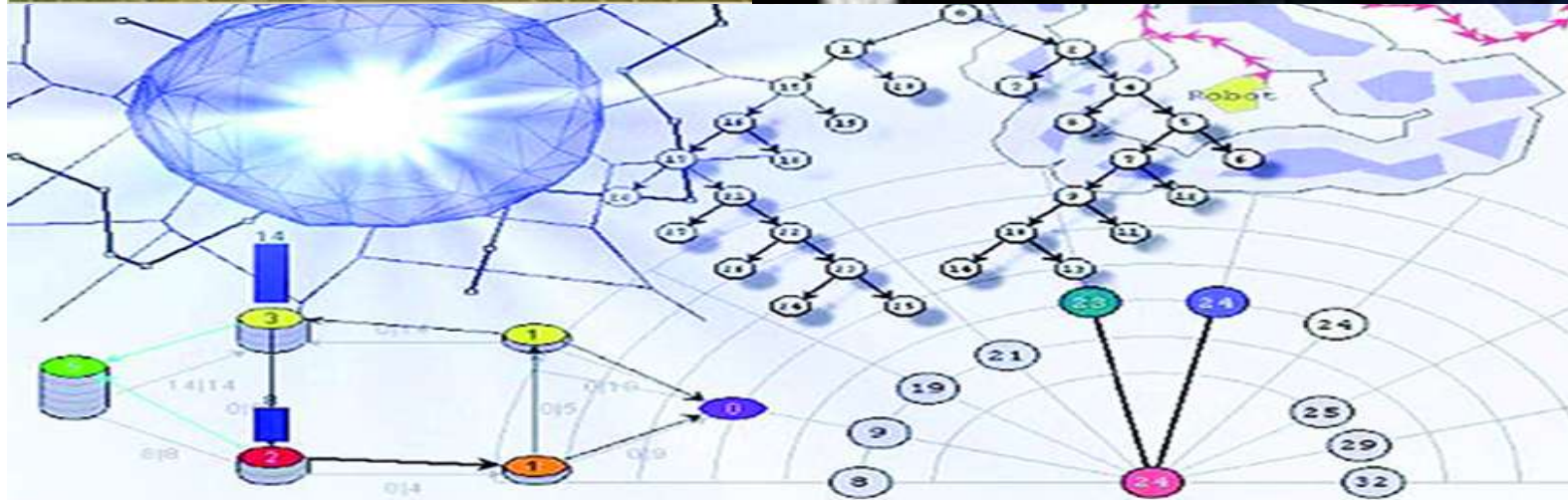# DATA STRUCTURES

Study Outline

ABU SALEH MUSA MIAH(ABID)

A Guidebook of

Data Structure

Engr. Abu Saleh Musa Miah (Abid)
Lecturer, Rajshahi Engineering Science and Technology College(RESTC)
B.Sc.Engg. in Computer Science and Engineering(**First Class First**)
M.Sc.Engg.(CSE-Pursuing),University of Rajshahi.
Email:abusalehcse.ru@gmail.com
Cell:+88-01734264899

Data Structure

===

Writer	:	**Engr. Abu Saleh Musa Miah (Abid).**
		B.Sc.Engg. in Computer Science and Engineering(**First Class First**)
		M.Sc.Engg.(CSE-Pursuing),University of Rajshahi.
		Cell:+88-01734264899
Email	:	abusalehcse.ru@gmail.com
Publisher	:	**Abu Syed Md Mominul Karim Masum.**
		Chemical Engr.
		Chief Executive Officer (CEO)
		Swarm Fashion, Bangladesh.
		Mohakhali,Dhaka.
Email	:	**swarm.fashion@gmail.com**
Cover page Design	:	Md. Kaiyum Nuri
		Chief Executive Officer (CEO)
		Jia Shah Rich Group(Textile)
		MBA,Uttara University
First Publication	:	**Octobor-2016**
Copyright	:	**Writer**
Computer Compose	:	**Writer**
Print	:	**Royal Engineering Press & Publications.**
		Meherchandi, Padma Residential Area, Boalia, Rajshahi.
Reviewer Team	:	Engr. Syed Mir Talha Zobaed.
		B.Sc. Engg. (First Class First)
		M.Sc. Engineering (CSE)
		University of Rajshahi
		Omar Faruk Khan (Sabbir)
		M. Engineering (CSE) University of Rajshahi
PRICE	:	**400.00 TAKA (Fixed Price).**
		US 05 Dollars (**Fixed Price**).

OOP CPP	:	Engr. Abu Saleh Musa Miah (Abid).
Published by	:	Rajshahi Engineering Science and Technology College.
		Mirjapur,Binodpur,Rajshahi.

Dedicated

To

All my Student's of Rajshahi Engineering Science and Technology college.

[This Page is Left Blank Intentionally]

লেখকের কথা

Data Structure কোর্সটি বাংলাদেশের প্রায় সকল বিশ্ববিদ্যালয়ের সিলেবাসে স্থান লাভ করেছে,বাজারে টেক্সটবই থাকলেও গভীর ভাবে অনুধাবন ও পরীক্ষায় ভাল মার্কস উঠানোর মত সাজানো টেক্সটবই সহজলভ্য নয় । পরীক্ষার প্রস্তুতির স্বার্থে তাই, এই গাইডবইটি আপনাকে সহযোগিতা করবে বলে আশা করা যায় ।

এই বইয়ের পাঠক হিসেবে, আপনিই হচ্ছেন সবচেয়ে গুরুত্বপূর্ণ সমালোচক বা মন্তব্যকারী । আর আপনাদের মন্তব্য আমার কাছে মূল্যবান, কারন আপনিই বলতে পারবেন আপনার উপযোগী করে বইটি লেখা হলো কিনা অর্থাৎ বইটি কিভাবে প্রকাশিত হলে আরও ভাল হতো । সামগ্রিক ব্যাপারে আপনাদের যে কোন পরামর্শ আমাকে উৎসাহিত করবে ।

Engr. Abu Saleh Musa Miah (Abid)
Writer

ACKNOWLEDGEMENTS

I wish to express my profound gratitude to all those who helped in making this book a reality; especially to my families, the classmates, and authority of Royal Engineering Publications for their constant motivation and selfless support. Much needed moral support and encouragement was provided on numerous occasions by my families and relatives. I will always be grateful, to the numerous web resources, anonymous tutorials hand-outs and books I used for the resources and concepts, which helped me build the foundation.

