

GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

Login/Register

- [Home](#)
- [Algorithms](#)
- [DS](#)
- [GATE](#)
- [Interview Corner](#)
- [Q&A](#)
- [C](#)
- [C++](#)
- [Java](#)
- [Books](#)
- [Contribute](#)
- [Ask a Q](#)
- [About](#)

[Array](#)

[Bit Magic](#)

[C/C++](#)

[Articles](#)

[GFacts](#)

[Linked List](#)

[MCQ](#)

[Misc](#)

[Output](#)

[String](#)

[Tree](#)

[Graph](#)

Function to check if a singly linked list is palindrome

Given a singly linked list of characters, write a function that returns true if the given list is palindrome, else false.



METHOD 1 (Use a Stack)

A simple solution is to use a stack of list nodes. This mainly involves three steps.

- 1) Traverse the given list from head to tail and push every visited node to stack.
- 2) Traverse the list again. For every visited node, pop a node from stack and compare data of popped node with currently visited node.
- 3) If all nodes matched, then return true, else false.

Time complexity of above method is $O(n)$, but it requires $O(n)$ extra space. Following methods solve this with constant extra space.

METHOD 2 (By reversing the list)

This method takes $O(n)$ time and $O(1)$ extra space.

- 1) Get the middle of the linked list.
- 2) Reverse the second half of the linked list.
- 3) Check if the first half and second half are identical.
- 4) Construct the original linked list by reversing the second half again and attaching it back to the first half

To divide the list in two halves, method 2 of [this](#) post is used.

When number of nodes are even, the first and second half contain exactly half nodes. The challenging thing in this method is to handle the case when number of nodes are odd. We don't want the middle node as part of any of the lists as we are going to compare them for equality. For odd case, we use a separate variable 'midnode'.

```

/* Program to check if a linked list is palindrome */
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

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

void reverse(struct node**);
bool compareLists(struct node*, struct node *);

/* Function to check if given linked list is
palindrome or not */

```

```

bool isPalindrome(struct node *head)
{
    struct node *slow_ptr = head, *fast_ptr = head;
    struct node *second_half, *prev_of_slow_ptr = head;
    struct node *midnode = NULL; // To handle odd size list
    bool res = true; // initialize result

    if (head!=NULL && head->next!=NULL)
    {
        /* Get the middle of the list. Move slow_ptr by 1
           and fast_ptr by 2, slow_ptr will have the middle
           node */
        while (fast_ptr != NULL && fast_ptr->next != NULL)
        {
            fast_ptr = fast_ptr->next->next;

            /*We need previous of the slow_ptr for
              linked lists with odd elements */
            prev_of_slow_ptr = slow_ptr;
            slow_ptr = slow_ptr->next;
        }

        /* fast_ptr would become NULL when there are even elements in list.
           And not NULL for odd elements. We need to skip the middle node
           for odd case and store it somewhere so that we can restore the
           original list*/
        if (fast_ptr != NULL)
        {
            midnode = slow_ptr;
            slow_ptr = slow_ptr->next;
        }

        // Now reverse the second half and compare it with first half
        second_half = slow_ptr;
        prev_of_slow_ptr->next = NULL; // NULL terminate first half
        reverse(&second_half); // Reverse the second half
        res = compareLists(head, second_half); // compare

        /* Construct the original list back */
        reverse(&second_half); // Reverse the second half again
        if (midnode != NULL) // If there was a mid node (odd size case) whi
                               // was not part of either first half or second
        {
            prev_of_slow_ptr->next = midnode;
            midnode->next = second_half;
        }
        else prev_of_slow_ptr->next = second_half;
    }
    return res;
}

/* Function to reverse the linked list Note that this

