Ds

Linear Array.

- A linear Array is a list of a finite number n of homogeneous data elements.

Length = UB - LB + 1.

upper bond, lower bond

A[K]

subscript or index

Subscripted variable

Representation of linear array in memory

LOC LA[K] = address of the element LA[K] of the array LA

First element of the address La (Base).

Formula.

$$LOC(LA[K]) = BASE(LA) + w(K - LB)$$

- $LOC(LA[K])$ → La array ka K elements en address.
- $BASE(LA)$ → Starting point
- w → words per memory cell
- LB → lower bound.

Exp pdf

Lower bound = 1965

base = 200

w = 4

K = 1965 32

$$LOC(auto[1932]) = base + w(K - LB)$$

$$= 200 + 4(1965 - 1932)$$

$$= 332$$ Ans

Traversing Linear Array

[visiting array]

Accessing each element of array once so that it can be processed.

Step 1 : Start

2 : initialize counter.

Set K = LB

3. repeat steps 4th and 5th (while K <= UB).

4. Apply process to LA[K].

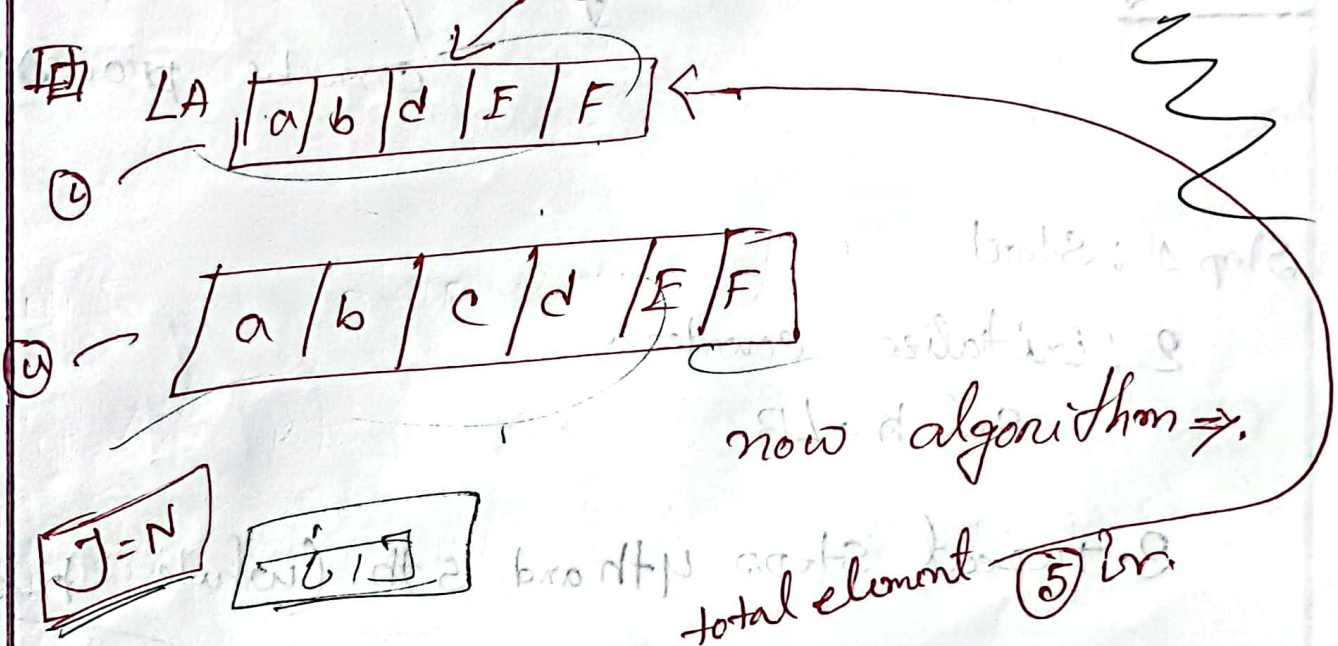
5. Increment counter

K = K+1.

6. Stop [যদি K <= UB না থাকে, তাহলে K মান UB থেকে বড় হবে]

Inserting into LA

c duskabo (insert c)



Step 1, Set J := N

2. Repeat Steps 3 and 4 while J >= K

3. Set LA[J+1] := LA[J] → (ii) মানের জন্য

4. Set J := J - 1 → ① এর মানে back যেতে যাতে 2য় অবস্থানে যেতে পারি

5. Set LA[K] := Item. → যেখানে যাবে বা বসাবে

6. Set N := N+1 → বসাই দিবে

7. Exit.

Read করার জন্য এরো সাইজ দে

Deleting Linner array.

