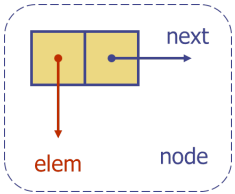
Liste

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

2022.

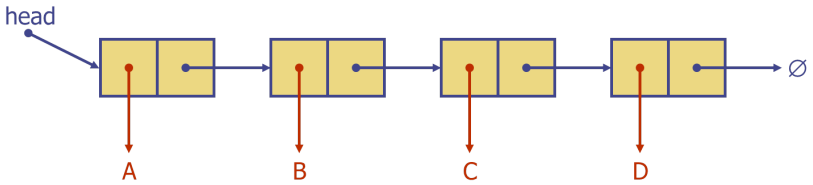
Jednostruko spregnuta lista

- 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



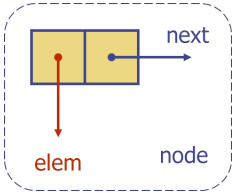
Jednostruko spregnuta lista

- 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



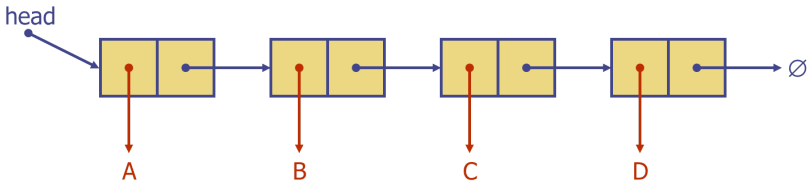
Element jednostruko spregnute liste u Pythonu

```
class Node:
  def __init__(self, value, next):
    self._value = value
    self._next = next
```



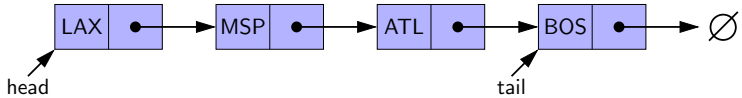
Iterator: obilazak svih elemenata liste

$current \leftarrow head$
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)

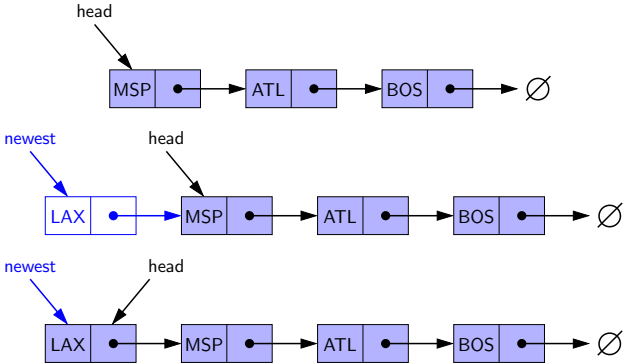


Granični slučajevi

- 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`

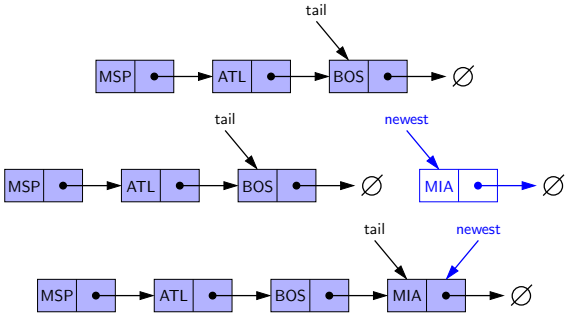
Dodavanje elementa na početak liste

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

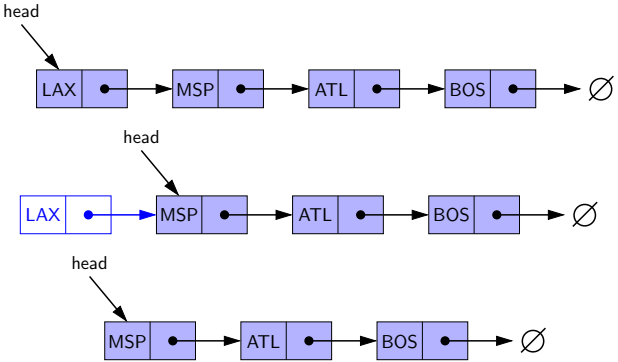
1. kreiraj novi čvor
2. upiši podatak u čvor
3. link na sledeći novog čvora je `None`
4. poslednji→sledeći pokazuje na novi čvor
5. `tail` pokazuje na novi čvor



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 [1]

```
class SingleList:
  def __init__(self):
    self._head = None
    self._tail = None
    self._size = 0

  def __len__(self):
    return self._size

  def is_empty(self):
    return self._size == 0

  def __iter__(self):
    current_node = self._head
    while current_node:
      yield current_node
      current_node = current_node.next
```

Implementacija jednostruko spregnute liste u Pythonu 2

```
def get_first(self):
  if self.is_empty():
    raise EmptyListException("Prazna lista!")
  return self._head

def get_last(self):
  if self.is_empty():
    raise EmptyListException("Prazna lista!")
  return self._tail
```

Implementacija jednostruko spregnute liste u Pythonu 3

```
def add_first(self, value):
  new_node = Node(value)
  if self.is_empty():
    self._tail = new_node
  else:
    new_node.next = self._head
  self._head = new_node
  self._size += 1

def add_last(self, value):
  new_node = Node(value)
  if self.is_empty():
    self._head = new_node
  else:
    self._tail.next = new_node
  self._tail = new_node
  self._size += 1
```

Implementacija jednostruko spregnute liste u Pythonu 4

```
def remove_first(self):
  if self.is_empty():
    raise EmptyListException("Prazna lista!")
  if self._size == 1:
    self._tail = None
  self._head = self._head.next
  self._size -= 1

def remove_last(self):
  if self.is_empty():
    raise EmptyListException("Prazna lista!")
  if self._size == 1:
    self._head = None
  for node in self:
    if node.next == self._tail:
      node.next = None
      self._tail = node
      break
  self._size -= 1
```

Implementacija jednostruko spregnute liste u Pythonu 5

```
def get_at(self, index):
  if not 0 <= index <= self._size-1:
    raise IndexError("Indeks van opsega!")
  counter = 0
  current_node = self._head
  while current_node:
    if counter == index:
      return current_node
    current_node = current_node.next
    counter += 1
```

Implementacija jednostruko spregnute liste u Pythonu 6

```
def insert_at(self, index, value):
  new_node = Node(value)
  if not 0 <= index <= self._size-1:
    raise IndexError("Indeks van opsega!")
  if index == 0:
    self.add_first(value)
    return
  previous_node = self.get_at(index-1)
  new_node.next = previous_node.next
  previous_node.next = new_node
  self._size += 1
```

Implementacija jednostruko spregnute liste u Pythonu 7

```
def remove_at(self, index):
  if not 0 <= index <= self._size-1:
    raise IndexError("Indeks van opsega!")
  if index == 0:
    self.remove_first()
    return
  if index == self._size-1:
    self.remove_last()
    return
  previous_node = self.get_at(index-1)
  next_node = previous_node.next.next
  previous_node.next = next_node
  self._size -= 1
```

Operacije

Operacija	Niz	Lista
data[i]	*O(1)*	*O(n)*
add_last(value)	*O(1)*	*O(1)*
add_first(value)	*O(n)*	*O(1)*
insert_at(index, value)	O(n)	*O(n)*
remove_first()	*O(n)*	*O(1)*
remove_last()	*O(1)*	*O(n)*
remove_at(index)	*O(n)*	*O(n)*