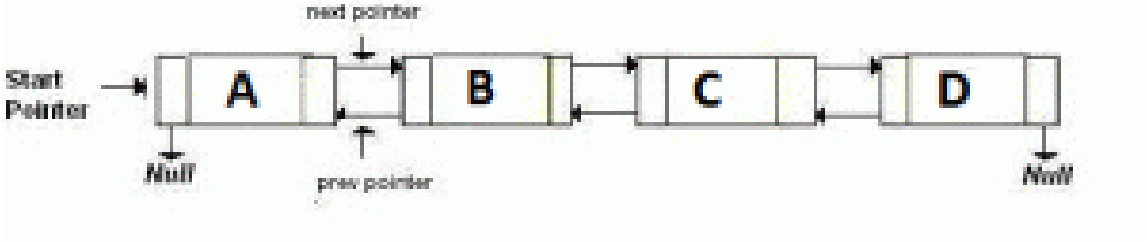
XOR Linked List – A Memory Efficient Doubly Linked List | Set 1

An ordinary Doubly Linked List requires space for two address fields to store the addresses of previous and next nodes. A memory efficient version of Doubly Linked List can be created using only one space for address field with every node. This memory efficient Doubly Linked List is called XOR Linked List or Memory Efficient as the list uses bitwise XOR operation to save space for one address. In the XOR linked list, instead of storing actual memory addresses, every node stores the XOR of addresses of previous and next nodes.

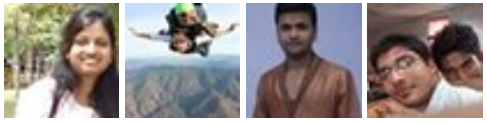


Consider the above Doubly Linked List. Following are the Ordinary and XOR (or Memory Effiecient) representations of the Doubly Linked List.

Ordinary Representation:
Node A:
prev = NULL, next = add(B) // previous is NULL and next is address of B

Node B:
prev = add(A), next = add(C) // previous is address of A and next is address of C

Node C:
prev = add(B), next = add(D) // previous is address of B and next is address of D



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

Node D:
prev = add(C), next = NULL // previous is address of C and next is NULL

XOR List Representation:
Let us call the address variable in XOR representation npx (XOR of next and previous)

Node A:
npx = 0 XOR add(B) // bitwise XOR of zero and address of B

Node B:
npx = add(A) XOR add(C) // bitwise XOR of address of A and address of C

Node C:
npx = add(B) XOR add(D) // bitwise XOR of address of B and address of D

Node D:
npx = add(C) XOR 0 // bitwise XOR of address of C and 0

Traversal of XOR Linked List:
We can traverse the XOR list in both forward and reverse direction. While traversing the list we need to remember the address of the previously accessed node in order to calculate the next node's address. For example when we are at node C, we must have address of B. XOR of add(B) and *npx* of C gives us the add(D). The reason is simple: npx(C) is "add(B) XOR add(D)". If we do xor of npx(C) with add(B), we get the result as "add(B) XOR add(D) XOR add(B)" which is "add(D) XOR 0″ which is "add(D)". So we have the address of next node. Similarly we can traverse the list in backward direction.

We have covered more on XOR Linked List in the following post.

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

References:
http://en.wikipedia.org/wiki/XOR_linked_list
http://www.linuxjournal.com/article/6828?page=0,0

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

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.

17 Comments GeeksforGeeks

Sort by Newest

devil • 11 months ago

Good havens, does anyone ever code in Java here. !crying

Reply • Share ›

subhin • 11 months ago

Can any one publish java code of above program

Reply • Share ›

devil ➦ subhin • 11 months ago

http://cocoadev.com/wiki/Desig... Some explanation. Just incorporate t

Reply • Share ›

devil ➦ subhin • 11 months ago

Don't worry. I am going to try and come up with it here. Hold on..

Reply • Share ›

Atul • 2 years ago

Somehow I didn't like the "node* next" in the given source code. Since it is the can't it be a simple number? Hence I implemented following.

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

typedef struct node
{
  int val;
  unsigned int pnx;  /* prev, next ptr XOR'ed value */
} NODE;
```

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 43 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

```
NODE *head, *tail;

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

see more

• Reply • Share ›

Sudha • 2 years ago

// Insertion, Deletion and Both direction traversal.

#include
#include

struct node
{
int num;
struct node *ptrdiff;
};

void insert(struct node**,struct node**,struct node*,struct node*);
void displayForward(struct node*,struct node*);
void displayBackward(struct node*,struct node*);
struct node* newnode(int);
struct node *XOR(struct node *, struct node *);
void delete_node(struct node **,struct node **,int);

int main()

see more

code1234 Sudha • 10 months ago

Works very well! Great, thanks! :)

• Reply • Share ›

raj • 3 years ago

```
/* */
[#include
#include
typedef struct node
{
int data;
struct node *npx;
}node;
void createlist(node **p)
{
node *prev=NULL,*next,*current;
int i,j,n,x;
current=*p;
while(1)
{
```

see more

• Reply • Share ›

Yogesh • 3 years ago

But we always need the prev node address in order to traverse from a given n list where we rem the prev node of a current node ptr.

kevindra • 3 years ago

I think there is something wrong with formatting of code. It's taking "" as >

• Reply • Share ›

GeeksforGeeks kevindra • 3 years ago

@kevindra: There seems to be some issue with formatting. We will loc
we have updated the code with pre tags and the code is readable.

• Reply • Share ›

kevindra • 3 years ago

```
#include < iostream >

using namespace std;

struct node{
  int v;
  node *next;
};
node *start = NULL;
node *end = NULL;

node *newNode(int v){
  node *np = new node;
  np->v = v;
  np->next  = NULL;
  return np;
}
```

see more

kevindra • 3 years ago

Here is the working code for insertion and traversal (both directions) in XOR lir

```
#include <iostream>

using namespace std;


struct node{
  int v;
  node *next;
};
node *start = NULL;
node *end = NULL;

node *newNode(int v){
  node *np = new node;
  np->v = v;
```

see more

• Reply • Share ›

ktanay • 3 years ago

//minor typo
npx = add(A) XOR add(C) // bitwise XOR of address of A and address of B

// bitwise XOR of address of A and address of C

• Reply • Share ›

GeeksforGeeks ktanay • 3 years ago

@ktanay: Thanks for pointing this out. We have corrected the typo.

• Reply • Share ›

rajcools • 3 years ago

we are able to save memory but per node time of execution is increasing!!!! ti

• Reply • Share ›

kl rajcools • 3 years ago

struct b
{
int a;
int b;
};

• Reply • Share ›