

# GeeksforGeeks

A computer science portal for geeks

Practice

IDE

Q&A

GeeksQuiz

## Sort a linked list of 0s, 1s and 2s

Given a linked list of 0s, 1s and 2s, sort it.

Source: [Microsoft Interview | Set 1](#)

Following steps can be used to sort the given linked list.

- 1) Traverse the list and count the number of 0s, 1s and 2s. Let the counts be n1, n2 and n3 respectively.
- 2) Traverse the list again, fill the first n1 nodes with 0, then n2 nodes with 1 and finally n3 nodes with 2.

### C/C++

```
// C Program to sort a linked list 0s, 1s or 2s
#include<stdio.h>
#include<stdlib.h>

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

// Function to sort a linked list of 0s, 1s and 2s
void sortList(struct node *head)
{
    int count[3] = {0, 0, 0}; // Initialize count of '0', '1' and '2' as 0
    struct node *ptr = head;

    /* count total number of '0', '1' and '2'
    * count[0] will store total number of '0's
    * count[1] will store total number of '1's
    * count[2] will store total number of '2's */
    while (ptr != NULL)
    {
        count[ptr->data] += 1;
        ptr = ptr->next;
    }

    int i = 0;
    ptr = head;

    /* Let say count[0] = n1, count[1] = n2 and count[2] = n3
    * now start traversing list from head node,
    * 1) fill the list with 0, till n1 > 0
```

```

    * 2) fill the list with 1, till n2 > 0
    * 3) fill the list with 2, till n3 > 0 */
while (ptr != NULL)
{
    if (count[i] == 0)
        ++i;
    else
    {
        ptr->data = i;
        --count[i];
        ptr = ptr->next;
    }
}

/* Function to push a node */
void push (struct node** head_ref, int new_data)
{
    /* allocate node */
    struct node* new_node =
        (struct node*) malloc(sizeof(struct node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Function to print linked list */
void printList(struct node *node)
{
    while (node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}

/* Driver program to test above function*/
int main(void)
{
    struct node *head = NULL;
    push(&head, 0);
    push(&head, 1);
    push(&head, 0);
    push(&head, 2);
    push(&head, 1);
    push(&head, 1);
    push(&head, 2);
    push(&head, 1);
    push(&head, 2);

    printf("Linked List Before Sorting\n");
    printList(head);

    sortList(head);

    printf("Linked List After Sorting\n");
    printList(head);
}

```

```
    return 0;
}
```

[Run on IDE](#)

## Java

```
// Java program to sort a linked list of 0, 1 and 2
class LinkedList
{
    Node head; // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    void sortList()
    {
        // initialise count of 0 1 and 2 as 0
        int count[] = {0, 0, 0};

        Node ptr = head;

        /* count total number of '0', '1' and '2'
        * count[0] will store total number of '0's
        * count[1] will store total number of '1's
        * count[2] will store total number of '2's */
        while (ptr != null)
        {
            count[ptr.data]++;
            ptr = ptr.next;
        }

        int i = 0;
        ptr = head;

        /* Let say count[0] = n1, count[1] = n2 and count[2] = n3
        * now start traversing list from head node,
        * 1) fill the list with 0, till n1 > 0
        * 2) fill the list with 1, till n2 > 0
        * 3) fill the list with 2, till n3 > 0 */
        while (ptr != null)
        {
            if (count[i] == 0)
                i++;
            else
            {
                ptr.data = i;
                --count[i];
                ptr = ptr.next;
            }
        }
    }

    /* Utility functions */

    /* Inserts a new Node at front of the list. */
}
```

```

public void push(int new_data)
{
    /* 1 & 2: Allocate the Node &
       Put in the data*/
    Node new_node = new Node(new_data);

    /* 3. Make next of new Node as head */
    new_node.next = head;

    /* 4. Move the head to point to new Node */
    head = new_node;
}

/* Function to print linked list */
void printList()
{
    Node temp = head;
    while (temp != null)
    {
        System.out.print(temp.data+" ");
        temp = temp.next;
    }
    System.out.println();
}

/* Driver program to test above functions */
public static void main(String args[])
{
    LinkedList llist = new LinkedList();

    /* Constructed Linked List is 1->2->3->4->5->6->7->
       8->8->9->null */
    llist.push(0);
    llist.push(1);
    llist.push(0);
    llist.push(2);
    llist.push(1);
    llist.push(1);
    llist.push(2);
    llist.push(1);
    llist.push(2);

    System.out.println("Linked List before sorting");
    llist.printList();

    llist.sortList();

    System.out.println("Linked List after sorting");
    llist.printList();
}
}
/* This code is contributed by Rajat Mishra */

