair` ne restituisce una coppia la cui differenza in valore assoluto è minima. La coppia è rappresentata da un array ordinato di due elementi. Esempio:

```
closestPair( new double[] {0.3, 0.1, 0.6, 0.8, 0.5, 1.1} ) → {0.5, 0.6}
```

Definisci in Java il metodo statico `closestPair`.