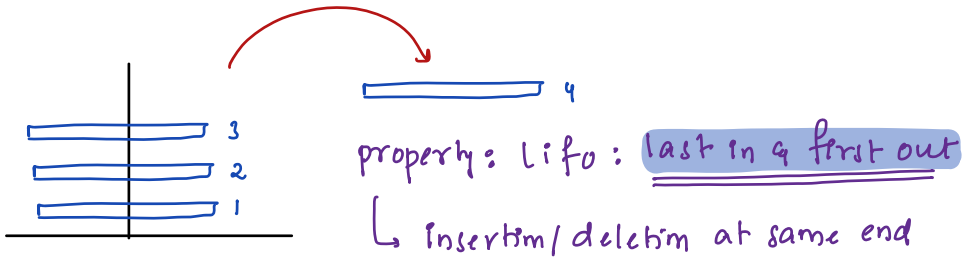
Todays Content:

- Stacks Basics
- Double Character trouble
- Expression Evaluation
    a) Infix → postfix : Idea
    b) Evaluate postfix : Code

Stacks:



property: Lifo: last in & first out

↳ Insertion/deletion at same end

Functnolities:

push(n): Insert n on top of stack

pop(): delete top most element

top(): return top ele

size(): return no: of ele in stack

Ex: 2 5 7 pop() →del 7 top() :5 9 11 8 pop() del 8 top() :11

8
11
9
~~7~~
5
2

Note: Only ele we can acess in stack is top most ele

Obs: Both push & pop happens from same side.

Stack implementation:

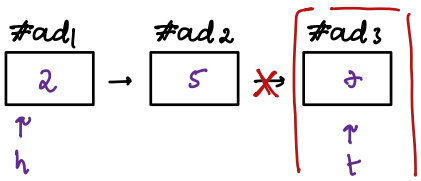
a) Using arrays

b) Using dynamic array } TODO

c) Using linked list ✓

Stack implementation using linked list:

Ex: 2 5 7 pop() top() 9 11 8 pop() top()

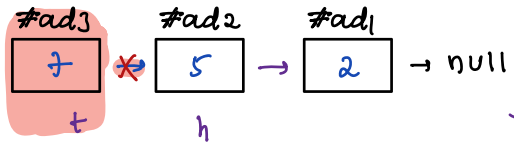


pop(): We need to delete tail node, traverse till last & break link to tail node: O(N)

→ a single pop: O(N) time

obs: When both push & pop are done from end, pop = O(N) ?

Ex2: 2 5 7 pop() top() 9 11 8 pop() top()



obs: push & pop are performed at head, it only takes O(1)

```
Class Node{
    int data;
    Node nent;
    Node(int x){
        data = x
        nent = null
    }
}
```

```
Node h = null // global head
int c = 0 //
```

```
__ push(int n){
    Node nn = new Node(n)
    nn.nent = h
    h = nn;
    c = c+1
}
```

```
__ pop(){
    if(h==null){ return }
    Node t = h
    h = h.nent
    t.nent = null
    c = c-1
}
```

//deletion not possible

```
int top(){
    if(h==null){ return }
    return h.data
}
```

```
int size(){
    return c
}
```

{Please refer them in your language}

Stack library:

```
stack<int> st;
```

(stack: declared stack; int: type of data; st: var name)

→ st.push(n) st.pop() st.top() st.size()

A single operation takes O(1)

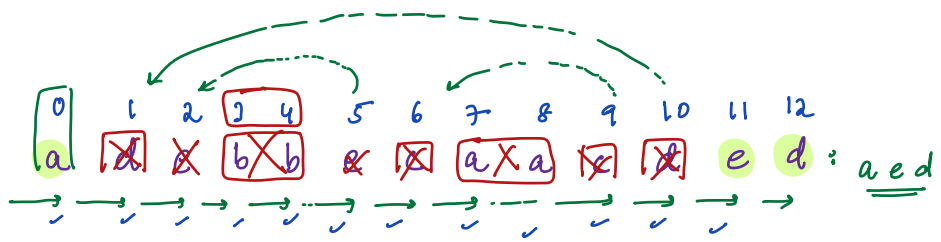
Q) Double Character Trouble

Given a string s, Remove equal pair of adjacent characters
Return the string without adjacent duplicates

