කැමති විෂය
කැමති වෙලාවක
කැමති තැනක
නිදහසේ ඉගන ගන්න
පාඩම් සහ ප්‍රශ්න

Shilpa64.lk

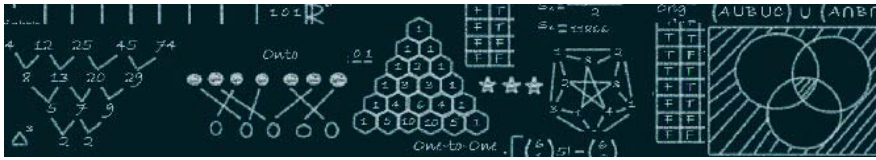
ශිල්පසයුර Google for Education CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka..

AlgoHack #10

PROGRAMING DATA STRUCTURES II

AlgoHack #10



PROGRAMING DATA STRUCTURES II

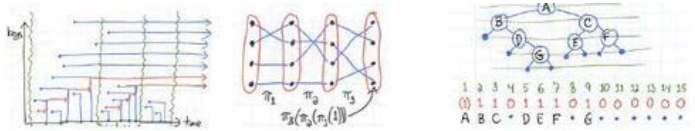
Authors
Niranjan Meegammana
N P Vishwa Kumara

Reviewers
Ravindu Ramesh Perera, Devanjith De Silva, Prabhashana Hasthidhara, Yamuna Ratnayake.

ශිල්පසයුර Google for Education CSSL COMPUTER SOCIETY OF SRI LANKA

AlgoHack aims to teach Computer Science and Programing to young people, initiated by Shilpa Sayura Foundation, supported by GOOGLE RISE and Computer Society of Sri Lanka.

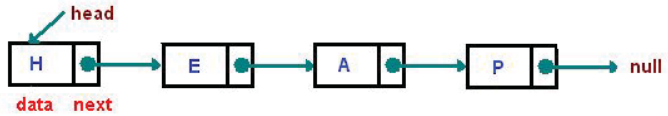
Complex Data Structures



They are used to store large and connected data.
We already studied arrays that is simple list of data.

Linked List

Linked lists are made up of several **nodes**.
Each node contains a reference to the next node.
Each node contains a unit of data called the cargo.
The first element of the list is called head.
We can traverse the list using references.



We can add, remove elements on demand.
But it takes additional memory to keep a pointers.
We also can't access data directly like xarray[n].

A node is the building block of a linked list. Following is a node class with methods to manipulate data.

The Node class defines the node.

```
class Node:
    def __init__(self, data=None, next=None):
```



```
    print(a)
    b = q.dequeue()
    print(b)
    c = q.dequeue()
    print(c)
    d = q.dequeue()
    print(d)
    q.enqueue(5)
    q.enqueue(6)
    print(q.dequeue())
```

Problem :

Shamil took a job as customer services manager in a supermarket. He wanted to improve customer service by reducing number of people in a queue. The supermarket has 2 counters. There are n number of people each taking t_n time to serve.

Design an algorithm to serve customers fast by assigning them to queue to serve fast.

Time taken to serve customer
4, 3, 7,13, 11, 4, 6, 4, 8, 11, 24, 12, 4, 9, 6, 14, 3, 6, 12

You can swap people between queues .

```
    def size(self):
        return len(self.elements)

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

class CreateQueueWithTwoStacks:

    def __init__(self):
        self.stack_1 = Stack()
        self.stack_2 = Stack()

    def enqueue(self, item):
        self.stack_1.push(item)

    def dequeue(self):
        if not self.stack_1.is_empty():
            while self.stack_1.size() > 0:
                self.stack_2.push(self.stack_1.pop())
            res = self.stack_2.pop()
            while self.stack_2.size() > 0:
                self.stack_1.push(self.stack_2.pop())
            return res

if __name__ == '__main__':
    q = CreateQueueWithTwoStacks()
    q.enqueue(1)
    q.enqueue(2)
    q.enqueue(3)
    a = q.dequeue()
```



```
        self.cargo = data
        self.next  = next

    def __str__(self):
        return str(self.data)
```

We can create many nodes

```
node1 = Node(1)
node2 = Node(2)
node3 = Node(3)
```

But they are not linked yet.

To link the nodes, we have to make the first node refer to the second and the second node refer to the third and son on.

```
node1.next = node2
node2.next = node3
```

The **node3.next** is None. It means end of the list. Following function prints the list.

```
def printList(node):

    while node:
          print (node)
          node = node.next

printlist(node1)
```

To invoke this method, we need to pass the reference to

the first node: What will be the output?
Following is a complete implementation of a linked list should include the head of the list, It has following methods:
size() - returns the number of nodes in the list
insert(data) - insert data at the head of the list
search(data) - returns the node that has the data, returns None if not found
delete(data) - delete a node with the data, returns the node if found, None if not found

print()- prints the whole Linked List.

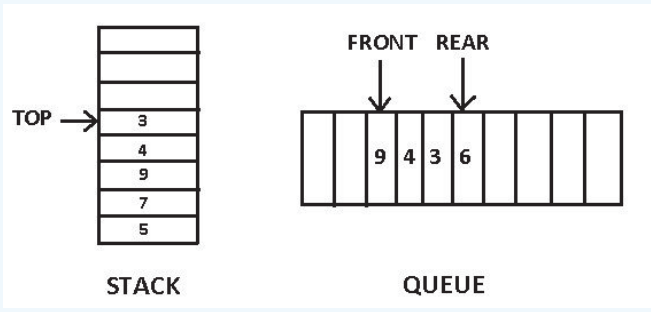
```
class Node(object):

    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next

    def get_data(self):
        return self.data

    def get_next(self):
        return self.next

    def set_next(self, new_next):
        self.next = new_next

class LinkedList(object):
```



```
q.enqueue(3)
q.dequeue()
```

Stack and Queue

Both Stack and Queue can be implemented using arrays or linked lists. Following figure gives a rough visualization of the stack and queue data structures :



Implementing A Queue using Two Stacks Python

```
class Stack:
    def __init__(self):
        self.elements = []

    def push(self, item):
        self.elements.append(item)

    def pop(self):
        return self.elements.pop()
```

```
while not s.is_empty():
    print(s.pop())
```

Explain above program.

