Reverse a stack using recursion

You are not allowed to use loop constructs like while, for..etc, and you can only use the following ADT functions on Stack S:
isEmpty(S)
push(S)
pop(S)

Solution:
The idea of the solution is to hold all values in Function Call Stack until the stack becomes empty. When the stack becomes empty, insert all held items one by one at the bottom of the stack.

For example, let the input stack be

```
    1  <-- top
    2
    3
    4

First 4 is inserted at the bottom.
    4 <-- top

Then 3 is inserted at the bottom
    4 <-- top
    3

Then 2 is inserted at the bottom
    4 <-- top
    3
    2

Then 1 is inserted at the bottom
    4 <-- top
    3
    2
    1
```

So we need a function that inserts at the bottom of a stack using the above given basic stack function.
//**Below is a recursive function that inserts an element at the bottom of a stack.**

```
void insertAtBottom(struct sNode** top_ref, int item)
{
   int temp;
   if(isEmpty(*top_ref))
   {
      push(top_ref, item);
   }
   else
   {

     /* Hold all items in Function Call Stack until we reach end of
       the stack. When the stack becomes empty, the isEmpty(*top_ref)
       becomes true, the above if part is executed and the item is
       inserted at the bottom */
     temp = pop(top_ref);
     insertAtBottom(top_ref, item);

     /* Once the item is inserted at the bottom, push all the
          items held in Function Call Stack */
     push(top_ref, temp);
   }
}
```

//**Below is the function that reverses the given stack using insertAtBottom()**

```
void reverse(struct sNode** top_ref)
{
  int temp;
  if(!isEmpty(*top_ref))
  {

    /* Hold all items in Function Call Stack until we reach end of
     the stack */
    temp = pop(top_ref);
    reverse(top_ref);

    /* Insert all the items (held in Function Call Stack) one by one
       from the bottom to top. Every item is inserted at the bottom */
    insertAtBottom(top_ref, temp);
  }
}
```

//**Below is a complete running program for testing above functions.**

```
#include<stdio.h>
#include<stdlib.h>
#define bool int

/* structure of a stack node */
struct sNode
{
   char data;
   struct sNode *next;
};

/* Function Prototypes */
void push(struct sNode** top_ref, int new_data);
int pop(struct sNode** top_ref);
bool isEmpty(struct sNode* top);
void print(struct sNode* top);

/* Driveer program to test above functions */
int main()
{
  struct sNode *s = NULL;
  push(&s, 4);
  push(&s, 3);
  push(&s, 2);
  push(&s, 1);

  printf("\n Original Stack ");
  print(s);
  reverse(&s);
  printf("\n Reversed Stack ");
  print(s);
  getchar();
}

/* Function to check if the stack is empty */
bool isEmpty(struct sNode* top)
{
  return (top == NULL)? 1 : 0;
}

/* Function to push an item to stack*/
void push(struct sNode** top_ref, int new_data)
{
  /* allocate node */
  struct sNode* new_node =
            (struct sNode*) malloc(sizeof(struct sNode));

  if(new_node == NULL)
  {
     printf("Stack overflow \n");
     getchar();
     exit(0);
  }

  /* put in the data  */
  new_node->data  = new_data;

  /* link the old list off the new node */
  new_node->next = (*top_ref);

  /* move the head to point to the new node */
  (*top_ref)    = new_node;
}

/* Function to pop an item from stack*/
int pop(struct sNode** top_ref)
{
  char res;
  struct sNode *top;

  /*If stack is empty then error */
  if(*top_ref == NULL)
  {
     printf("Stack overflow \n");
     getchar();
     exit(0);
  }
  else
  {
     top = *top_ref;
     res = top->data;
     *top_ref = top->next;
     free(top);
     return res;
  }
}

/* Functrion to pront a linked list */
void print(struct sNode* top)
{
  printf("\n");
  while(top != NULL)
  {
    printf(" %d ", top->data);
    top =  top->next;
  }
}
```

Please write comments if you find any bug in above code/algorithm, or find other ways to solve the same problem.

