

2013

DATA STRUCTURES

JANUARY

IMPORTANT

Introduction

WEEK-01 | 002-363

WEDNESDAY

02

- Data → sets of values

Data item refers to single unit of values.



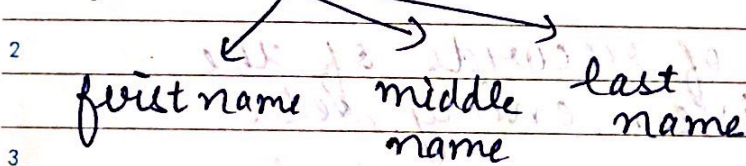
Group items

Elementary items

- Data items that are divided into subitems are called group items.

- Data items that are not divisible are called elementary items.

eg Employee's name



eg Social security number or Roll number.

* Basic Terminology :-

- * Entity → An entity is something that has certain attributes or properties which may be assigned values. The values may be

numeric

non-numeric

Attributes :	Name	Age	Gender
Values :	John	34	M

FEBRUARY

THINGS TO DO

* Entity Set :- Entities with similar attributes (eg. all the employees in an organization) form an entity set.

* Field → is a single elementary unit of information representing an attribute of an entity.

Record is the collection of field values of a given entity.

* File is the collection of records of the entities in a given entity set.

Records may be classified acc to length

fixed-length records variable-length records

• all the records contain the same data items with the same amount of space assigned to each data item.

• file-records may contain different lengths.

e.g. Student records usually have variable lengths, since different students take different numbers of courses.

DECEMBER '12							JANUARY						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
31					1	2		1	2	3	4		
3	4	5	6	7	8	9	7	8	9	10	11		
10	11	12	13	14	15	16	14	15	16	17	18		
17	18	19	20	21	22	23							

* Data Structures →

• The logical or mathematical model of a particular organization of data is called a data structure.

• The structure should be simple enough that one can effectively process the data when necessary.

* Array :-

• Simplest type of data structure is a linear array. (one-dimensional array)

• Linear array means a list of finite number n of similar or data elements referenced resp. by a set of n consecutive numbers, usually $1, 2, 3, \dots, n$.

If A is array, its elements can be written as: 1) Subscript notation

$A_1, A_2, A_3, \dots, A_n$

~~or~~ or

2) Parenthesis notation

$A(1), A(2), A(3), \dots, A(N)$

3) Bracket notation

$A[1], A[2], A[3], \dots, A[N]$

FEBRUARY '13

T	W	T	F	S	S
			1	2	3
5	6	7	8	9	10
12	13	14	15	16	17
19	20	21	22	23	24
26	27	28			

MARCH '13

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

THINGS TO DO

• Multi dimensional arrays :-
Such as matrices.

eg of linear Array: -

STUDENT	
1	John
2	Tom
3	Mary
4	Smith

STUDENT [1] = John
STUDENT [2] = Tom

eg of Two dimensional array :-

Store ↓	Dept.	1	2	3
1		40	100	40
2		50	10	60
3		70	20	100

name of array = SALES

SALES [1, 1] = 40

SALES [2, 1] = 50

Size of array = $m \times n$

= rows $\times$ columns

= 3×3

= 9

* LINKED LIST :- Array is a collection of similar data types that are stored into a common variable.

array [4] = [5 6 7 2]

DECEMBER '12							JANUARY						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
31					1	2		1	2	3	4		
3	4	5	6	7	8	9	7	8	9	10	11		
10	11	12	13	14	15	16	14	15	16	17	18		
17	18	19	20	21	22	23	21	22	23	24	25		

* Data Structure Operations:-

- 1 (1) Traversing:- Accessing each record exactly once so that certain items in the record may be processed. (Processing each element in the list.)
- 2 (2) Searching:- Finding the location of the record with a given key value, or finding the locations of all records which satisfy one or more conditions.
- 3 (3) Inserting:- Adding a new record to the structure.
- 4 (4) Deleting:- Removing a record from the structure.
- 5 (5) Sorting:- Arranging the records in some logical order.
eg: alphabetically acc to some

FEBRUARY '13						
M	T	W	T	F	S	S
					1	2
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

MARCH '13						
M	T	W	T	F	S	S
					1	2
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

THINGS TO DO

NAME Key

- (6) Merging:- Combining the records in two
[Combining two lists into a single list.]

different sorted files into a single sorted file.

* Data Types:-

- Each variable in any ~~of~~ algorithm or program has a data type which determines the code that is used for storing its value.

Data types are :-

- (i) Character :- Here data are coded using some character code such as ~~EXCITE~~ ASCII.

A =

- (ii) Integer :- Data are coded using int values.

- (iii) Floating :- decimal values.

- (iv) Logical :- variable can have only two values i.e. true or false.

13 SUNDAY or 1 for true, 0 for false.

* Algorithm:-

THINGS TO DO An algorithm is a well-defined list of steps for solving a particular problem.

- Time and space are two measures to calculate how efficient the algorithm is.

DECEMBER '12							JANUARY '13						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
31					1	2		1	2	3	4	5	6
3	4	5	6	7	8	9	7	8	9	10	11	12	13
10	11	12	13	14	15	16	14	15	16	17	18	19	20
17	18	19	20	21	22	23	21	22	23	24	25	26	27
24	25	26	27	28	29	30	28	29	30	31			

The complexity of an algorithm is the function which gives the running time and space in terms of the input size.

