CHAPTER - 3 STACK

- PROF. TRUPESH PATEL

STACK : Definition and concept :

"A linear list which allows insertion and deletion of an element at one end only is called STACK."

PUSH - Insertion operation, POP - Deletion operation
Top - Most accessible element, Bottom - least element
LIFO (Last In First Out) : The last inserted element must be removed first as operation is possible from one end only hence this mechanism is called LIFO (last in first out).

Beside Image is the demonstration of stack phenomenon, a pile of trays in a cafeteria. Here, a person desiring a tray finds that only one is available to him or her at the surface of the tray counter.

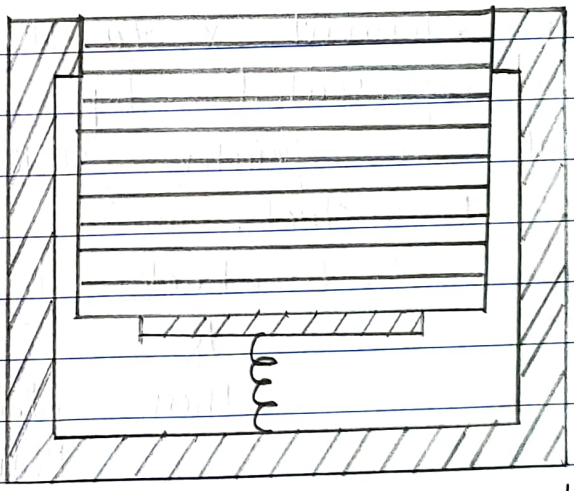
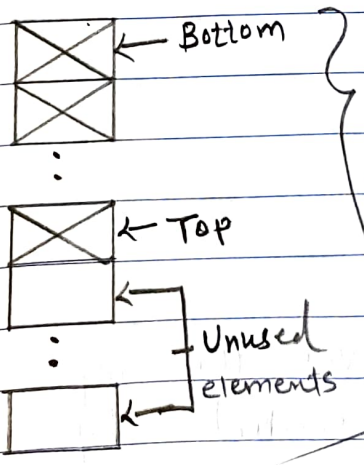


fig : A - Cafeteria tray holder



A pointer Top keeps track of the top element in the stack. Initially, when the stack is empty, Top has a value of "one" and so on.

— PROF. TRUPESH PATEL

Each time a new element is inserted in the stack, the pointer is incremented by "one" before, the element is placed on the stack. The pointer is decremented by "one" each time a deletion is made from the stack.

Applications of stack:

1. Recursion
2. Evaluation of Expression
3. Reversing characters
4. Handling Hardware Interrupts
5. Use of backtracking

STACK Algorithm: 1. PUSH(S, TOP, X) ⇝ It inserts an element X to the top of a stack which is represented by vector S containing N elements with a pointer TOP denoting the top element in the stack.

1. [Check for stack overflow]
   If TOP ≥ N
   Then Write ("STACK OVERFLOW")
   Return
2. [Increment Top]
   Top ← Top + 1
3. [Insert Element]
   S[Top] ← X
4. [Finished]
   Return.

2. POP(S, Top) ⇝ This function removes the top element from a stack which is represented by

a vector S and returns this element. Top is a pointer to the top element of the stack.

1. [check for underflow of stack]
   If Top = 0,
   Then Write ('STACK UNDERFLOW ON POP')
   Take action in response to underflow
   Return.
2. [Decrement Pointer]
   Top ← Top - 1
3. [Return former top element of stack]
   Return (S [Top + 1])

3. PEEP (S, Top, I) : This functions returns the value of the i^{th} element from the TOP of the stack which is represented by a vector S containing N elements. The element is not deleted by this function.

1. [check for stack Underflow]
   If Top - I + 1 ≤ 0
   Then Write ('STACK UNDERFLOW ON PEEP')
   Take action in response to Underflow
   Exit
2. [Return I^{th} element from top of stack]
   Return (S [Top - I + 1])

4. CHANGE (S, Top, X, I) : This procedure changes the value of the I^{th} element from the top of the stack to the value containing X.

– PROF. TRUPESH PATEL

1. [check for stack Underflow]
   If Top - I + 1 ≤ 0
   Then Write('STACK UNDERFLOW ON CHANGE')
   Return
2. [change I^{th} element from top of stack]
   S[Top - I + 1] ← X
3. [Finished]
   Return

- **Applications of stack:**
  - Reverse string:

Write an algorithm which will check that the given String belongs to following grammar or not.

$L = \{ WCW^R \mid W \in \{a,b\}^* \}$ { Where W^R is the reverse of W }.

Algorithm: Recognise

Here, Given an input string name STRING on alphabet { a, b, c }

NEXTCHAR – function which returns the next symbol in STRING.

VECTOR S – S represents stack

Top – It is a pointer to the top element of the stack.