I would also like to express my profound gratitude to Engr. Syed Mir Talha Zobaed for his outstanding contribution in inspiring, editing and proofreading of this guidebook. I am also grateful to Omar Faruque Khan (Sabbir) for his relentless support and guideline in making this book a reality.

I am thankful to the following readers those have invested their valuable times to read this book carefully and have given suggestion to improve this book:
Abu Hurayra (Principle,RESTC)
Engr. Mainuddun Maruf (B.Sc. Engg in Electrical and Electronics Engineering. IUT) .
Md.Moyeed Hossain (B.Sc.Engr. Computer Science and Engineering ,RUET, Lecturere, RESTC).
Md.Shamim Akhtar (B.A Honours,English,M.A. Lecturere, RESTC).
MD.Sharafat Hossain(B.Sc. Engg in Electrical and Electronics Engineering. RUET).
………………………….. And numerous anonymous readers.
I am also thankful to the different hand notes from where I have used lots of solutions, such as Dynamic Memory Allocation,Operator overloading etc

I wish to express my profound gratitude to the following writers whose books I have used in my Guidebooks:

1.	E. Horowitz and S. Sahni	:	Fundamentals of Data Structures, *Galgotia.*
2.	Edward M. Reingold &Wilfred J. Hansen	:	Data Structures, *Addison Wesley Publishers*
3.	Niklaus Wirth	:	Algorithms + Data Structures = Programs, *Prentice Hall*
4.	Robert L. Kruse	:	Data Structures and Program Design, *Prentice Hall*
5.	Seymour Lipshultz	:	Data Structures (Schaum's Outline Series), *Tata McGraw-Hill*
6.	E. Horowitz and S. Sahni	:	Computer Algorithms, *Galgotia.*
7.	Seymour E. Goodman & S. T. Hedetniemi	:	Introduction to Design and Analysis of Algorithms, *McGraw-Hill.*

……………………... and Numerous anonymous Power Point Slides and PDF chapters from different North American Universities.

Data Structure
Syllabus

CSE2121: Data Structure
75 Marks [70% Exam, 20% Quizzes/Class Tests, 10% Attendance]
3 Credits, 33 Contact hours, Exam. Time: 4 hours

Arrays: Maximization, ordered lists, sparse matrices, representation of arrays.

Stacks, Queues and Recursion: Different types of stacks and queues: Circular, dequeues, etc; evaluation of expressions, multiple stacks and queues;

Recursion: Direct and indirect recursion, depth of recursion; Simulation of Recursion, Removal of recursion; Towers of Hanoi.

Links Lists: singly linked lists, linked stacks and queues, the storage pool, polynomial addition, equivalence relations, sparse matrices, doubly linked lists and dynamic storage management, generalized lists, garbage collection and compaction.

Trees: Basic terminology, binary trees, binary tree representations, binary tree traversal; Extended binary trees: 2-trees, internal and external path lengths, Huffman codes/algorithms; threaded binary trees, binary tree representation of trees; Application of Trees: Set representation, decision trees, games trees: Counting binary trees.

Graphs: Introduction, definitions and terminology, graph representations, traversals, connected components and spanning trees, shortest path and transitive closure, activity networks, topological sort and critical paths, enumerating all paths.

Symbol Tables: static tree tables, dynamic tree tables; Hash Tables: Hashing functions overflow handling, theoretical evaluation of overflow techniques.

Files: file, queries and sequential organizations: Indexing Techniques: Cylinder-surface indexing hashed indexes, tree indexing-B-trees; Tree indexing.

Books Recommended:

1.	E. Horowitz and S. Sahni	:	Fundamentals of Data Structures, *Galgotia.*
2.	Edward M. Reingold &Wilfred J. Hansen	:	Data Structures, *Addison Wesley Publishers*
3.	Niklaus Wirth	:	Algorithms + Data Structures = Programs, *Prentice Hall*
4.	Robert L. Kruse	:	Data Structures and Program Design, *Prentice Hall*
5.	Seymour Lipshultz	:	Data Structures (Schaum's Outline Series), *Tata McGraw-Hill*
6.	E. Horowitz and S. Sahni	:	Computer Algorithms, *Galgotia.*
7.	Seymour E. Goodman & S. T. Hedetniemi	:	Introduction to Design and Analysis of Algorithms, *McGraw-Hill.*

Index

Part-A   Marks-26.25

Part-B Marks-26.25

Study Outline of Computer Programming

Part-A

Marks-35

Cha pter **0**

Introductio n to Data Structure
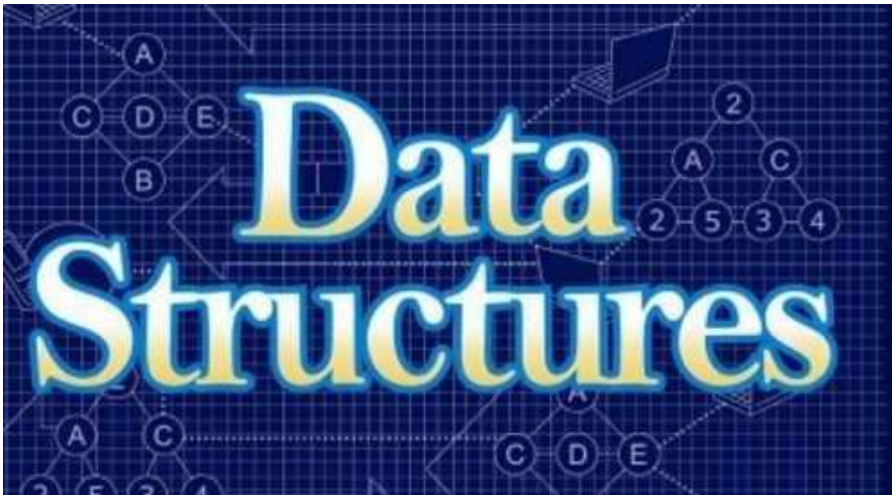
Important Question

1. Define data structure. Why is data structure necessary? **Exam-2016**
2. What is data structure? What are the differences between linear and nonlinear data structure? **Exam-2014,2013**
3. What do mean by Linear data structure and Nonlinear data structure? Given Example. Exam-**2016**
4. Why data structure is needed? Differentiate between linear and nonlinear data structure. **Exam-2015**

Question: Define data structure. Why is data structure necessary? Exam-2016

Data Structure:

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

Necessary of Structure:

Data structures are important for the following reasons:

- Data structures are used in almost every program or software system.
- Specific data structures are essential ingredients of many efficient algorithms, and make possible the management of huge amounts of data, such as large integrated collection of databases.
- Some programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design.