1 Set item = LA[K]

2 Repeat for j = K to (N-1)

Set LA[J] := LA[J+1]

3 Set N := N-1;

4 Exit

here, LA is Linner array
N elements.
K positive integer.
Such K ≤ N
delete (K-th) element from LA.

---

data may be organized different way, logical and mathematical manner of particular organisation of data is ds.

Bubble Sort

- here data is an array N
- Sort to element in Data

Repeat steps 2 and 3 for K = 1 to N-1.

Set PTR := 1.

Repeat while PTR <= N-K

a) If Data[PTR] > Data[PTR+1], then, interchange Data[PTR] and Data[PTR+1]

b) Set PTR := PTR+1

Exit.)

worst case - $O(n \times n) = O(n^2)$

Best n - $\Theta(n)$

$O(n^2)$

Searching

Linear Search

A Linear array data with N elements and a specific Item information are given. find location K

1. Set K := 1 and LOC := 0
2. Repeat steps 3 and 4 while LOC = 0 and K ≤ N
3. If Item = Data[K] then Set LOC = K.
4. Set K = K + 1.
5. If LOC = 0 then.
   print item is not in the array Data.
   else,
   print Loc is the location of Item.
6. Exit

* Binary search.

~~right(+)~~ left (-)

— 0 —

• Array middle part

Int $\frac{B+E}{2}$ — Bigger End.

left (-) Beg | right (+) end

(1) BEG = LB, End = UB and MID = Int((BEG+End)/2)

(2) Step 3 and 4 while BEG ≤ End and

element → Begene end এর সমান বা ছোট হওয়া পর্যন্ত 3,4 loop চলবে

Data [MID] ≠ Item

1st বার মিড এর মধ্যে find হয়ে গেলে বাইরে হয়ে যাবে

(3) If Item < Data [mid] then : set End = Mid - 1. else BEG = mid + 1.

left হলে এটা — Right হলে এটা

(4) Set mid = Int((BEG + End)/2).

আবার middle বের.

5. If Data [mid] = Item set Loc := Mid

Else Loc := Null.

Beg > End হলে exist হবে না

log 2n

O/

Efficiency of algorithm → Complexity

• Complexity is the function that gives the running time and/or space for some input.

which complexity f(n)

Best case : The minimum value [1st 2r]

worst case : The maximum value [last 2r]

average case : The expected value (half).

not minimum not maximum

• $\log n < n < n\log n < n^2 < n^3 < 2^n < n!$

$O(1) < \log n < n < n\log n < n^2 < n^3 < 2^n < n!$

big o nation : It is a way to describe the speed or complexity of given algorithm

2D Array Memory Representation

Row major :-

~~Loc A[i][j] = Base(A) + w[N(j-1) + (k-1)]~~

$$Loc\ A[i][j] = Base(A) + w[n(i-L_1) + (j-L_2)]$$

$$[n = U_2 - L_2 + 1]$$

Sub: A 2D array is defined as A[3:7, -1:4] requires 4 words per memory cycle. Find the location A[5,2] if array is implemented in Row major order.

A[3:7, -1:4]
$L_1 = 3$ $U_1 = 7$
$L_2 = -1$ $U_2 = 4$.

Base(A) = 200
w = 4.
i = 5, j = 2

$n = 4 - (-1) + 1$
$= 6$

here Row major.

$$Loc\ A[i][j] = Base(A) + w[n(j-1) + (k-L_2)]$$

$$= 200 + 4[6(5-3) + (2-(-1))]$$

$$= 200 + 4[6(5-1) + (2-1)]$$

$$= 200 + 4[6(4) + 1]$$

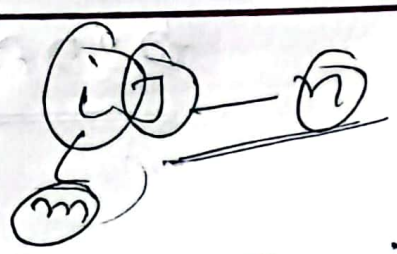
$$= 200 + 4[24 + 1]$$

$$= 200 + 4[25]$$

$$= 200 + 100$$

$$= 300$$

Column major:



$$Loc\ A[i][j] = Base(A) + w[(i-L_1) + m(j-L_2)]$$

m (k-1

rows — column.