1. [Initialize stack by placing a letter 'c' on the top]
   Top ← 1
   S[TOP] ← 'c'

2. [Get and stack symbols either 'c' or blank is encountered]
   NEXT ← NEXTCHAR(STRING)
   Repeat While NEXT ≠ 'c'
   If NEXT = ' '
   Then Write ('Invalid string')
   Exit
   Else Call PUSH(S, TOP, NEXT)
   NEXT ← NEXTCHAR(STRING)

3. [Scan characters following 'c'; Compare them to the characters on stack]
   Repeat While S[TOP] ≠ 'c'
   NEXT ← NEXTCHAR(STRING)
   X ← POP(S, TOP)
   If NEXT ≠ X
   Then Write ('INVALID STRING')
   EXIT

4. [Next symbol must be blank]
   If NEXT ≠ ' '
   Then Write ('VALID STRING')
   Else Write ('INVALID STRING')

5. [Finished]
   EXIT

~ PROF. TRUPESH PATEL

- Write an algorithm to determine if an input character string is of the form $a^i b^i$ where $i >= 1$.

i.e. no. of a = no. of b.

Algorithm RECOGNIZE

1. [Initialize stack and counter]
   TOP ← 0
   COUNTER_B ← 0

2. [Get and stack character 'a' from whole string and count the occurance of 'b']
   NEXT ← NEXTCHAR(STRING)
   Repeat While NEXT != ' '
   IF NEXT = 'a'
   Then PUSH(S, TOP, NEXT)
   Else COUNTER_B ← COUNTER_B + 1
   NEXT ← NEXTCHAR(STRING)

3. [Pop the stack until empty and decrement the COUNTER_B]
   Repeat While TOP != 0
   POP(S, TOP)
   COUNTER_B ← COUNTER_B - 1

4. [check for grammer]
   If COUNTER_B != 0
   Then Write('INVALID STRING')
   Else Write('VALID STRING')

- PROF. TRUPESH PATEL

Trace of contents of stack for RECOGNIZE.

Input String	character Scanned	contents of Stack
abcba□	none	c
	a	ca
	b	cab
	c	cab
	b	ca
	a	c
	□	c

Valid string

aabcaab□	none	c
	a	ca
	a	caa
	b	caab
	c	caab
	a	caa

Invalid string since a ≠ b

aabcbaaa□	none	c
	a	ca
	a	caa
	b	caab
	c	caab
	b	caa
	a	ca
	a	c
	a	c

Invalid string since c is Top and NEXT ≠ '!'.

— Prof. Trupesh Patel

- **Recursion:** "Recursion is the technique of defining a set or a process in terms of itself."

"A procedure that contains a procedure call to itself, or a procedure call to a second procedure which eventually causes the first procedure to be called, is known as a recursive procedure."

There are two types of recursion.
1. Recursively defined functions: (Primitive)
   Factorial Function
2. Recursive use of a procedure: (Non-Primitive)
   Find GCD of given numbers.

Algorithm to find factorial of given no. using recursion:

For, integer N, we can compute factorial of N. stack A is used to store an activation record associated with each recursive call. Each activation record contains the current value of N and the current return address RET_ADDR. TEMP_REC is also a record which contains two variables PARAM & ADDRESS. TOP is a pointer to the top element of stack A. Initially return address is set to the main calling address. PARAM is set to N initial value.

~ PROF. TRUPESH PATEL

1. [Save N and return address]
   call PUSH (A, TOP, TEMP_REC)

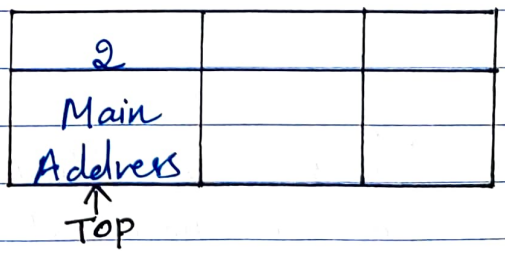
2. [Is the base criterion found ?]
   If N = 0
   then FACTORIAL ← 1
   Go to step 4
   else PARM ← N-1
   ADDRESS ← step 3
   Go to step 1

3. [calculate N !]
   FACTORIAL ← N * FACTORIAL

4. [Restore previous N and return address]
   TEMP_REC ← POP (A, TOP)
   (i.e. PARM ← N, ADDRESS ← RET_ADDR, POP STACK)
   Go to ADDRESS

– TRACE OF Algorithm FACTORIAL :-

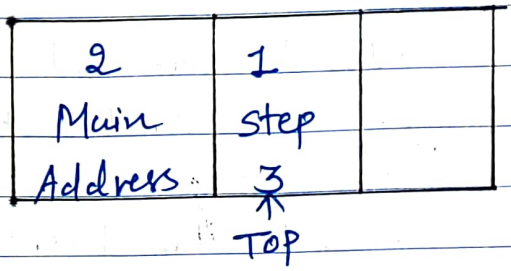
Level	Description	Stack contents
Level 1	1. PUSH (A, 0, (2, main address))	
	2. N ≠ 0	
	PARM ← 1, ADDR ← stp 3	