Question: What do mean by Linear data structure and Nonlinear data structure? Given Example. Exam-2016

Answer:

Linear Data Structure:

A data structure is said to be linear if its elements form a sequence or a linear list.

Examples:

Array
Linked List
Stacks
Queues

Non Linear Data Structure:

A non-linear data structure is a data structure in which a data item is connected to several other data items. So that a given data item has the possibility to reach one-or-more data items.

Examples

Graphs and
Trees.

Question: What is data structure? What are the differences between linear and nonlinear data structure? Exam-2014,2013

Question: Why data structure is needed? Differentiate between linear and nonlinear data structure. Exam-2015

Answer:

Linear data structure	Non-linear data structure
Data is arranged in linear sequence.	Data is not arranged in sequence.
They are easy to implement in computer's memory since they are organized sequentially.	They are difficult to implement in computer's memory since the data element can be attached to various other data elements.
Example: List, Stacks, Queue etc.	Example: Tree, Graph etc.

Question: Explain the Operation of Data Structure?

Answer:

The operations one normally performs on any linear structure, whether it he an array or a linked
list, include the following:

**Traversal :** Visit every part of the data structure
Search : Traversal through the data structure for a given element
Insertion : Adding new elements to the data structure
Deletion : Removing an element from the data structure.
Sorting : Rearranging the elements in some type of order(e.g Increasing or Decreasing)
Merging : Combining two similar data structures into

Cha pter

1

Aarrays
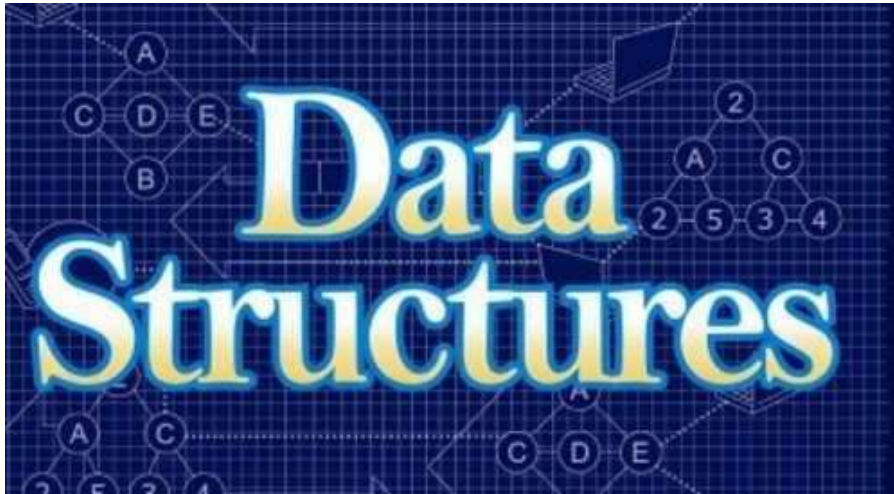
Important Question

1. Explain linear array representation in memory. **Exam-2015**
2. How can you traverse an array? **Exam-2013**
3. Simulate the binary search algorithm on the following data:10,20,50,70,80,90,100,110(suppose we search for item 110). **Exam-2015**
4. Simulate the binary search algorithm on the following data:12 14 16 18 20 22 24 26 28 30(suppose we search for iteam 23). **Exam-2013**
5. Is it possible to apply binary search on unsorted data? Justify your answer. **Exam-2013**
6. For column major order find out the address of the element score[15,3] from a 20X5 matrix array score with base value 100 and w=4. **Exam-2015**
7. For column major order find out the address of the element score[5,2] from a 20X4 matrix array score with base value 150 and w=4. **Exam-2013**
8. What do you mean by internal and external sorting ? List some names of each type. **Exam-2014**
9. For column major order find out the address of the element marks[12,3] from a 25X4 matrix array marks with base value 250 and w=8. **Exam-2014**
10. What is pointer and pointer array? **Exam-2014**
11. How a pointer can save memory space to store a 2D array? **Exam-2014**
12. How does a pointer array can save memory when stores a variable sized group of data?Discuss with necessary figures. **Exam-2015**
13. What is a record? What is the difference between a record and a linear array? **Exam-2015**
14. What is sparse matrix? What is the difference between triangular matrix and tridiagonal matrix? **Exam-2013,2014**
15. How can you locate the element a_{ij} of a sparse matrix from a ID array? **Exam-2013**
16. What is sparse matrix? What is the difference between triangular matrix and Tridiagonal matrix? **Exam-2016**
17. What are the disadvantages of array implementation of list? **Exam-2013**

Question: Write the Basic Operation of an arrays?
Answer:
Basic Operations of an array:
Following are the basic operations supported by an array.

Traverse − print all the array elements one by one.
Insertion − Adds an element at the given index.
Deletion − Deletes an element at the given index.
Search − Searches an element using the given index or by the value.
Update − Updates an element at the given inde

Question (1) : Explain linear array representation in memory. Exam-2015
Answer:

Let LA be a linear array in the memory of the computer. In **Fig. 4-2**. Let us use the notation

$$LOC(LA[K]) = \text{address of the clement } LA[K] \text{ of the array LA}$$

Computer needs to keep track only of the address of the first element of LA, denoted by
Base(LA)
and called the base address of LA. Using this address Base(LA), the computer calculates the address of any element of LA by the following formula:

$$LOC(LA[K]) = Base(LA) + w(K - \text{lower bound})$$
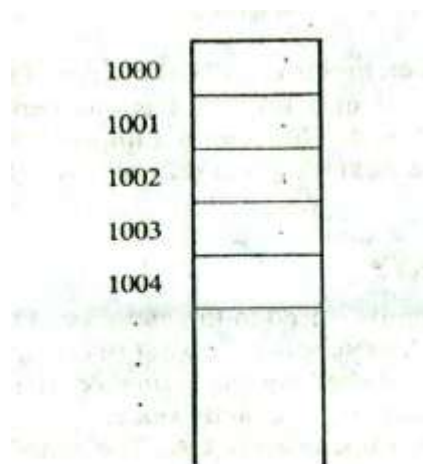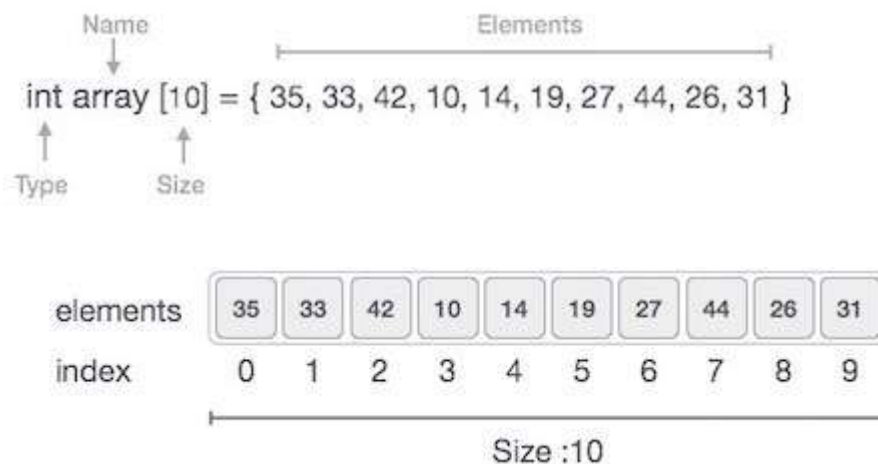where **w** is the number of words per memory cell for the array LA.