Ex: defined as A[20][5], 4 words. Find the location A[12,3], use column major.

⇒ Base = 200, w = 4, i = 12, j = 3

m = number of rows.

here

$$Loc\ A[i][j] = Base(A) + w[(i-L_1) + m(j-L_2)]$$

$$Loc\ A[12][3] = 200 + 4 + [(12-0) + 20(3-0)]$$

$$= 488$$

clm $Loc\ A[j,k] = Base(A) + w[M(k-1) + (j-1)]$.

row $Loc\ A[j,k] = Base(A) + w[M(j-1) + (k-1)]$

Matrix

Matrix multiplication.

let

$A[M \cdot P] \Rightarrow C[M \times N]$

$B[P \cdot N]$

row column.

a এর row b এর কলাম same থাকতে হবে

m = row
N = column

Algorithm :

1. Repeat step 2 to 4 for I = 1 to m.
2. Repeat step 3 and 4 for J = 1 to n
3. Set C[I, J] := 0
   initialize করে দিতে.
4. Repeat for K = 1 to P:

$$C[I,J] := C[I,J] + A[I,K] * B[K,J]$$

Sparse matrix

array representation

*যেখানে 0 ছাড়া অন্য value গুলোর জায়গা Row কলাম এবং Value

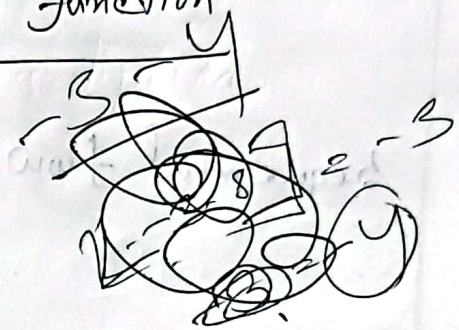
	0	1	2	3	4
0	0	0	4	0	5
1	0	0	0	3	6
2	0	0	0	2	0
3	0	2	3	0	0

Row	0	0	1	1	2	3	3
Column	2	4	3	4	3	1	2
Value	4	5	3	6	2	2	3

- Matrix with a relatively high proportion of zero entites are called sparse matrix

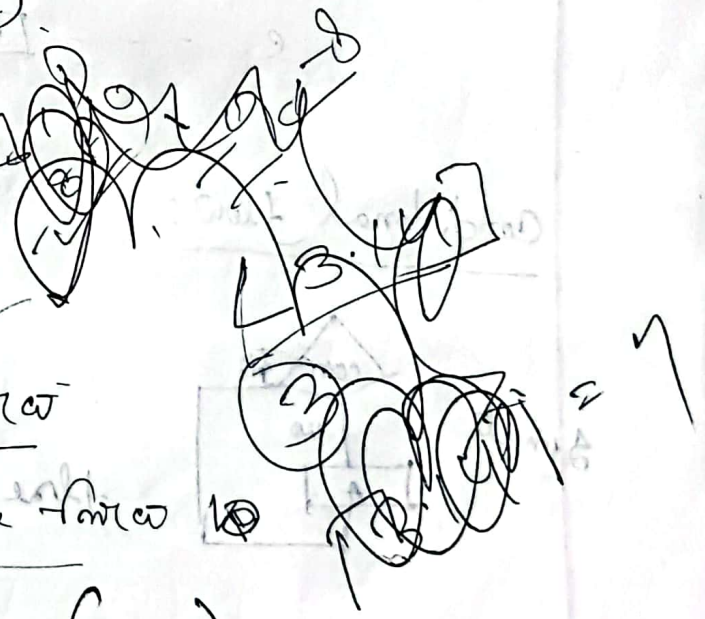
Mathematical and function

- Exponents and logarithm.



- common logarithm (base 10), natural logarithm (base e)
  Binary " (base 2)

- algorimathic notation:



- step, control, exit.
- comments. [---] square
- variable - upper case square
- assigenment statements (:=)
- input and output

⌊3.14⌋ = 3

⌊-8.5⌋ = -9.

floor

⌈-8.5⌉ = -8

⌈3.4⌉ = 4.

ceiling

Control structure

Sequential flow:

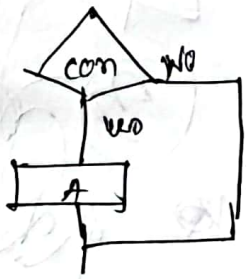
module A --- A Serially যে

B --- B

C --- C

Conditional flow:

Simple



false হলে কিছু করবে না.

multiple if (A)

A1

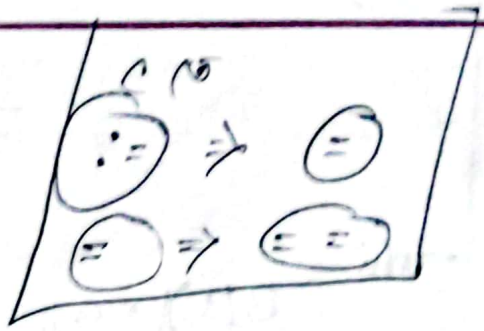
Else if

A2

Else if A1A2

else B.