```

```
function may change the head */
void reverse(struct node** head_ref)
{
    struct node* prev = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
}

/* Function to check if two input lists have same data*/
bool compareLists(struct node* head1, struct node *head2)
{
    struct node* temp1 = head1;
    struct node* temp2 = head2;

    while (temp1 && temp2)
    {
        if (temp1->data == temp2->data)
        {
            temp1 = temp1->next;
            temp2 = temp2->next;
        }
        else return 0;
    }

    /* Both are empty reurn 1*/
    if (temp1 == NULL && temp2 == NULL)
        return 1;

    /* Will reach here when one is NULL
    and other is not */
    return 0;
}

/* Push a node to linked list. Note that this function
changes the head */
void push(struct node** head_ref, char 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 pochar to the new node */
(*head_ref)    = new_node;
}

// A utility function to print a given linked list
void printList(struct node *ptr)
{
    while (ptr != NULL)
    {
        printf("%c->", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}

/* Drier program to test above function*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;
    char str[] = "abacaba";
    int i;

    for (i = 0; str[i] != '\0'; i++)
    {
        push(&head, str[i]);
        printList(head);
        isPalindrome(head)? printf("Is Palindrome\n\n"):
                           printf("Not Palindrome\n\n");
    }

    return 0;
}

```

Output:

a->NULL
Palindrome

b->a->NULL
Not Palindrome

a->b->a->NULL
Is Palindrome

c->a->b->a->NULL
Not Palindrome

a->c->a->b->a->NULL
Not Palindrome

b->a->c->a->b->a->NULL
Not Palindrome

a->b->a->c->a->b->a->NULL
Is Palindrome

Time Complexity $O(n)$

Auxiliary Space: $O(1)$

METHOD 3 (Using Recursion)

Use two pointers left and right. Move right and left using recursion and check for following in each recursive call.

- 1) Sub-list is palindrome.
- 2) Value at current left and right are matching.

If both above conditions are true then return true.

The idea is to use function call stack as container. Recursively traverse till the end of list. When we return from last NULL, we will be at last node. The last node to be compared with first node of list.

In order to access first node of list, we need list head to be available in the last call of recursion. Hence we pass head also to the recursive function. If they both match we need to compare (2, n-2) nodes. Again when recursion falls back to (n-2)nd node, we need reference to 2nd node from head. We advance the head pointer in previous call, to refer to next node in the list.

However, the trick in identifying double pointer. Passing single pointer is as good as pass-by-value, and we will pass the same pointer again and again. We need to pass the address of head pointer for reflecting the changes in parent recursive calls.

Thanks to [Sharad Chandra](#) for suggesting this approach.

```
// Recursive program to check if a given linked list is palindrome
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
```

```
/* Link list node */
struct node
{
    char data;
    struct node* next;
};
```

```
// Initial parameters to this function are &head and head
bool isPalindromeUtil(struct node **left, struct node *right)
{
```

```
    /* stop recursion when right becomes NULL */
    if (right == NULL)
        return true;
```

```
    /* If sub-list is not palindrome then no need to
```

```

        check for current left and right, return false */
bool isp = isPalindromeUtil(left, right->next);
if (isp == false)
    return false;

/* Check values at current left and right */
bool isp1 = (right->data == (*left)->data);

/* Move left to next node */
*left = (*left)->next;

return isp1;
}

// A wrapper over isPalindromeUtil()
bool isPalindrome(struct node *head)
{
    isPalindromeUtil(&head, head);
}

/* Push a node to linked list. Note that this function
changes the head */
void push(struct node** head_ref, char 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;
}

// A utility function to print a given linked list
void printList(struct node *ptr)
{
    while (ptr != NULL)
    {
        printf("%c->", ptr->data);
        ptr = ptr->next;
    }
    printf("NULL\n");
}

/* Driver program to test above function*/
int main()
{
    /* Start with the empty list */

```

```

struct node* head = NULL;
char str[] = "abacaba";
int i;

for (i = 0; str[i] != '\0'; i++)
{
    push(&head, str[i]);
    printList(head);
    isPalindrome(head)? printf("Is Palindrome\n\n"):
                       printf("Not Palindrome\n\n");
}

return 0;
}

```

Output:

a->NULL
Not Palindrome

b->a->NULL
Not Palindrome

a->b->a->NULL
Is Palindrome

c->a->b->a->NULL
Not Palindrome

a->c->a->b->a->NULL
Not Palindrome

b->a->c->a->b->a->NULL
Not Palindrome

a->b->a->c->a->b->a->NULL
Is Palindrome

Time Complexity: $O(n)$

Auxiliary Space: $O(n)$ if Function Call Stack size is considered, otherwise $O(1)$.

Please comment if you find any bug in the programs/algorithms or a better way to do the same.

