

# Liste

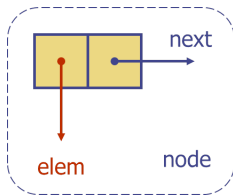
© Goodrich, Tamassia, Goldwasser

Katedra za informatiku, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

2023.

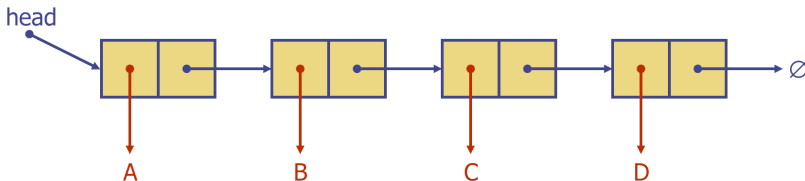
# Jednostruko spregnuta lista

- predstavlja sekvencu elemenata
- elementi su sadržani u „čvorovima“ liste (nodes)
- susedstvo između elemenata se opisuje vezama/referencama/pokazivačima
- svaki čvor sadrži
  - podatak koji se čuva
  - link prema sledećem čvoru



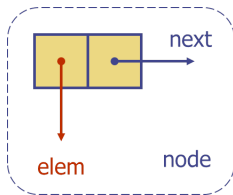
# Jednostruko spregnuta lista

- čvorovi ne zauzimaju susedne memorijske lokacije – mogu biti „razbacani“ po memoriji
- redosled se održava pomoću veza između čvorova
- svaki čvor ima vezu prema sledećem
- koji je prvi?
  - potrebna nam je posebna referenca na prvi element liste („glava“)
- na koga pokazuje poslednji element?
  - njegova referenca na sledećeg je None



# Element jednostruko spregnute liste u Pythonu

```
class Node:  
    def __init__(self, element, next):  
        self._element = element  
        self._next = next
```



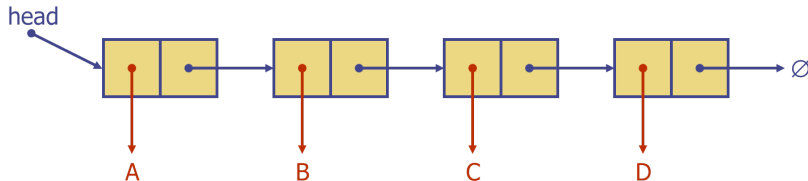
# Iterator: obilazak svih elemenata liste

$current \leftarrow head$

**while**  $current$  is not None **do**

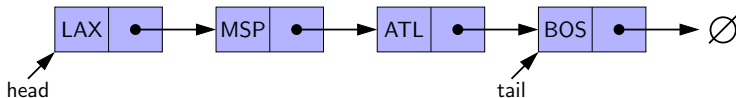
    obradi  $current$

$current \leftarrow current._next$



# Poslednji element liste

- kako doći do **poslednjeg** elementa liste?
  - krenemo od glave dok ne dođemo do elementa čiji `_next` je `None`
  - ovaj postupak je  $O(n)$
- bilo bi zgodno čuvati referencu na poslednji element liste
  - analogno glavi, referenca se zove „rep“ (tail)

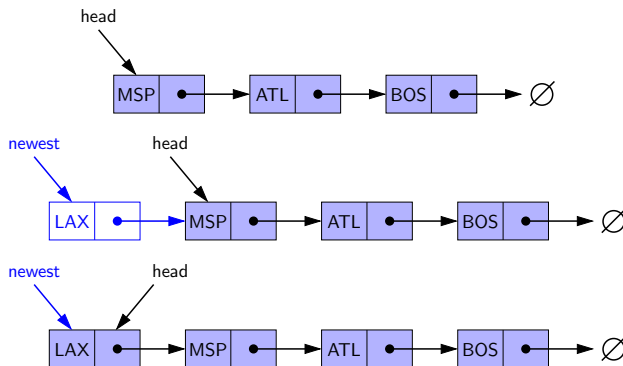


# Granični slučajevi

- kako predstaviti praznu listu?
  - `head = tail = None`
- kako predstaviti punu listu?
  - lista nema ograničenje na maksimalan broj elemenata :)
- ako lista ima jedan element?
  - `head == tail`

