

## Union and Intersection of two Linked Lists

Given two Linked Lists, create union and intersection lists that contain union and intersection of the elements present in the given lists. Order of elements in output lists doesn't matter.

Example:

Input:

List1: 10->15->4->20

List2: 8->4->2->10

Output:

Intersection List: 4->10

Union List: 2->8->20->4->15->10

### Method 1 (Simple)

Following are simple algorithms to get union and intersection lists respectively.

*Intersection (list1, list2)*

Initialize result list as NULL. Traverse list1 and look for its each element in list2, if the element is present in list2, then add the element to result.

*Union (list1, list2):*

Initialize result list as NULL. Traverse list1 and add all of its elements to the result.

Traverse list2. If an element of list2 is already present in result then do not insert it to result, otherwise insert.

This method assumes that there are no duplicates in the given lists.

Thanks to [Shekhu](#) for suggesting this method. Following is C implementation of this method.

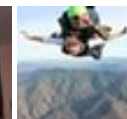
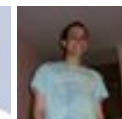
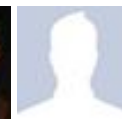
Google™ Custom Search



GeeksforGeeks



53,527 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

```

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

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

/* A utility function to insert a node at the beginning of a linked list */
void push (struct node** head_ref, int new_data);

/* A utility function to check if given data is present in a list */
bool isPresent (struct node *head, int data);

/* Function to get union of two linked lists head1 and head2 */
struct node *getUnion (struct node *head1, struct node *head2)
{
    struct node *result = NULL;
    struct node *t1 = head1, *t2 = head2;

    // Insert all elements of list1 to the result list
    while (t1 != NULL)
    {
        push(&result, t1->data);
        t1 = t1->next;
    }

    // Insert those elements of list2 which are not present in result
    while (t2 != NULL)
    {
        if (!isPresent(result, t2->data))
            push(&result, t2->data);
        t2 = t2->next;
    }

    return result;
}

/* Function to get intersection of two linked lists head1 and head2 */
struct node *getIntersection (struct node *head1, struct node *head2)
{
    struct node *result = NULL;
    struct node *t1 = head1;

    // Traverse list1 and search each element of it in list2. If the element

```

**Build Web Apps in Minutes**

NEBULON SYSTEMS

The ACME FOOD COMPANY

Free Download!

IRON SPEED

## 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

```
// is present in list 2, then insert the element to result
while (t1 != NULL)
{
    if (isPresent(head2, t1->data))
        push (&result, t1->data);
    t1 = t1->next;
}

return result;
}
```

```
/* A utility function to insert a node at the begining of a linked 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;
}
```

```
/* A utility function to print a linked list*/
```

```
void printList (struct node *node)
{
    while (node != NULL)
    {
        printf ("%d ", node->data);
        node = node->next;
    }
}
```

```
/* A utility function that returns true if data is present in linked list
else return false */
```

```
bool isPresent (struct node *head, int data)
{
    struct node *t = head;
    while (t != NULL)
    {
        if (t->data == data)
            return 1;
    }
}
```

# HIGH-PERFORMANCE COMPUTING ON A UNIVERSITY BUDGET

*The quest to find tailored server capabilities on a shoestring*

To conduct successful university research and crunch massive amounts of data, you need the highest-performing hardware on the market. Unfortunately, the price tag of high performance can be daunting for researchers who rely on grants to fund their projects.

There are many off-the-shelf server options available, of course, and your department may even have a standardized, prescribed server.

The best bang for your buck, however, may be to configure a custom server to fit your exact high-performance computing needs.

Unsure of how to get the best performance out of your application? Do you need assistance answering any of these questions?

WHAT NETWORKING INTERFACES ARE NEEDED?

☐ 1GbE ☐ InfiniBand  
☐ 10GbE ☐ Unsure  
☐ 40GbE

HOW MANY CPU CORES DO YOU WANT?

☐ 4 ☐ 16  
☐ 8 ☐ Unsure  
☐ 12


HOW MUCH STORAGE DO YOU WANT?

