Università di Roma Tor Vergata

Tutoraggio
Liste, code, pile

Laura Trivelloni
23 Ottobre 2017

Ingegneria degli algoritmi
A.A. 2017/2018

Dalla scorsa lezione...

Implementazione in Python:

Liste

- **L.append(<elem>)**
- **L.extend(<lista>)**
- **L.insert(<indice>,<elem>)**
- **L.pop()**
- **L.pop(<indice>)**
- **L.sort()**
- **L.index(<index>)**

Dizionari

- **<dict>.get(<key>,<default>)**
- **<dict>.pop(<key>,<default>)**
- **<dict>.items()**
- **<dict>.values()**
- **<dict>.keys()**
- **<dict>.clear()**
- **len()**

Dalla scorsa lezione...

Implementazione in Python:

Liste

- **L.append(<elem>)**
- **L.extend(<lista>)**
- **L.insert(<indice>,<elem>)**
- **L.pop()**
- **L.pop(<indice>)**
- **L.sort()**
- **L.index(<index>)**

Dizionari

- **<dict>.get(<key>,<default>)**
- **<dict>.pop(<key>,<default>)**
- **<dict>.items()**
- **<dict>.values()**
- **<dict>.keys()**
- **<dict>.clear()**
- **len()**

Quali sono le differenze?

Strutture dati

Strutture dati

Lista

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre

Dizionario

- oggetto **mutabile**
- **<chiave>:<valore>**
- accesso ai dati con la **chiave** (unica, immutabile)
- collezione non ordinata (**mappa**)
- parentesi graffe

Strutture dati

Lista

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre

Dizionario

- oggetto **mutabile**
- **<chiave>:<valore>**
- accesso ai dati con la **chiave** (unica, immutabile)
- collezione non ordinata (**mappa**)
- parentesi graffe

E cos'è una **tupla**?

Strutture dati

Lista

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre

Tupla

- **immutabile**
  - modifica → **conversione** in lista
  - es. indici di dizionari
- accesso ai dati tramite **posizione**
- collezione ordinata (**sequenza**)
- parentesi tonde

Strutture dati

Lista

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre

Tupla

- **immutabile**
  - modifica → **conversione** in lista
  - es. indici di dizionari
- accesso ai dati tramite **posizione**
- collezione ordinata (**sequenza**)
- parentesi tonde

E cos'è un **array**?

Strutture dati

Lista

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre
- **dynamic space allocation**
  - allocazione **sparsa**
  - se nodo eliminato → deallocato
  - da mantenere riferimento a nodo precedente e successivo

Array

- oggetto **mutabile**
- elementi di qualsiasi tipo
- accesso ai dati tramite **posizione** (intero)
- collezione ordinata (**sequenza**)
- parentesi quadre
- **static space allocation**
  - allocazione **contigua**
  - se elemento eliminato → non deallocato
  - sfruttamento peggiore della memoria

Strutture dati

Ora che abbiamo le idee un po’ più chiare...

Strutture dati

Ora che abbiamo le idee un po’ più chiare...

Cos’è una **coda**?

Strutture dati

Ora che abbiamo le idee un po’ più chiare...

Cos’è una **coda**?

Strutture dati

Ora che abbiamo le idee un po’ più chiare...

Cos’è una **coda**?

Cos’è una **pila**?

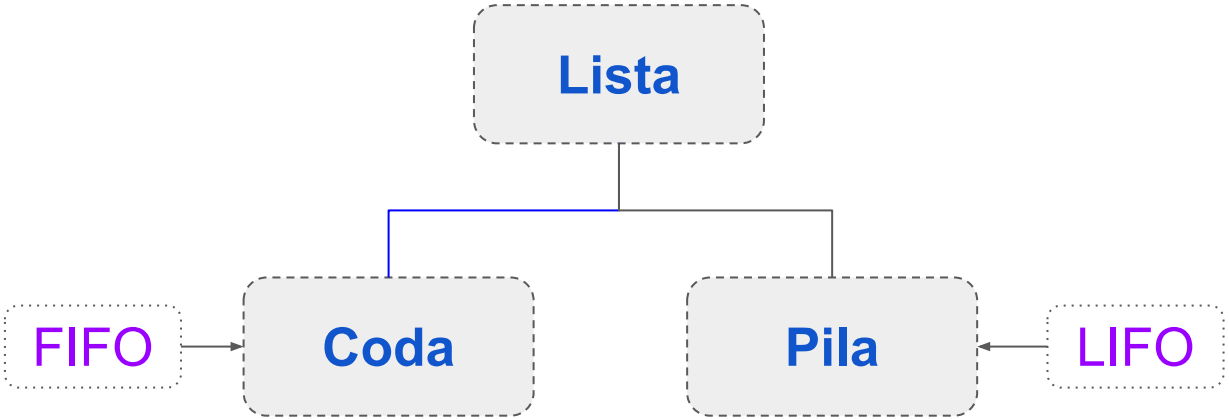
Strutture dati

Ora che abbiamo le idee un po’ più chiare...

Cos’è una **coda**?

Cos’è una **pila**?

Strutture dati



Implementazione

- Lista
  - Liste **collegate**
  - Liste **doppiamente collegate**
- Coda
  - Usando liste **collegate**
  - Usando l'implementazione delle **liste** di Python
  - Usando il tipo **deque** presente nel modulo **collections**
- Pila (o stack)
  - Usando le liste **collegate**
  - Usando l'implementazione delle **liste** di Python

Esercizio 1.1

Implementare nel modulo **generatoreCasuale.py** un algoritmo in grado di creare in output un file di testo contenente *N* numeri casuali compresi tra *min* e *max* ognuno diviso da uno spazio dal successivo).

NB:

- Il nome del file di output, *N*, *min* e *max* devono essere parametri che l'utente deve poter scegliere in modo interattivo
- Per generare i numeri casuali usate la funzione **randint(min, max)** appartenente al modulo **random**

Esercizio 1.2

Usando il file ottenuto dall'esecuzione di **generatoreCasuale.py**, considerare il problema di stampare la sequenza di valori interi letti da tale file in modo che tutti i *dispari precedano i pari*, e soffermarsi sulle tre varianti:

1. Sia i valori pari che dispari devono essere stampati nello *stesso ordine* in cui compaiono nella sequenza di input;
2. I valori pari devono essere stampati in *ordine opposto* a quello della sequenza di input, e i dispari nello *stesso ordine*;
3. Sia i valori pari che i dispari devono essere stampati in *ordine opposto* a quello della sequenza di input;

Utilizzare come strutture dati di appoggio **pile** e **code**, e discutere quale scelta è più opportuna in ciascun caso.

Esercizio 2

Provate a creare un algoritmo in grado di **ordinare una lista** contenente elementi interi.
Testatene la correttezza.

Esercizio 2

Provate a creare un algoritmo in grado di **ordinare una lista** contenente elementi interi.
Testatene la correttezza.

Buon lavoro! ☺