Fondamenti di Informatica A.A. 2017-2018

Esercizi proposti

Ovidiu Daniel Barba
ovi.daniel.b@gmail.com

Laura Trivelloni
laura.trivelloni@gmail.com

Emanuele Vannacci
emanuele.vannacci@gmail.com

Giugno 2018

1 Esercizi

1. Data una lista `L` di interi con possibili liste annidate al suo interno, scrivere una funzione **ricorsiva** in Python per ognuno dei seguenti punti:

   (a) determinare la somma di tutti i valori contenuti in `L` (anche quelli nelle liste annidate)

   (b) determinare il massimo valore contenuto in `L` (considerando anche i valori nelle liste annidate)

   Per esempio, data la lista:

   ```
   L = [1, 5, [2, [1, 1], 4] ]
   ```

   la funzione `(a)` deve ritornare `14`, mentre `(b)` `5`.

   Suggerimento: usare la funzione `type(obj)` che ritorna il tipo di dato di `obj`.

2. Scrivere una funzione **ricorsiva** in Python che, dati due interi positivi `a` e `b`, calcola a^b.

3. Scrivere una funzione **ricorsiva** in Python che, data una string `S`, la inverte.

4. Scrivere una funzione **ricorsiva** in Python che, data un intero positivo `n`, ritorna i coefficienti della riga `n`-esima del triangolo di Tartaglia. Ad esempio, per $n = 4$, ritorna $[1, 3, 3, 1]$.

5. Definire in Python una classe `Iterator` che prende come parametro del costruttore una lista qualsiasi. L'`Iterator` permette di attraversare la lista in entrambe le direzioni, aggiungere e rimuovere un elemento subito dopo la posizione in cui si trova lungo la lista e inoltre tiene traccia della posizione attuale nella lista. I metodi devono essere:

   (a) **hasNext** : ritorna vero se esiste un elemento dopo la posizione attuale, falso altrimenti

(b) **hasPrevious**: ritorna vero se esiste un elemento prima della posizione attuale, falso altrimenti
(c) **next**: ritorna l'elemento successivo alla posizione attuale
(d) **previous**: ritorna l'elemento precedente alla posizione attuale
(e) **add**: aggiunge un elemento subito dopo la posizione attuale
(f) **remove**: rimuove l'elemento subito dopo la posizione attuale e lo ritorna

6. Scrivere una funzione Python che, data un'immagine `P` e un float `avg`, si scorre la lista (usando l'`Iterator` definito nell'esercizio [5]) dei pixel di `P` e ritorna vero se almeno un pixel ha la media delle componenti del proprio colore strettamente maggiore di `avg` e falso altrimenti.
Suggerimento: la funzione `getPixels(pict)` ritorna una lista di pixel che puó essere passata all'`Iterator`.

7. Definire in Python una classe `Knapsack` (zaino) che prende come parametro del costruttore un numero intero positivo `C` che indica la capacitá massima dello zaino. La classe ha come attributo una lista di `Item` (a sua volta una classe con un nome e un numero intero positivo che indica l'occupazione nello zaino) e permette l'inserimento di un `Item` solo se non viene ecceduta la sua capacitá massima aggiungendo l'occupazione dell'`Item`. Inoltre il `Knapsack` deve tenere traccia e ritornare su richiesta lo spazio rimanente, il numero di elementi al suo interno, l'elemento con occupazione massima e quello con occupazione minima.

8. Presi in considerazione il `Knapsack` e i relativi `Item` definiti nell'esercizio precedente, una matrice (rappresentata come lista di liste) `A` di dimensioni `NxM` che ha come elementi istanze di `Item` e un numero intero positivo `T`, scrivere una funzione in Python che per ogni colonna di `A` crea un `Knapsack` di capacitá `T` e ritorna vero se, per ogni colonna di `A`, **tutti** gli item della relativa colonna possono essere inseriti nell'i-esimo `Knapsack`, altrimenti falso.

9. Data una lista l di interi, scrivere una funzione **ricorsiva** in Python che ordini gli elementi di l in modo crescente.
Suggerimento: Se la lista contiene n elementi, posizionare al primo posto l'elemento massimo e ordinare i restanti n-1 elementi.