Laboratorio di Programmazione a.a. 2019-20

Problema 8

25 / 28 Novembre 2019

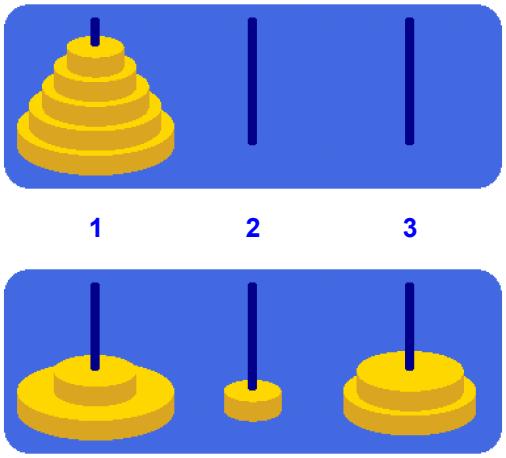
Descrizione

Il celebre rompicapo della *Torre di Hanoi* è costituito da tre asticelle (1, 2 e 3), in una delle quali (1) sono inizialmente inanellati n dischi di diametro diverso e decrescente dal basso verso l'alto. Per risolvere il rompicapo si deve trasferire la torre di n dischi in corrispondenza a una diversa asticella (2), spostando un solo disco per volta da un'asticella ad un'altra e rispettando il vincolo che un disco non può mai essere appoggiato sopra un altro disco di diametro inferiore.

Il rompicapo ammette un'elegante soluzione ricorsiva: per trasferire una torre di n dischi dall'asticella s (sorgente) all'asticella d (destinazione) è necessario innanzitutto trasferire una torre di $n–1$ dischi—tutti tranne quello alla base—dall'asticella s all'asticella t (transito), rispettando le regole del gioco; quindi si può spostare la base—il disco di diametro maggiore fra gli n considerati—da s a d; infine ci si cimenta di nuovo con il sottoproblema di trasferire una torre di $n–1$ dischi, questa volta da t a d, per riportarla sopra la base. Il procedimento che ne risulta realizza la strategia ottimale, che risolve il rompicapo con il minor numero di mosse, pari a $2^n–1$ per una torre di n dischi.

Dato un intero positivo n, la procedura `hanoi-moves` restituisce la sequenza (lista) di mosse che risolve il rompicapo della *Torre di Hanoi* per n dischi, dove ciascuna mossa è descritta dalla coppia (lista di due elementi) di asticelle in cui il disco spostato si trova immediatamente prima e immediatamente dopo quella mossa:

```
(define hanoi-moves      ; val: lista di coppie
  (lambda (n)            ; n > 0 intero
    (hanoi-rec n 1 2 3)
    ))

(define hanoi-rec    ; val: lista di coppie
  (lambda (n s d t)  ; n intero, s, d, t: posizioni
    (if (= n 1)
        (list (list s d))
        (let ((m1 (hanoi-rec (- n 1) s t d))
              (m2 (hanoi-rec (- n 1) t d s))
              )
          (append m1 (cons (list s d) m2))
          ))
    ))
```



Nel caso di una torre iniziale di altezza $n = 3$, per esempio, il rompicapo viene risolto in 7 mosse, rappresentate dalla seguente lista di coppie che risulta dall'applicazione della procedura `hanoi-moves`:

```
(hanoi-moves 3)  ⟶  '((1 2) (1 3) (2 3) (1 2) (3 1) (3 2) (1 2))
```

Il matematico francese Édouard Lucas inventò questo gioco nel 1883 e lo commercializzò accompagnandolo con una leggenda che narra di un monaco Indù alle prese con il rompicapo per una torre di 64 dischi d'oro e di una profezia che nel momento in cui avrà completato tutte le mosse il mondo finirà. In effetti il numero minimo di mosse per spostare una torre di 64 dischi rispettando le regole stabilite è enorme, al punto che non basterebbe il tempo di vita dell'universo (secondo le stime attuali) per portare a termine il compito.