• Quadratic equation.



① read A, B, C

② Set D := $B^2 - 4AC$

3. If D > 0 then

a) Set x_1 := $(-B + \sqrt{D})/2A$ and x_2 := $(-B - \sqrt{D})/2A$.

b. write : x_1, x_2

else if D = 0

Set x := $-B/2A$.

write : "unique solution", x.

else.

write : 'no real solution'

Ex#1

Matrix

→ যদি $B[1] = a_{11}$ $B[2] = a_{23}$

i J

তবে index বাহির করো.

$$L = \frac{J(J-1)}{2} + K.$$

Eire.

$3,2 \Rightarrow L = \frac{3(3-1)}{2} + 2$

$= 3 + 2 = 5$

$B[5] = 32.$

Multidimensional array:

⇒ যে array more than one level or dimention.

General multidimensional Array

যেখানে কোন প্রমাণ LB = 1 ধরে নেয়া

Maze (2:8), (-4:1), (6:10)

LB UB

[UB - LB + 1].

Length of dimension

$b_1 = 8 - 2 + 1$

$= 7$

$b_2 = 1 + 4 + 1$

$= 6$

$b_3 = 10 - 6 + 1$

$= 5$

M N

7, 6

So here $7 \times 6 \times 5$ elements 210.

$= 210$ — elements total.

Math

now Base 200, w = 4 [4 bit এর]

(5, -1, 8)

effective index.

E_1 5 - 2 = 3 E_2 = -1 - (-4) = 3 E_3 = 8 - 6 = 2.

↑ এর যেখানে শুরুতে (-) Lower bound.

Row major order.

$E_1 L_2 = 3 \cdot 6 = 18$

$E_1 \cdot L_2 + E_2 = 18 + 3 = 21$.

$(E_1 L_2 + E_2) L_3 = 21 \cdot 5 = 105$

$(E_1 L_2 + E_2) L_3 + E_3 = 105 + 2$

$= 107$

$E_1 L_2$

$E_1 L_2 + E_2$.

order.

$(E_1 L_2 + E_2) L_3$

$(E_1 L_2 + E_2) L_3 + E_3$

$(E_1 L_2 + E_2) L_3 + E_3) L_4$

Therefore

$Loc(Maze[5,-1,8)$

$= Base + w[\ ]$ $\quad (E_1 L_2 + E_2) L_3 + L_3$

$= 200 + 4(107) =$

$= 628.$

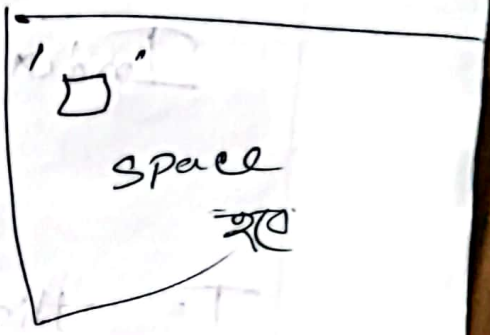
String

Concatenation

A = 'The'
B = 'man'

A || B ⇒ 'The man'

[add করা



Sub string

The substring The end
substring.

initial ⇒ start

[string, initial, lenght)

The man, 5, 2
start len

⇒ 'ma'

String Operation

Index (text, pattern)

T = this function to the program

Index (T, 'the' = 7.

Index (T, 'thm') = 0

Index (T'

word processing

Insert (text, position, string)

Ex

Insert (ABCDEFGR, 3, XYZ)

3 no position add ~~ABC~~ XYZ

= ABCXYZDEFGR.

