Swap nodes in a linked list without swapping data

Given a linked list and two keys in it, swap nodes for two given keys. Nodes should be swapped by changing links. Swapping data of nodes may be expensive in many situations when data contains many fields.

It may be assumed that all keys in linked list are distinct.

Examples:

```
Input:  10->15->12->13->20->14,  x = 12, y = 20
Output: 10->15->20->13->12->14

Input:  10->15->12->13->20->14,  x = 10, y = 20
Output: 20->15->12->13->10->14

Input:  10->15->12->13->20->14,  x = 12, y = 13
Output: 10->15->13->12->20->14
```

This may look a simple problem, but is interesting question as it has following cases to be handled.
1) x and y may or may not be adjacent.
2) Either x or y may be a head node.
3) Either x or y may be last node.
4) x and/or y may not be present in linked list.

How to write a clean working code that handles all of the above possibilities.

We strongly recommend to minimize your browser and try this yourself first.

The idea it to first search x and y in given linked list. If any of them is not present, then return. While searching for x and y, keep track of current and previous pointers. First change next of previous pointers, then change next of current pointers. Following are C and Java implementations of this approach.

C

```
/* This program swaps the nodes of linked list rather
   than swapping the field from the nodes.*/

#include<stdio.h>
```

```
#include<stdlib.h>

/* A linked list node */
struct node
{
    int data;
    struct node *next;
};

/* Function to swap nodes x and y in linked list by
   changing links */
void swapNodes(struct node **head_ref, int x, int y)
{
   // Nothing to do if x and y are same
   if (x == y) return;

   // Search for x (keep track of prevX and CurrX
   struct node *prevX = NULL, *currX = *head_ref;
   while (currX && currX->data != x)
   {
       prevX = currX;
       currX = currX->next;
   }

   // Search for y (keep track of prevY and CurrY
   struct node *prevY = NULL, *currY = *head_ref;
   while (currY && currY->data != y)
   {
       prevY = currY;
       currY = currY->next;
   }

   // If either x or y is not present, nothing to do
   if (currX == NULL || currY == NULL)
       return;

   // If x is not head of linked list
   if (prevX != NULL)
       prevX->next = currY;
   else // Else make y as new head
       *head_ref = currY;

   // If y is not head of linked list
   if (prevY != NULL)
       prevY->next = currX;
   else  // Else make x as new head
       *head_ref = currX;

   // Swap next pointers
   struct node *temp = currY->next;
   currY->next = currX->next;
   currX->next  = temp;
}

/* Function to add a node at the begining of List */
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 nodes in a given linked list */
void printList(struct node *node)
{
    while(node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Druver program to test above function */
int main()
{
    struct node *start = NULL;

    /* The constructed linked list is:
     1->2->3->4->5->6->7 */
    push(&start, 7);
    push(&start, 6);
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    printf("\n Linked list before calling swapNodes() ");
    printList(start);

    swapNodes(&start, 4, 3);

    printf("\n Linked list after calling swapNodes() ");
    printList(start);

    return 0;
}
```

Run on IDE

Java

```
// Java program to swap two given nodes of a linked list
class LinkedList
{
    Node head; // head of list

    class Node
    {
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
    }
```

```
/* Function to swap Nodes x and y in linked list by
   changing links */
public void swapNodes(int x, int y)
{
    // Nothing to do if x and y are same
    if (x == y) return;

    // Search for x (keep track of prevX and CurrX)
    Node prevX = null, currX = head;
    while (currX != null && currX.data != x)
    {
        prevX = currX;
        currX = currX.next;
    }

    // Search for y (keep track of prevY and currY)
    Node prevY = null, currY = head;
    while (currY != null && currY.data != y)
    {
        prevY = currY;
        currY = currY.next;
    }

    // If either x or y is not present, nothing to do
    if (currX == null || currY == null)
        return;

    // If x is not head of linked list
    if (prevX != null)
        prevX.next = currY;
    else //make y the new head
        head = currY;

    // If y is not head of linked list
    if (prevY != null)
        prevY.next = currX;
    else // make x the new head
        head = currX;

    // Swap next pointers
    Node temp = currX.next;
    currX.next = currY.next;
    currY.next = temp;
}

/* Function to add Node at beginning of list. */
public void push(int new_data)
{
    /* 1. alloc the Node and put the data */
    Node new_Node = new Node(new_data);

    /* 2. Make next of new Node as head */
    new_Node.next = head;

    /* 3. Move the head to point to new Node */
    head = new_Node;
}

/* This function prints contents of linked list starting
   from the given Node */
public void printList()
{
    Node tNode = head;
    while (tNode != null)
    {
```

```
            System.out.print(tNode.data+" ");
            tNode = tNode.next;
        }
    }

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

        /* The constructed linked list is:
            1->2->3->4->5->6->7 */
        llist.push(7);
        llist.push(6);
        llist.push(5);
        llist.push(4);
        llist.push(3);
        llist.push(2);
        llist.push(1);

        System.out.print("\n Linked list before calling swapNodes() ");
        llist.printList();

        llist.swapNodes(4, 3);

        System.out.print("\n Linked list after calling swapNodes() ");
        llist.printList();
    }
}
// This code is contributed by Rajat Mishra
```