* Complexity of Algorithms :-

* The complexity of an algorithm M is the function $f(n)$ which gives the running time and / or storage space requirements of the algorithm in terms of the size n of the input data.

time $\rightarrow$ no. of comparisons

1) Worst case :- the maximum value of $f(n)$ for any possible input.

2) Average case :- the expected value of $f(n)$

eg Worst case :- Worst case occurs when ITEM is the last element in the array DATA or is not there at all.

$$C(n) = n$$

in case of linear search algo.

FEBRUARY '13						
M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

MARCH '13						
M	T	W	T	F	S	S
					2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

THINGS TO DO

eg Average Case :- we assume ITEM does appear in DATA.

and is equally likely to occur at any position in the array.

Probability $P = \frac{1}{n}$

$$C(n) = 1 \cdot \frac{1}{n} + 2 \cdot \frac{1}{n} + \dots + n \cdot \frac{1}{n}$$

$$= (1+2+\dots+n) \cdot \frac{1}{n}$$

$$= \frac{n(n+1)}{2} \cdot \frac{1}{n} = \frac{n+1}{2}$$

Thus, average no. of comparisons are needed to find the location of ITEM in DATA list:

Rate of Growth; Big O Notation

Suppose M is an algorithm, and n is the size of the input data.

Clearly the complexity $f(n)$ of M increases as n increases.

It is usually the rate of increase of $f(n)$ that we want to examine.

THINGS TO DO

This is done by comparing $f(n)$ with some standard function, such as

$\log_2 n$, n , $n \log_2 n$

DECEMBER						
M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JANUARY '13						
M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

$\log_2 n$ grows most slowly
 2^n grows most rapidly.

One way to compare the function $f(n)$ with these standard functions is to use the functional O notation defined as follows:-

$$8n^3 - 576n^2 + 832n - 248 = O(n^3)$$

Complexities :-

- 1) Linear search $\rightarrow O(n)$
- 2) Binary Search $\rightarrow O(\log n)$
- 3) Bubble sort $\rightarrow O(n^2)$
- 4) Merge-sort $\rightarrow O(n \log n)$

Big O notation defines an upper bound function ~~g(n)~~ which represents the time/space complexity of the algorithm on an input characteristic n .

Search :- Omega notation & theta notation
data structure vs data types.

* Difference b/w Data structure and data type?

6. A data structure is ~~an abstract~~ description of a way of organizing data to allow certain operations on it to be performed efficiently.

20 SUNDAY

7. A data type is a class of ~~com~~ objects that all share some property.

THINGS TO DO

eg. integer is a data type containing all of the infinitely many ~~store~~ integers.

DECEMBER '12						
M	T	W	T	F	S	S
31					1	2
3					8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JANUARY '13						
M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

* Trees? - Data frequently contain a hierarchical relationship b/w various elements.

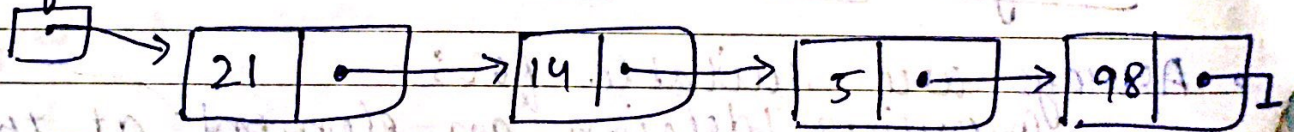
The data structure which reflects this relationship is called a rooted tree graph or simply a tree.

* Linked List: -

A linked list is a structure in which objects refer to the same kind of object, and where

the objects called nodes are linked in a linear sequence.

eg front or head



• Successive elements are connected by pointers.

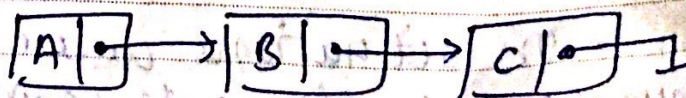
• Last element points to NULL.

• It can be made just as long as required.

• Must know the pointer to the first element of the list (called start, head, etc.).

• Insertion and Deletion to a linked list is done as follows:-

THINGS TO DO



[X] item to be inserted b/w A and B

FEBRUARY							MARCH						
M	T	W	T	F	S		M	T	W	T	F	S	
						1	2	3					
4	5	6	7	8	9	10	4	5	6	7	8	9	10
11	12	13	14	15	16	17	11	12	13	14	15	16	17
18	19	20	21	22	23	24	18	19	20	21	22	23	24
25	26	27	28				25	26	27	28	29	30	31

* Data Structure Types:- array, linked list,

a) Stack:- A stack, is also called a last-in first-out (LIFO) system,
It is a linear list in which insertions and deletions can take place only at one end, called the top.

eg Stack of ^{books} ~~diaries~~. in which ~~diaries~~ books are inserted only at the top of the stack and books can be deleted only from the top of the stack.

b) Queue:- A queue, also called as first-in first-out (FIFO) system,
It is a linear list in which deletions can take place only at one end of the list, the front of the list and insertions can take place only at the other end of the list, the 'rear' end of the list.

