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 elments in output lists doesn't matter.

Example:

```
Input:
   List1: 10->15->4->20
   lsit2:  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.

```
#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 begining of a linked list*/
void push (struct node** head_ref, int new_data);

/* A utilty function to chec 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 list
    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
    // 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 utilty 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;
        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;
}
```

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(mLogm) time. Refer this post for details of this step.
2) Sort the second Linked List using merge sort. This step takes O(nLogn) 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(mLogm + nLogn) 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.

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 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

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

Rekha Hindwar • 3 days ago

we can improve the complexity... by traversing both list simultaneously it will take o(m) or o(n) whichever is bigger instead of o(m+n).

Karthik Sagar • 2 months ago

In this programme union will fail if there are repeated elements in first list

lovey • 3 months ago

union and intersection of two list when we have duplicates then use visited array concept i.e. Hashing

ideone link of the code:http://ideone.com/4guGQ8

温发琥 • 3 months ago

here's a simple example of method 2, a, b are the head nodes of 2 sorted linked list

```
void inter(Node* a, Node* b, struct Linkedlist* res)

{

if(a!=NULL&&b!=NULL)

{

if(a->data == b->data)

{
```

```
nodePushBack(res, a->data);

inter(a->next, b->next, res);

}

else if(a->data < b->data)
```

see more

• Reply • Share ›

random_dude • 5 months ago

Third method can be easily modified to work with duplicates as well. Instead of storing whether the element is present in the hash, store the number of occurrences of each element...

2 • Reply • Share ›

xxxxx • 5 months ago

<script>alert(1)</script>

• Reply • Share ›

Manraj Singh ➦ xxxxx • 3 months ago

Fail.

• Reply • Share ›

super • 5 months ago

please tell me how hashing work

• Reply • Share ›

Jun • 8 months ago

can anybody please share implementation of method3

• Reply • Share ›

Abhinav Bhardwaj ➦ Jun • 8 months ago

This is Java implementation of Intersection of two Linked List using Hashing

```
public Node findIntersection(Node head1, Node head2) {
HashSet<integer> myhash = new HashSet<integer>();

Node firstCurrent = head1;
Node secondCurrent = head2;
Node IntersectionHead = null;

Node prev = null;
while (firstCurrent != null) {
myhash.add(firstCurrent.data);
```

```
firstCurrent = firstCurrent.next;
}
while (secondCurrent != null) {
if (myhash.contains(secondCurrent.data)) {
Node newNode = new Node(secondCurrent.data);
if (prev == null)
IntersectionHead = newNode;
```

see more

• Reply • Share ›

pradeep kumar • 9 months ago

why do we use these type of assignment in many functions:-

struct node *t1 = head1, *t2 = head2;

And we start use t1 and t2.....plz tell me.....

• Reply • Share ›

GOPI GOPINATH pradeep kumar • 9 months ago

We are assigning the heads of the lists to temporary pointers in order to traverse the list, yet preserving the heads of the list .

• Reply • Share ›

ashish jaiswal • 10 months 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

• Reply • Share ›

ashish jaiswal ashish jaiswal • 3 months ago

sorry complexity is more..

• Reply • Share ›

Himanshu Dagar • a year 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 will be done in O(mn) .So is it true??

• Reply • Share ›

nanda Himanshu Dagar • a year 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 time with in one single loop, the loop condition would be to check if both are not nulls.

As an alternative, you can traverse the first list, put each in a hash table and output all of them. And then traverse the second and output the ones not present in the first.

1 • Reply • Share ›

ryan nanda • 6 months ago

it will be O(max(m,n))

• Reply • Share ›

Nishanth • a year ago

what will be the size of hash array? that is , the integer can be -ve too right?? then how can we hash it??

• Reply • Share ›

Sumit Poddar • 2 years ago

I think there is one more method which will take time complexity of O(n) where n is the number of elements in the list 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

• Reply • Share ›

Arun • 2 years ago

1 more method to find the intersection:

1. Sort the 1 list with n elements -> O(nlogn).
2. Use binary search for finding each element of list 2(m) in the sorted list1 -> O(mlogn).
Hence we get the results in overall O((m+n)logn) which is better than the Method 2 suggest above using merge sort.

• Reply • Share ›

Aryaa ➚ Arun • 8 months ago

binary search is not possible with linked list as it needs random access

1 • Reply • Share ›

eragon ➚ Arun • a year ago

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

1 • Reply • Share ›

gaurav ➚ Arun • 2 years 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..

• Reply • Share ›

Gupta ➚ gaurav • 2 years ago

I think it is also asymptotically better than above merge sort method... coz we will select the list to sort which has less number of nodes, by checking

countnodes1 and countnodes2... so time complexity will become O((m+n)logn), where n<m, which="" is="" better="" than="" o(mlogm+nlogn).="" please="" correct="" me="" if="" i="" am="" wrong..="">

• Reply • Share ›

linux.kindle • 3 years ago

how abt. we maintain a BST for implementing intersection of LLs? We can then traverse the tree to get the output.

• Reply • Share ›

Rediff • 3 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

• Reply • Share ›

GeeksforGeeks • 3 years ago

@Shekhu: The method 1 assumes that there are no duplicates in the input lists. You can modify the intersection 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;
}
```

