Python Data Structures

Grammy Jiang

Jun 27, 2019

CONTENTS:

CHAPTER

ONE

NODES

class data_structures.linked_list.**Node**(*value: Any*, *next_: Optional[data_structures.linked_list.Node] = None*)

The most simple Node class, only contains *value* and *next* properties, and there is no any security mechanism for node properties modification

Any enhanced Node class should inherit this class

Parameters

- **value** (*Any*) –
- **next** (*Optional[Node]*) –

abstract classmethod after_node(*value: Any*, *node: data_structures.linked_list.Node*) → data_structures.linked_list.Node

Create a node with the given value, and after the given node

Parameters

- **value** (*Any*) – The value that the node contains
- **node** (*Node*) – The node which *next* points to this node

Returns The node created

Return type *Node*

class data_structures.linked_list.nodes.**SinglyNode**(*value: Any*, *next_: Optional[data_structures.linked_list.Node] = None*)

The most simple SinglyNode class, only contains *value* and *next* properties, and there is no any security mechanism for node properties modification

Any enhanced SinglyNode class should inherit this class

Parameters

- **value** (*Any*) –
- **next** (*Optional[Node]*) –

classmethod after_node(*value: Any*, *node: data_structures.linked_list.nodes.SinglyNode*) → data_structures.linked_list.nodes.SinglyNode

Create a node with the given value, and after the given node

Parameters

- **value** (*Any*) – The value that the node contains
- **node** (*SinglyNode*) – The node which *next* points to this node

Returns The node created

Return type *SinglyNode*

class data_structures.linked_list.nodes.**SecureSinglyNode**(*value: Any*, *next_: Optional[data_structures.linked_list.nodes.SecureSin = None*, *frozen: bool = True*)

A node with secure mechanism - when it is frozen, the value and next cannot be changed

Initial a node with secure mechanism, and as default the node is frozen to change after initialized

Parameters

- **value** (*Any*) –
- **next** (*Optional[Node]*) –
- **frozen** (*bool*) –

classmethod after_node(*value: Any*, *node: data_structures.linked_list.nodes.SecureSinglyNode*, *frozen: bool = True*) → data_structures.linked_list.nodes.SecureSinglyNode

Create a node with the given value, and after the given node

Parameters

- **value** (*Any*) – The value that the node contains
- **node** (SinglyNode) – The node which *next* points to this node
- **frozen** (*bool*) –

Returns The node created

Return type *SinglyNode*

freeze() → None

Freeze this node - the value and next cannot be changed

Returns

Return type None

unfreeze() → None

Unfreeze this node - the value and next can be changed

Returns

Return type None

property next

Returns

Return type Optional[*SecureSinglyNode*]

property value

Returns

Return type Any

class data_structures.linked_list.nodes.**DoublyNode**(*value: Any*, *previous: Optional[data_structures.linked_list.nodes.DoublyNode] = None*, *next_: Optional[data_structures.linked_list.nodes.DoublyNode] = None*)

The most simple DoublyNode class, only contains *value* and *next* properties, and there is no any security mechanism for node properties modification

Any enhanced DoublyNode class should inherit this class

Create a node with the given value, previous and next nodes

Parameters

- **value** (*Any*) –
- **previous** (*Optional[DoublyNode]*) –
- **next** (*Optional[DoublyNode]*) –

classmethod after_node(*value: Any*, *node: data_structures.linked_list.nodes.DoublyNode*) → data_structures.linked_list.nodes.DoublyNode

Create a node with the given value, and after the given node

Parameters

- **value** (*Any*) – The value that the node contains
- **node** (DoublyNode) – The node which *next* points to this node

Returns The node created

Return type *DoublyNode*

CHAPTER

TWO

SINGLY LINKED LIST

class data_structures.linked_list.singly.**SinglyLinkedList**(**args*)

This is the simple singly linked list:

- have only one head point, no tail pointer

Parameters **args** –

append(*value: Any*) → None

Append a node after the tail of this singly linked list

Parameters **value** (*Any*) –

Returns

Return type None

insert_after(*value: Union[data_structures.linked_list.nodes.SinglyNode, Any], node: Optional[data_structures.linked_list.nodes.SinglyNode] = None*) → None

If after_node is not provided, the given value will be insert to the beginning of this singly linked list

Parameters

- **value** (*Union[Node, Any]*) –
- **node** (*Optional[Node]*) –

Returns

Return type None

pop() → data_structures.linked_list.nodes.SinglyNode

Pop the last node of this singly linked list

Returns

Return type *SinglyNode*

remove_after(*node: Optional[data_structures.linked_list.nodes.SinglyNode] = None*) → None

Remove one node after the give node

Parameters **node** (*Optional[Node]*) – If after_node is not provided, the first node of this singly linked list will be removed

Returns

Return type None

replace(*old: Any*, *new: Any*, *max_: Optional[int] = None*) → None

In-place replace the node old value with the given new one

Complexity:

- Space: (n), (n), (1)
- Time: (n), (n), (1)

Parameters

- **old** (*Any*) – The old value to be replaced
- **new** (*Any*) – The new value to replace the old one
- **max** (*Optional[int]*) – if max is not provided all of nodes equaled to old will be changed to new

Returns This method is a in-place change and returns None

Return type None

reverse() → None
In-place reverse

Returns

Return type None

search_iter(*value: Any*) → data_structures.linked_list.singly.SinglyLinkedListSearchIterator
Search for a given value, return a iterator

Parameters **value** (*Any*) –

Returns

Return type Iterator

class data_structures.linked_list.singly.**CircularSinglyLinkedList**(**args*)

Parameters **args** –

append(*value: Any*) → None

Parameters **value** (*Any*) –

Returns

Return type None

pop() → data_structures.linked_list.nodes.SinglyNode

Returns

Return type *SinglyNode*

reverse() → None
In-place reverse

Returns

Return type None

search_iter(*value: Any*)

Parameters **value** (*Any*) –

Returns

CHAPTER

THREE

DOUBLY LINKED LIST

class data_structures.linked_list.doubly.**DoublyLinkedList**(**args*)

Parameters **args** –

append(*value: Any*) → None

Parameters **value** (*Any*) –

Returns

Return type None

pop() → Optional[data_structures.linked_list.nodes.DoublyNode]

Returns

Return type Optional[*Node*]

reverse() → None
In-place reverse

Returns

Return type None

search_iter(*value: Any*) → data_structures.linked_list.doubly.DoublyLinkedListSearchIterator
Search for a given value, return a iterator

Parameters **value** (*Any*) –

Returns

Return type Generator[*Node*, None, None]

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

INDEX