# Dodavanje elementa na početak liste

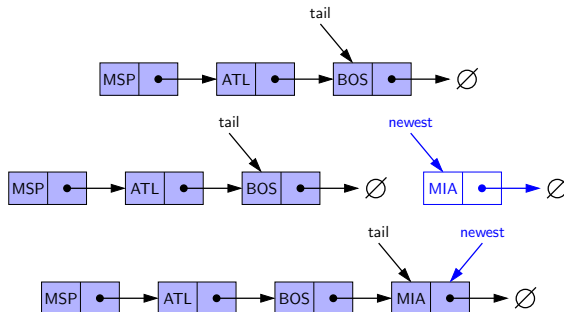
- 1 kreiraj novi čvor
- 2 upiši podatak u čvor
- 3 link na sledeći novog čvora pokazuje na glavu
- 4 glava pokazuje na novi čvor





# Dodavanje elementa na kraj liste

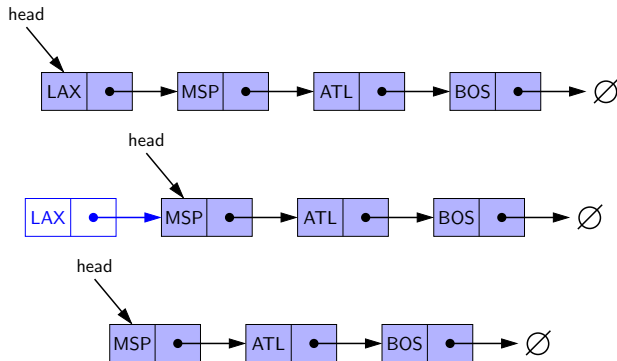
- 1 kreiraj novi čvor
- 2 upiši podatak u čvor
- 3 link na sledeći novog čvora je None
- 4 poslednji→sledeći pokazuje na novi čvor
- 5 tail pokazuje na novi čvor



# Uklanjanje elementa sa početka liste

- 1 head treba da pokazuje na drugi element liste

`head = head._next`



# Uklanjanje elementa sa kraja liste

- za vežbu ;)

# Implementacija jednostruko spregnute liste u Pythonu <sub>1</sub>

```
class SingleList:
    def __init__(self):
        self._head = self._tail = None

    def add_first(self, elem):
        newest = Node(elem, self._head)
        self._head = newest
        if self._tail is None:      # ako je bila prazna
            self._tail = self._head # sada ima jedan element

    def add_last(self, elem):
        newest = Node(elem, None)
        if self._tail is not None:
            self._tail._next = newest
        self._tail = newest
        if self._head is None:      # ako je bila prazna
            self._head = newest      # sada ima jedan element
```

# Implementacija jednostruko spregnute liste u Pythonu 2

```
def remove_first(self):
    if self._head is None:    # već je prazna
        return
    if self._head == self._tail:
        self._head = self._tail = None
    self._head = self._head._next

def remove_last(self):
    if self._tail is None:
        return
    if self._head == self._tail:
        self._head = self._tail = None
    current = self._head
    while current._next != self._tail:
        current = current._next
    current._next = None
    self._tail = current
```

# Implementacija jednostruko spregnute liste u Pythonu 3

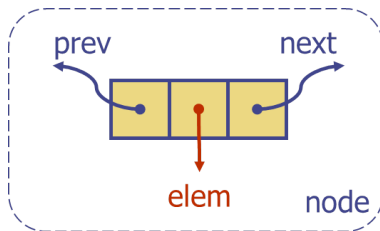
```
def get_first(self):  
    if self._head is not None:  
        return self._head._element  
    else:  
        return None
```

```
def get_last(self):  
    if self._tail is not None:  
        return self._tail._element  
    else:  
        return None
```

```
def __len__(self):  
    # ???
```

# Dvostruko spregnuta lista

- kretanje „unazad“ (od repa prema glavi) u jednostruko spregnutoj listi je nemoguće
- rešenje: čvorovi treba da sadrže referencu i na prethodni i na sledeći element liste



