

Risolvi i seguenti esercizi giustificando sinteticamente le risposte.

1. Programmazione e argomenti procedurali in Scheme

Data una lista s di interi positivi e diversi fra loro, la procedura `lis-list` restituisce la *lista* delle sottosequenze crescenti più lunghe di s . Per esempio:

`(lis-list '(27 7 18 49 8 15 53 1 28 6))` \rightarrow `((7 8 15 28) (7 8 15 53) (7 18 49 53))`

Completa il programma riportato nel riquadro sottostante, che si basa sull'idea sviluppata nell'esercizio di laboratorio in relazione al problema in esame.

```
(define lis-list ; s: lista non vuota di interi positivi e diversi fra loro
  (lambda (s)
    (lis-rec s 0)
  ))

(define lis-rec
  (lambda (s t)
    (cond ((null? s)
            ..... )
          ((< (car s) t)
            ..... )
          (else
            (better (lis-rec (cdr s) t)
                     (map .....
                           (lis-rec (cdr s) (car s))
                           )))
          )))

(define better
  (lambda (u v)
    (cond ( .....
            v)
          ( .....
            u)
          (else
            ..... )
          )))
```

2. Verifica formale della correttezza

```
(define square      ; val: intero
  (lambda (n)       ; n ≥ 0 intero
    (sqr-rec n 1 0)
  ))

(define sqr-rec
  (lambda (x y z)
    (if (> x 0)
        (sqr-rec (- x 1) (+ y 2) (+ z y))
        z
    )))
```

In relazione alla procedura `sqr-rec`, riportata qui sopra a destra, si può dimostrare che:

$$\forall i, j, k \in \mathbf{N} . (\text{sqr-rec } i \ j \ k) \rightarrow k + i \cdot (i + j - 1)$$

Dimostra questa proprietà per induzione sul valore di i attenendoti allo schema impostato qui sotto.

- Formalizza la proprietà che esprime il caso / i casi base:
- Formalizza l'ipotesi induttiva:
- Formalizza la proprietà da dimostrare come passo induttivo:
- Dimostra il caso / i casi base:
- Dimostra il passo induttivo:
- Infine, in relazione alla procedura `square` riportata sopra a sinistra, dimostra che $(\text{square } n) \rightarrow n^2$:

3. Programmazione in Java

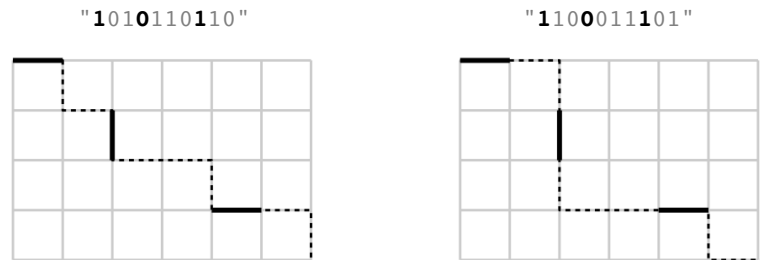
Gli argomenti del metodo statico `commonStretches` sono due stringhe `u`, `v` composte solo dai caratteri `'0'` e `'1'`, entrambe con esattamente m occorrenze di `'0'` ed n occorrenze di `'1'`. Tali stringhe rappresentano due *percorsi di Manhattan* in un reticolo di $m \times n$ tratti di strada, dove `'0'` denota uno spostamento verticale e `'1'` orizzontale. Date le stringhe `u`, `v`, il metodo `commonStretches` calcola il numero di tratti di strada comuni ai due percorsi, numero che può variare da 0 a $m+n$, nel secondo caso quando i due percorsi (e quindi le due stringhe) coincidono. Per esempio:

```
commonStretches( "1010110110", "1100011101" )    →    3    //  m = 4, n = 6
```

Affinché un certo tratto sia comune a due percorsi, questi devono passare per lo stesso nodo (incrocio) e lo spostamento successivo deve chiaramente procedere nella stessa direzione (orizzontale o verticale). I due percorsi passano per lo stesso nodo se una parte iniziale (prefisso) delle stringhe che li rappresentano è composta dallo stesso numero di `'0'` e dallo stesso numero di `'1'`; inoltre, lo spostamento che segue va nella stessa direzione se il carattere successivo delle due stringhe coincide.

Nell'esempio questa situazione si verifica in corrispondenza ai caratteri in posizione 0 , 3 e 7 come illustrato dalla figura qui a lato.

Definisci in Java il metodo statico `commonStretches` in accordo con le specifiche indicate.



4. Programmi ricorsivi in Java

```
public static long q( int i, int j, int k ) { //i,j,k >= 0
    long x = ( i < 2 ) ? i : q( i-2, j, k );
    long y = ( j < 2 ) ? j : q( i, j-2, k );
    long z = ( k < 2 ) ? k : q( i, j, k-2 );
    long m = x + y + z;
    return ( m == 0 ) ? 1 : m;
}
```

In relazione al metodo statico riportato sopra, determina il risultato restituito dalle seguenti invocazioni:

q(0,0,0) →	q(3,2,1) →
q(1,1,1) →	q(2,1,5) →
q(1,3,1) →	q(3,3,3) →

5. Memoization

Applica la tecnica *top-down* di *memoization* per realizzare una versione più efficiente del metodo `q` dell'esercizio 4.