Fig. 4-2 Computer memory.

Example:
Arays can be declared in various ways in different languages. For illustration, let's take C array declaration.

As per the above illustration, following are the important points to be considered.

Index starts with 0.
Array length is 10 which means it can store 10 elements.
Each element can be accessed via its index. For example, we can fetch an element at index 6 as 9.

Question (2): How can you traverse an array? Exam-2013
Answer:
Traverse Operation

In traversing operation of an array, each element of an array is accessed exactly for once for processing . This is also called visiting of an array.

Let **A** be a collection of data elements stored in the memory of the computer**. Suppose we want to print the contents of each clement of A. This can be accomplished by traversing A, that is, by accessing and processing (frequently called visiting)** each clement of A exactly once.

Algorithm of Traversing an Array:
(Traversing a Linear Array) Here LA is a linear array with lower bound LB and upper bound UB. This algorithm traverses LA applying an operation PROCESS to each element of LA.

I. [Initialize counter.] Set K: = LB.
2. Repeat Steps 3 and 4 while K UB.
3. [Visit element.] Apply PROCESS to LA[K].
[Increase counter.] Set K := K + 1.
[End of Step 2 loop.]
  4.Exit.

Question(5): What are the disadvantages of array implementation of list? Exam-2013
Answer:
Disadvantage of array implementation:

 If a data entry is added to or removed from an array-based list, data needs to be shifted to update the list.
 In the worst case, for an array-based list with n data entries, an add and a remove takes o(n) time.
 Also, all data in the array-based list must be stored sequentially in memory. Large lists will require significant contiguous blocks of memory.

Question: What is Binary Search algorithm and How its Works Explain with Example?
Binary Search Algorithm:

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.

How Binary Search Works:
The binary search algorithm applied to our array DATA works as follows. During each stage of our algorithm, our search for ITEM is reduced to a segment of elements of DATA

DATA[BEG], DATALBEG + 11, DATA[BEG + 2],..., DATA[END]

Note that the variables **BEG and END denote, respectively, the beginning and end locations of the segment under consideration**. The algorithm compares **ITEM** with the middle clement **DATA[MID]** of the segment, **where MID** is obtained by

$$MID = INT((BEG + END)/2)$$ (We use INT(A) for the integer value of A.)
If DATA[MID] = ITEM, then the search is successful and
we set **LOC : = MID.** Otherwise a new segment of DATA is obtained as follows:

(a) If **ITEM < DATA[MID]**, then ITEM can **appear only in the left half of the segment**:
DATA[BEG], DATA[BEG +11 .....DATA[MID —1]
So we reset END : = MID - 1 and begin searching again.
(b) If **ITEM> DATA[MID],** then -ITEM can appear only in the right half of the segment:
DATA[MID + fl DATA[MID + 2]..... DATA [END]
So we reset BEG := MID + 1 and begin searching again.

<u>EXAMPLE</u>

Let DATA be the following sorted 13-clement array:
DATA: 11, 22, 30, 33, 40, 44, , 60, 66, 77, 80, 88, 99
We apply the binary search t o DATA for 'different values of ITEM.

Suppose **ITEM = 40**. The search for ITEM in the array DATA is pictured in Fig. 4-6, where the values of **DATA[BEG]** and **DATA[END]** in each stage of the algorithm are indicated by circles and the value of DATA Specifically, BEG, END and MID will have the following successive values:

(I) Initially, **BEG= 1** and **END = 13**. Hence
MID = INT((1 + 13)/2] =7 and so DATA(MIDJ = 55

(2) **Since 40< 55,   END has its value changed by END = MID - 1 =6**
Hence **MID = INT [(1 + 6)/2 = 3 and so DATALMID] = 30**

(3) **Since 40>30, BEG has its value changed by BEG = MID + 1=4**
Hence MID INT[4 + 6)/2]=5 and so DATA[MID]=40
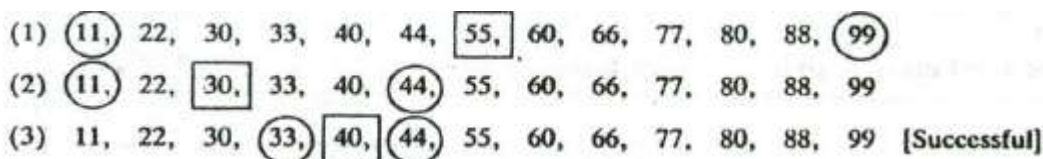We have found ITEM in location LOC MID= 5.

Fig. 4-6 Binary search for ITEM = 40.

Suppose **ITEM = 85. The binary search for ITEM is pictured in Fig. 4-7. Here BEG. END and MID will have the following successive values:**

(1) Again initially, BEG=1, END=13, MID=7 and DATA[MID]=55.