La procedura `hanoi-rec`, che applica una *ricosione ad albero*, è un modo elegante di rappresentare la soluzione del rompicapo. Tuttavia, per sapere quale sarà la disposizione dei dischi dopo la k-ima mossa (dove $k \le 2^n–1$) è sufficiente una *ricorsione di coda*! A tal fine si può infatti osservare che, delle $2^n–1$ mosse necessarie a trasferire una torre di n dischi, quella esattamente a metà, cioè la 2^{n-1}-ima, permette di spostare il disco più grande. A seconda che $k < 2^{n-1}$ oppure $k \ge 2^{n-1}$, si può quindi stabilire dove è collocato l'n-imo disco, cioè il disco di diametro maggiore fra quelli considerati, e di conseguenza ci si può concentrare su uno solo dei due sottoproblemi che hanno come obiettivo trasferire una torre di $n–1$ dischi, dove cambiano solo le asticciole sorgente e destinazione. Questo schema ricorsivo determina di volta in volta la collocazione di un nuovo disco, procedendo dal più grande al più piccolo.

Cerca di realizzare i programmi richiesti nei punti (i) e (ii) seguenti *senza* usare la ricorsione ad albero.

(i) Definisci una procedura `hanoi-disks` che, dati due interi n, k, con $n > 0$ e $0 \le k \le 2^n–1$, restituisce la configurazione al termine della k-ima mossa, rappresentata dal numero di dischi per ciascuna asticella. Per $k = 0$ la procedura restituisce la configurazione iniziale in cui tutti i dischi si trovano in corrispondenza all'asticella 1.

Una configurazione può essere descritta in questi termini da una terna di coppie (lista di tre liste), relative alle tre asticciole: ogni coppia associa la posizione dell'asticella (1, 2 o 3) al corrispondente numero di dischi (un intero compreso fra *0* ed *n*). A tal fine conviene utilizzare come parametri della procedura ricorsiva, oltre alle posizioni *s*, *d* e *t* delle asticelle, i (tre) numeri dei dischi di cui si conosce la collocazione in corrispondenza alle rispettive asticelle. Si suggerisce inoltre di non preoccuparsi che l'ordine delle tre liste nella terna sia predeterminato, in quanto comunque ciascuna asticciola è identificata dal primo elemento di una coppia. Per esempio, la configurazione illustrata dalla figura in basso nella pagina precedente può essere descritta dalla terna `'((3 2) (2 1) (1 2))` oppure, indifferentemente, da `'((1 2) (2 1) (3 2))`, `'((2 1) (1 2) (3 2))`, ecc.

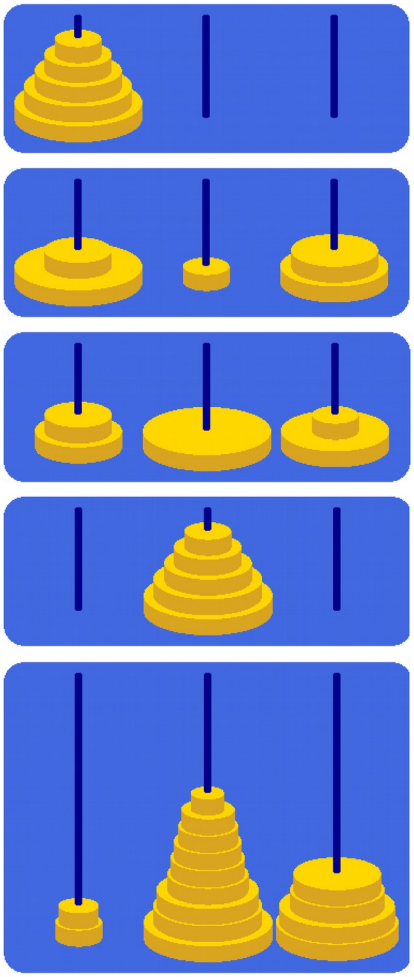
Esempi

```
(hanoi-disks 3 0)  ⟶ '((1 3) (3 0) (2 0))
(hanoi-disks 3 1)  ⟶ '((3 0) (2 1) (1 2))
(hanoi-disks 3 2)  ⟶ '((2 1) (1 1) (3 1))
(hanoi-disks 3 3)  ⟶ '((1 1) (3 2) (2 0))
(hanoi-disks 3 4)  ⟶ '((3 2) (2 1) (1 0))
(hanoi-disks 3 5)  ⟶ '((2 1) (1 1) (3 1))
(hanoi-disks 3 6)  ⟶ '((1 1) (3 0) (2 2))
(hanoi-disks 3 7)  ⟶ '((3 0) (2 3) (1 0))

(hanoi-disks 5 13)  ⟶ '((3 2) (2 1) (1 2))

(hanoi-disks 15 19705)
                    ⟶  '((3 4) (2 9) (1 2))

(hanoi-disks 15 32767)
                    ⟶ '((3 0) (2 15) (1 0))
```

