XOR Linked List – A Memory Efficient Doubly Linked List | Set 2

In the previous post, we discussed how a Doubly Linked can be created using only one space for address field with every node. In this post, we will discuss implementation of memory efficient doubly linked list. We will mainly discuss following two simple functions.

1) A function to insert a new node at the beginning.
2) A function to traverse the list in forward direction.

In the following code, *insert()* function inserts a new node at the beginning. We need to change the head pointer of Linked List, that is why a double pointer is used (See this). Let use first discuss few things again that have been discussed in the previous post. We store XOR of next and previous nodes with every node and we call it npx, which is the only address member we have with every node. When we insert a new node at the beginning, npx of new node will always be XOR of NULL and current head. And npx of current head must be changed to XOR of new node and node next to current head.

printList() traverses the list in forward direction. It prints data values from every node. To traverse the list, we need to get pointer to the next node at every point. We can get the address of next node by keeping track of current node and previous node. If we do XOR of curr->npx and prev, we get the address of next node.

```
/* C/C++ Implementation of Memory efficient Doubly Linked List */
#include <stdio.h>
#include <stdlib.h>

// Node structure of a memory efficient doubly linked list
struct node
{
```


```
    int data;
    struct node* npx;  /* XOR of next and previous node */
};

/* returns XORed value of the node addresses */
struct node* XOR (struct node *a, struct node *b)
{
    return (struct node*) ((unsigned int) (a) ^ (unsigned int) (b));
}

/* Insert a node at the begining of the XORed linked list and makes th
   newly inserted node as head */
void insert(struct node **head_ref, int data)
{
    // Allocate memory for new node
    struct node *new_node  = (struct node *) malloc (sizeof (struct no
    new_node->data = data;

    /* Since new node is being inserted at the begining, npx of new no
       will always be XOR of current head and NULL */
    new_node->npx = XOR(*head_ref, NULL);

    /* If linked list is not empty, then npx of current head node will
       of new node and node next to current head */
    if (*head_ref != NULL)
    {
        // *(head_ref)->npx is XOR of NULL and next. So if we do XOR o
        // it with NULL, we get next
        struct node* next = XOR((*head_ref)->npx,  NULL);
        (*head_ref)->npx = XOR(new_node, next);
    }

    // Change head
    *head_ref = new_node;
}

// prints contents of doubly linked list in forward direction
void printList (struct node *head)
{
    struct node *curr = head;
    struct node *prev = NULL;
    struct node *next;

    printf ("Following are the nodes of Linked List: \n");

    while (curr != NULL)
    {
```


Popular Posts

```
        // print current node
        printf ("%d ", curr->data);

        // get address of next node: curr->npx is next^prev, so curr->
        // will be next^prev^prev which is next
        next = XOR (prev, curr->npx);

        // update prev and curr for next iteration
        prev = curr;
        curr = next;
    }
}

// Driver program to test above functions
int main ()
{
    /* Create following Doubly Linked List
       head-->40<-->30<-->20<-->10   */
    struct node *head = NULL;
    insert(&head, 10);
    insert(&head, 20);
    insert(&head, 30);
    insert(&head, 40);

    // print the created list
    printList (head);

    return (0);
}
```

Output:

```
Following are the nodes of Linked List:
40 30 20 10
```

Note that XOR of pointers is not defined by C/C++ standard. So the above implementation may not work on all platforms.

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

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

12 Comments GeeksforGeeks

Sort by Newest

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

Join the discussion…

neelabhsingh • 4 months ago

It is possible to make memory efficient Linked List in java?

Reply • Share ›

Ashish Saxena • 6 months ago

Is it possible to delete a node from this list if only that node address is known ?

Reply • Share ›

Ekta ➦ Ashish Saxena • 4 months ago

Can u please provide the code for deletion in case u did.. i found on ma
understand it working way!!

Reply • Share ›

rajat6875 • 10 months ago

I tried it this way and it is working
and i tried insertion at end normal way
please tell me if this can have any error in any case.

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

class XorLinkedList{

private:

   int data;
   int npx;
```

```
xorLinkedList(){
}
```

see more

• Reply • Share ›

Rajendra Kumar Roul • a year ago

nice

• Reply • Share ›

Arindam Sanyal • a year ago

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

struct node{
int info;
struct node *next;.
struct node *prev;.
};
struct node * addtoempty(struct node *, int);.
struct node * addtoend(struct node *, int);.
void display(struct node *);.

void main(){
clrscr();
struct node *start=NULL;.
int n, d;
printf("nenter the number of nodes in the doubly linked list...");.
scanf("%d",&n);
```

see more

Nishant • 2 years ago

i think in this ur displaying the reversed list as ur pointers are moving in each ir
points to the start position.

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

• Reply • Share ›

Vikrant • 2 years ago

Will it make a difference if in 7th line of insert() function:

```
(*head_ref)->npx=XOR(newnode,next);
```

next is replaced by (*head_ref)->npx
I think it will be the same.

• Reply • Share ›

Kumar Sukhani • 2 years ago

http://en.wikipedia.org/wiki/X...

• Reply • Share ›

Kamal • 2 years ago

a nice article is also at http://www.ritambhara.in/memor...

• Reply • Share ›

Game • 2 years ago

XOR of two pointer types is an undefined behaviour in C/C++. Just to prove m

page
http://en.wikipedia.org/wiki/X...
and Ctrl-F for "XOR of pointers is not defined in some contexts (e.g., the C lan
Let's make this site a clean place :)
Also, for further info on pointers and pointer arithmetic http://www.eskimo.com

1 ^ | v • Reply • Share ›

GeeksforGeeks ➚ Game • 2 years ago

@Game: Thanks for pointing this out. We will add a note for this.

^ | v • Reply • Share ›