Algoritmi i strukture podataka

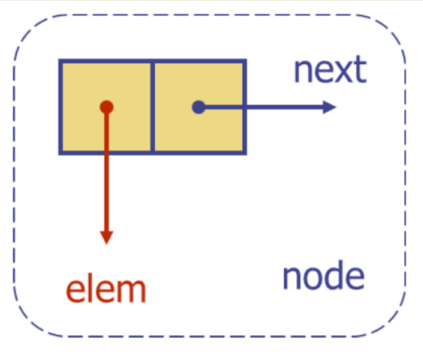
04 Lista

Katedra za informatiku, Fakultet tehničkih nauka, Novi Sad

2021

Lista

- Sekvenca elemenata
- Elementi sadržani u čvorovima liste
- Element sadrži vezu ka susednom elementu
- Čvorovi se ne raspoređuju na uzastopne memorijske lokacije



Python __Underscore__ metode

- Pored `__init__` i `__len__`, na ovom terminu će biti potrebne i sledeće metode:
- `__iter__(self)`
  - Koristi se za implementaciju iterator protokola odnosno za omogućavanje pristupa elementima kolekcije
  - Metoda vraća iterator objekat
  - ```
    for item in list:
        …
    ```
- `__eq__(self, other)`
  - Koristi se za poređenje dva objekta iste klase
  - `node1 == node2`
    ```

Python __Underscore__ metode

- `__delitem__(self, index)`
  - Koristi se za brisanje elemenata liste po indeksu
  - `del my_list[5]`
- `__setitem__(self, index, value)`
  - Koristi se za izmenu vrednosti elementa liste na zadatom indeksu *index*
  - `my_list[5] = "abc"`
- `__getitem__(self, index)`
  - Koristi se za pristup elementu po indeksu
  - `my_list[5]`

Jednostruko spregnuta lista

- (Singly) Linked List
- Osnovne operacije nad listom L:
  - `L.add_first(e)` – Element *e* se dodaje na početak liste
  - `L.add_last(e)` – Element *e* se dodaje na kraj liste
  - `L.remove_first(e)` – Uklanja prvi element. Izuzetak ukoliko ne postoji.
  - `L.remove_last(e)` – Uklanja poslednnji element. Izuzetak ukoliko ne postoji.
  - `L.get_first()` – Dobavlja prvi element liste. Izuzetak ukoliko ne postoji.
  - `L.get_last()` – Dobavlja poslednji element liste. Izuzetak ukoliko ne postoji.
  - `L[index] = e` – Izmena elementa po indeksu. Izuzetak ukoliko *index* ne postoji.
  - `L.is_empty()` – Proverava da li je lista prazna (rezultat tipa boolean)
  - `len(L)` – Pronalazi i vraća broj elemenata liste *L*

Jednostruko spregnuta lista

- (Singly) Linked List
- Dodatne operacije koje omogućuju istovetne sintaksne odlike kao kod ugrađene python liste.
- Operacije nad listom L:
  - `L.append(e)` – Element *e* se dodaje na kraj liste
  - `L.remove(e)` – Pronalazi se element *e* i on se uklanja. Izuzetak ukoliko ne postoji.
  - `L.insert(index,e)` – Element *e* se umeće na poziciju *index* liste *L.* Izuzetak ukoliko *index* ne postoji.
  - `del L[index]` – Uklanja element na poziciji *index.* Izuzetak ukoliko *index* ne postoji.
  - `L[index]` – Pristup elementu po indeksu. Izuzetak ukoliko *index* ne postoji.
  - `L[index] = e` – Izmena elementa po indeksu. Izuzetak ukoliko *index* ne postoji.
  - `L.extend(L2)` – Lista *L* se proširuje elementima liste *L2*

Dvostruko spregnuta lista

- Doubly Linked List
- Operacije:
  - ... sve definisane za jednostruko spregnutu listu i još:
  - `L.insert_after(existing_element, new_element)`
  - `L.insert_before(existing_element, new_element)`

Zadatak 1

- Implementirati klasu **Node** koja odgovara jednom elementu liste.

Zadatak 2

- Implementirati klasu **LinkedList**.

Zadatak 3

- Implementirati klasu **DoublyLinkedList.**