Run on IDE

Python

```
# Python program to swap two given nodes of a linked list
class LinkedList(object):
    def __init__(self):
        self.head = None

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

    # Function to swap Nodes x and y in linked list by
    # changing links
    def swapNodes(self, x, y):

        # Nothing to do if x and y are same
        if x == y:
            return

        # Search for x (keep track of prevX and CurrX)
        prevX = None
        currX = self.head
        while currX != None and currX.data != x:
            prevX = currX
            currX = currX.next

        # Search for y (keep track of prevY and currY)
```

```
        prevY = None
        currY = self.head
        while currY != None and currY.data != y:
            prevY = currY
            currY = currY.next

        # If either x or y is not present, nothing to do
        if currX == None or currY == None:
            return
        # If x is not head of linked list
        if prevX != None:
            prevX.next = currY
        else: #make y the new head
            head = currY

        # If y is not head of linked list
        if prevY != None:
            prevY.next = currX
        else: # make x the new head
            head = currX

        # Swap next pointers
        temp = currX.next
        currX.next = currY.next
        currY.next = temp

    # Function to add Node at beginning of list.
    def push(self, new_data):

        # 1. alloc the Node and put the data
        new_Node = self.Node(new_data)

        # 2. Make next of new Node as head
        new_Node.next = self.head

        # 3. Move the head to point to new Node
        self.head = new_Node

    # This function prints contents of linked list starting
    # from the given Node
    def printList(self):
        tNode = self.head
        while tNode != None:
            print tNode.data,
            tNode = tNode.next

# Driver program to test above function
llist = LinkedList()

# The constructed linked list is:
# 1->2->3->4->5->6->7
llist.push(7)
llist.push(6)
llist.push(5)
llist.push(4)
llist.push(3)
llist.push(2)
llist.push(1)
print "Linked list before calling swapNodes() "
llist.printList()
llist.swapNodes(4, 3)
print "\nLinked list after calling swapNodes() "
llist.printList()

# This code is contributed by BHAVYA JAIN
```

Run on IDE

Output:

```
Linked list before calling swapNodes() 1 2 3 4 5 6 7
Linked list after calling swapNodes() 1 2 4 3 5 6 7
```

Optimizations: The above code can be optimized to search x and y in single traversal. Two loops are used to keep program simple.

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

23 Comments Category: Linked Lists

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
- Generic Linked List in C

(Login to Rate and Mark)

3 Average Difficulty : **3/5.0**
Based on **9** vote(s)

Add to TODO List
Mark as DONE

Like Share 2 people like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

23 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

Amit gupta • 14 days ago
C# Code (with optimized solution): http://ideone.com/NXDE0E

@Geek4Geek: Kindly enable .Net code editor.

• Reply • Share ›

dbcoder • 16 days ago
can we use doubly linked list?

• Reply • Share ›

Rajan Kalra • 23 days ago
Program with finding both nodes to be swapped in single traversal and thus better time complexity!
http://ideone.com/4E3RKt

• Reply • Share ›

peshavar_rayzyada • a month ago
void swapNodes(struct node **head, int x, int y)

{

struct node *xp=*head;

struct node *yp=*head;

while(xp->data!=x)

xp=xp->next;

while(yp->data!=y)

yp=yp->next;

int pol=xp->data;

int sol=yp->data;

xp->data=sol;

yp->data=pol;

}

will this not work>>everytime??? assumed that data is there in linked list

• Reply • Share ›

devdutt jiwan • a month ago

what is wrong in it? this code is unable to swap first node with any other node

void swap(node** head,int x,int y)

{

if(x==y)

{

return;

}

int i(1),j(1);

node* prex= NULL;

node* prey = NULL;

node* temp;

see more

• Reply • Share ›

Saurabh • a month ago

http://code.geeksforgeeks.org/...