Related Topics:

- [Clone a linked list with next and random pointer | Set 2](#)
- [Given a linked list of line segments, remove middle points](#)
- [Construct a Maximum Sum Linked List out of two Sorted Linked Lists having some Common nodes](#)
- [Given a linked list, reverse alternate nodes and append at the end](#)
- [Pairwise swap elements of a given linked list by changing links](#)
- [Self Organizing List | Set 1 \(Introduction\)](#)
- [Merge a linked list into another linked list at alternate positions](#)

- [QuickSort on Singly Linked List](#)

Like 11

Tweet 0

g+1 1

Writing code in comment? Please use ideone.com and share the link here.

166 Comments

GeeksforGeeks

 Login ▼ Recommend Share

Sort by Newest ▼



Join the discussion...

**THE SAINT** • 2 days ago

Recursive Solution:

```
int palindrome = 1;
int main()
{
    struct node *q;
    .
    .
    q = head;
    if(palindrome_rec(head))
        printf("List is palindrome.\n");
    else
        printf("List is not palindrome.\n");
    return 0;w
}

int palindrome_rec(struct node *p)
{
```

[see more](#) |  • Reply • Share ›**prk** • 9 days ago

To do it recursively in java, you can use a wrapper class to store left node and boolean result at each level. Like this:

```
class NodeWrapper{

    public boolean result;

    public Node left;

    public NodeWrapper(boolean result,Node left){
```

```

this.result=result;

this.left=left;

}

}

public boolean checkPalindrome(Node head){

```

[see more](#)

^ | v • Reply • Share ›



Rohit Singla • 11 days ago

in method 3, I think the comparison is going twice.

If the list is a->b->c->b->a->NULL,

first a is compared with a

then b is compared with b

then c is compared with c

then b is again compared with b

then a is again compared with a

Can we stop this 2nd time comparison by using this condition:-

if(*left == right || (*left)->next == right)

i.e., for odd and even length palindrome cases????

^ | v • Reply • Share ›



Aditya Goel • 15 days ago

Method 1(Stack) using recursion -

```

int isPalindrome(struct node *head)
{
    static struct node *ptr=head;
    if(head==NULL)
        return 1;
    if(!isPalindrome(head->next))
        return 0;
    if(head->next){
        if(head->next->data!=ptr->data)
            return 0;
        else {
            ptr=ptr->next;
            return 1;
        }
    }
}

```

```
}
}
```

Ideone link - <http://ideone.com/O4RHwn>

Note - Compile it as cpp program and call this function only once in the program as I'm using static variables.

Edit - Just noted that similar approach is already been used in method 3. :D

^ | v • Reply • Share ›



Vamos • 19 days ago

// Modified isPalindromeUtil method with just one boolean check :)

```
bool isPalindromeUtil(struct node **left, struct node *right)
{
    if(!right) return true;
    bool isp = isPalindromeUtil(left, right->next) && (right->data == (*left)->data);
    if(!isp) return false;
    *left = (*left)->next;
    return true;
}
```

2 ^ | v • Reply • Share ›



Ok → Vamos • 19 days ago

Thanks a ton!! :)

1 ^ | v • Reply • Share ›



shellk007 • 21 days ago

- 1.Take two pointers: single_step, Double step.
- 2.single_step takes one step and pushes value on stack. double_step takes two steps.
- 3.when double_step reaches end,single_step reaches mid.
- 4.Now stop pushing and start popping stack and keep single_step moving.
- 5.compare single_step and popped element.
- 6.if all same, return true else false.

space: $O(n/2)$

time: $O(n)$

P.S.: haven't coded it yet, please rectify the mistakes.

Thanks,

SK

^ | v • Reply • Share ›



guest → shellk007 • 8 days ago



bt if in case we have string like aaa then hw will it wrk ?

1 ^ | v • Reply • Share ›



shellk007 → guest • 7 days ago

Right... the above mentioned work only for list having even number of nodes.... we can keep a count of nodes also... if count is odd then middle element is not compared... rest is the same...