(2) Since **85>55**, BEG has its value changed by **BEG = MID + 1= 8**. Hence
MID INTL(8 + 13)/21 10 and so DATALMIDJ 77
(3) Since **85>77,** BEG has its value changed by BEG **MID + 1= 11**.
Hence   **MID = INT[(1 I + 13)/2]= 12 and so DATA(MID]=88**
(4) Since *85<*88, **END has its value changed by END = MID – 1=11.**
Hence MID INT[(1I + 11)/2] =11 and so DATA[MID]= 80
(Observe that now BEG = END   MID = 11.)
Since 85> 80, BEG has its value changed by BEG = MID + 1=12. But now BEG> END. Hence ITEM does not belong to DATA.
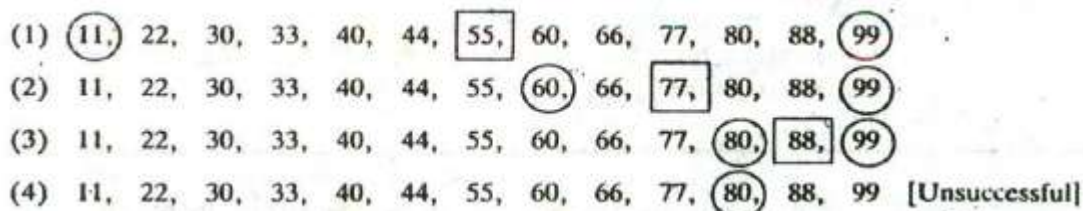
Fig. 4-7 Binary search for ITEM = 85.

Or

The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

First, we shall determine half of the array by using this formula −
mid = low + (high - low) / 2
Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We change our low to mid + 1 and find the new mid value again.
low = mid + 1
mid = low + (high - low) / 2
Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

The value stored at location 7 is not a match, rather it is less than what we are looking for. So, the value must be in the lower part from this location.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Hence, we calculate the mid again. This time it is 5.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We compare the value stored at location 5 with our target value. We find that it is a match.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We conclude that the target value 31 is stored at location 5.

Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

Question(3): Simulate the binary search algorithm on the following data:10,20,50,70,80,90,100,110 (suppose we search for item 110). Exam-2015

Answer:
 Here item=110

10	20	50	70	80	90	100	110
BEG=1	2	3	MID=4	5	6	7	END=8

(1) Initially, **BEG= 1** and **END = 8**. Hence
$$MID = INT((1 + 8)/2] = 4 \text{ and so } DATA(MIDJ = 70$$

(2) Since 110>70 , BEG has its value changed by BEG= MID + 1 =5
     Hence **MID = INT [(5 + 8)/2 = 6 and so DATA [MID] = 90**

(3) Since 110>90 , BEG has its value changed by BEG= MID + 1 =7
     Hence **MID = INT [(7 + 8)/2 = 7 and so DATA [MID] = 100**
(4) Since 110>100 , BEG has its value changed by BEG= MID + 1 =8
     Hence **MID = INT [(8 + 8)/2 = 8 and so DATA [MID] = 110**
          We have found ITEM in location LOC MID=8.

Question(4): Simulate the binary search algorithm on the following data:12 14 16 18 20 22 24 26 28 30(suppose we search for iteam 23). Exam-2013

Answer:
 Here item=23

12	14	16	18	20	22	24	26	28	30
BEG=1	2	3	4	MID=5	6	7	8	9	END=10

(1) Initially, **BEG= 1** and **END = 10**. Hence
$$MID = INT((1 + 10)/2] = 5 \text{ and so } DATA(MIDJ = 20$$

(2) Since 23>20 , BEG has its value changed by BEG= MID + 1 =6
     Hence **MID = INT [(6+ 10)/2 = 8 and so DATA [MID] = 26**

(3) Since 23<26 , END has its value changed by END= MID -1 =7
     Hence **MID = INT [(6 + 7)/2 = 6 and so DATA [MID] = 22**

(4) Since 23>22 , BEG has its value changed by BEG= MID + 1 =7
     Hence **MID = INT [(7 + 7)/2 = 7 and so DATA [MID] = 24**
          We have not found ITEM in location LOC MID=7

Question: Is it possible to apply binary search on unsorted data? Justify your answer. Exam-2013
Answer:

The answer is Yes.

We could apply. But we most probably would miss the element you're searching for. Because binary search works based on the assumption/ invariant (for elements in increasing order) that whenever we are in a position all the elements to its left lower than itself and all the elements to its right are greater than itself. Which is useful to decide which side to search further.

But in the case of unsorted list, the element you're searching may be in any side of the array from your current position.So we've to search the both sides of the list, which is linear.

Question: For column major order find out the address of the element score[15,3] from a 20X5 matrix array score with base value 100 and w=4. Exam-2015

Question: For column major order find out the address of the element score[5,2] from a 20X4 matrix array score with base value 150 and w=4. Exam-2013

Question: What do you mean by internal and external sorting ? List some names of each type. Exam-2014

Question: For column major order find out the address of the element marks[12,3] from a 25X4 matrix array marks with base value 250 and w=8. Exam-2014

Answer:

Consider the 25 x 4 matrix array SCORE in Example 4.11. Suppose Base(SCORE) = 200 and there are $w = 4$ words per memory cell. Furthermore, suppose the programming language stores two-dimensional arrays

using row-major order. Then the address of SCORE[12. 31, the third test of the twelfth student, follows:

LOC(SCORE[12. 3]) = 200 + 4(4(12 - 1) + (3– 1)] = 200 + 446] = 384

Observe that we have simply used Eq. (4.5).

Question: What is pointer and pointer array? Exam-2014

Answer:

Pointer:

Variable P is called a pointer if P "points" to an element in DATA, i.e. if P contains the address of an element in DATA.

Pointer Array:

An array PTR is called a pointer array if each element of PTR is a pointer.

Pointers and pointer arrays are used to facilitate the processing of the information in DATA. This section discusses- this useful tool in the context of a specific example.

Question: How a pointer can save memory space to store a 2D array? Exam-2014

Question: How does a pointer array can save memory when stores a variable sized group of data? Discuss with necessary figures. Exam-2015

Question: What is a record? What is the difference between a record and a linear array? Exam-2015
Answer:
Record:
        A record is a data structure made up of a fixed number of information items, not necessarily of the same type.
Or
    A record is a collection of related data items, each of which is called a field or Attribute, and it file is a collection of similar records.

Difference between a record and a linear array:

Linear array	record
An array is suitable for homogeneous data. You cannot store different type of data in array as integer, float, double, etc.	Record may have different data type
in array, ordering always start from index	In a record, there may not be a natural ordering in opposed to linear array
A linear array does not hierarchical structure.	A record form a hierarchical structure, as an example a record may be parent child relationship, so it will create hierarchical structure.
The operation of processing each element in the list is known as traversing the link list.	The operation of processing each element in the list is known as traversing the link list.

