Design a stack with operations on middle element

How to implement a stack which will support following operations in **O(1) time complexity**?
1) push() which adds an element to the top of stack.
2) pop() which removes an element from top of stack.
3) findMiddle() which will return middle element of the stack.
4) deleteMiddle() which will delete the middle element.
Push and pop are standard stack operations.

The important question is, whether to use a linked list or array for implementation of stack?

Please note that, we need to find and delete middle element. Deleting an element from middle is not O(1) for array. Also, we may need to move the middle pointer up when we push an element and move down when we pop(). In singly linked list, moving middle pointer in both directions is not possible.

The idea is to use Doubly Linked List (DLL). We can delete middle element in O(1) time by maintaining mid pointer. We can move mid pointer in both directions using previous and next pointers.

Following is C implementation of push(), pop() and findMiddle() operations. Implementation of deleteMiddle() is left as an exercise. If there are even elements in stack, findMiddle() returns the first middle element. For example, if stack contains {1, 2, 3, 4}, then findMiddle() would return 2.

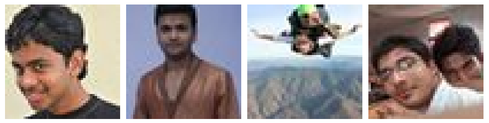
```
/* Program to implement a stack that supports findMiddle() and deleteM
   in O(1) time */
#include <stdio.h>
#include <stdlib.h>

/* A Doubly Linked List Node */
```

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

```c
struct DLLNode
{
    struct DLLNode *prev;
    int data;
    struct DLLNode *next;
};

/* Representation of the stack data structure that supports findMiddle
   in O(1) time.  The Stack is implemented using Doubly Linked List. I
   maintains pointer to head node, pointer to middle node and count of
   nodes */
struct myStack
{
    struct DLLNode *head;
    struct DLLNode *mid;
    int count;
};

/* Function to create the stack data structure */
struct myStack *createMyStack()
{
    struct myStack *ms =
               (struct myStack*) malloc(sizeof(struct myStack));
    ms->count = 0;
    return ms;
};

/* Function to push an element to the stack */
void push(struct myStack *ms, int new_data)
{
    /* allocate DLLNode and put in data */
    struct DLLNode* new_DLLNode =
               (struct DLLNode*) malloc(sizeof(struct DLLNode));
    new_DLLNode->data  = new_data;

    /* Since we are adding at the begining,
      prev is always NULL */
    new_DLLNode->prev = NULL;

    /* link the old list off the new DLLNode */
    new_DLLNode->next = ms->head;

    /* Increment count of items in stack */
    ms->count += 1;

    /* Change mid pointer in two cases
       1) Linked List is empty
```

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

```
       2) Number of nodes in linked list is odd */
    if (ms->count == 1)
    {
         ms->mid = new_DLLNode;
    }
    else
    {
        ms->head->prev = new_DLLNode;

        if (ms->count & 1) // Update mid if ms->count is odd
           ms->mid = ms->mid->prev;
    }

    /* move head to point to the new DLLNode */
    ms->head  = new_DLLNode;
}

/* Function to pop an element from stack */
int pop(struct myStack *ms)
{
    /* Stack underflow */
    if (ms->count  ==  0)
    {
        printf("Stack is empty\n");
        return -1;
    }

    struct DLLNode *head = ms->head;
    int item = head->data;
    ms->head = head->next;

    // If linked list doesn't become empty, update prev
    // of new head as NULL
    if (ms->head != NULL)
        ms->head->prev = NULL;

    ms->count -= 1;

    // update the mid pointer when we have even number of
    // elements in the stack, i,e move down the mid pointer.
    if (!((ms->count) & 1 ))
        ms->mid = ms->mid->next;

    free(head);

    return item;
}
```

```
// Function for finding middle of the stack
int findMiddle(struct myStack *ms)
{
    if (ms->count  ==  0)
    {
        printf("Stack is empty now\n");
        return -1;
    }

    return ms->mid->data;
}
```

```
// Driver program to test functions of myStack
int main()
{
    /* Let us create a stack using push() operation*/
    struct myStack *ms = createMyStack();
    push(ms, 11);
    push(ms, 22);
    push(ms, 33);
    push(ms, 44);
    push(ms, 55);
    push(ms, 66);
    push(ms, 77);

    printf("Item popped is %d\n", pop(ms));
    printf("Item popped is %d\n", pop(ms));
    printf("Middle Element is %d\n", findMiddle(ms));
    return 0;
}
```

Output:

```
Item popped is 77
Item popped is 66
Middle Element is 33
```

This article is contributed by **Chandra Prakash**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 46 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

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
- Swap Kth node from beginning with Kth node from end in a Linked List
- QuickSort on Doubly Linked List

40 Tweet 3 2

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

23 Comments GeeksforGeeks

Sort by Newest

bartender · 10 days ago

Just an observation, the previous node of the head is pointing to some randorr
createMyStack() function.
So, if you ever have to iterate, do it based on count value and next pointers.

Reply · Share ›

moltu bartender · 10 days ago

right. or i would say it would be best to explicitly point the prev node to l
convention i think, end of Link Lists should be marked by Null pointers.