```

[Run on IDE](#)

## Python

```

# Python program to sort a linked list of 0, 1 and 2
class LinkedList(object):
    def __init__(self):
        # head of list

```

```

self.head = None

# Linked list Node
class Node(object):
    def __init__(self, d):
        self.data = d
        self.next = None

def sortList(self):

    # initialise count of 0 1 and 2 as 0
    count = [0, 0, 0]

    ptr = self.head

    # count total number of '0', '1' and '2'
    # * count[0] will store total number of '0's
    # * count[1] will store total number of '1's
    # * count[2] will store total number of '2's
    while ptr != None:
        count[ptr.data] += 1
        ptr = ptr.next

    i = 0
    ptr = self.head

    # Let say count[0] = n1, count[1] = n2 and count[2] = n3
    # * now start traversing list from head node,
    # * 1) fill the list with 0, till n1 > 0
    # * 2) fill the list with 1, till n2 > 0
    # * 3) fill the list with 2, till n3 > 0
    while ptr != None:
        if count[i] == 0:
            i += 1
        else:
            ptr.data = i
            count[i] -= 1
            ptr = ptr.next

# Utility functions
# Inserts a new Node at front of the list.
def push(self, new_data):

    # 1 & 2: Allocate the Node &
    # Put in the data
    new_node = self.Node(new_data)

    # 3. Make next of new Node as head
    new_node.next = self.head

    # 4. Move the head to point to new Node
    self.head = new_node

# Function to print linked list
def printList(self):
    temp = self.head
    while temp != None:
        print str(temp.data),
        temp = temp.next
    print ''

# Drier program to test above functions
l1 = LinkedList()
l1.push(0)

```

```
l1list.push(1)
l1list.push(0)
l1list.push(2)
l1list.push(1)
l1list.push(1)
l1list.push(2)
l1list.push(1)
l1list.push(2)

print "Linked List before sorting"
l1list.printList()

l1list.sortList()

print "Linked List after sorting"
l1list.printList()

# This code is contributed by BHAVYA JAIN
```

[Run on IDE](#)

Output:

```
Linked List Before Sorting
2  1  2  1  1  2  0  1  0
Linked List After Sorting
0  0  1  1  1  1  2  2  2
```

Time Complexity:  $O(n)$

Auxiliary Space:  $O(1)$

This article is compiled by **Narendra Kangralkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



The advertisement features a blue header with a magnifying glass icon over the text 'T-SQL'. Below this is a green banner with the text 'A Free Course From Microsoft'. The main title 'Querying with Transact-SQL' is in large black font. To the left is the 'edX' logo with 'edx.org' below it. To the right is a green button with the text 'Enroll now' and 'Self-Paced' below it.

124 Comments Category: [Linked Lists](#) [Sorting](#)

## Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

(Login to Rate and Mark)

2.6

Average Difficulty : **2.6/5.0**  
Based on **3** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Like Share 28 people like this.

Writing code in comment? Please use [code.geeksforgeeks.org](http://code.geeksforgeeks.org), generate link and share the link here.

@geeksforgeeks, Some rights reserved

Contact Us!

About Us!

Advertise with us!