☐ 1TB ☐ 1PB+  
☐ Unsure

WHAT OPERATING SYSTEM ARE YOU RUNNING?

☐ Linux Variant  
☐ Windows Server  
☐ VMWare  
☐ Unsure

Download To See Entire Infographic

 **SERVERS**DIRECT.

```

        t = t->next;
    }
    return 0;
}

/* Drier program to test above function*/
int main()
{
    /* Start with the empty list */
    struct node* head1 = NULL;
    struct node* head2 = NULL;
    struct node* intersecn = NULL;
    struct node* unin = NULL;

    /*create a linked lits 10->15->5->20 */
    push (&head1, 20);
    push (&head1, 4);
    push (&head1, 15);
    push (&head1, 10);

    /*create a linked lits 8->4->2->10 */
    push (&head2, 10);
    push (&head2, 2);
    push (&head2, 4);
    push (&head2, 8);

    intersecn = getIntersection (head1, head2);
    unin = getUnion (head1, head2);

    printf ("\n First list is \n");
    printList (head1);

    printf ("\n Second list is \n");
    printList (head2);

    printf ("\n Intersection list is \n");
    printList (intersecn);

    printf ("\n Union list is \n");
    printList (unin);

    return 0;
}

```

## Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 44 minutes ago

**Aman** Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

**Sanjay Agarwal** bool

tree::Root\_to\_leaf\_path\_given\_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago


**GOPI GOPINATH** @admin Highlight this

sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

**newCoder3006** If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

AdChoices 

[▶ Linked List](#)

[▶ And Union](#)

[▶ Map Union](#)

```
}
```

Output:

```
First list is
10 15 4 20
Second list is
8 4 2 10
Intersection list is
4 10
Union list is
2 8 20 4 15 10
```

Time Complexity:  $O(mn)$  for both union and intersection operations. Here  $m$  is the number of elements in first list and  $n$  is the number of elements in second list.

### Method 2 (Use Merge Sort)

In this method, algorithms for Union and Intersection are very similar. First we sort the given lists, then we traverse the sorted lists to get union and intersection.

Following are the steps to be followed to get union and intersection lists.

- 1) Sort the first Linked List using merge sort. This step takes  $O(m \log m)$  time. Refer [this post](#) for details of this step.
- 2) Sort the second Linked List using merge sort. This step takes  $O(n \log n)$  time. Refer [this post](#) for details of this step.
- 3) Linearly scan both sorted lists to get the union and intersection. This step takes  $O(m + n)$  time. This step can be implemented using the same algorithm as sorted arrays algorithm discussed [here](#).

Time complexity of this method is  $O(m \log m + n \log n)$  which is better than method 1's time complexity.

### Method 3 (Use Hashing)

*Union (list1, list2)*


Initialize the result list as NULL and create an empty hash table. Traverse both lists one by one, for each element being visited, look the element in hash table. If the element is not present, then insert the element to result list. If the element is present, then ignore it.

AdChoices 

[► Print Union](#)

[► Java Array](#)

[► Intersection](#)

AdChoices 

[► Union of 2 Set](#)

[► Union Public](#)

[► Union Union](#)

### *Intersection (list1, list2)*

Initialize the result list as NULL and create an empty hash table. Traverse list1. For each element being visited in list1, insert the element in hash table. Traverse list2, for each element being visited in list2, look the element in hash table. If the element is present, then insert the element to result list. If the element is not present, then ignore it.

Both of the above methods assume that there are no duplicates.

Time complexity of this method depends on the hashing technique used and the distribution of elements in input lists. In practical, this approach may turn out to be better than above 2 methods.

Source: <http://geeksforgeeks.org/forum/topic/union-intersection-of-unsorted-lists>

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



### Related Tpoics:

---

- [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
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- Swap Kth node from beginning with Kth node from end in a Linked List



6



0



2

Writing code in comment? Please use [ideone.com](https://ideone.com) and share the link here.