• Reply • Share ›

Shekhu • 3 years ago

The first Method given above would not work when the first list is having duplicate values.

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.

• Reply • Share ›

Ritesh Shekhu • 3 years ago

Dude ,when they are talking about union and intersection assume that the list given will follow the property of a set.

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

• Reply • Share ›

Nitin Gupta • 3 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 , put it on the left of that element)
complexity O(n logn)

then traverse second list (complexity O(n)) and and take each element of Second list and match to element in BIF tree...if it found then put it in a Intersection Linked and if not found then connect that element in BIF tree. Complexity (n logn)
On Result you have a Intersection Linked List and A BIF Tree which is Your Union List ...Build a Linked List using BIF 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 can reduce complexity to O(n)

complexity to O(n).

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

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

• Reply • Share ›

coder Nitin Gupta • 3 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 List so 1st 2 of the List 2 will be added to the Binary Tree so your BT will have 4 6 8 9 2 now for the next 2 of the List 2 element will be found in the BT and that will be added to the intersection (this is wrong) because intersection of both list is {} , not {2}.
so better unmatched elements should be added to BT only after list 2 elements are exhausted

• Reply • Share ›

anonymous Nitin Gupta • 3 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 writing code) */
```

• Reply • Share ›

anonymous Nitin Gupta • 3 years ago

very nice solution. :)

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

• Reply • Share ›

Shouri • 3 years ago

Hashing the two sets would yield better time complexity right.

• Reply • Share ›

Nishanth Shouri • a year ago

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

• Reply • Share ›

GeeksforGeeks ➚ Shouri • 3 years ago

@Shouri: Thanks for suggesting the hashing method. We have added it to the original post.

▲ | ▼ • Reply • Share ›

bharatkrayra • 3 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, if element exists put it into second vector (no need to keep it sorted this time).
Compelexity (nlogn)

Finally you will get first vector in sorted order (UNION)
and second vector may be unsorted (who need the sorted one) (INTERSECTION).

Overall complexity = (nlogn)

If you have integer data in list what you can do is sort first and second list in an array using any sorting technique of O(n).
then using merge sort merge procedure to separate out UNION and INTERSECTION in O(n).
Complexity = O(n)... (Happy Now!)

▲ | ▼ • Reply • Share ›

numid ➚ bharatkrayra • 2 years ago

good one!

▲ | ▼ • Reply • Share ›

gauravjain • 3 years ago

I think there is some mistake above in the example. You have swapped union list and intersection list.

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

▲ | ▼ • Reply • Share ›

GeeksforGeeks ➚ gauravjain • 3 years ago

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

▲ | ▼ • Reply • Share ›

-
  - 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

- acemrek

  We can also do it with keeping parent....

  Given a binary tree, how do you remove all the half nodes? · 3 hours ago

- acemrek

  R1 Q1: http://ideone.com/EUHNGh

  Snapdeal Interview Experience | Set 12 (For Senior Software Developer) · 4 hours ago

- Mrinmay Mukherjee

  Hope it helps :) Please share your suggestions!...

  C++ Programming Language · 4 hours ago

- Praveen Dara

  //Author Praveen Dara (AITP) //counts the...

  Write a function that counts the number of times a given int occurs in a Linked List · 5 hours ago

- Praveen Dara

  //Author Praveen Dara (AITP) // deletion of SLL...

  Write a function to delete a Linked List · 5 hours ago

- Aditya Goel

  Method#4 Traverse array once and find out min,...

  Check if array elements are consecutive | Added Method 3 · 5 hours ago