~ PROF. TRUPESH PATEL

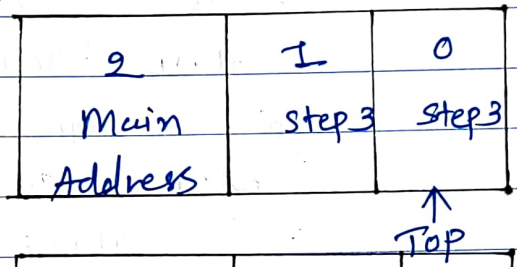
Level 2 (1st Recursive call)

1. PUSH (A, 1, (1, step 3)
2. N ≠ 0
   PARM ← 0, ADDR ← step 3

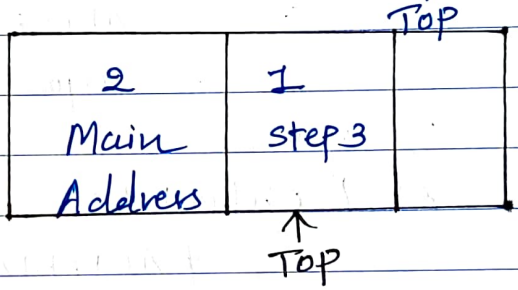


Level 3 (2nd Recursive call)

1. PUSH (A, 2, (0, step 3)
2. N = 0
   FACTORIAL ← 1

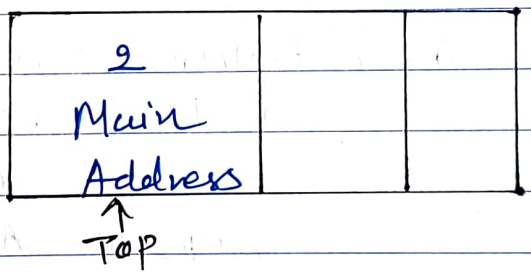


4. POP (A, 3)
   go to step 3



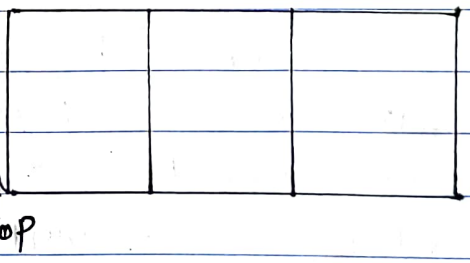
Return to level 2

3. FACTORIAL ← 1 * 1
4. POP (A, 2)
   go to step 3.



Return to level 1

3. FACTORIAL ← 2 * 1
4. POP (A, 1)
   go to main address



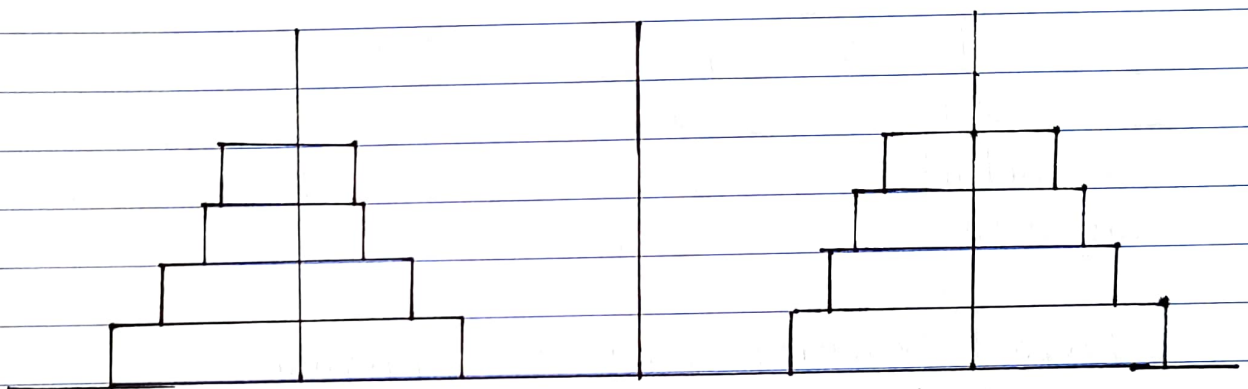
Tower of Hanoi : It is a recursive problem which has a historical basis in the ritual of the ancient Tower of Brahma.

problem ~ Given N discs of decreasing size stacked on one needle and two empty

needles. It is required to stack all the discs onto a second needle in decreasing order of size. The third needle may be used as temporary storage.

Following rules are applicable for discs movement.