(ii) Definisci una procedura `hanoi-picture` che, dati due interi *n*, *k*, con $n > 0$ e $0 \le k \le 2^n-1$, restituisce un'immagine della disposizione dei dischi al termine della *k*-ima mossa.

Esempi

```
(hanoi-picture 5  0)  ⟶
(hanoi-picture 5 13)  ⟶
(hanoi-picture 5 22)  ⟶
(hanoi-picture 5 31)  ⟶
(hanoi-picture 15 19705) ⟶
```



Per perseguire l'obiettivo (ii) sulla base della struttura del programma per `hanoi-disks` è utile introdurre un ulteriore parametro della procedura ricorsiva: l'immagine relativa ai dischi di cui si conosce la collocazione. (Probabilmente è opportuno prevedere altri parametri aggiuntivi per rendere disponibili tutte le informazioni necessarie alla costruzione delle immagini attraverso le procedure indicate qui di seguito: tali integrazioni sono lasciate alla tua cura.)

Il *teachpack* `hanoi.ss` è stato definito al fine di facilitare la costruzione di immagini come quelle mostrate negli esempi. Aggiungendo all'ambiente di programmazione tale *teachpack*[1] hai a disposizione tre procedure specializzate:

- `(towers-background` *n*`)` restituisce l'immagine del fondo e delle tre asticelle, dimensionando questi elementi in modo appropriato per una torre di *n* dischi.
- `(disk-image` *d n p t*`)` restituisce l'immagine del *d*-imo disco fra gli *n* utilizzati ($1 \le d \le n$; la dimensione dei dischi cresce con *d*) quando questo è collocato in corrispondenza all'asticella di posizione *p* ($1 \le p \le 3$) all'altezza in cui si troverebbe se posto sopra altri *t* dischi.
- `(above` <*immagine*$_1$> <*immagine*$_2$>`)` immagine che si ottiene sovrapponendo <*immagine*$_1$> a <*immagine*$_2$>, prestando attenzione al fatto che l'ordine dei due argomenti non è indifferente, in quanto la prima immagine si sovrappone alla seconda nascondendone alcune parti.

Per capire meglio il ruolo di ciascuna delle procedure introdotte può essere di aiuto sperimentare la valutazione di qualche semplice espressione come le seguenti:

```
(towers-background 2)      →  ?

(towers-background 5)      →  ?

(disk-image 5  5  2  2)    →  ?

(above (disk-image 5  5  2  2) (towers-background 5))   →  ?

(above (disk-image 1  5  2  2) (towers-background 5))   →  ?

(above (disk-image 3  5  2  2)
       (above (disk-image 1  5  2  0) (towers-background 5)))   →  ?

(above (disk-image 3  5  3  1)
       (above (disk-image 4  5  3  0) (towers-background 5)))   →  ?
```

1 Il *teachpack* deve essere scaricato e salvato in una cartella personale, quindi integrato nell'ambiente DrRacket selezionando la voce "**Add Teachpack**" del Menù "**Language**". Benché si tratti di un programma, il file **non** deve essere aperto o copiato nella finestra delle definizioni di DrRacket perché l'editor vi scriverebbe alcune informazioni nascoste compromettendone l'integrità.