Corso di Programmazione

Esame del 25 Luglio 2019

cognome e nome

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Programmazione in Scheme

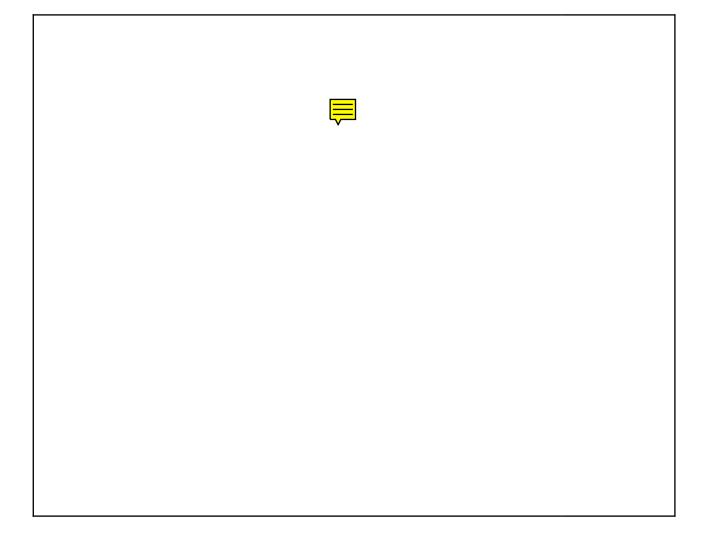
Il seguente programma, basato sulla procedura les-list, applica una variante dello schema ricorsivo discusso a lezione per calcolare la *lista* delle sottosequenze comuni più lunghe delle stringhe u e v. Per esempio:

```
(lcs-list "arto" "atrio") \rightarrow ("ato" "aro")
```

In particolare, il primo argomento di les-rec rappresenta un prefisso di qualche soluzione.

Si intende che la lista restituita da lcs-rec contenga stringhe *diverse fra loro* (senza ripetizioni) e, chiaramente, tutte della stessa lunghezza. Date due liste con queste caratteristiche, il compito della procedura best è quindi di selezionare tutte e sole le stringhe che faranno parte della soluzione per gli argomenti p, u, v.

Scrivi un programma in Scheme per realizzare la procedura best, tenendo conto del contesto in cui è applicata.



2. Dati procedurali in Scheme

La procedura paths restituisce la *lista* dei percorsi di Manhattan relativi a una mappa in cui gli spostamenti in basso sono rappresentati da caratteri consecutivi della stringa u e gli spostamenti a destra da caratteri di v. Per esempio:

```
(paths "12" "ab") \rightarrow ("12ab" "1a2b" "1ab2" "a12b" "a1b2" "ab12")
```

Più precisamente, il *k*-imo spostamento in basso è associato al *k*-imo carattere di u; l'*n*-imo spostamento a destra all'*n*-imo carattere di v. Nell'esempio, in particolare, gli spostamenti in basso sono associati a cifre e quelli a destra a lettere minuscole: la terza stringa della lista risultante indica quindi gli spostamenti in questo ordine: uno spostamento iniziale in basso (1), due spostamenti successivi a destra (ab) e uno spostamento finale in basso (2).

Completa la definizione della procedura paths riportata nel riquadro.

Programmazione in Java

Gli esercizi successivi fanno riferimento al seguente programma per calcolare il numero di percorsi di Manhattan lungo i collegamenti di un reticolo che si estende in tre dimensioni (problema affrontato anche in Laboratorio):

Una particolarità di questa soluzione è che il programma utilizza un oggetto "contatore" di tipo Counter, condiviso da tutte le invocazioni ricorsive di manhattan3DRec, per contare i cammini — intuitivamente, per contare quante volte si raggiunge il nodo destinazione seguendo tutti i possibili cammini diversi.

Il protocollo della classe Counter comprende (*):

un costruttore per creare un nuovo contatore inizializzato a zero; un metodo void incr() per incrementare di una unità il contatore; un metodo void add(long n) per aggiungere un valore intero non negativo n al contatore; un metodo void reset() per ri-azzerare il contatore; un metodo long count() per acquisire il valore del contatore (il conteggio); un metodo String toString() che restituisce la stringa che rappresenta il valore corrente del contatore.

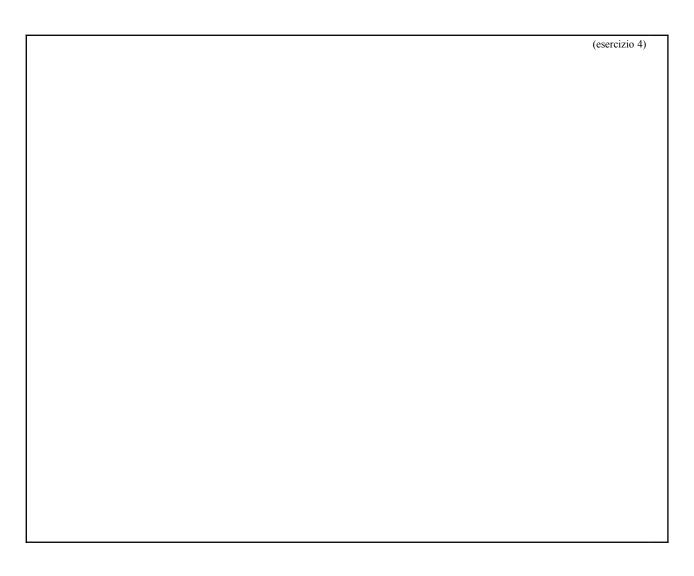
3. Memoization

Completa il programma riportato nel riquadro, che applica la tecnica top-down di memoization per realizzare una versione più efficiente di manhattan3D.

```
public static long manhattan3D( int i, int j, int k ) { \#i,j,k \ge 0
 Counter v = new Counter();
 h = new ;
 mem( i, j, k, v, h );
 return v.count();
private static void mem( int i, int j, int k, Counter v, _____ h ) {
 if ( == UNKNOWN ) {
  if ( (i == 0) && (j == 0) && (k == 0) ) {
   v.incr(); h _____ = 1;
  } else {
   long n = v.count();
   if ( i > 0 ) { mem( _______); }
   if ( k > 0 ) { mem( \ldots ); }
   h _____ = ____;
 } else {
  v.add( h _____);
private static final long UNKNOWN = -1;
```

4. Classi in Java

Definisci la classe Counter scegliendo una rappresentazione interna appropriata e realizzando il costruttore e i metodi del protocollo (*) in accordo a quanto specificato sopra.



5. Ricorsione e iterazione

Completa la definizione del metodo statico manhattan3D, riportato nel riquadro sottostante, che trasforma la struttura ricorsiva di manhattan3DRec in una struttura iterativa basata su uno *stack*.