Queue

Jie Wang

Contents

- The queue interface.
- Implementation.
- Applications - BFS.
- Homework.

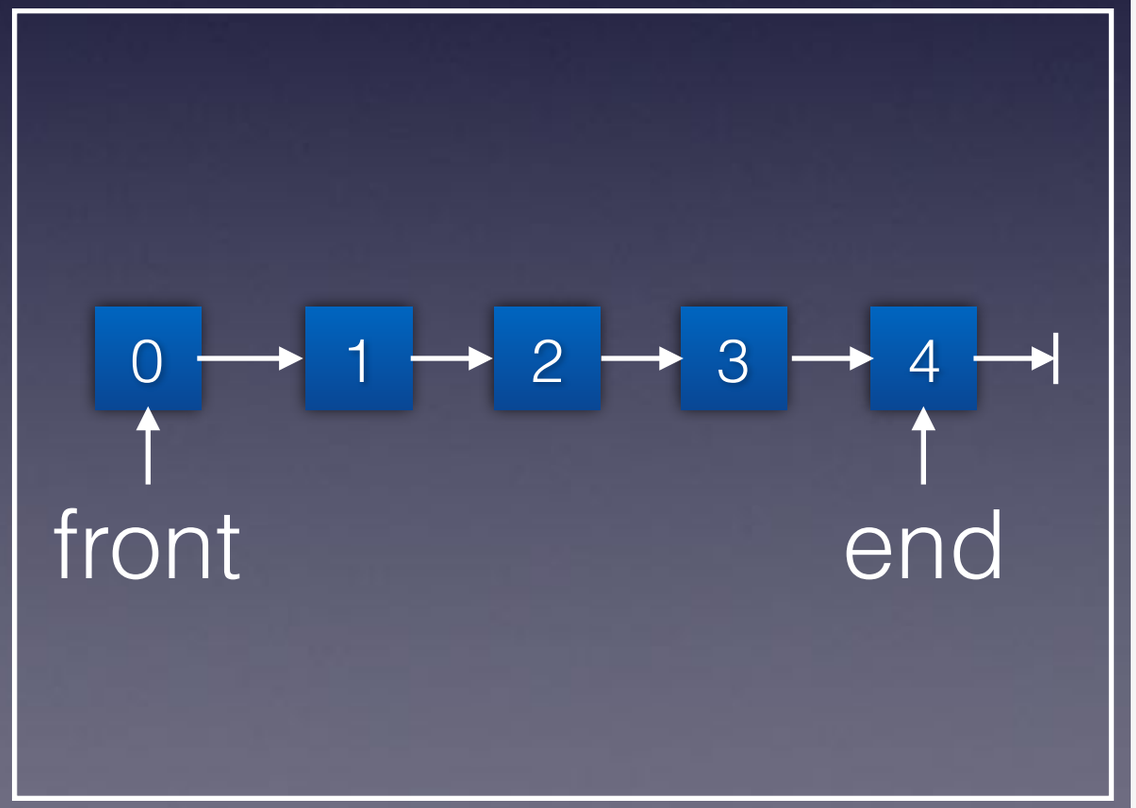
Queue interface

- FIFO data structure
- enQueue(x) : insert x into the back of the queue.
- deQueue(): retrieve and remove the front element of the queue.
- peek(): retrieve, but not remove the front element of the queue.

Implementation

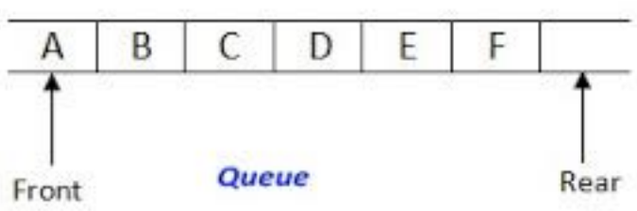
- Linked list implementation.
- Circular Array implementation.

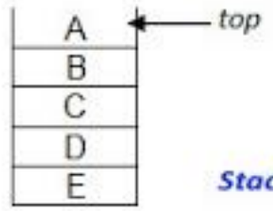
Linked List Implementation




```
class ListNode:
    def __init__(self, x):
        self.val = x
        self.next = None

class Queue:
    def __init__(self):
        self.size = 0
        self.front = None
        self.end = None

    def isEmpty(self):
        return self.front is None

    def enQueue(self, x):
        node = ListNode(x)
        if self.isEmpty():
            self.front = node
        else:
            self.end.next = node

        self.end = node
        self.size += 1

def deQueue(self):
    if self.isEmpty():
        raise Exception('Queue is Empty!')

    temp = self.front.val
    self.front = self.front.next
    if self.front is None:
        self.end = None
    self.size -= 1
    return temp
```

Circular Array Implementation





shyam prakash

sspcprogram.blogspot.com

Applications- BFS

- Traverse a graph.
- Shortest path.

BFS example 1

- 102. Binary Tree Level Order Traversal.

- Given a binary tree, return the *level order* traversal of its nodes' values.

```
class Solution:
    def levelOrder(self, root):
        """
        :type root: TreeNode
        :rtype: List[List[int]]
        """
        result = []
        if root is None:
            return result

        q = []
        q.append(root)
        while len(q) > 0:
            l = len(q)
            oneLevel = []
            for i in range(l):
                node = q.pop(0)
                oneLevel.append(node.val)
                if node.left is not None:
                    q.append(node.left)
                if node.right is not None:
                    q.append(node.right)
            result.append(oneLevel)

        return result
```

BFS example 2

- 133. Clone Graph

```python
class Solution:
    def cloneGraph(self, node):
        if node is None:
            return None

        nodes = self.getNodes(node)
        cache = {}
        for myNode in nodes:
            cache[myNode.label] = UndirectedGraphNode(myNode.label)
        for myNode in nodes:
            clone = cache[myNode.label]
            for neighbor in myNode.neighbors:
                clone.neighbors.append(cache[neighbor.label])

        return cache[node.label]

    def getNodes(self, node):
        q = []
        myHash = set()
        nodes = []
        if node is None:
            return nodes
        q.append(node)
        myHash.add(node)
        while len(q) > 0:
            top = q.pop(0)
            nodes.append(top)
            for neighbor in top.neighbors:
                if neighbor not in myHash:
                    q.append(neighbor)
                    myHash.add(neighbor)

        return nodes
```

HW 4

622. Design Circular Queue

232. Implement Queue using Stacks

107. Binary Tree Level Order Traversal II

323. Number of Connected Components in an Undirected Graph

406. Queue Reconstruction by Height