Question: What is sparse matrix? What is the difference between triangular matrix and tridiagonal matrix? Exam-2013,2014,2016
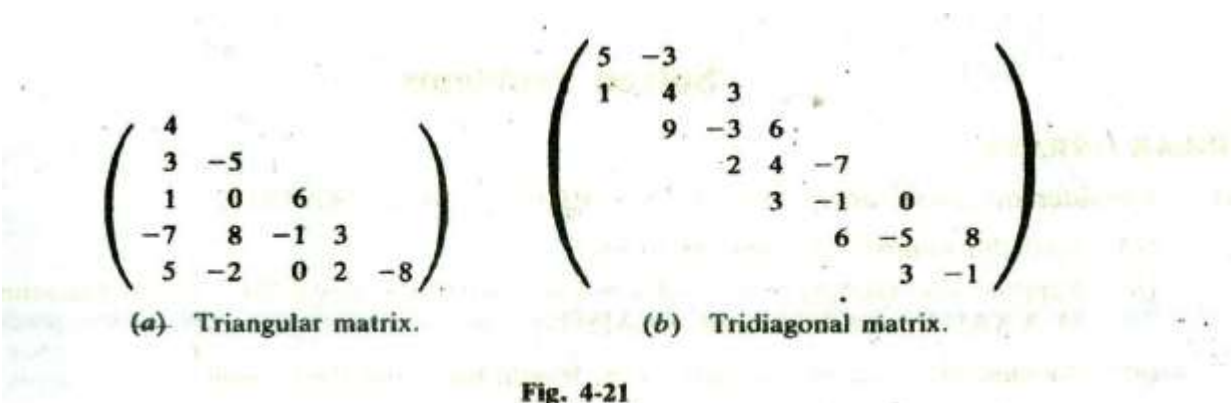Answer:

Sparse Matrices
        Matrices with a relatively high proportion of zero entries are called sparse matrices. Two general types of n-square sparse matrices which occur in various applications.

Triangular matrix
        The first matrix, where **all entries above the main diagonal are zero or, equivalently, where nonzero entries can only occur on or below the main diagonal, is called a (lower) triangular matrix.**

Tridiagonal Matrix
        The second matrix, **where nonzero entries can only occur on the diagonal or on elements immediately above or below the diagonal, is called a tridiagonal matrix**

$$\begin{pmatrix} 4 & & & & \\ 3 & -5 & & & \\ 1 & 0 & 6 & & \\ -7 & 8 & -1 & 3 & \\ 5 & -2 & 0 & 2 & -8 \end{pmatrix} \qquad \begin{pmatrix} 5 & -3 & & & & \\ 1 & 4 & 3 & & & \\ 9 & -3 & 6 & & & \\ & 2 & 4 & -7 & & \\ & & 3 & -1 & 0 & \\ & & & 6 & -5 & 8 \\ & & & & 3 & -1 \end{pmatrix}$$

(a) Triangular matrix.   (b) Tridiagonal matrix.

Fig. 4-21

Question: How can you locate the element a_{ij} of a sparse matrix from a ID array? Exam-2013

Answer:

Suppose we want to place in memory the triangular array *A* in Fig. 4-22. Clearly it would be wasteful to store those entries above the main diagonal of *A,* since we know they are all zero; hence we store only the other entries of *A* in a linear array *B* as indicated by the arrows. That is, we let

B[1J=a11, B[2]=a21, B[31=a22 , B(3]=a31,

Observe first that *B* will contain only

1+2+3+4++nn(n+l)

elements, which is about half as many elements as a two-dimensional *n* X *n* array. Since we will require the value of a, in our programs, we will want the formula that gives us the integer L in terms of J and K where B[L]a,K

Observe that L represents the number of elements in the list up to and including a_{JK}. Now there are

$$1 + 2 + 3 + \cdots + (J - 1) = \frac{J(J - 1)}{2}$$

elements in the rows above a_{JK}, and there are K elements in row J up to and including a_{JK}. Accordingly,

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

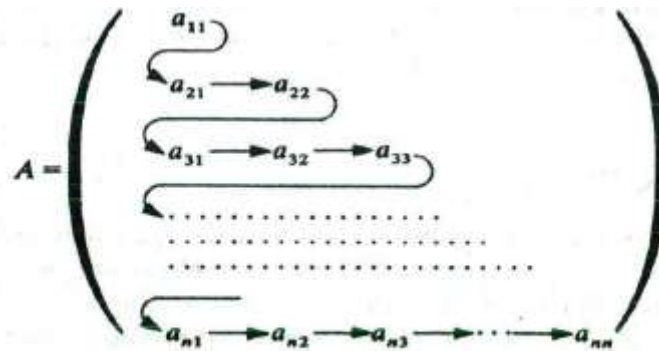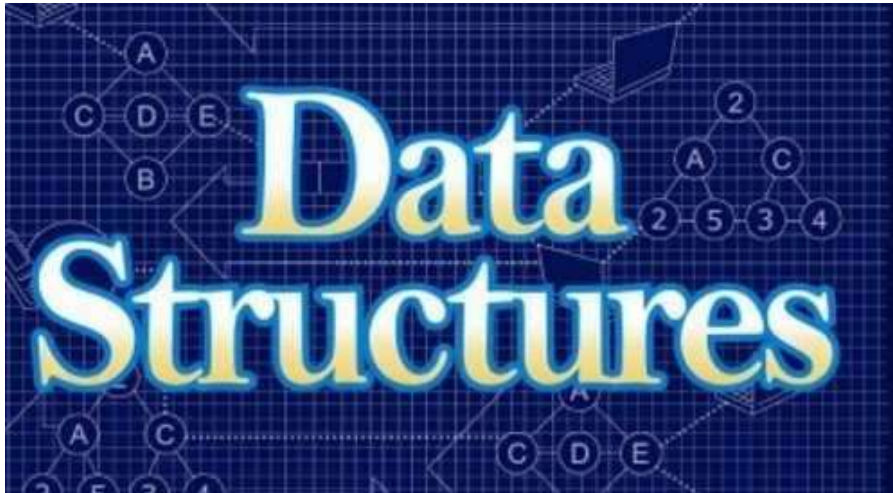yields the index that accesses the value a_{JK} from the linear array B.