25 Comments

GeeksforGeeks

Sort by Newest ▼



Join the discussion...



ashish jaiswal · 5 days ago

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

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

```
struct node;
```

```
void push(struct node**, int);
```

```
void print(struct node*);
```

```
struct node* getunion(struct node*, struct node*);
```

```
struct node* getintersect(struct node*, struct node*);
```

```
typedef struct node
```

\

int data;

struct node\*next:

[see more](#)

^ | v • Reply • Share ›



**Himanshu Dagar** • 3 months ago

In the third method of Using Hash Table what will be its complexity??  
Will it be  $O(m+n)$  or  $O(mn)$ ??

If searching for an element in hash table ,it will take  $O(n)$  time then surely it wil

^ | v • Reply • Share ›



**nanda** → Himanshu Dagar • 3 months ago

No. Intersection is quite clear as it says two different traversals.

Union would be  $O(m+n)$ , it is like traversing both the lists at the same t  
condition would be to check if both are not nulls.

As an alternative, you can traverse the first list, put each in a hash tabl  
traverse the second and output the ones not present in the first.

^ | v • Reply • Share ›



**Nishanth** • 7 months ago

what will be the size of hash array? that is , the integer can be -ve too right?? t

^ | v • Reply • Share ›



**Sumit Poddar** • a year ago

I think there is one more method which will take time complexity of  $O(n)$  where  
which has maximum number of elements. Below is the code in Java.



```
public class SplitList {
```

```
    public static void main(String[] args) {  
        Node n1 = new Node(20, null);  
        Node n2 = new Node(4, n1);  
        Node n3 = new Node(15, n2);  
        Node n = new Node(10, n3);  
  
        Node m1 = new Node(10, null);  
        Node m2 = new Node(2, m1);  
        Node m3 = new Node(4, m2);  
        Node m = new Node(8, m3);  
  
        split(n, m);  
    }
```

[see more](#)

^ | v • Reply • Share ›



**Arun** • a year ago

1 more method to find the intersection:

1. Sort the 1 list with n elements ->  $O(n \log n)$ .
  2. Use binary search for finding each element of list 2(m) in the sorted list1 ->
- Hence we get the results in overall  $O((m+n) \log n)$  which is better than the Met

^ | v • Reply • Share ›



**eragon** → Arun • 2 months ago

And how do you do a logn binary search on a singly linked list?

^ | v • Reply • Share ›



**gaurav** → Arun • a year ago

I think we can get union also by this method with little modification

Initialize union with first list  
initialize intersection with null

while performing binary search -->  
1.element is there in first list-->add to intersection list  
2.element not present-->add to union list

correct if i am wrong..

^ | v • Reply • Share ›



**Gupta** → gaurav • 8 months ago

I think it is also asymptotically better than above merge sort method which has less number of nodes, by checking count nodes 1 and 2 will become  $O((m+n)\log n)$ , where  $n < m$ , which is better. please correct me if i am wrong..>

^ | v • Reply • Share ›



**linux.kindle** • 2 years ago

how abt. we maintain a BST for implementing intersection of LLs? We can then

^ | v • Reply • Share ›



**Rediff** • 2 years ago

/\* Paste your code here (You may **delete** these lines **if not** writing code)

```
public class UnionIntersection{

    public static Node unionOfTwoLists ( Node a, Node b){

        HashMap commons = new HashMap();

        Node union = null;
```

```

while(a!=null){
    if(union == null){
        union = new Node(a.data);
        commons.put(a.data, null);
    }

    if(!commons.containsKey(a.data) && union != null)
    {
        commons.put(a.data, null);
    }
}

```

[see more](#)

^ | v • Reply • Share ›



**GeeksforGeeks** • 2 years ago

@Shekhu: The method 1 assumes that there are no duplicates in the input list function to following to handle duplicates.

```

/* Function to get intersection of two linked lists head1 and head2
struct node *getIntersection (struct node *head1, struct node *head2)
{
    struct node *result = NULL;
    struct node *t1 = head1;

    while (t1 != NULL)
    {
        // Note the second condition
        if (isPresent(head2, t1->data) && !isPresent(result, t1->data)
            push (&result, t1->data);
        t1 = t1->next;
    }

    return result;
}

```

```
}
```

^ | v • Reply • Share ›



**Shekhu** • 2 years ago

The first Method given above would not work when the first list is having duplic

e.g.