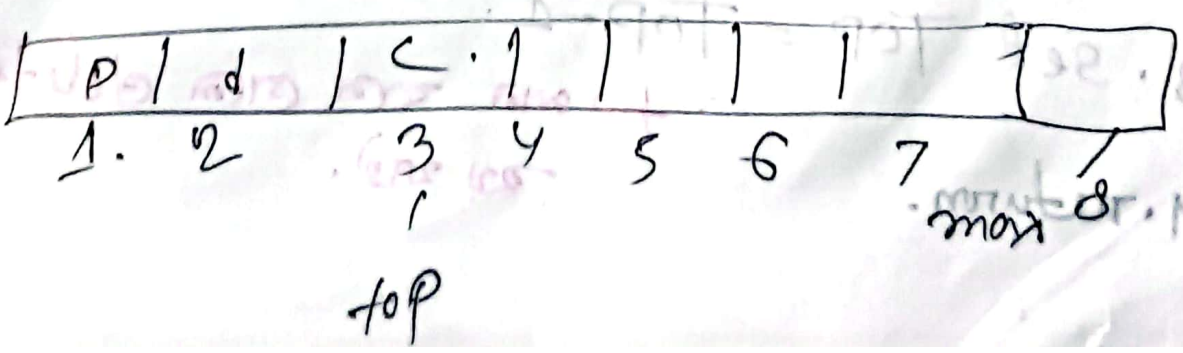
Stack

- A stack is a list of elements in which an element may be inserted or deleted only one end called the top of stack.

- Stack means portion of last element
- Top initially set 0 { portion jaha top stord with 1.

- Push = insert element } operation of stack.
- pop = delete

Array representation.



• How to push algorithm

1. If top = MAXSTK then : print :
   overflow and return.
   full হলে insert হবে না.

2. Set TOP := TOP+1

3. Set STACK[TOP] = item
   insert new item.

4. return

• How to POP:

1. If top = 0 then: print : underflow
   and return.
   খালি হলে delete হবে না

2. Set ITem := Stack[Top]

3. Set TOP = TOP-1.
   = top মান কমে গেলে delete
   হয়ে যাবে.

4. return.

STACK. A B C D E _ _ _

Push L

Pop ITem.

top 6

1 2 3 4 5 6

A B C D E L [Push करने पर add]

A B C D E . ⊙ [POP करने पर last को ir delete]

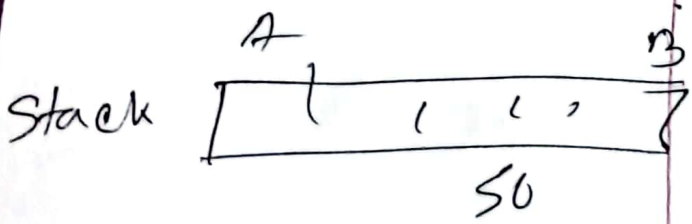
Minimizing overflow

Application of stack.

- Evalation of Arithmatic.
- Back-tracing
- Delimiter Checking
- Reverse of Data.
- Processing a function call.

A_{20} B_{30}

A + B = 50



polish nation

(before placed operands)

A+B কে +AB লিখাকে polish nation.

Prefix

Inverse polish nation.

A+B কে ~~AB~~ হবে

postfix হবে

AB+

Prefix +AB, postfix AB+

#

① : (A+B)*C

⇒ [+AB] *C

⇒ *+ABC

3rd যুক্ত করতে then

(*) আগে হবে

দুটি করে আগে

[ছোট আগে]

② # A+B*C

⇒ A + [*BC]

⇒ +A*BC

$(A+B)*/(C-D)$

$= [+AB] / [-CD]$

$= */+ \cdot AB - CD$

(↑ or ^) right to left

(*, /) left to right

+ - " "

Reverse / postpix notation:

① $(A+B) * C$

$= AB+C*$

Am ② $ABC*+$

Am ③ $AB+CD-/$

Post of reverse + - / order

infix	postfix	prefix / polish
$A+B$	$AB+$	$+AB$
	$A*B+C/D$	$A*B+C/D$
	$=AB*+C/D$	$=*AB+C/D$
	$=AB*CD/+$	$=*AB+/CD$
		$=+*AB/CD$

Evalution of a postfix nation:

- postfix এ · বিবেচনার Ans বের কর!

let given postfix P·

Step

(1). Add a right paranthesis ")" at the end P

P")" তোলে বসাও শেষে হবে·

(2). Scan P from left to right and repeat step 3 and 4 for each element until ")" find

")" না পাওয়া পর্যন্ত 3, 4 step.

(3). If an operand, put on stack.

(4) If and operator ⊗ is encountered.