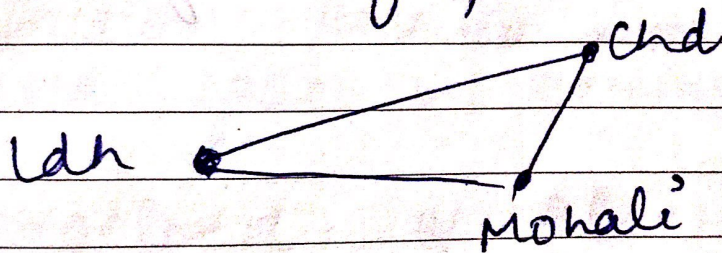
eg: line of people waiting at a bus stop.

c) Graph:- Data sometimes contain a relationship b/w pairs of elements which is not necessarily hierarchical in nature.

eg suppose an airline flies only b/w the cities connected by lines

DECEMBER '12							JANUARY '13						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
31													
3	4	5	6	7	8	9	1	2	3	4	5	6	
10	11	12	13	14	15	16	7	8	9	10	11	12	13
17	18	19	20	21	22	23	14	15	16	17	18	19	20
24	25	26	27	28	29	30	21	22	23	24	25	26	27
							28	29	30	31			

This type of data structure is called a graph.



• Data Structures are classified into

linear

nonlinear

• If elements of data structure forms a sequence or a linear list.

• Such as trees and graphs.

• There are two ways of representing such linear structures in memory.

To have the linear relationship b/w the elements represented by means of sequential memory locations such as arrays.

To have the linear relationship b/w the elements represented by means of pointers or links. such as linked list.

* LINEAR ARRAY:-

A linear array is a list of finite number n of homogeneous data elements (i.e. data elements of the same type) such that:-

1) The elements of the array are referenced resp. by an index set consisting of n consecutive numbers.

2) The elements of the array are stored resp. in successive memory locations.

THINGS TO DO

FEBRUARY '13						
M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10

MARCH '13						
M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10

The number n of elements is called the length or size of the array.

$$\text{length} = \text{UB} - \text{LB} + 1$$

To find the number of data elements in array.

UB $\rightarrow$ Upper bound is the largest index
LB $\rightarrow$ Lower bound is the smallest index of the array.

$$\text{Length} = \text{UB} \quad ; \quad \text{if } \text{LB} = 1$$

Example: $4-3$

* Automobile company uses an array AUTO to record the number of automobiles sold each year from 1932 through 1984.

AUTO[K] = no. of automobiles sold in the year K.

$$\text{LB} = 1932, \quad \text{UB} = 1984$$

$$\text{Length} = \text{UB} - \text{LB} + 1$$

$$= 1984 - 1932 + 1$$

Thus AUTO contains 55 elements.

THINGS TO DO

* Representation of linear Arrays in memory :-

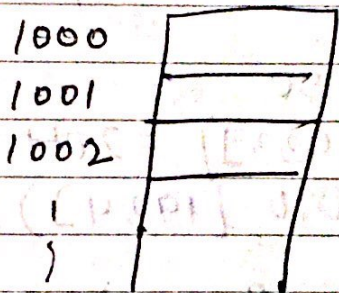
LA = Linear Array

DECEMBER						
	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JANUARY						
M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27

- Memory of the computer is simply a sequence of addressed locations.

$LOC(LA[K])$ = address of element $LA[K]$ of the array LA .



Computer Memory.

Elements of LA are stored in successive memory cells.

Computer does not need to keep track of the address of every element of LA , but needs to keep track only the address of first element of LA denoted by $Base(LA)$
Base address.

Computer calculates the address of any element of LA by the following formula:-

$$LOC(LA[K]) = Base(LA) + w(K - \text{Lower bound})$$

$w \rightarrow$ no. of words per memory cell for the array LA .

Ques: Consider the array AUTO :-
~~records~~ the no. of automobiles sold
 each year from 1932 through 1984.

Base (AUTO) = 200, $w = 4$ words per memory cell

$$\text{LOC}(\text{AUTO}[1932]) = 200$$

$$\text{LOC}(\text{AUTO}[1933]) = 204$$

$$\text{LOC}(\text{AUTO}[1934]) = 208 \dots$$

Ans: $\text{LOC}(\text{AUTO}[1965]) = \text{Base}(\text{AUTO}) + w(1965 - \text{lower bound})$

$$= 200 + 4(1965 - 1932) = 332$$

* Operations On Linear Array :-

① Traversing Linear Array :-

LA is a linear array with lower bound LB and upper bound UB.

This algo traverse LA applying an operation PROCESS to each element of LA.

- 1° [Initialize counter] Set $K := LB$
- 2° Repeat steps 3 and 4 while $K \leq UB$

- 3° [Visit element] Apply PROCESS to LA[K]

- 4° [Increase counter] Set $K := K + 1$

DECEMBER '12						
M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JANUARY						
M	T	W	T	F	S	S
	1	2	3			
7	8	9	10	1		
14	15	16	17	2		
21	22	23	24	3		
28	29	30	31			

[End of step 2 loop]
5. Exit.

WEEK-05 028-337
MONDAY

28

② Insert an element to Linear Array:-

INSERT (LA, N, K, ITEM)
LA is Linear Array with N elements
K is a +ve integer such that $K \leq N$

This algo inserts an element ITEM into Kth position in LA with N elements.