Related Topics:

- Visa Interview Experience | Set 6 (On-Campus)
- Count number of ways to reach a given score in a game
- Greedy Algorithm for Egyptian Fraction
- Find length of period in decimal value of 1/n
- How to check if an instance of 8 puzzle is solvable?
- How to Implement Forward DNS Look Up Cache?
- Birthday Paradox
- Job Sequencing Problem | Set 1 (Greedy Algorithm)

Tags: Recursion

Like 1 Tweet 0 g+1 1

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

41 Comments **GeeksforGeeks** Login

Recommend Share Sort by Newest

Join the discussion…

Aditya Goel • 3 days ago

It is just simple problem of reversing a link list.

Reply • Share ›

Rahul Ranjan • a month ago

My implementation.....

#include<bits stdc++.h="">

using namespace std;

void reverse(stack<int> &s,queue<int> &q)

{

if(!s.empty())

{

cout<<s.top()<<endl; q.push(s.top());="" s.pop();="" reverse(s,q);="" }="" s.push(q.front());="" q.pop();="" return;="" }="" int="" main()="" {="" stack="" <int=""> s;

queue <int> q;

s.push(1);

see more

• Reply • Share ›

Rahul Ranjan • a month ago

if we store the popped elements from the stack in a queue and pop elements from the front end of the queue recursively till the queue is empty,the same goal will be achieved with lesser time complexity.

Time complexity using a queue = O(n)(to empty the stack)+O(n)(to empty the queue and push elements in reverse order into the stack)=O(n)

Time complexity without a queue = O(n)(to empty the stack)+O(n*(n-1)/2)(to push elements into the stack in the desires order).

Correct me if i am missing any intricate detail of this question?

• Reply • Share ›

Anymosity • 2 months ago

Do we use 2 array doubling in reversing this array stack?

• Reply • Share ›

Ashish Jaiswal • 2 months ago

By swapping data from top of stack to bottom of stack we can reverse data of stack...although does not comply with abpve question but can another solution to question above

http://codepad.org/6AC1Y3GY

• Reply • Share ›

Ashish Jaiswal • 2 months ago

Another approach using without recursion is here

#include<stdio.h>

#include<stdlib.h>

```
typedef struct stack

{

int top2;

unsigned capacity;

int*array;

}Stack;

Stack*createstack(unsigned capacity)

{
```

see more

^ | v • Reply • Share ›

Chirag • 3 months ago

```
ReverseStack(stack *s)
{
int temp;
if(s->isEmpty())
return;
temp = s->pop();
ReverseStack(&s);
s->push(temp);
}
```

Can't we solve problem this way?

2 ^ | v • Reply • Share ›

The_Geek → Chirag • 2 months ago

Dude, check output before posting on GFG, its the same stack, not the reverse one ! Anyway nice try !

3 ^ | v • Reply • Share ›

Rajat Panjwani • 4 months ago

Another way to solve this problem can be as follows:-

```
void swap(top1,top2){
if(top2>top1){
t = Stack[top1];
Stack[top1] = Stack[top2]
Stack[top2] = t
```

top1++
top2--
swap(top1,top2)
}
}

where top1 and top2 are the two ends of the stack initially i.e, top1 starts from 0 and top2 equals the top element index. t is the temporary variable used for swapping.

1 • Reply • Share ›

Ashish Jaiswal Rajat Panjwani • 2 months ago

Thats nice approach but not as efficient as using LL above...still another alternate...good keep it up...bro

• Reply • Share ›

Harshdeep Gupta Rajat Panjwani • 3 months ago

this is wrong because u cannot access arbitrary elements of stack.You can only access its top element.

• Reply • Share ›

Rajat Panjwani Harshdeep Gupta • 3 months ago

I think you need to rectify your concepts related to stacks.
Please note that whether we implement stacks using linked list or using arrays, we can access any member of stack at any time. On the other hand we can only pop the top element of stack.