যেকোনো operator পেলে

a. remove the two elements of stack. where A is the top B is the next top.

(5) b. Evaluate B ⊗ A

B ⊗ A format লিখবো

. place the result of (b) back stack.

Stack push করে দিবে

[End of if.]

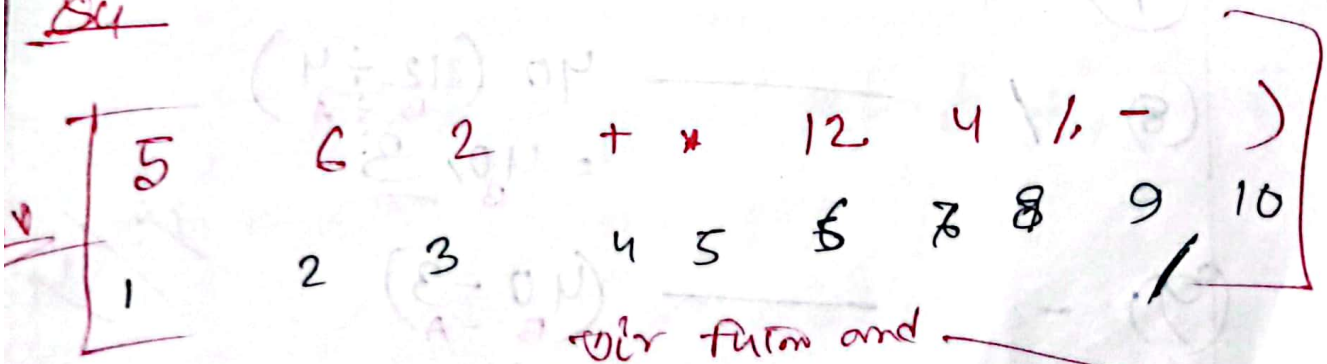
[end of step 2 loop] Step 2 পর্যন্ত loop

5. Set value equal to the top element on.

Stack.

6. Exit.

Ans



Dir form and

5 * (6+2) - 12/4.

= (37) Ans

5 * (6+2) - 12/4

Symbol scan | stack

(1) (5) 5 ——— 5

(2) 6 ——— 5, 6

(3) 2 ——— 5, 6, 2.

(4) + ——— 5 (6+2)
B A
= 5, 8.

(5) * ——— 5 * 8.
= 40

(6) 12 ——— 40, 12 (12=4)

(7) 4 ——— 40, 12, 4 = 8

(8) / ——— 40 (12 ÷ 4)
B ÷ A
= 40, 3
B A

(9) - ——— (40 - 3)
B - A
= 37.

(10))

Operator হলে operand বন্ধ হবে.

Transforming infix to postfix

- Q infix , P postfix

1. push "(" onto stack, and add ")" to the end of Q.

2. Scan Q from left to right repeat step (3-6) for each element of Q until the stack empty

3. If an operand is encountered, add it to P
   operand পাওয়া গেলে P তে push.

4. If a left parenthesis is encountered, push it into stack. (stack এ রাখো)

5. If an operator (X) is encountered then.

a. Repeatedly pop from STACK and add to P each operator (on the top stack) which

has the same precedence as or higher precedence than ⊗. until left parenthesis is encountered

- ⊗ সমান বা higher হলে stack থেকে pop করে P তে push করব + তারপর ⊗ push দিব

b. add ⊗ to stack
তার পরে stack duki dibo

6. Rp parenthesis is encountered then:

a. Repeatedly pop from stack and add to P each operator (on the top of stack) until a left parenthesis is encountered
(stack থেকে Left parenthesis না পাওয়া পর্যন্ত pop করে যাবো, then P তে add করবো)

b. remove the left parenthesis,
Left parenthesis টা মুছে দিবো, Rp parenthesis ও discard করবো।

7. Exit

Infix to Postfix

Example - (C - left parenthesis)

Infix expression

Q: A + (B * C - (D / E ↑ F) * G) * H

A + (B * C - (D / E ↑ F) * G) * H)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20.