List1={1->4->5->1->4}

List2={1->4->9->78}

Intersection result would be {1->4->1->4}

whereas it should have been {1->4} only.

^ | v • Reply • Share ›



**Ritesh** ➔ Shekhu • 2 years ago

Dude ,when they are talking about union and intersection assume that a set.

```
/* Paste your code here (You may delete these lines if not wri
```

^ | v • Reply • Share ›



**Nitin Gupta** • 2 years ago

Well i have another solution for this....

First traverse First linked list and build a binary tree (say it BIF) ( for duplicate element )

complexity  $O(n \log n)$

then traverse second list ( complexity  $O(n)$  ) and and take each element of Set tree...if it found then put it in a Intersection Linked and if not found then connect it to null  
complexity  $O(n \log n)$

On Result you have a Intersection Linked List and A BIF Tree which is Your Ur

Tree..... ( Note when you traverse whole BIF tree then delete it)

Total complexity  $O(n \log n)$

This technique will also handle duplicate element . Using this technique you ca

```
/* Paste your code here (You may delete these lines if not writing co
```

```
/* Paste your code here (You may delete these lines if not writing co
```

^ | v • Reply • Share ›



**coder** → Nitin Gupta • 2 years ago

there is one problem if input is like below

List 1 :- 4 6 8 9

List 2 :- 5 2 2 10

Now as you said if element is not found then add to the BT of the 1st Li  
the Binary Tree so your BT will have 4 6 8 9 2 now for the next 2 of the  
and that will be added to the intersection (this is wrong) because inters  
so better unmatched elements should be added to BT only after list 2 €

^ | v • Reply • Share ›



**anonymous** → Nitin Gupta • 2 years ago

what should be intersection & union list when,

list1 : 1->2->3

list2 : 2->2->2

```
/* Paste your code here (You may delete these lines if not wr
```

^ | v • Reply • Share ›



**anonymous** → Nitin Gupta • 2 years ago



very nice solution. :)

```
/* Paste your code here (You may delete these lines if not wri
```

^ | v • Reply • Share ›



**Shouri** • 2 years ago

Hashing the two sets would yield better time complexity right.

^ | v • Reply • Share ›



**Nishanth** → Shouri • 7 months ago

how to hash when negative integers are there?? what ll be the size of t

^ | v • Reply • Share ›



**GeeksforGeeks** → Shouri • 2 years ago

@Shouri: Thanks for suggesting the hashing method. We have added

^ | v • Reply • Share ›



**bharatkrayra** • 2 years ago

Create a vector by scanning first list and putting elements in sorted order.

Complexity (nlogn)

Now scan second list and put its element in sorted order in the previous vector (no need to keep it sorted this time).

Complexity (nlogn)

Finally you will get first vector in sorted order (UNION)

and second vector may be unsorted (who need the sorted one) (INTERSECTI

Overall complexity = (nlogn)

If you have integer data in list what you can do is sort first and second list in ar  $O(n)$ .

then using merge sort merge procedure to separate out UNION and INTERSE

Complexity =  $O(n)$ ... (Happy Now!)

^ | v • Reply • Share ›



**numid** → bharatkraya • 11 months ago

good one!

^ | v • Reply • Share ›



**gauravjain** • 2 years ago

I think there is some mistake above in the example. You have swapped union |

```
/* Paste your code here (You may delete these lines if not writing c
```

^ | v • Reply • Share ›



**GeeksforGeeks** → gauravjain • 2 years ago

@gauravjain: Thanks for pointing this out. We have updated the post.

^ | v • Reply • Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

**Contact Us!**

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team