Fig. 4-22

Cha pter 2

Stack ,Queues and Recursion

Important Question

1. What is stack ?What are the operations on stack? Explain with example . **Exam-2015,2013**
2. Discuss the array representation mechanism of stack**. Exam-2016,2013**
3. What is polish notation? What are the benefits of polish notation? **Exam-2016,2014**
4. Simulate the postfix expression evaluation algorithm using 12,6,7,6,2,+,*,12,4,/,- by showing Stact's contents as each element is scanned. **Exam-2016,2014**
5. Simulate the infix to postfix transformation algorithm for- A+(B*C-(D/E↑F)*G)*H by showing the stack's contents as each element is scanned. **Exam-2013**
6. Convert the following infix expression to its equivalent prefix and postfix expression.
   (i) A*B/C+D↑ (E-F*G)/H
   (ii) 1+2*3/4↑ 5*6-7*8   Exam-2016
7. Convert the following infix expression to its equivalent prefix and postfix expression.
   (i) A*B+(C*D/E)*F+(G↑ H)
   (ii) 1*2/3+(4-5↑6)+7-8   **Exam-2014**
8. Convert the following infix expression to its equivalent prefix and postfix expression
   (i) A+B*C/D-E(F/G+H↑K)
   (ii) (1+2)↑3/4*5+7-8↑ 9 **Exam-2015**
9. Convert the following infix expression into their equivalent prefix and postfix expressions:

   Exam-2013
   (i) A+B*C/(D-E)+F/G*H
   (ii)10*B+5/D-(12+F)/50

10. Explain the operation on queue with example**. Exam-2016**
11. Define deque and priority queue with example. **Exam-2016**
12. What is priority queue? Why is it important? **Exam-2015**
13. What is priority queue? Why is it important? **Exam-2013**
14. What is overflow and underflow? How can you handle them? **Exam-2014**
15. What is garbage collection? When does it place**? Exam-2016**
16. What is garbase collection and compaction? **Exam-2014**

Question: What is stack ?What are the operations on stack? Explain with example . Exam-2015,2013
Answer:
Stack:
        Stack is an ordered list of similar data type its a LIFO structure. (Last in First out).
 A stack is a list of elements in which an element may be inserted or deleted only at one end, called the lop of the stack. This means, in particular, that element removed from a stack in the reverse order of that in which they were inserted into the stack.
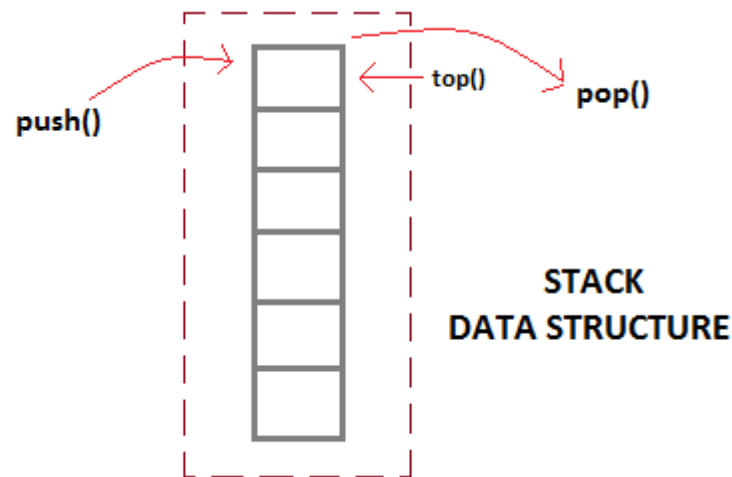
Operation of Stack:
                Special terminology is used for two basic operations associated with stacks:

                (a) "Push" is the term used to insert an element into a stack.
                (b) "Pop" is the term used to delete an element front stack.
We emphasize that these terms arc used only with stacks, not with other data structures.
        📖 Both insertion and deletion are allowed at only one end of Stack called **Top**.
        📖 Stack is said to be in **Overflow** state when it is completely full and is said to be in **Underflow** state if it is completely empty
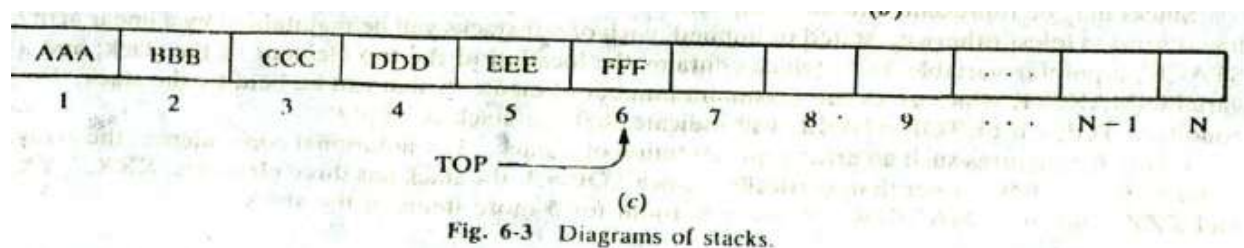
EXAMPLE 6.1
Suppose the following 6 elements arc pushed, in order, onto an empty stack:
            AAA, BBB, CCC, ODD, EEE, FFF
Figure 6-3 shows three waxs of-picturing such a stack. For notational stack by writing:
            STACK: AAA, BBB CCC, ODD, EEE, FFF
The implication is that the **right-most clement is the top clement.**

Fig. 6-3 Diagrams of stacks.

.

Question: Discuss the array representation mechanism of stack. Exam-2016,2013
Array Representation Of Stacks:
                Stacks may be represented in the computer in various ways, usually by means of a one-way list or a linear array. a pointer variable TOP, which contains the location of the top clement of the stack; and a variable MAXSTK which gives the maximum number of elements that can be held by the stack. The condition TOP = 0 or TOP = NULL will indicate that the stack is empty.
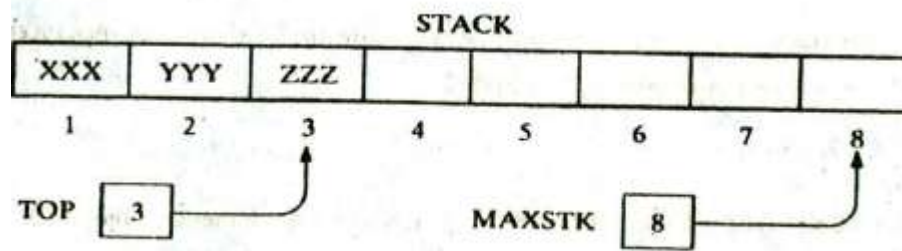
Fig. 6-5

Figure 6-5 pictures.

Such an array representation of it (For notational convenience, the array is drawn horizontally rather than vertically.) Since TOP 3, the stack has three demerits, XXX, YYY and ZZZ; and since MAXSTK = 8, there is room for 5 more items in the stack.

The operation of adding (pushing) an item onto a stack and the operation. of removing (popping) an item from a stack may be implemented, respectively, by the following procedures, called PUSH and POP.