Symbol	Stack	Expression P
(1) A	(	A
(2) +	(+	A
(3) (	(+(	A
(4) B	(+(B	AB
(5) *	(+(B*	AB
(6) C	(+(*	ABC
(7) -	(+(-	ABC*
(8) (	(+(-(	ABC*
(9) D.	(+(-(	ABC*D
10. /	(+(-(/	ABC*D
11. E	(+(-(/	ABC*DE
12. ↑	(+(-(/↑	ABC*DEF
13. F	(+(-(/↑	ABC*DEF

A এলে operand.

left এলে stack থেকে.

higher এলে বের হয়ে যাবে

এই স্থানে (আসার পূর্বে P তে দিবো and (কে remove করো

(14))	(+(-	ABC*DEF*/
(15) *	(+(-*	ABC*DEF*/
(16) G	(+(-*	ABC*DEF*/G
(17))	(+	ABC*DEF*/G*-
(18) *	(+* * higher তাই just push করো.	ABC*DEF*/G*-
(19) H	(+*	ABC*DEF*/G*-H
(20))		ABC*DEF*/G*-H*+

এই parenthesis থেকে (পর্যন্ত সব
[illegible] (→ এর পরে P তে push করো.
Higher wise
(কে remove

Introduction.

Data : Data means simply value or set of values.

Entity : Each patient is an entity.

Attributes : The properties - Name, N.Id, Room etc are attributes.

Entity set : The collection of patients in the entity set.

Information : It is processed, organized and structured data.

• Data process করলে information পাওয়া যায়

• Data type :

Build data type (char, double, int) ব্যবহার করে user defined.

Abstract " " (

user defined data type

5, 4, 8, 5

A=5, B=2.12

5, 4, 8, 5

A B

5 2 3

5 + 8,

Abstract data type.

It is a type ~~of~~ for objects whose behaviour is defined by a set of value and set of operation.

- সরাসরি abilevl না (built in data যা available) নিজের তৈরি হচ্ছে না বানাতে হবে.

called user defined data type.

Data Structure

(*) The mathematical and logical model of particular organization of data is called a data structure.

Data stuture.

- Non linear
  - tree
  - graph
- linear (+)
  - array
  - linked list
  - stack
  - Queue

Array

- collection of same type of data.

Linked list

It is an ordered collection of finite, homogeneous data elements, called nodes, where the linear order is maintained by means links or pointer.

-ওর তথ্যের সাথে নির্দিষ্ট লোকেশনে আরেকটার adress

পরের রয়ে adress বলে দিবে

Stack

last in first out

LIFO

Bve mega
Zuta mega
তারপর ove, zuta pori
পরে ঘুমাই
time zuta wali

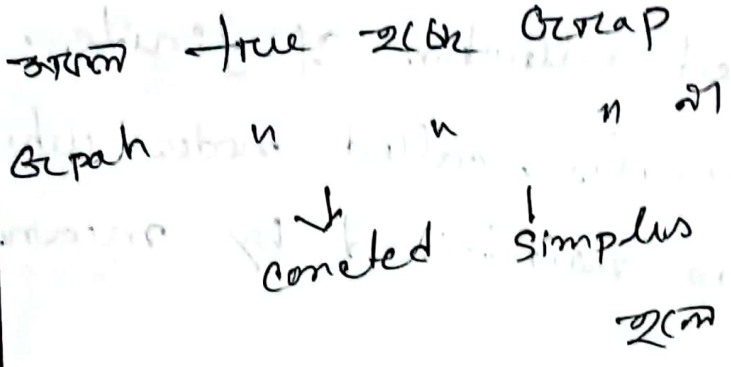
Queue

FIFO

First in first out

Tree

Contain ~~hierical~~ hierachical relationship between various elements



DS operation.

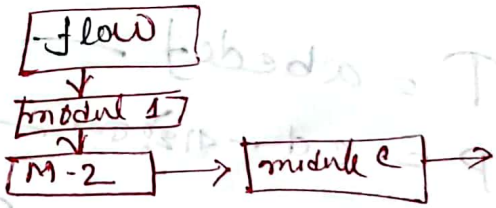
- Traversing - Procesing each element in the list
- Search - find the exect element given value
- Insertion - Adding new element in the list
- Deletion - Removing an
- Sorting - Arranging element some type of order
- Merging - combining two sorted list in the single list

Difference

Data	Information
It means value or set of value.	• data with attributes is called information.
• It has individual figures, number or groups.	• Information is processed data.

Control structure: ⁂

i. **sequential logic:** unless the instruction are given to traverse. the moduls are executed in obvious sequence.



(ii) **Selection logic:** The structures which implement this logic are called conditional structure or if structures.

