Week 1

Wednesday, March 8, 2023 8:33 AM

Array:

- Contiguous area of memory consisting of equal size elements indexed by contiguous integers
- Constant time access to any element
- Constant time to add/remove at the end
- Linear time to add/remove at an arbitrary location

Time for common operations

	Add	Remove
Beginning	O(n)	O(n)
End	O(1)	O(1)
Middle	O(n)	O(n)

Link List:

- Node contains two elements, a <u>key</u> and <u>next pointer</u>

List API

PushFront(Key)	Add to front
Key TopFront()	Return front item
PopFront()	Remove front item
PushBack(Key)	Add to back (also know as append)
PopBack()	Remove back item
Key TopBack()	Return back item
Boolean Find(Key)	Is key in list?
Erase(key)	Remove key from list
AddBefore(Node,Key)	Add key before node

<u>PushFront(key)</u>

```
    New node. = node
    Key = node.key
    Head = node.next
    Head = nod
    If tail = null:
        Tail = head
```

<u>PopFront()</u>

```
    If head = null: ERROR
    Head = head.next
    If head = null:
        Tail = null
```

```
PushBack(key)
    New node = node
    Key = node.key
    Node.next = null
    If tail = null:
        Node = tail
        Tail  = head
    Else:
        Node = tail.next
        Node = tail

PopBack()
    If head = null: ERROR
    If head = tail:
        Tail = null
        Head. = tail
    Else:
        Head = p
        While p.next.next != null
            P.next. = p
        Null = p.next; P = tail

AddAfter(node, key)
    New node = node2
    Key = node2.key
    Node2.next = node.next
    Node.next. = node 2
    If tail = node
        Node2 = tail
```

Singly Linked list	No tail	With tail
PushFront	O(1)	
TopFront	O(1)	
PopFront	O(1)	
PushBack	O(n)	O(1)
TopBack	O(n)	O(1)
PopBack	O(n)	
Find	O(n)	
Erase	O(n)	
Empty	O(1)	
AddBefore	O(n)	
AddAfter	O(1)	

Doubly Linked List

Node contains

- Key
- Next pointer
- Prev pointer

```
PushBack(key)
    New node = node
    Key = node.key
    If tail = null
        Node = tail
        Tail = head
    Else
        Node = tail.next
        Tail = node.prev
        Node = tail

PopBack()
    If head = null: ERROR
    If head. = tail
        Null = tail
        Tail = head
    Else
        Tail.prev = tail
        Null = tail.next

AddAfter(node, key)
    New node. = node2
    Key = node2.key
    Node.next = node2.next
    Node = node2.prev
    Node2 = node.next
    If node2.next != null
        Node2 = Node2.next.prev
    If tail = node
        Node2 = head
```

Doubly Linked list	No tail	With tail
PushFront	O(1)	
TopFront	O(1)	
PopFront	O(1)	
PushBack	O(n)	O(1)
TopBack	O(n)	O(1)
PopBack	O(n) O(1)	
Find	O(n)	
Erase	O(n)	

Empty	O(1)	
AddBefore	O(n) O(1)	
AddAfter	O(1)	

Stacks

Stack: Abstract data type with the following operations:

- Push(Key): adds key to collection
- KeyTop(): returns most recently added key
- KeyPop(): removes and returns most recently added key

```
IsBalanced(str)
Stack stack
For char in str
    If char in ['(", '[']:
        Stack.Push(char)
    Else:
        If stack.empty(): return false
        Stack.pop() = top
        If (top = '[' and char != ']') or
        (top = '(' and char !+ ')'):
        Return false
Return stack.empty()
```

Queue

Queue: Abstract data type with the following operations:

- Enqueue(Key): adds key to collection
- Dequeue(): removes and returns least recently added key

FIFO: First in First Out