^ | v • Reply • Share ›



Guest • 25 days ago

If all algorithms discussed above are taking $O(n)$ time, can we do this by the method suggested below. It seems easier-

1. take two pointers i j , one pointing to the start of the list and other pointing to the end of the list.
2. move these one by one, and compare values pointed by these pointers. if at any stage there is a difference in the pointed values, break
3. continue till $i \leq j$

Please comment if you find any flaw in my approach

^ | v • Reply • Share ›



Raj Chopra → Guest • 25 days ago

As question mentions that It is a Singly Linked List. It is not possible to move back from the second pointer (From 'end' towards 'start' as you have mentioned in your approach).

1 ^ | v • Reply • Share ›



rahul singh • a month ago

```
void palindrome(struct node **head_ref)
{
    struct node *slow=*head_ref;
    struct node *fast=*head_ref;
    struct node *current=*head_ref;
    struct node *mid=NULL;
    int flag=1;
    while(slow!=NULL&&fast!=NULL&&fast->next!=NULL)
    {
        slow=slow->next;
        fast=fast->next->next;
    }
    mid=slow;
    printf("mid data %d",mid->data);
    reverse(&mid->next);
    mid=mid->next;
    while(mid->next!=NULL)
```

{

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) ›**darkprotocol** • a month ago

I would suggest to use while loop instead of stack since most of the cases it leads to stackoverflow when dealt with |length| as big number :\\

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Nikita Chhabra** • a month ago

in case of only 1 element: we should check
if(right==NULL || right->next==NULL) instead of
if(right==NULL)

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Tejwinder** • a month ago

using static variable

Ideas welcome :)

<http://ideone.com/sBqbcU>

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**Ansuraj Khadanga** ➔ Tejwinder • 21 days ago

My implementation using static variable in C -

<http://ideone.com/ndONRD>

I guess this code takes auxiliary space of $O(n/2)$. Is it worth it?

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**dinesh** • 2 months ago

recursion method has flaw.....each time left node is at head position so ans is always not a palindrome..... plz suggest some way except global `_/_`

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**AYUSH KUMAR** • 3 months ago

<http://ideone.com/zmEwrF>

code in c++ of checking palindrome

[^](#) | [v](#) • [Reply](#) • [Share](#) ›**neelabhsingh** • 4 months ago

Java implementation <http://ideone.com/AaiHty>

2 ^ | v • Reply • Share ›



Akshith Bhatia • 4 months ago

a simple method also exist to check if a string is a possible palindrome or not just like tests can be a palindrome(stets or tset)

insert all pairs of characters (ie if aaa is given in a string then only a and a is entered in the string while third a wont be counted) in an array and take the length of the array. 1 st point the array length(2) must be even 2nd original string length minus the new array length must not be greater than 1

if these conditions are met the string can be a palindrome.

^ | v • Reply • Share ›



Ankit Srivastava • 4 months ago

This is the C# implementation which can be written in any language with no ugly double pointer needed and we need not find the length in a separate iteration as well.

```
.
struct LinkedList
{
    public int Data;
    public LinkedList Next;
};

static bool IsPalindrome(LinkedList head)
{
    if (head == null && head.Next == null)
        return true;

    LinkedList first = head;
    LinkedList last = head.Next;

    if (Compare(first, last, 2) == null)
        return false;
```

[see more](#)

^ | v • Reply • Share ›



Rajat Panjwani • 4 months ago

while using Stack,it is inefficient to insert all the n elements into the stack and then pop them up. Just push half elements till the middle element and then continue with the pop operation from middle element till the end of the list.

4 ^ | v • Reply • Share ›



karthik ➔ Rajat Panjwani • 2 months ago

How do u determine the middle of stack here ?

^ | v • Reply • Share ›

**neo** → karthik • 23 days ago

fast and slow ptr concept.push slow ptr elements into stack and when fastptr reaches null it means odd no of elements , so pop element,and then continue to pop until stack is empty or list is empty,may be this vague idea . I think we can build a concrete implementation of it, please correct me if i am wrong

1 ^ | v • Reply • Share ›

**Guest** • 4 months ago

Third method is really nice.Got to learn a new concept.

