Lecture 4 - Dynamic Arrays

Foundations of Computer Science (4CC505)

Dr Sam O'Neill

Summary

Data Structure	`create(X)`	`get(i)` `set(i,x)`	`insert(i,x)` `delete(i)`	`insert_first(i,x)` `delete_first()`	`insert_last(i,x)` `delete_last()`	Space
Array	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Dynamic Array	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$**	$O(n)$
Singly Linked List	$O(n)$	$O(n)$	$O(n)^\dagger$	$O(1)$	$O(n)$	$O(n)$
Doubly Linked List	$O(n)$	$O(n)$	$O(n)^\dagger$	$O(1)$	$O(1)$	$O(n)$

Worse-case Complexity

**Amoritized Time, † assumes traversal to ith node

Data Structure	`create(X)`	`get(i)` `set(i,x)`	`insert(i,x)` `delete(i)`	`insert_first(i,x)` `delete_first()`	`insert_last(i,x)` `delete_last()`	Space
Doubly Linked List	$O(n)$	$O(n)$	$O(n)^\dagger$	$O(1)$	$O(1)$	$O(n)$

Worse-case Complexity

† assumes traversal to ith node

Data Structure	`create(X)`	`get(i)` `set(i,x)`	`insert(i,x)` `delete(i)`	`insert_first(i,x)` `delete_first()`	`insert_last(i,x)` `delete_last()`	Space
Singly Linked List	$O(n)$	$O(n)$	$O(n)^\dagger$	$O(1)$	$O(n)$	$O(n)$

Worse-case Complexity

† assumes traversal to ith node

Python Code - With `insert_first()`

```
# Linked list implementation in Python

class Node:
    """ Represents a single node"""
    def __init__(self, item):
        self.item = item
        self.next = None

class LinkedList:
    """ The whole linked list"""
    def __init__(self):
        self.head = None
        self._size = 0

    def insert_first(self, x):                  # O(1)
        new_node = Node(x)                      # create a new node with item x
        new_node.next = self.head               # set the new node to point to the current head
        self.head = new_node                    # replace the linked list head to be new node
        self._size += 1                         # increase the size

if __name__ == '__main__':

    linked_list = LinkedList()

    # Assign item values
    linked_list.insert_first(1)
    linked_list.insert_first(2)
    linked_list.insert_first(3)

    # list will now contain 3 -> 2 -> 1
```

Python Code

```
# Linked list implementation in Python
# Programiz https://www.programiz.com/dsa/linked-list

class Node:
  """ Represents a single node"""
  def __init__(self, item):
      self.item = item
      self.next = None

class LinkedList:
  """ The whole linked list"""
  def __init__(self):
      self.head = None

if __name__ == '__main__':

  linked_list = LinkedList()

  # Assign item values
  linked_list.head = Node(1)
  second = Node(2)
  third = Node(3)

  # Connect nodes
  linked_list.head.next = second
  second.next = third

  # list will not contain 1 -> 2 -> 3
```