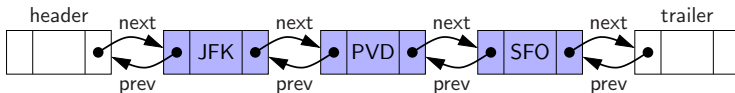
# Element dvostruko spregnute liste u Pythonu

```
class Node2:
    def __init__(self, element, prev, next):
        self._element = element
        self._prev = prev
        self._next = next
```

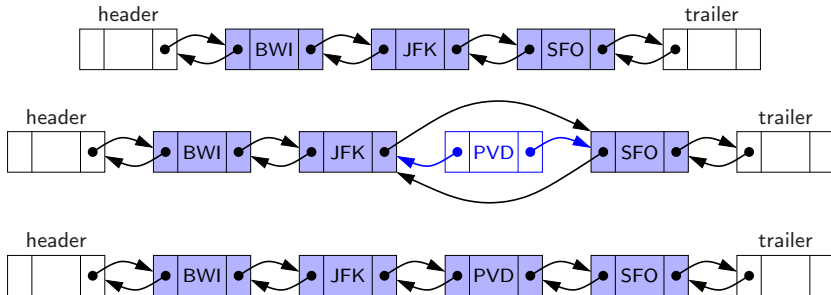


# Dvostruko spregnuta lista: glava i rep

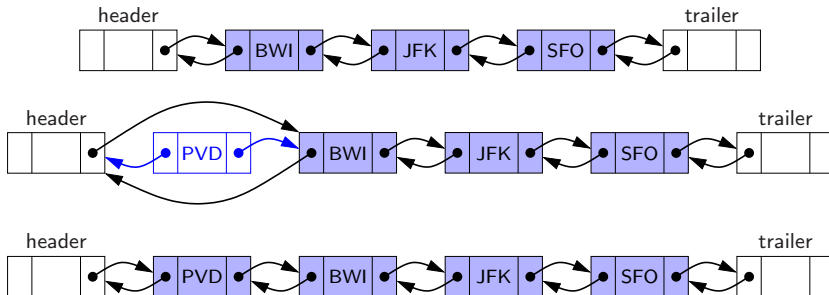
- prvi i poslednji element imaju poseban status
- ne koriste se za čuvanje podataka
- prazna lista: `head.next == tail` and `tail.prev == head`



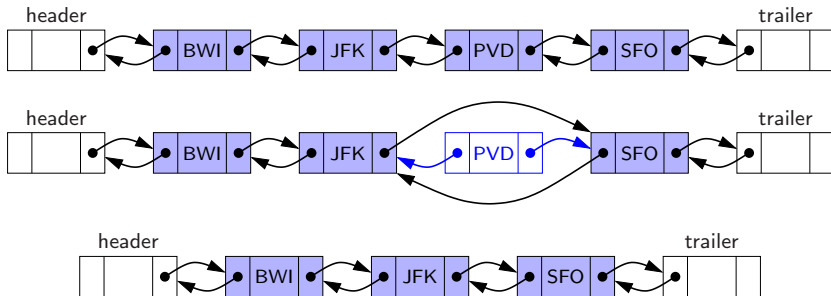
# Ubacivanje elementa u listu



# Dodavanje elementa na početak liste



# Uklanjanje elementa iz liste



# Dvostruko spregnuta lista u Pythonu

```
class DoubleList:
    def __init__(self):
        self._head = self.Node2(None, None, None)
        self._tail = self.Node2(None, None, None)
        self._head._next = self._tail
        self._tail._prev = self._head

    def is_empty(self):
        return self._head._next == self._tail

    def insert_before(self, element, successor):
        newest = self.Node2(element, successor._prev, successor)
        successor._prev._next = newest
        successor._prev = newest
        return newest

    def delete(self, node):
        if self.is_empty():
            return
        predecessor = node._prev
        successor = node._next
        predecessor._next = successor
        element = node._element
        node._prev = node._next = node._element = None
        return element
```