Ex1: 0 1 2 3
a b b d → a d

Ex2: 0 1 2 3 4 5 6
a b c c b d e → a d e

Ex3: 0 1 2 3 4
a b b b e → a b e

Ex4: 

Idea: : Say we pop all ele from stack: d e a

d x1
e x2
a x3

: Note: We need to revern final string: a e d
↳ obs1: Do it from right → left to get exact output

Pseudocode:

```
stack<char> st;
Iterate on string str      :   TC: O(N)  SC: O(N)
    : If st is Empty : st.push(ch)   ↳ no 2 adj character are same.
    : ch == st.top():                10:05 → 10:15pm
          st.pop
      else
          st.push(ch)
pop all ele from stack & revern them.
```

Expression Evaluation: →

Ex1: $8 * 5 + 4 =$

$= 40 + 4 = 44$

Ex2: $10 + 3 * 4 - 6 / 3 =$

$10 + 12 - 6 / 3$

$10 + 12 - 2 = 20$

Ex3: $7 \times 1 + 2 - 8 * 3 + 10 / 5 = ?$

$7 + 2 - 8 * 3 + 10 / 5$

$7 + 2 - 24 + 10 / 5$

$7 + 2 - 24 + 2 = -13$

operator precedence:

- / * : same predencesa
- + - : same predesa

In above cas, if 2 operatrs have same precedence, operatr which comes first from left we do that.

a (operatr) b ⇒ // operand

Infin: operatr between operaods

Infin —convert→ postfin —evaluate→

postfin: operatr after operarads

Infin Expressions ⟶ Post fin Expressions

a + b	⟶	a b +
a − b	⟶	a b −
a / b	⟶	a b /
a * b	⟶	a b *

Infin: → Postfin

: 4 + 8 * 7

4 + (8 7 *)

4 8 7 * +

: 10 + 3 * 4 − 7

10 + (3 4 *) − 7

(10 3 4 * +) − 7

10 3 4 * + 7 −

: 10 / (4 − 2) * 6 + 9

10 / (4 2 −) * 6 + 9

(10 4 2 − /) * 6 + 9

(10 4 2 − / 6 *) + 9

10 4 2 − / 6 * 9 +

Infin: A + B / C * (D + E) - F

Stack

Postfin: A B C / D E + * + F -

A + B * C - D * (F + G * K) + L * M

A B C x + D F G k * + * - L M * +

Given infin: TC: O(N), → Infin length SC: O(N)

Create a stack st

Iterating in infin

: operand : add it at end of postfin

: '(' : push inside stack

: ')' : pop all characters & them in postfill until we get an open bracked '('

: operator :

while (st.size() > 0 && st.top() != '(' && predence(st.top()) >= predence(operatr))

{
  char ch = st.top() // get top operatr
  add ch at back of postfin
  st.pop()
}

push new operatr in stack

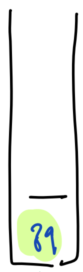
Once we iterate on infin:

→ pop all ele from stack add it in postfin.

```
int precedence(char ch)
{
    if (ch == '+' || ch == '-') { return 1; }
    if (ch == '/' || ch == '*') { return 2; }
}
```

Infix: ⟶ Postfix: {Execution order operators}

10 / (4 - 2) * 6 + 9 ⟺ 10 4 2 - / 6 * 9 +



operator	a ⊕ b	
-	4 - 2	2 push in stack
/	10 / 2	5 push in stack
x	5 * 6	30 push in stack
+	30 + 9	39 push in stack

↳ Single top ele left in stack is ans

Evaluating Postfix, We took a stack s1

Iterate on expression TC: O(N) SC: O(N)

- : Operand → push in stack
- : Operator : get 1st top elem = b & pop it
  - get 2nd top ele = a & pop it
  - push result of a ⊕ b in stack

Single top ele left in stack is ans

Postfix : 100_80_21_*_+_ // Input is given in string format

Note: Both operators & operands are separated using space