• Reply • Share ›

bhuvi • 2 months ago

```
struct node *prevX = NULL, *currX = NULL;

struct node *prevY = NULL, *currY = NULL;

struct node* temp1 = *head_ref;

while(temp1)

{
optimized to search x and y in single traversal
if(temp1->data == x)

{

currX = temp1;

}

if(temp1->data == y)
```

see more

• Reply • Share ›

Ravi • 3 months ago

I think its simpler to insert a dummy node between X and Y if they are adjacent and proceed as usual

• Reply • Share ›

Adithya G • 3 months ago

Inserting dummy nodes at the start and end of the list would take care if one or both of the elements happened to be a head/tail,right?

• Reply • Share ›

Shubham Sharma • 3 months ago

in line while (currX && currX->data != x)

i mean when currX->data!=x is sufficient so y v have written currX what does it represent

• Reply • Share ›

Shiju T ➚ Shubham Sharma • 3 months ago

If currX is NULL, program will crash if we access currX->data

1 • Reply • Share ›

abhishek chattopadhyay • 6 months ago

Consider the following solution. I believe this would be faster. Let me know what you guys

think.

If the call is made like this
swapNodes(&start, m,n)

then take two pointers (x and y) to the structure and set them as
x = m-1 th position
y = n-1 th position
in the linked list.

operation 1:
x->next = x->next ^ y->next
y->next = x->next ^ y->next
x->next = x->next ^ y->next

operation 2:
x = x->next
y = y->next

operation 3:
repeat operation 1

1. Solution works for adjacent nodes also
2. This solution doesn't work if one of the nodes is a head node. Some more tweaking would be needed .

• Reply • Share ›

shacha • 6 months ago

to handle adjacent node we need to check a condition in swap
struct node *temp = currY->next;
if(currX=prevY)
currY->next = currX;
else
currY->next = currX->next;
currX->next = temp;

• Reply • Share ›

Mitari • 8 months ago

Gurudatha,i think it works for adjacent nodes also,see listnode 1-> becomes Y,n X->X..then,after swapping, X->2 and Y->X..finaly 1->Y->X->2

• Reply • Share ›

Gurudatha • 9 months ago

The case of two nodes being adjacent is not taken care in the example.
Consider linked list like this headnode->listnode1->X->Y->listnode2->NULL.

• Reply • Share ›

dharmendra kumar • 9 months ago

Dear Sir ,

I am a regular reader of your blog from long time , i had also started my blog www.mysmartsupport.com and using , godaddy hosting , but the problem is that my sites load very slowly , kindly suggest me which hosting service is best and fast , and please also post a article on how to migrate your blog from one hosting company to another.

Thank you very very much sir

• Reply • Share ›

erol yeniaras • 9 months ago

Nice code! I could not see where you handled the adjacent inputs though.

• Reply • Share ›

erol yeniaras • 9 months ago

Here is c++ code. a little different approach than above:
http://ideone.com/ez6pZ5

• Reply • Share ›

Jongi • 9 months ago

I dont think so this is handling adjacent nodes.

• Reply • Share ›

varun • 9 months ago

Java version of solution, similar to the one described above

http://ideone.com/5xdv7r

• Reply • Share ›

Kerem Sahin • 9 months ago

Solution by using sentinel nodes:

http://ideone.com/RX44qe

• Reply • Share ›

Rishabh Rusia • 9 months ago

```
// If x is not head of linked list
if (prevX != NULL)
prevX->next = currY;
else // Else make y as new head
*head_ref = currY;
```

```
// If y is not head of linked list
if (prevY != NULL)
prevY->next = currX;
else // Else make x as new head
*head_ref = currX;
```

can u xplain the meaning of this part ?? anyone ??

• Reply • Share ›

Rohit Kujur Rishabh Rusia • 9 months ago

this block checks if X or Y are the head nodes being swapped...
if they are not then it makes the prevX or prevY points to currY or currX respectively...

if any of them is the head node then the pointer to head node needs to be changed(head_ref).... coz after swap new head will become X if Y was earlier head and Y if X was earlier head

1 • Reply • Share ›