If condition
[module A]
else
[module B]

3. Iteration logic: Do start with a repeat statement and ends followed by a module.

#multidimarrtial Array

→ length of dimension and elmnt
↓
(UB-LB+1)

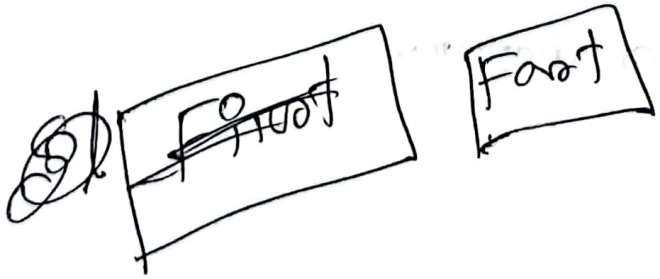
যতগুলো array হবে তত

Pattern matching

T = abcdef — length - (5)

P = cd - যাকে খুঁজব — কি খুঁজব — R

max = স্টপ S-R+1



format

1. Set K:=1 and max := S-R+1

K=1 থেকে, max পর্যন্ত complexity হবে(6)

2. Repeat step 3 to 5 while K ≤ max

3. Repeat for L=1 to R

Pattern এর length

if P[L] ≠ T [K+L-1]

1 [1,2,3,4] 1 একটা একটা ck করবে

4. Set index = K and exit — না মিললে বাইরে হবে

5. Set K:=K+1 —— যদি না পায় তবে ইন

① যোগ হবে.

6. index=0 set.

7. Exit.

Example

P = XXXXY.

$Q_0 = \epsilon$, $Q_1 = X$, $Q_2 = XX$ $Q_3 = XXX$ $Q_4 = XXXY$

$Q_5 = XXXXY = P$

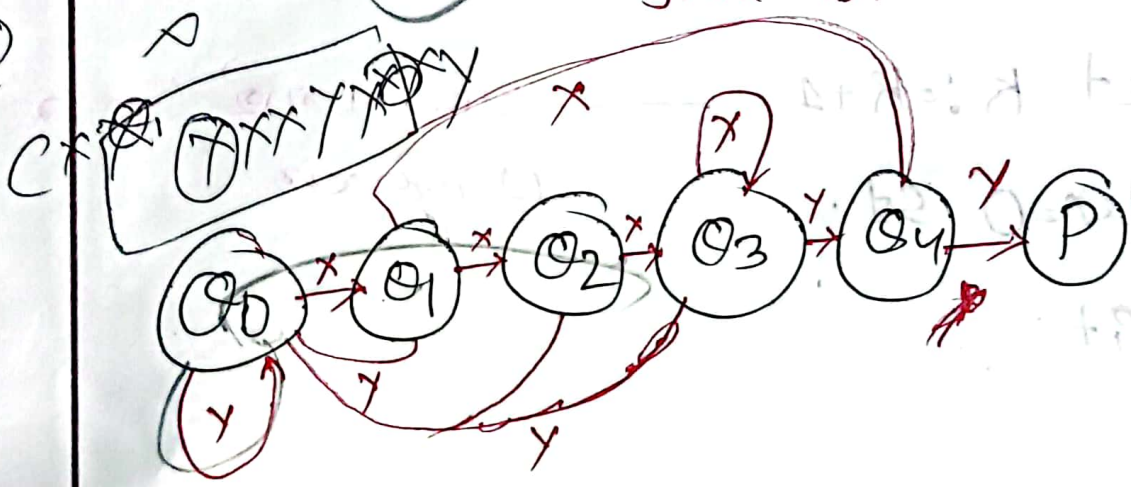
table.

$\epsilon \times Q_1$

- যদি ϵx থাকে
- ϵ নাই x থাকে $\Rightarrow Q_1$
- ফিল্ড না থাকলে Q_0.

Q	x	y
Q_0	Q_1	Q_0
Q_1	Q_2	Q_0
Q_2	Q_3	Q_0
Q_3	Q_3	Q_4
Q_4	Q_4	P

final state থাকে



xxxyxx

=

#Slow pattern matching

→ Length(T) − Length(P) + 1

T = (ab)5 P = abc

T = ababababab = Lof T = 10

P = abc = To P = 3

→ 10 − 3 + 1

⇒ 8.

find (3) টা P এর মধ্যে কতবার T থেকে 3 টা করে

abc এভাবে মিল নাই

1) aba = abc 3

2) bab = abc 1 6. bab = abc = 1

3. aba = abc 3 7. aba = abc 3

4. bab = abc 1 8. bab = abc 1

5. aba = abc 3

C = 3 + 1 + 3 + 1 + 3 + 1 + 3 + 1

= 16.