1. Only one disc may be moved at a time.
2. A disc may be moved from any needle to any other.
3. At no time may a larger disc rest upon a smaller disc.



Needle A (start of problem) Needle B (intermediate) Needle C (completion of problem)

Fig: Tower of Hanoi

Solution: move one disc from needle A to C. (disc = 1)

if disc = 2 then,

move the first disc to needle B
move the second from needle A to C.
move disc from B to needle C.

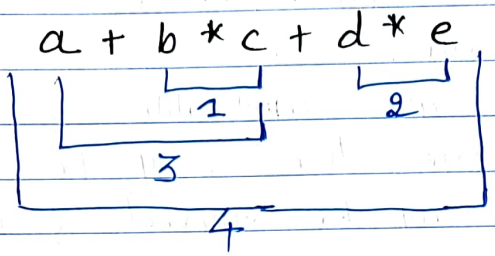
In general,

1. move N-1 discs from A to B.
2. move disc N from A to C.
3. move N-1 discs from B to C.

– PROF. TRUPESH PATEL

– POLISH EXPRESSION & THEIR COMPILATION :-

– Evaluating Infix Expression :



- A repeated scanning from left to right is needed as operators appears inside the operands.

- Repeated scanning is avoided if the infix expression is first converted to an equivalent parenthesis free prefix or postfix expression.

- prefix Expression : operator , operand , operand
- postfix Expression : operand , operand , operator

This types of notation is known polish notation or Reverse polish Notation due to polish logician Jan Lukasiewicz.

- In both prefix & postfix equivalents of an infix expression , the variables are in same relative position.

- The expressions in postfix or prefix form are parenthesis free and operators are rearranged according to rules of precedence.

~ Prof. Trupesh Patel

• Polish Notation :

No.	INFIX	POSTFIX	PREFIX
1	a	a	a
2	a+b	ab+	+ab
3	a+b+c	ab+c+	++abc
4	a+(b+c)	abc++	+a+bc
5.	a+(b*c)	abc*+	+a*bc
6.	a*(b+c)	abc+*	*a+bc
7.	a*b*c	ab*c*	**abc

• Convert INFIX To POSTFIX Expression :

Precedence

Symbol	Input precedence function f	Stack precedence function g	Rank function r
+, -	1	2	-1
*, /	3	4	-1
↑	6	5	-1
Variables	7	8	1
(	9	0	-
)	0	-	-

Algorithm : REVPOL

INFIX string contains an infix expression.
POLISH string contains converted reverse polish expression
Symbol have precedence value as per above table.
S denotes vector to represent stack, Top denotes the top of the stack
PUSH, POP are used for stack manipulation.
NEXTCHAR returns the next Symbol in given input.
TEMP used for temporary storage purpose.

~ PROF. TRUPESH PATEL

1. [Initialize stack]
   TOP ← 1
   S[TOP] ← '('

2. [Initialize output string and rank count]
   POLISH ← ''
   RANK ← 0

3. [Get first input symbol]
   NEXT ← NEXTCHAR(INFIX)

4. [Translate the infix expression]
   Repeat thru step 7 while NEXT ≠ ')'

5. [Remove symbols with greater precedence from stack]
   If TOP < 1
   then Write('INVALID')
   Exit
   Repeat while f(NEXT) < g(S[TOP])
   TEMP ← POP(S, TOP)
   POLISH ← POLISH ○ TEMP
   RANK ← RANK + r(TEMP)
   If RANK < 1
   then Write('INVALID')
   Exit

6. [Are there matching parentheses ?]
   If f(NEXT) ≠ g(S[TOP])
   then Call PUSH(S, TOP, NEXT)
   else POP(S, TOP)

— PROF. TRUPESH PATEL

7. [Get the next input symbol]
   NEXT ← NEXTCHAR (INFIX)

8. [Is the expression valid ?]
   If TOP ≠ 0 OR RANK ≠ 1
   then Write ('INVALID')
   else Write ('VALID')
   Exit

Translation : (a+b↑c↑d) * (e+f/d) (INFIX to POSTFIX)

Character Scanned	Contents of stack	Reverse polish Expression	Rank
	(		
(	((		
a	((a		
+	((+	a	1
b	((+b	a	1
↑	((+↑	ab	2
c	((+↑c	ab	2
↑	((+↑↑	abc	3
d	((+↑↑d	abc	3
)	(	abcd↑↑+	1
*	(*	abcd↑↑+	1
(	(*(	abcd↑↑+	1
e	(*(e	abcd↑↑+	1
+	(*(+	abcd↑↑+e	2
f	(*(+f	abcd↑↑+e	2
/	(*(+/	abcd↑↑+ef	3
d	(*(+/d	abcd↑↑+ef	3
)	(*	abcd↑↑+efd/+	2
)		abcd↑↑+efd/+*	1