# Liste

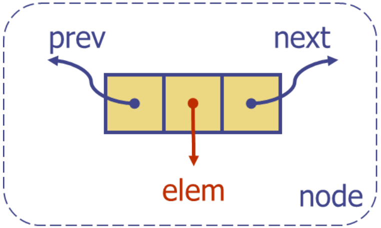
© Goodrich, Tamassia, Goldwasser

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

2022.

## 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

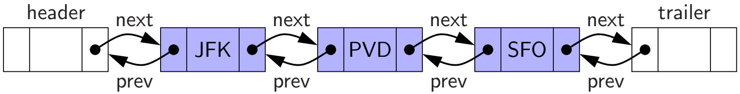


## Element dvostruko spregnute liste u Pythonu

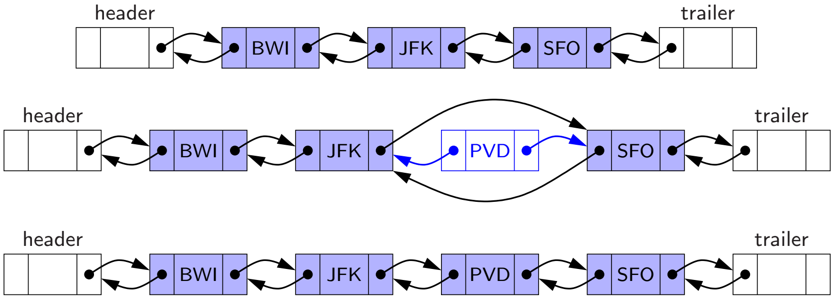
```python
class Node:
  def __init__(self, value, previous, next):
    self._value = value
    self._previous = previous
    self._next = next
```

## Dvostruko spregnuta lista: glava i rep

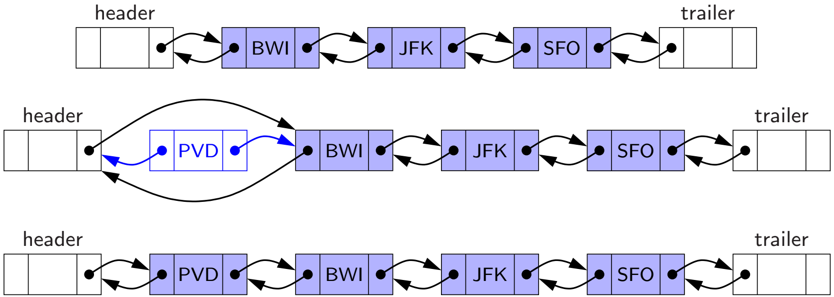
- prvi i poslednji element imaju poseban status
- ne koriste se za čuvanje podataka
- prazna lista: `head.next == tail and tail.prev == head`



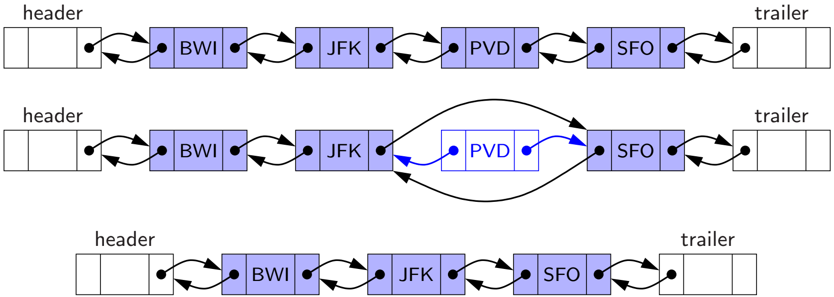
## Ubacivanje elementa u listu



## Dodavanje elementa na početak liste



## Uklanjanje elementa iz liste



## Implementacija dvostruko spregnute liste u Pythonu 1

```
class DoublyList:
  def __init__(self):
    self._head = Node(None, None, None)
    self._tail = Node(None, self._head, None)
    self._head.next = self._tail
    self._size = 0

  def __len__(self):
    return self._size

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

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

## Implementacija dvostruko spregnute liste u Pythonu $_2$

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

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

## Implementacija dvostruko spregnute liste u Pythonu $_3$

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

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

## Implementacija dvostruko spregnute liste u Pythonu $_4$

```
def remove_first(self):
  if self.is_empty():
    raise EmptyList("Prazna lista!")
  to_remove = self._head.next
  if self._size == 1:
    self._head.next = self._tail
    self._tail.previous = self._head
  else:
    new_first = self._head.next.next
    new_first.previous = self._head
    self._head.next = new_first
  self._size -= 1
  return to_remove

def remove_last(self):
  if self.is_empty():
    raise EmptyList("Prazna lista!")
  to_remove = self._tail.previous
  if self._size == 1:
    self._tail.previous = self._head
    self._head.next = self._tail
  else:
    second_last = self._tail.previous.previous
    second_last.next = self._tail
    self._tail.previous = second_last
  self._size -= 1
  return to_remove
```

## Implementacija dvostruko spregnute liste u Pythonu $_5$

```
def insert_after(self, node1, value):
  new_node = Node(value)
  node1.next.previous = new_node
  new_node.next = node1.next
  node1.next = new_node
  new_node.previous = node1
  self._size += 1
  return new_node

def insert_before(self, node1, value):
  new_node = Node(value)
  node1.previous.next = new_node
  new_node.previous = node1.previous
  new_node.next = node1
  node1.previous = new_node
  self._size += 1
  return new_node
```

## Implementacija dvostruko spregnute liste u Pythonu $_6$

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

## Implementacija dvostruko spregnute liste u Pythonu 7

```
def insert_at(self, index, value):
  if not 0 <= index <= self._size:
    raise IndexError("Nedozvoljen index!")
  if index == 0:
    return self.add_first(value)
  if index == self._size:
    return self.add_last(value)
  current_node = self.get_at(index)
  new_node = self.insert_before(current_node, value)
  self._size += 1
  return new_node
```

## Implementacija dvostruko spregnute liste u Pythonu 8

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