Queue

Data added at end and removed from top in a queue.

```
class Queue:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def enqueue(self, item):
        self.items.insert(0,item)

    def dequeue(self):
        return self.items.pop()

    def size(self):
        return len(self.items)

Create a Queue with 3 items.
q = Queue()
q.enqueue('hello')
q.enqueue('dog')
```



```
def __init__(self, head=None):
        self.head = head

 def insert(self, data):
    new_node = Node(data)
    new_node.set_next(self.head)
    self.head = new_node

 def size(self):
    current = self.head
    count = 0
    while current:
        count += 1
        current = current.get_next()
        return count

 def search(self, data):
       current = self.head
       while current:
            if current.get_data() == data
                  return current
             else:
                  current = current.get_next()
                  return None

 def delete(self, data):
      current = self.head
      prev = None
      while current:
          if current.get_data() == data:
               if current == self.head:
                    self.head = current.get_next()
               else:
```

```
                prev.set_next(current.get_next())
                return current

        prev = current
        current = current.get_next()
        return None

    def print(self):

        lst = []
        current = self.head
        while current:
            lst.append(str(current.get_data())
            current = current.get_next()
            print(join(lst))
```

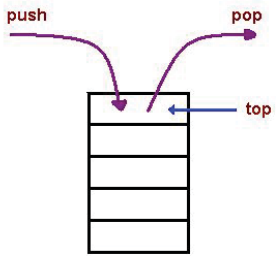
Explain what happens in above code

```
#irst we create linked List
LL1=LinkedList() // empty list
LL1.insert(10) // insert first item
LL1.insert(20)
LL1.insert(30)
LL1.insert(40)
LL1.print()
```

What does this output?
How the list is ordered? Why?
How do you delete 30?
How do you find the new list size?

Remove the node
Insert next element from array to the list
Exit when all nodes are processed.

Stack



Stack is list where data added and removed from top.
Stack is Last in First Out (LIFO) data structure. The last item added is the first to be removed.

```
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self): # remove and return top element
        return self.items.pop()

    def is_empty(self):
        return (self.items == [])

s = Stack()
s.push(54)
s.push(45)
s.push("+")
```

Its is a linked list whose tail.next points to the head.

```
insert(self, data):
      new_node = Node(data)
       new_node.set_next(self.head)
       self.head = new_node
```

Explain what happens in above code block?

Circular lists are useful in applications to repeatedly go around the list. Operating systems running multiple applications on a list.

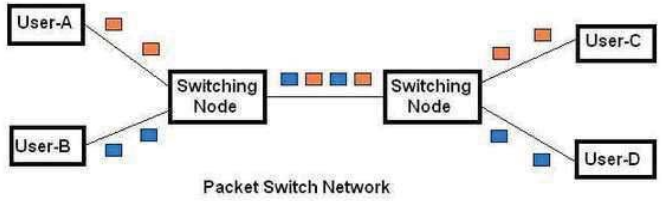
It cycles through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. When it reaches the end of the list it can cycle around to the front of the list.

Design an algorithm for a Circular Linked List.
What kind of use you can get from it ?

Split a string 50 characters to create an array.
Create a circular list
Insert 10 elements to the list from array.
Select a node
Print its content

How do you search 40?
What happens if you search for 70?

Write a function to find 3rd elementI
nsert a Node at the Tail of a Linked List
Insert a node at the head of a linked list
Insert a node at a specific position in a linked list
Delete first node linked list
Delete last node linked list
Reverse a linked list
Sort a linked List
Compare two linked lists
Merge two sorted linked lists



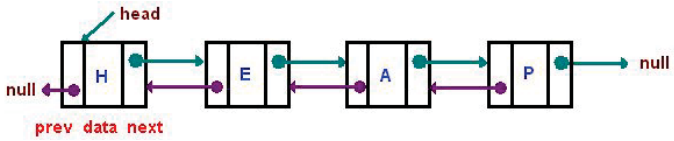
When messages are delivered on **networks,** the message is broken into packets and each packet has a key of the next one so that at the receiver's end , it will recreate message.

Consider the history section of **web browsers**, where it creates a linked list of web-pages visited, so that when you check history (traversal of a list) or press back

button, the previous node's data is fetched.

Discuss other uses of Linked List.

A **doubly linked list** has two references, one to the next node and another to previous node. So that we can traverse in two ways using two pointers at each node.



Doubly-linked list implementation in python

```
class Node(object):
        def __init__(self, data, prev, next):
            self.data = data
            self.prev = prev
            self.next = next

class DoubleList(object):
      head = None
      tail = None
```

Can you modify Linked List program to manipulate a doubly linked list.

What functions you have to modify?
What are the advantages of doubly linked lists?

Design an algorithm and write a function to reverse the doubly linked list. It is not sufficient to just swap values between nodes. All pointers in the linked list should be correctly updated.

Design an algorithm and write a function to remove the middle element(s) from the doubly linked list. If the list has an odd number of elements then the middle is one element, if the list has an even number of elements then the middle is two elements. All pointers should be correctly updated.

Circular Linked List

Head is the first node in list. The last node points to the head of the list.