1. [Initialize counter] Set $J := N$
2. Repeat steps 3 and 4 while $J \geq K$
3. [Move Jth element downward]
Set $LA[J+1] := LA[J]$
4. [Decrease counter] Set $J := J - 1$
- [End of Step 2 loop]
5. [Insert element] Set $LA[K] := ITEM$
6. [Set N.] Set $N := N + 1$
7. Exit.



Description:- ① Create space in LA by moving downward one location

THINGS TO DO each element from the Kth position.

② These elements are moved in reverse order

FEBRUARY '13							MARCH '13						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
					1	2	1					1	2
4	5	6	7	8	9	10	4	5	6	7	8	9	10
11	12	13	14	15	16	17	11	12	13	14	15	16	17
18	19	20	21	22	23	24	18	19	20	21	22	23	24
25	26	27	28				25	26	27	28	29	30	31

i.e. first $LA[N]$, then $LA[N-1]$, ... last $LA[K]$;

otherwise data might be erased.

③ First set $J := N$

J as a counter decrease J each time the loop is executed until J reaches K .

④ ~~insert~~ Insert $GTEM$ into the array in the space just created. Before the exit from algo, the number N of elements in LA is increased by 1 to account for the new element.

③ Deletion: - DELETE ($LA, N, K, GTEM$)

Here LA is a linear array with N elements and K is a +ve integer such that $K \leq N$.

This algo deletes the K^{th} element from LA and assigns it to a variable $GTEM$.

1. Set $GTEM := LA[K]$

2. Repeat for $J = K$ to $N-1$:

[Move $J+1^{st}$ element upward]

set $LA[J] := LA[J+1]$

THINGS TO DO

[End of loop]

3. [Reset the number N of elements in LA]
Set $N := N - 1$

4. End.

DECEMBER '12							JANUARY '13						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	2	3	4	5	6	7	1	2	3	4	5	6	7
8	9	10	11	12	13	14	8	9	10	11	12	13	14
15	16	17	18	19	20	21	15	16	17	18	19	20	21
22	23	24	25	26	27	28	22	23	24	25	26	27	28
29	30	31					29	30	31				

* Linear Search

A linear array DATA with N elements and a specific ITEM of information are given.

This algorithm finds the location LOC of ITEM in the array DATA or sets $LOC = 0$.

(1) [Initialize] Set $K := 1$ and $LOC := 0$

(2) Repeat steps 3 and 4 while $LOC = 0$ and $K \leq N$

(3) If $ITEM = DATA[K]$ then Set $LOC := K$

(4) Set $K := K + 1$ [increments counter]
[End of step 2 loop]

(5) [Successful?]

If $LOC = 0$ then:
Write: ITEM is not in the array DATA

Else:

Write: LOC is the location of ITEM
[End of if structure]

(6) Exit.

THINGS TO DO

Best case complexity = $O(1)$

Avg case = $O(\frac{n+1}{2})$

Worst case = $O(n)$

APRIL '13						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY			
M	T	W	T
	1	2	
6	7	8	9
13	14	15	16
20	21	22	23
27	28	29	30

* BINARY SEARCH

BINARY (DATA, LB, UB, ITEM, LOC)

Here DATA is a sorted array with LB and UB + ITEM is a given item of information.

The variables BEG, END and MID denote resp. the beginning, end and middle locations of a segment of elements of DATA.

This algo finds the location LOC of ITEM in DATA or sets LOC = NULL.

① [Initialize segment variables]

Set $BEG := LB$, $END := UB$

$MID = INT((BEG + END) / 2)$

② Repeat steps 3 and 4 while $BEG \leq END$ and $DATA[MID] \neq ITEM$

③ If $ITEM < DATA[MID]$ then
Set $END := MID - 1$

Else Set $BEG := MID + 1$

[End of 2nd structure]

THINGS TO DO

④ Set $MID = INT((BEG + END) / 2)$

[End of step 2 loop]

JUNE '13						
M	T	W	T	F	S	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JULY '13						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

⑤ If $DATA[MID] = GTEM$ then
Set $LOC := MID$

Else:

Set $LOC := NULL$

[End of structure]

⑥ Exit.

eg $DATA = 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99$

(a) $GTEM = 40$

⑩ $BEG = 1$ $END = 13$

$$MID = \frac{1+13}{2} = 7$$

$DATA[7] = 55$

$40 < 55$

so $END = MID - 1 = 6$

$$MID = \frac{1+6}{2} = 3$$

$DATA[3] = 30$

THINGS TO DO

Since $40 > 30$

so $BEG = MID + 1$

$END = 6$

APRIL '13						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

MAY						
M	T	W	T	F	S	S
		1	2	3	4	
6	7	8	9	10	11	
13	14	15	16	17	18	
20	21	22	23	24	25	
27	28	29	30	31		

$$MID = \frac{4+6}{2} = 5$$

$$DATA[MID] = 40$$

$$LOC = MID = 5$$

(b) Suppose $ITEM = 85$
 $BEG = 1$ $END = 13$

$$MID = \frac{1+13}{2} = 7$$

$$DATA[MID] = 55$$

$$85 > 55$$

$$SO \quad BEG = MID + 1 = 8$$

$$MID = \frac{8+13}{2} = 10$$

$$DATA[MID] = 77$$

$$85 > 77$$

$$BEG = MID + 1 = 11$$

$$MID = INT \left[\frac{(11+13)}{2} \right] = 12$$

$$DATA[MID] = 88$$

$$85 < 88$$

$$END = MID - 1 = 11$$

$$MID = \frac{11+11}{2} = 11$$

JUNE '13						
M	T	W	T	F	S	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

JULY '13						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

THINGS TO DO

11

131-234 WEEK-19

SATURDAY

2013
IMPORTANT $DATA[MID] = 80$ $85 > 80$ $BEG = MID + 1 = 12$ Since $BEG > END$ Hence, $ITEM$ does not belong to $DATA$. $Complexity = O(\log n)$ Limitations of Binary Search Algo:-

- ① List must be sorted.
- ② One must have direct access to the middle element in any sublist.