1 ^ | v • Reply • Share ›

**Guest** • 5 months ago

```
temp = forward = backward = head;
```

```
int i =0;
```

```
while(temp.getNext()!=null){
```

```
if( (i & 1) !=0)
```

```
forward = backward = temp.getNext();
```

```
temp = temp.getNext();
```

```
i++;
```

```
}
```

```
// move forward in front and backward in back direction from center
```

```
while(forward!=null && backward !=null && forward.getData() ==backward.getData()){
```

```
System.out.println("forward is "+forward.getData()+" backward is "+backward.getData());
```

[see more](#)

^ | v • Reply • Share ›

**JavaImplementation** • 5 months ago

```
private boolean checkPalindromeRecursiveUsingStatic(Node end) {
```

```
if(end==null){
```

```
return true;
```

```
}
```

```
boolean flag = checkPalindromeRecursiveUsingStatic(end.getNextNode());
```

```

if(start==null){

return flag;

}

if(start.getNextNode()==end || start==end ||flag==false){

start=null;

return flag;

```

[see more](#)

^ | v • Reply • Share ›



Demon • 5 months ago

@GeeksforGeeks <http://goo.gl/EsSvhZ>

Please pay attention

Method 3: need modification

I just copy and paste the copy in ideone.com

and It giving the wrong OUTPUT (Not same as yours)

For every input it gives output "NOT PALINDROME"

3 ^ | v • Reply • Share ›



Harinder Singh ➔ **Demon** • 4 months ago

in some compilers it'll give that output

just try this

```

bool isPalindrome(struct node *head)
{
return isPalindromeUtil(&head, head);
}

```

1 ^ | v • Reply • Share ›



Guest ➔ **Demon** • 4 months ago

i too got the same results <http://ideone.com/xg3sfy>

^ | v • Reply • Share ›



Zeus ➔ **Demon** • 5 months ago

Hi Demon,

here is my java implementation:-

```

public boolean isPalindrome(Node<item> right) {

```

isPalindromeUtil(right, right)


```

if(right == null)

return true;

boolean isp = isPalindrome(right.next);

if(isp == false) //middle check

return false;

boolean isp1 = (left.item == right.item)? true:false;

left = left.next;

return isp1;

}

```

but with a minor modification i.e i made left as global Node , with this little change everything works fine.

^ | v • Reply • Share ›



coder.girl • 5 months ago

Can someone please explain the bool variable isp here? how does it work?

^ | v • Reply • Share ›



Demon → coder.girl • 5 months ago

isp is checking structure wise (like fist node && last, second && second last and so on)

and isp1 is checking the data inside the nodes.

1 ^ | v • Reply • Share ›



Guest • 5 months ago

@GeeksforGeeks why in Method 2 and Method 3 for the input:- a->NULL coming different Answers?

1 ^ | v • Reply • Share ›



money • 5 months ago

can't we reverse the linked list and compare with the existing one....if the element are same they should be palindrome

^ | v • Reply • Share ›



The_Geek → money • 5 months ago

Method 1, i.e. stack implementation is same as what u want to say.

^ | v • Reply • Share ›

**ntk18** • 5 months ago

I think this is the better implementation :

<http://ideone.com/rkyDBn>

^ | v • Reply • Share ›

**codecrecker** • 6 months ago

#include<stdio.h>

#include<stdlib.h>

typedef struct nd node;

struct nd{

int d;

node *n;

};

node *head,*ptr,*nx;

node *createNode(int d)

{

node *tmp;

[see more](#)

^ | v • Reply • Share ›

**valluri** • 6 months ago

consider the LL Ex : 1234321.

When node with value 4 is reached "isPalindromeUtil" the left and right pointers criss-cross and the test continues. Is there a way this can be emulated.

1 ^ | v • Reply • Share ›

**vee*** • 6 months ago

#include<iostream>

#include<string.h>

#include<cstdlib>

using namespace std;

// structure for linkedlist node

```
struct linkedlist
```

```
{
```

```
int value;
```

```
struct linkedlist * next;
```

```
}*head=NULL;
```

```
// function that manipulate node values and invoked by helper function
```

[see more](#)

