Progetto “Unrolled Linked List”

Appello del 29 Giugno 2020

Descrizione del progetto

Lo scopo del progetto è la realizzazione di una lista concatenata *singola* “srotolata”. Ovvero, una lista concatenata in cui i nodi possono contenere un numero m di valori invece di un solo valore (il valore m è uguale per tutti i nodi). Un esempio di questa variante di lista concatenata è presentato in Figura 1.

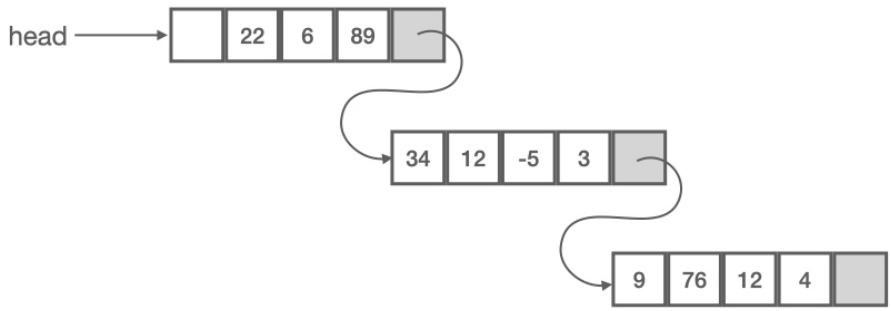

Figura 1: Rappresentazione di una lista concatenata singola “srotolata” in cui ogni nodo contiene $m = 4$ elementi.

Si richiede che questa struttura dati supporti le seguenti operazioni:

- Inserimento in testa. Inserisce una chiave in testa alla lista.
- Rimozione in testa. Rimuove l’elemento in testa alla lista ritornandolo. Se la lista è vuota viene ritornato `None`.
- Ricerca di un elemento. Metodo che ritorna `True` se l’elemento cercato è nella lista e `False` altrimenti.
- Accesso ad un elemento dato l’indice. Da un indice i, viene ritornato (senza rimuoverlo) l’elemento contenuto in posizione i nella lista, con gli indici che partono da zero. Viene ritornato `None` se l’indice è minore di zero o maggiore o uguale della lunghezza della lista. Per esempio, gli elementi di indice 0, 1, 2, 3 e 4 nella lista di Figura 1 sono, rispettivamente, 22, 6, 89, 34 e 12. In particolare, l’ultimo elemento inserito (e non ancora rimosso) nella lista avrà indice 0.
- Calcolo del numero di elementi contenuti nella lista. Ritorna il numero di elementi contenuti nella lista. Questo non conta le posizioni vuote che possono esserci all’interno del primo nodo. Per esempio, la lista di Figura 1 ha lunghezza 11 (non 12).

Si impongono inoltre i seguenti vincoli:

- L’inserimento e la rimozione in testa devono avere costo $O(1)$.

- verificare se un elemento è contenuto, l'accesso ad un indice ed il calcolo del numero di elementi contenuti nella lista deve avere costo $O(n)$, dove n è il numero di elementi contenuti nella lista.
- Tutti i nodi della lista ad eccezione del primo devono contenere esattamente m chiavi

Si presti particolare attenzione all'ultimo vincolo. Nell'esempio di Figura 1, l'inserimento di un altro valore, per esempio -4, non comporta la creazione di un nuovo nodo (dato che c'è ancora una posizione libera nel primo nodo della lista prima del 22). Dopo l'inserimento di -4, il precedente elemento di indice 0, il 22, avrà ora indice 1.

Codice

Viene fornito uno scheletro del codice che deve essere implementato, con i metodi che devono essere scritti:

```
class UnrolledLinkedList:

    def __init__(self, m):
        # Implementazione del costruttore. Tutti i nodi
        # dovranno essere in grado di contenere m chiavi

    def inserisci_testa(self, chiave):
        # inserimento di una chiave in testa alla lista

    def rimuovi_testa(self):
        # Rimozione della chiave in testa alla lista. Ritorna
        # la chiave rimossa e None se la lista è vuota

    def contiene(self, chiave):
        # Ritorna true se la chiave fornita come argomento è
        # contenuta all'interno della lista e false altrimenti

    def accesso_indice(self, indice):
        # Ritorna l'elemento della lista che si trova all'indice
        # dato come argomento. Il primo elemento ha indice 0.
        # Nel caso non sia contenuto nessun elemento in
        # quell'indice il metodo ritorna None

    def lunghezza(self):
        # ritorna il numero di elementi contenuti nella lista
```

Nel progetto è consentito avere funzioni aggiuntive che testano il buon funzionamento della lista concatenata. È inoltre consentito (e consigliato) avere una classe che rappresenti i singoli nodi della lista.

Si ricorda di commentare adeguatamente il codice, approfittandone per spiegare le scelte implementative effettuate.

Esempi d'uso

Il seguente frammento di codice chiama tutti i metodi la cui implementazione è richiesta dal progetto. Come commento è indicato il valore atteso in una implementazione funzionante:

```
xs = UnrolledLinkedList(3)
xs.inserisci_testa(4)
xs.inserisci_testa(7)
xs.inserisci_testa(8)
xs.inserisci_testa(2)
xs.inserisci_testa(9)
print(xs.contiene(4)) # True
print(xs.contiene(2)) # True
print(xs.contiene(12)) # False
for i in range(0, xs.lunghezza()):
    print(xs.accesso_indice(i)) # 9 2 8 7 4
print(xs.lunghezza()) # 5
print(xs.rimuovi_testa()) # 9
print(xs.rimuovi_testa()) # 2
print(xs.rimuovi_testa()) # 8
print(xs.lunghezza()) # 2
xs.inserisci_testa(3)
for i in range(0, xs.lunghezza()):
    print(xs.accesso_indice(i)) # 3 7 4
```

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno 19/06/2020 secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del 29/06/2020*.
- L'email deve avere come allegato il progetto in un singolo file in codice sorgente Python versione 3, dal nome `Nome_Cognome_matricola.py`, quindi, per esempio Mario Rossi di matricola 12345 consegnerà un file dal nome `Mario_Rossi_12345.py`.
- Il file deve contenere sotto forma di commento le seguenti linee indicanti nome, cognome e numero di matricola:

  ```
  # Nome: Mario
  # Cognome: Rossi
  # Matricola: 12345
  ```

Note aggiuntive

Ci si potrebbe chiedere perché potrebbe essere utile utilizzare questa variante di liste concatenate rispetto alle più semplici liste tradizionali. Principalmente per due motivi: si occupa lo spazio necessario per un reference solo ogni m chiavi invece che per ogni chiave e, più rilevante, nei computer moderni strutture che accedono a blocchi contigui di memoria sono più veloci (questo diminuisce la costante, non cambia il comportamento asintotico).