④ Sorting an Array :-

Bubble sort :- in ascending order
 compare 1, 2 and ~~interchange~~ $1 < 2$
 $2, 3, 2 < 3$
 $3, 4, 3 < 4$
 Pass = One less than total elements

Bubble (DATA, N)

Here DATA is an array with N elements.

This algo sorts the elements in DATA.

- 1° Repeat steps 2 and 3 for $K=1$ to $N-1$
2. Set $PTR := 1$ [initializes pass pointer PTR]
3. Repeat while $PTR \leq N-K$: [Executes pass].
 - a) If $DATA[PTR] > DATA[PTR+1]$; then
 interchange $DATA[PTR]$ and $DATA[PTR+1]$
 - [End of if structure]
 - b) Set $PTR := PTR + 1$.
 - [End of inner loop]
- [End of step 1 outer loop]

THINGS TO DO

Complexity = $O(n^2)$

FEBRUARY '13						
M	T	W	T	F	S	S
					1	2
					3	

MARCH '13						
M	T	W	T	F	S	S
					1	2
					3	

* Multidimensional Arrays:-

Arrays where elements are referenced by two or ~~more~~ three subscripts are called multi-dimensional arrays.

Two dimensional Arrays:-

A two dimensional array $m \times n$ array A is a collection of $m \cdot n$ data elements such that each element is specified by a pair of integers called subscripts (such as J, K).

element of A $A[J, K]$ can be denoted as:-
 $1 \leq J \leq m$ and $1 \leq K \leq n$

Two dimensional arrays are also called matrices in Maths.

Size of array = $m \times n$

length of 1 dimension = $UB - LB + 1$

* Representation of Two-Dimensional Arrays in Memory:-

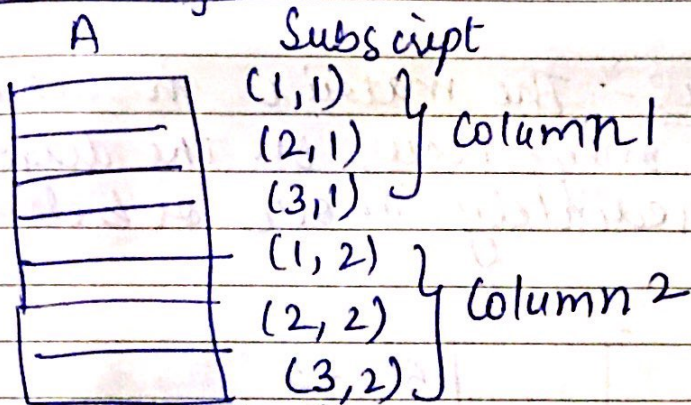
Let A be 2 dimensional $m \times n$ array. It can be represented either column by column or row by row.

JANUARY '13						
M	T	W	T	F	S	S
				4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

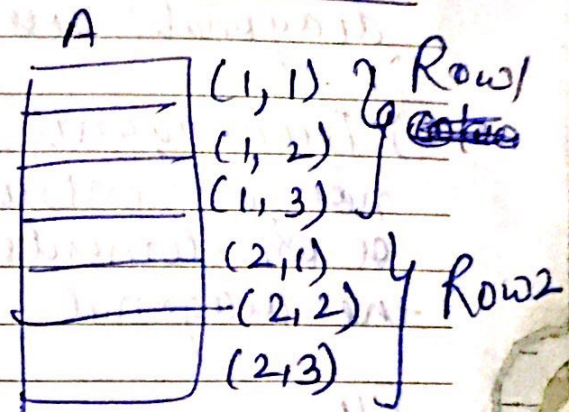
FEBRUARY						
M	T	W	T	F	S	S
						1
4	5	6	7	8		
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28			

column which is called column-major order or row by row which is called row-major order.

Column major order.



Row-major order.



Computer just stores the address of 1st element and find address of any other element.

(Column-major order) →

$$LOC(A[J,K]) = Base(A) + w [M(K-1) + (J-1)]$$

Row-major order →

$$LOC(A[J,K]) = Base(A) + w [N(J-1) + (K-1)]$$

4.12 example from pg. 21 Book.

* Sparse Matrices :-

Matrices with a relatively high proportion of zero entries are called sparse matrices.