• Reply • Share ›

neelabhsingh • 5 months ago

Please check I am correct or not?
Here we are doing in reverse() function we are reaching the bottom of stack using recursion. When we reach the bottom of stack we call insertAtBottom() which is also recursive. In this function we pop all elements which are in new stack. Now insert new element and push all elements which are poped. For Example input 1->2->3->4, where top is element is 1. We want like 4->3->2->1. Top element is 4.

First we will pop all element using reverse(). when we reach bottom of stack we will get 4 now call insertAtBottom(), Here top_ref is Null so top will become new element 4.

new stack is 4->

So come again in reverse function now pop element is 3. Again call insertAtBottom(). Now in this function stack is not empty. So we go else part and pop element till bottom now new stack is empty now push 3 in new stack then push 4 in the stack in new stack, when new element is come pop all element then push new element, then push older element due to which top is always points to 4. which we wants.

Reply • Share ›

xyz • 6 months ago

Can't we do it this way: Take two stacks and then push elements into one stack in the order given and then pop them one by one and push into the other stack so the order will be the reverse one...

Reply • Share ›

Nitin Gupta xyz • 5 months ago

Then it would not be inplace

2 • Reply • Share ›

Batman xyz • 6 months ago

One of the stacks here essentially is the function call stack !

Reply • Share ›

Vishal Srivastava • 7 months ago

```
#include<iostream>

#include<stack>

using namespace std;

void insertIntoStack(stack<int> &st, int temp)

{

if(st.empty())

st.push(temp);

else

{

int alt=st.top();

st.pop();
```

see more

2 • Reply • Share ›

still learning • 7 months ago

can someone identify error for me in this code ..please??

link:http://ideone.com/yuWYDZ

• Reply • Share ›

Gaurav Suman still learning • 4 months ago

in main(): instead of reversestack(s) it should be reversestack(&s);

• Reply • Share ›

guest • 7 months ago

why dont we simply solve like reversing a link list using recursion.stack is similar to link list.we just update the start pointer to newnode.

5 • Reply • Share ›

Ashish Jaiswal guest • 2 months ago

Because as quesion says use ADT functions on Stack S:isEmpty(S)
push(S)
pop(S)
only

• Reply • Share ›

Goutham • 7 months ago

I think using another stack of same size would be a better idea, as the complexity would be O(n)
Coming to the space complexity it is O(n)
But even in the above procedure the values in recursive calls were stored in a stack.
So using another stack is a better idea than a recursive function

1 • Reply • Share ›

Ashish Jaiswal Goutham • 2 months ago

Another function call stack apart from reverse() is itself a another stack used to implement above problem....

• Reply • Share ›

guest • 7 months ago

void reverse_stack(stack s)

{
if(temp1==0)
return;

else
{

temp1--; //temp1 is the count of number of elements pushed into the stack

```
s.top--;
reverse_stack(s);
printf("%d\n",s.stk[s.top+1]);
}

}
```

• Reply • Share ›

ANA • 8 months ago

c++ implementation
https://ideone.com/8TKcGX

1 • Reply • Share ›

newbie • a year ago

I'm sorry but I'm very bad at recursions
I cannot visualise exactly how the two functions reverse and insertATBottom are working .
please help me with it.
I tried searching internet for recursion examples , but all i get is the same factorial example .

• Reply • Share ›

Abhinav Aggarwal • 2 years ago

A better code with time complexity O(n).

```
 #include<iostream>
#include<stack>

using namespace std;

int reverse(stack<int> s, int num)
{
        if(s.empty())
                return num;

        int val=s.top();
        s.pop();
        s.push(reverse(s,val));
        return val;
}
```

see more

• Reply • Share ›

rahul23 ➦ Abhinav Aggarwal • a year ago

dude it is not reversing , op o f ur program is 11 10 9 ..... 0 , it is the same stack not dude,it is not reversing ..op o f ur program is 11-10-9......0...it is the same stack not reverse...plz check code before posting.:)

8 • Reply • Share ›

pefullarton Abhinav Aggarwal • 2 years ago

I think, you are wrong,
you would end up getting the same stack again. Try to run it for any sample input.