^ | v • Reply • Share ›



Sumit Saurabh • 7 months ago

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
typedef struct node{
```

```
char data;
```

```
struct node *next;
```

```
}NODE;
```

```
void append(NODE **head, int data){
```

```
NODE *n = (NODE*)malloc(sizeof(NODE));
```

```
n->data = data;
```

```
n->next = *head;
```

```
*head = n;
```

[see more](#)

^ | v • Reply • Share ›



Neyaz Ahmad • 7 months ago

Method2 : Java Code

<http://ideone.com/QMaDAC>

^ | v • Reply • Share ›



Mohan Raja • 8 months ago

This also can be used to find whether a singly linked list is palindrome or not

<http://ideone.com/Wrmt7O>

Idea:

1. Get the count of nodes
2. Move till $N/2$ or $(N+1)/2 - 1$ depending on even or odd count
3. From $N/2$ onwards push the element in the array.
4. Traverse again from beginning of the list and compare with the array values.
5. Show palindrome or not.

Time Complexity : $O(n)$

^ | v • Reply • Share ›



reeetesh11 → Mohaan Raja • 7 months ago

space complexity(n)

^ | v • Reply • Share ›



Kim Jong-il → Mohaan Raja • 7 months ago

Nothing new, Like method 1.

1 ^ | v • Reply • Share ›



Mohan Raja → Kim Jong-il • 7 months ago

Acceptable.. No need for stack.. Thats it..

1 ^ | v • Reply • Share ›



Harinder Singh → Mohaan Raja • 4 months ago

yes but instead of using a stack you are using an array...just an alternate version for solving it in $O(n)$ space and $O(n)$ time complexity

^ | v • Reply • Share ›



VAIBHAV GUPTA • 8 months ago

Method 3 gives single letter as palindrome i.e. output for "a->NULL" is "Is Palindrome" in third method...

whereas in Method 2 output for "a->NULL" is "Not Palindrome"

so slight correction in either of two output is required.

but i want to know whether single letter is considered a palindrome or not???

^ | v • Reply • Share ›



Kim Jong-il → VAIBHAV GUPTA • 7 months ago

Single letter is always a palindrom.

1 ^ | v • Reply • Share ›



np • 8 months ago

@GeeksforGeeks correct the output

Given:

a->NULL

Not Palindrome

it should be palindrome.....

Program is correct.

^ | v • Reply • Share ›

Load more comments



Subscribe



Add Disqus to your site



Privacy



GeeksforGeeks

Like

93,235 people like GeeksforGeeks.



Feedback social plugin

- [Interview Experiences](#)
- [Advanced Data Structures](#)
- [Dynamic Programming](#)
- [Greedy Algorithms](#)

- [Backtracking](#)
- [Pattern Searching](#)
- [Divide & Conquer](#)
- [Mathematical Algorithms](#)
- [Recursion](#)
- [Geometric Algorithms](#)

• Popular Posts

- [All permutations of a given string](#)
- [Memory Layout of C Programs](#)
- [Understanding “extern” keyword in C](#)
- [Median of two sorted arrays](#)
- [Tree traversal without recursion and without stack!](#)
- [Structure Member Alignment, Padding and Data Packing](#)
- [Intersection point of two Linked Lists](#)
- [Lowest Common Ancestor in a BST.](#)
- [Check if a binary tree is BST or not](#)
- [Sorted Linked List to Balanced BST](#)

Follow @GeeksforGeeks



[Subscribe](#)

• Recent Comments

- [lebron](#)

since the array size is 5, it takes constant...

[K'th Smallest/Largest Element in Unsorted Array | Set 3 \(Worst Case Linear Time\)](#) · 3 hours ago

- [lebron](#)

merge sort

[K'th Smallest/Largest Element in Unsorted Array | Set 3 \(Worst Case Linear Time\)](#) · 3 hours ago

- [Shubham Sharma](#)

You saved my time :)

[Searching for Patterns | Set 2 \(KMP Algorithm\)](#) · 3 hours ago

- [Prakhar](#)

Why so many LOCs, if I'm not wrong (please...

[Largest Sum Contiguous Subarray](#) · 4 hours ago

- [Aayush Gupta](#)

For R4 Q3, Another solution would be to use a...

[Amazon Interview Experience | Set 168](#) · [5 hours ago](#)

- [EigenHarsha](#)

For Power Of 2, We Simply Doing.. var1 =

[Practo Interview Experience | Set 2 \(Off-Campus\)](#) · [5 hours ago](#)

•

@geeksforgeeks, [Some rights reserved](#) [Contact Us!](#)

Powered by [WordPress](#) & [MooTools](#), customized by geeksforgeeks team