THINGS TO DO

There exists 2 types of sparse matrices :-

MARCH '13							APRIL '13						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
				1	2	3	1	2	3	4	5	6	7
4	5	6	7	8	9	10	8	9	10	11	12	13	14
11	12	13	14	15	16	17	15	16	17	18	19	20	21
18	19	20	21	22	23	24	22	23	24	25	26	27	28
25	26	27	28	29	30	31	29	30					

a) Triangular matrix:- The matrix in which all entries ~~are~~ above the main diagonal are zero. or where nonzero entries can only occur on or below the main diagonal ~~are~~.

b) Tridiagonal Matrix:- The matrix in which nonzero entries can only occur on the diagonal or on elements immediately above or below the diagonal.

$$\begin{bmatrix} 4 & & & \\ 3 & -5 & & \\ 1 & 0 & 6 & \\ -7 & 8 & -1 & 3 \\ 5 & -2 & 0 & 2 & -8 \end{bmatrix}$$

Triangular matrix

$$\begin{bmatrix} 5 & -3 & & & \\ 1 & 4 & 3 & & \\ & 9 & -3 & 6 & \\ & & 2 & 4 & -7 \\ & & & 3 & -10 \\ & & & & 6 & -5 & 8 \\ & & & & & & -3 & -1 \end{bmatrix}$$

Tridiagonal matrix.

* Consider the 25×4 matrix array SCORE

Suppose $\text{Base}(\text{SCORE}) = 200$ and there are $w = 4$ words per memory cell.

furthermore, suppose the programming language stores 2-dimensional arrays using row-major order.

Then find the address of

3rd test of 12th student.

JANUARY '13													FEBRUARY '13						
SCORE													T W T F S S						
1	2	3	4	5	6														
7	8	9	10	11	12	13													
14	15	16	17	18	19	20							4	5	6	7	8	9	10
21	22	23	24	25	26	27							11	12	13	14	15	16	17
28	29	30	31										18	19	20	21	22	23	24
													25	26	27	28			

$$\text{LOC}(\text{SCORE}[12,3]) = 200 + 4[4(12-1) + (3-1)]$$

$$= 200 + 4[46] = \underline{\underline{384}} \text{ Ans.}$$

* Space-Time Tradeoffs

MONDAY

06

Complexity of an algo is a function $f(n)$ which measures the time and/or space used by an algo in terms of the input size n .

The space-time tradeoff refers to a choice b/w algorithmic solutions of a data processing problem that, allows one to decrease the running time of an algorithmic solution by increasing the space to store the data and vice-versa.

10
Examples:-

Operations on Linear Array

11
(1) TRAVERSING:- accessing each element of an array exactly once.

Suppose we want to print all the elements of an array.
let LA be a linear array with N elements.

eg K is a counter

A	B	C	D	E
1	2	3	4	5

Name of an array = LA
LB = 1
UB = 5

Step 1 Set $K := 1$

28 SUNDAY Step 2 Repeat while $K \leq 5$

Step 3 Process LA[1]
Print A

THINGS TO DO Step 4 Set $K := K + 1$

$K := 2$

$2 \leq 5$ ✓

Step 5 Process LA[2]
Print B

JUNE '13							JULY						
M	T	W	T	F	S	S	M	T	W	T	F	S	S
3	4	5	6	7	8	9	8	9	10	11	12	13	14
10	11	12	13	14	15	16	15	16	17	18	19	20	21
17	18	19	20	21	22	23	22	23	24	25	26	27	28
24	25	26	27	28	29	30	29	30	31				

Step 6 Set $K := K + 1$

Now $K := 3$

Check condition: $- 3 \leq 5$

condition true

Step 7

Process $LA[3]$

Print C

Step 6

Set $K := K + 1$

Now $K = 4$

check condition: $- 4 \leq 5$ condition true

Step 7

Process $LA[4]$

Print D

Step 8

Set $K := K + 1$

Now $K = 5$

Check condition: $- 5 \leq 5$

Condition true

Step 9

Process $LA[5]$

Print E

Step 10

Set $K := K + 1$, $K = 6$

Check condition: $-$

Now $6 \leq 5$

~~condition~~

condition false

THINGS TO DO

Step 11

End of while loop.

Step 12

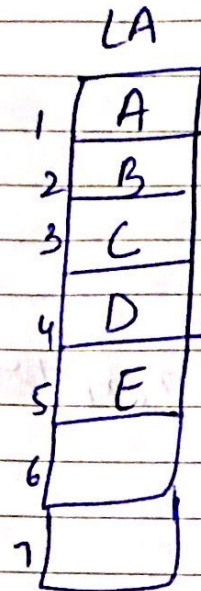
Exit.

AUGUST '13						
M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEPTEMBER '13						
M	T	W	T	F	S	S
30						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

(2) INSERTION AT K^{th} position in a linear array :-

let us take an array of 5 data elements.



$$K = 3$$

$$\text{ITEM} = Z$$

J is counter
 $N = 5$

(1)

Set $J := 5$

(2)

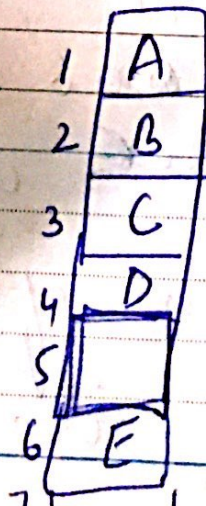
while $5 \geq 3$

condition true

(3)