• Reply • Share ›

zyfo2 • 2 years ago

if you can use another stack, it will be simpler. the two recursion way is tricky

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

• Reply • Share ›

Jun zyfo2 • 8 months ago

ofcourse,,but that would increase the space complexity na

• Reply • Share ›

Emmanuel Livingstone zyfo2 • a year ago

Yeah something like having another empty stack and in every call, we pop the top element of the first stack and push it into the second stack and then recursively call the same function again.

• Reply • Share ›

vkjk89 • 3 years ago

Can anyone please explain the Time complexity of this algorithm ? Whats time complexity of this algorithm ?

• Reply • Share ›

prakhar vkjk89 • 3 years ago

I think it's of the order of O(n^2).

1 • Reply • Share ›

srinivas • 4 years ago

```
 void ReverseStack(Stack s)
{
   if(isEmpty(s) return;
   x=pop(s);
   ReverseStack(s);
   RecursivePush(s,x);
}
RecursivePush(Stack s,int x);
```

```
 {
  int temp;
  if(isEmpty(s)) {push(s,x) return;}
  temp=pop(s);
  recursivepush(s,x);
  push(s,temp);
 }
```

1 • Reply • Share ›

Kolo • 5 years ago

[sourcecode language="java"]
public class ReverseStackByRecursion<T> {

public void reverse(Stack<T> s) {
if (s.isEmpty()) return;
T last = getLast(s);
// reverse the remaining stack
reverse(s);
// put the last element on the top
s.push(last);
}

// retrieve and remove the deepest element of the stack
public T getLast(Stack<T> s) {
T a = s.pop();
if (s.isEmpty()) {
return a;
} else {

see more

1 • Reply • Share ›

vasu Kolo • 4 years ago

```
 void ReverseStack(Stack s)
{
  if(isEmpty(s) return;
   x=pop(s);
   ReverseStack(s);
   RecursivePush(s,x);
}
RecursivePush(Stack s,int x);
{
  int temp;
```

```
    if(isEmpty(s)) {push(s,x) return;}
    temp=pop(s);
    recursivepush(s,x);
    push(s,temp);
  }
```

4 • Reply • Share ›

erman vasu • a year ago

what you call "recursivepush" is the same thing as "InsertAtBottom" in the original article. so you're just writing the same thing

• Reply • Share ›

shreya erman • 10 months ago

can nyone make me understand what actually hapening here n how..m unable to visualise :(

1 • Reply • Share ›

GOPI GOPINATH shreya • 10 months ago

A simple solution.....

```
void reverse(Stack s)
{
int num = (int)s.Pop();
if (s.Count != 1)
reverse(s);
insertbottom(s , num);
}
void insertbottom(Stack s , int a)
{
int num = (int)s.Pop();
if (s.Count != 0)
insertbottom(s , a);
else
s.Push(a);
s.Push(num);
}
```

• Reply • Share ›

RAHUL JAIN Kolo • 4 years ago

awesome answer

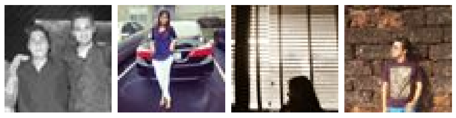
• Reply • Share ›

Subscribe Add Disqus to your site Privacy

-
- GeeksforGeeks

  Like

  92,235 people like GeeksforGeeks.

  Facebook social plugin
-
-
    - [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

- Humble_Learner

  The question itself is about without the use of...

  Linked complete binary tree & its creation · 24 minutes ago

- Haikent

  bool isbst(node<t> *rm ,int min=INT_MIN,int...

  A program to check if a binary tree is BST or not · 31 minutes ago

- Haikent

  int isbst(struct node *tm) {...

  A program to check if a binary tree is BST or not · 1 hour ago

- Gaurav pruthi

  yup...... Author is trying to explain...

  When are static objects destroyed? · 3 hours ago

- Nitin Pandey

  Round 1 - Question 2

  https://ideone.com/5A8Ffv

  LinkedIn Interview Experience | Set 5 (On campus) · 3 hours ago

- Vito

  To print the sub-array also, please note this...

  Largest Sum Contiguous Subarray · 3 hours ago

-