Reply · Share ›

Gopi Shankar · 2 months ago

```
int deleteMiddle( struct myStack* stack ) {
if( stack->count == 0 ) {
printf( "Stack is Empty" );
return -1;
}

stack->mid->prev->next = stack->mid->next;
stack->mid->mext->prev = stack->mid->prev;

if( stack->count & 1) {
stack->mid = stack->mid->prev;
}
stack->count--;
}
```

Reply · Share ›

venkat · 5 months ago

In push function Dll Node prev pointer is not updated properly, for old nodes

Reply • Share ›

Aditya Raman • 6 months ago

We can use a vector of elements and do all the operations under O(1) without
push= push_back() in vector
pop=erase(vector[vector.size()-1])
mid operations with index vector.size()/2

Reply • Share ›

Yukang → Aditya Raman • 6 months ago

you have to update the vector after deleting mid,
that is not O(1)

Reply • Share ›

Vinodhini • 8 months ago

A function for deleteMiddle():
please correct me if I am wrong.
Algorithm:
1)if the count is 0 then there are no elements so return -1
2)
a)if count is 1, we need to update mid and head as NULL and decrement cour
b)else
we update the mid->prev's next and mid->next's prev pointers provided they e
>next would be null). if the count had been odd, we update mid as mid->next є
count and delete the mid.

```
int deleteMiddle(struct myStack *ms){
if(ms->count==0)
return -1;
if(ms->count==1){
int data=ms->mid->data;
struct DLLNode * temp=ms->mid;
```

ms->mid=NULL;

see more

• Reply • Share ›

tosif khan • 8 months ago

I just came across this article 1 evening before my interview, this question sav
microsoft, cant thank you enough

16 • Reply • Share ›

Pandian • 9 months ago

In push() function, mid should be updated only when count is even. But in cod

1 • Reply • Share ›

Sanjith Sakthivel • 9 months ago

```
/* Just adjust the middle pointer during Push and Pop ( Doubly Linked List Imp
Then you will find the middle element at simple (1) order.
*/

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct link
{
int element;
struct link *prev,*next;.
};
int count;
struct link *start,*end,*mid;.
void createStack()
{
```

count=0;.

see more

^ | ⌄ • Reply • Share ›

ritesh • 10 months ago

the explanation says if stack contains {1, 2, 3, 4}, then findMiddle() would retur

assuming the linked list is 1->2->3->4
the code return mid element as 3 an not 2

^ | ⌄ • Reply • Share ›

Srishti • 10 months ago

In push function, mid pointer should be updated to
ms->mid->next, since we are adding an element, same is the mistake in pop
>mid->prev.
Please correct me if i'm wrong.

^ | ⌄ • Reply • Share ›

skulldude • 11 months ago

In the push() function - mid is being updated only when the length becomes oc
elements in the stack say - {1,2} - mid will be pointing to 2. Now when we quer
as per the description given at the top of the page, getMiddle() should return 1.

So, mid has to be updated when count becomes even and the reverse has to

Similarly delMiddle() function seems to be missing.

^ | ⌄ • Reply • Share ›

coder • 11 months ago

there is a mistake in updating the mid pointer in the pop and push functions.

2 ^ | ⌄ • Reply • Share ›

Avi • 11 months ago

```
void deleteMiddle(struct myStack* ms) {
  if (ms->count == 0) return;
  assert(ms->mid != NULL);
  struct DLLNode* node = ms->mid;
  if (ms->count == 1) {
    assert(node->next == NULL);
    assert(node->prev == NULL);
    ms->head = NULL;
    ms->mid = NULL;
  } else {
    // If count is odd, the mid will now point to prev of mid.
    if (ms->count & 1) {
      assert(node->prev != NULL);
      ms->mid = node->prev;
      ms->mid->next = node->next;
      node->next->prev = ms->mid;
    } else {
```

see more

• Reply • Share ›

Aap NITian Hain • 11 months ago

/* link the old list off the new DLLNode */.

new_DLLNode->next = ms->head;.
instead it should be modified to check this condition.
if(ms->count==0)
new_DLL->next=ms->head;
ms->head=new__DLL;
else

new_DLLNode->next = ms->head;.

^ | ⌄ • Reply • Share ›

GeeksforGeeks • 11 months ago

Thanks for pointing this out. We have updated the code.

^ | ⌄ • Reply • Share ›

Xiaoge Yuan • 11 months ago

So in pop(), head shall be freed at last

^ | ⌄ • Reply • Share ›

Xiaoge Yuan • 11 months ago

there is a bug in pop(), if after pop the stack size is 0 then mid is same as hea
the memory not allocated to this program.

^ | ⌄ • Reply • Share ›

kind-of-beginner • 11 months ago

```
struct myStack *ms =
               (struct myStack*) malloc(sizeof(struct myStack));
// cant this just be 'struct myStack *ms = malloc(sizeof(*ms));' ?

/* in pop, if you want to delete -1, how will you know if you have de
Also for pop, cant we do something like:*/

int pop(struct myStack *ms)
{
  //empty head
  if (!ms->head)
    return -1; //whatever message
```

```
myNode *node = ms->head;

// single item only
```

see more

^ | v • Reply • Share ›

GeeksforGeeks • 11 months ago

@minoz: push() and pop() operations are like standard stack operations. We

^ | v • Reply • Share ›

minoz • 11 months ago

Are push and pop to be done from the middle of the stack?
If they are, the final status in the given example will be
22 44 55 33 11
and findMiddle() should return 55 but your program returns 33.

^ | v • Reply • Share ›

minoz • 11 months ago

Is push and pop done from the middle of the stack?

^ | v • Reply • Share ›

Subscribe Add Disqus to your site