$LA[6] := LA[5]$

$LA[6] := E$



THINGS TO DO

JUNE							'13
M	T	W	T	F	S	S	
					1	2	
3	4	5	6	7	8	9	
10	11	12	13	14	15	16	
17	18	19	20	21	22	23	
24	25	26	27	28	29	30	

JULY							
M	T	W	T	F	S	S	
1	2	3	4				
8	9	10	11				
15	16	17	18				
22	23	24	25				
29	30	31					

(4) Decrement counter

$$\begin{aligned} \text{Set } J &:= J - 1 \\ &= 5 - 1 \\ J &:= 4 \end{aligned}$$

(5) check condition 4 > 3

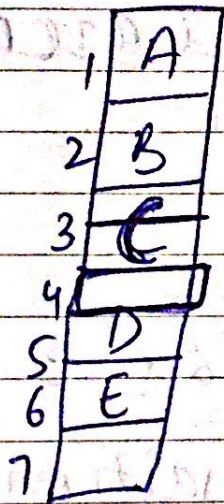
condition true

(6)

$$LA[5] := LA[4]$$

shift element downwards

$$\text{Now } LA[5] := D$$



(7)

$$\text{Set } J := J - 1 = 3$$

(8)

$$\text{Check } 3 > 3$$

condition true

(9)

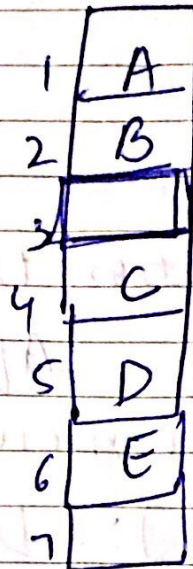
$$\text{THINGS TO DO } LA[4] := LA[3]$$

shift element downwards

AUGUST '13						
M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

SEPTEMBER '13						
M	T	W	T	F	S	S
30						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

Now $LA[4] = C$



we want to

← insert
ITEM
here.

(10)

~~Set~~ $LA[K] := \text{ITEM}$
Decrement counter

Set $J := J - 1$

$= 3 - 1 = 2$

check condition $2 \leq 3$

condition false

end of while
loop here

(11)

Now insert element at $K=3$

Set $LA[3] := \text{ITEM}$

THINGS TO DO

(12)

Set $N := N + 1$
 $N := 6$

JULY							'13	
M	T	W	T	F	S	S		
1	2	3	4	5	6	7		
8	9	10	11	12	13	14		
15	16	17	18	19	20	21		
22	23	24	25	26	27	28		

AUGUST								
M	T	W	T	F	S	S		
							1	2
5	6	7	8	9	10	11		
12	13	14	15	16	17	18		
19	20	21	22	23	24	25		

FRIDAY

Resultant array after insertion

1	A
2	B
3	Z
4	C
5	D
6	E
7	

③ Insertion at end of an array.

LA

1	A
2	B
3	C
4	D
5	

Name of an array LA
 $N = 4$

ITEM = Z

① Set $i := 4$

② Set $LA[i+1] := \text{ITEM}$

$LA[4+1] := Z$

Insertion $\rightarrow LA[5] := Z$

③ Reset N

$N := N + 1$

$= 4 + 1$

$N = 5$

④ Exit.

Resultant array:

1	A
2	B
3	C
4	D
5	Z

SUNDAY 04

THINGS TO DO

SEPTEMBER '13	OCTOBER '13
M T W T F S S	M T W T F S S
30 1	1 2 3 4 5 6
2 3 4 5 6 7 8	7 8 9 10 11 12 13
9 10 11 12 13 14 15	14 15 16 17 18 19 20
16 17 18 19 20 21 22	21 22 23 24 25 26 27
23 24 25 26 27 28 29	28 29 30 31

AUGUST

07

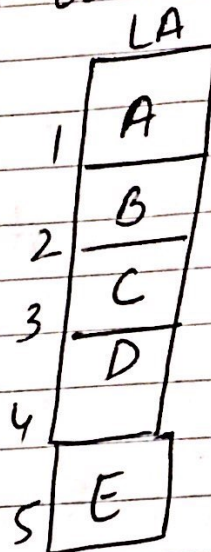
219-146 WEEK-32

WEDNESDAY

2013
IMPORTANT

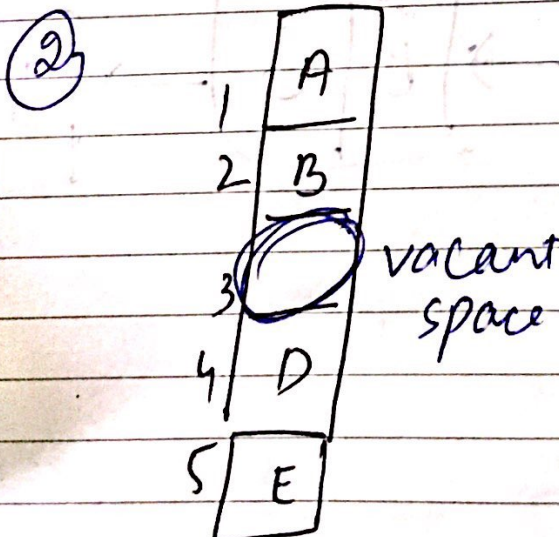
DELETION AT K^{th} position in
an array:

let us take an array of 5 elements



$K = 3$

① Set $\text{ITEM} := \text{LA}[3]$
i.e. $\text{ITEM} := C$



for $J := K$ to $N - 1$
set $J = 3$

$\text{LA}[3] := \text{SetA}[4]$

i.e. $\text{LA}[3] := D$

1	A
2	B
3	D
4	
5	E

Vacant
space

(3) Now increment counter
Set $J := J + 1$

$$J = 4$$

Set $LA[4] := LA[5]$

i.e. $LA[4] = E$

Resultant array:

1	A
2	B
3	D
4	E
5	

LINEAR SEARCH IN AN ARRAY

DATA:-

A	B	C	D	E
1	2	3	4	5

$$N = 5$$

Search ITEM = D

LOC is location for ITEM

① set $K := 1$ and $LOC := 0$

② while $LOC = 0$ and $K \leq N$

③ if $ITEM = DATA[K]$
Set $LOC := K$

Check if $ITEM = DATA[1]$
but $DATA[1] = A$

thus $ITEM \neq DATA[1]$

④ Set $K := K + 1$
 $K := 2$

⑤ check if $ITEM = DATA[2]$

But $DATA[2] = B$

Thus $ITEM \neq DATA[2]$

⑥ Increment counter

Set $K := K + 1$

$K := 3$

$K \leq N$ ✓

check condition

$3 \leq 5$

⑦ Check ~~if~~^{at} $DATA[3] =$

But $DATA[3] = C$

Thus increment counter

⑧

$K := 4$

$4 \leq 5$

condition true

⑨

$ITEM = DATA[4]$

as, $DATA[4] = D$

Thus $LOC := K$

THINGS TO DO

$LOC := 4$

4 is location of item

ER	'13	OCTOBER	'13
T F S S	M T W T F S S		
1	1 2 3 4 5 6		
5 6 7 8	7 8 9 10 11 12 13		
12 13 14 15	14 15 16 17 18 19 20		

SUNDAY 11

LINEAR search

eg for unsuccessful search

DATA

A	B	C	D
1	2	3	4

$$N = 4$$

Search ITEM = M

① Set $K := 1$ and $LOC := 0$

② while $LOC = 0$ and $K \leq N$

③ if $ITEM = DATA[K]$
Set $LOC := K$

~~DATA[1] = A~~

ITEM not found
so increment counter

④ Set $K := K + 1 = 2$

check condition $2 \leq 4$

so check $DATA[2] = B$

THINGS TO DO

JULY

13 AUGUST

Item not found

(5) Thus increment counter
K is now 3

$3 \leq 4$ condition true

But DATA (3) = C
ITEM not found

(6) increment counter
 $K := K + 1$
 $= 4$

$4 \leq 4$ condition true

But DATA (4) = D

Item not found

(7) increment counter

$K := 5$

$5 \leq 4$

condition false

exit from loop

SEPTEMBER '13							OCTOBER '13						
T	W	T	F	S	S		M	T	W	T	F	S	S
						1		1	2	3	4	5	6
3	4	5	6	7	8		7	8	9	10	11	12	13
10	11	12	13	14	15		14	15	16	17	18	19	20
17	18	19	20	21	22		21	22	23	24	25	26	27
24	25	26	27	28	29		28	29	30	31			

THINGS TO DO

(8)

LOC = 0

Item is not found
in array DATA

* ASYMPTOTIC NOTATIONS

Execution time of an algo depends on the instⁿ set, processor speed, disk I/O speed etc. Hence, we estimate the efficiency of an algo asymptotically.

Time function of an algo is represented by $T(n)$ where n is the output size.

Following asymptotic notations are used to calculate the running time complexity of an algorithm.

O - Big Oh

Ω - Big omega

Θ - Big theta

o - little oh

ω - little omega.

① Big Oh (O)

° It is commonly used notation.

° A function $f(n)$ can be represented is the order of $g(n)$ that is $O(g(n))$, if there exists a value of positive integer n as n_0 and a positive constant C , such that :-

MARCH 2014						
M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

$$f(n) \leq c \cdot g(n) \quad \text{for } n > n_0 \quad \text{in all case}$$

Hence function $g(n)$ is an upper bound for function $f(n)$, as $g(n)$ grows faster than $f(n)$.

• Big O notation defines an upper bound function.

② OMEGA NOTATION (Ω)

It is used when the function $g(n)$ defines a lower bound for the function $f(n)$.

$$f(n) = \Omega(g(n))$$

iff there exists a +ve integer n_0 and a +ve number c , such that

$$|f(n)| \geq c|g(n)| \quad \text{for all } n \geq n_0$$

③ Theta Notation (Θ): -

It is used when the function $f(n)$ is bounded both from above and below by the function $g(n)$.

$$f(n) = \Theta(g(n))$$

iff there exist two +ve constants c_1 and c_2 and a +ve integer n_0 such that

FEBRUARY 2014						
M	T	W	T	F	S	S
					1	2
					8	9
3	4	5	6	7	14	15
10	11	12	13	14	21	22
17	18	19	20	21	28	
24	25	26	27	28		

Wednesday

57-308 Week 9

40

February

$$c_1 |g(n)| \leq |f(n)| \leq c_2 |g(n)|$$

for all $n \geq n_0$

* Characteristics of an Algorithm:-

- ① Input / Output:- whenever any input is given then it should produce an output.
- ② Correctness:- Every step must be logically correct.
- ③ Definiteness:- It should give the same (fix) result at any instance of time
eg = $3+5=8$ (all time)
- ④ Effectiveness:- The same algorithm can be executed on any pen and paper. It should be executable on all machines.
- ⑤ Generality:- It should be applicable to classes of problems eg - $\frac{1}{2} \times b \times h$
(for all right angled triangle)
- ⑥ Finiteness:- No. of steps should be finite and result should be get in finite time.