In executing the procedure PUSH, one must first test whether there is room in the stack for the new item; if not, then we have the condition known as overflow. Frequently, TOP and MAXSTK arc global variables; hence the procedures may be called using only

PUSH(STACK, ITEM) and POP(STACK, ITEM) respectively.

We note that the value of TOP is hanged before the insertion in PUSH but the value of TOP is changed after the deletion in POP.

Question: What is polish notation? What are the benefits of polish notation? Exam-2016,2014
Polish Notation

Polish notation also known as prefix notation is defined as: In this notation system operator is written before the operands.

Example of polish notation or prefix notation is **+AB or /CD. In these two examples we can see that here A and B are two operands and + is an operator** which is written before operands, similarly we can say for the second example.

The conversion of an infix expression into prefix notation is shown belowConvert the infix expression (A + B) x (C + D) into prefix notation?
SOLUTION
(A + B) x (C + D)
= [+AB] x [+CD]
= x [+AB+CD]
= x+AB+CD
So prefix notation is x+AB+CD.

Benefits of Polish notation:
- Infix notation is easy to read for humans, whereas pre-/postfix notation is easier to parse for a machine.
- The big advantage in pre-/postfix notation is that there never arise any questions like operator precedence.
- Postfix notation, also known as RPN, is very easy to process left-to-right. An operand is pushed onto a stack; an operator pops its operand(s) from the stack and pushes the result. Little or no parsing is necessary. It's used by Forth and by some calculators (HP calculators are noted for using RPN).

For example, consider the infix expression 1 # 2 $ 3. Now, we don't know what those operators mean, so there are two possible corresponding postfix expressions: 1 2 # 3 $ and 1 2 3 $ #. Without knowing the rules governing the use of these operators, the infix expression is essentially worthless.

Or, to put it in more general terms: it is possible to restore the original (parse) tree from a pre-/postfix expression without any additional knowledge, but the same isn't true for infix expressions

Question: Simulate the postfix expression evaluation algorithm using 12,6,7,6,2,+,*,12,4,/,- by showing stack contents as each element is scanned. Exam-2016,2014

Question: Simulate the infix to postfix transformation algorithm for- A+(B*C-(D/E↑F)*G)*H by showing the stack's contents as each element is scanned. Exam-2013
We simulate Algoiithm 6.4 to transform 0 into its equivalent postlix cxprcsion P.
First we push "(" onto STACK, and then we add ")" to the end of Qto obtain:

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)

The elements of 0 have now been labeled from left to right for easy reference. Figure 6-8 shows the status of
STACK and of the string I' as each element of 0 is scanned. Observe tIt:iI
(I) Each operand is simply added to P and does not change STACK.
(2) The subtraction operator (-) in row 7 sends * front to P before it (-) is pushed onto STACK.
(3) The right parenthesis in row 14 scnds t and then / from STACK to P. and then removes the left parenthesis Ironi he top of STACK.
(4) The right parenthesis in row 20 sends * and then + from STACK to P. and (lien removes the left parenthesis from the top of STACK.
After Step 20 is executed, the STACK is empty and
P: ABCI)EF 1/G - 11* +
which is the required postlix equivalent of 0.
Question: Convert the following infix expression to its equivalent prefix and postfix expression.
       (i) A*B/C+D↑ (E-F*G)/H
       (ii) 1+2*3/4↑ 5*6-7*8 Exam-2016
Question: Convert the following infix expression to its equivalent prefix and postfix expression.
Question: (i) A*B+(C*D/E)*F+(G↑ H)
       (ii) 1*2/3+(4-5↑6)+7-8 Exam-2014
Question: Convert the following infix expression to its equivalent prefix and postfix expression
       (i) A+B*C/D-E(F/G+H↑K)
       (ii) (1+2)↑3/4*5+7-8↑ 9 Exam-2015
Question: Convert the following infix expression into their equivalent prefix and postfix expressions:
                                                            Exam-2013
            (i) A+B*C/(D-E)+F/G*H
            (ii)10*B+5/D-(12+F)/50

Question: Explain the operation on queue with example. Exam-2016
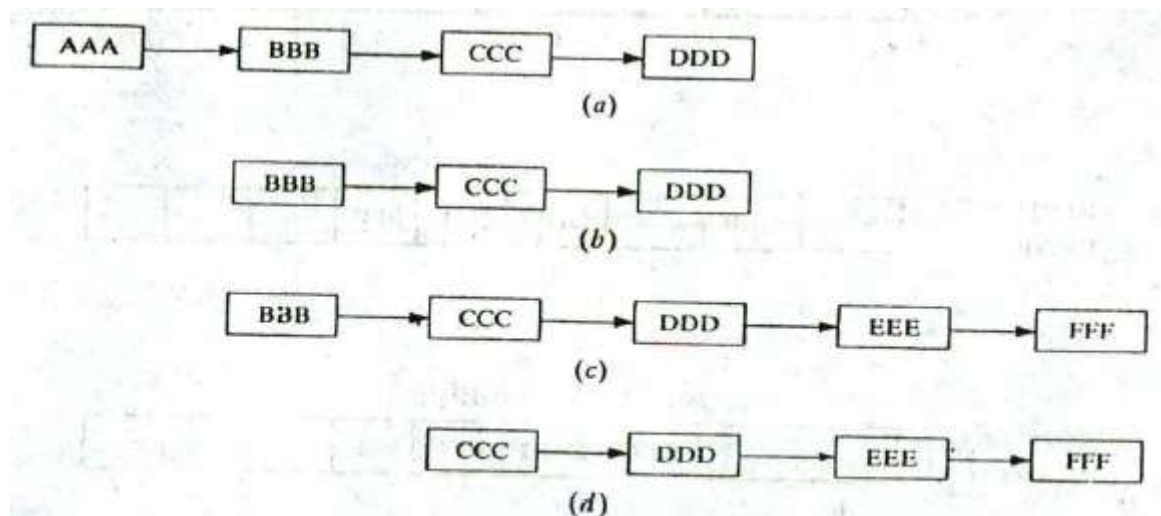**Answer:**
Queue:
        Queue is a linear list of elements in which deletions can take place only at one end, called the front, and insertions can take place only at the other end, called the rear. The terms "front" and "rear" are used in describing a linear list only when it is implemented as a queue.

Queues are also called first-in first-out (FIFO) lists, since the first clement i1 a queue will be the first element Out of the queue. In other words, the order in which elements enter a queue is the order in which they leave. This contrasts with stacks, which are last-in first-out (LIFO) lists.

EXAMPLE 6.9

📖 Figure 6-15(a) is a schematic diagram of a queue wilts 4 elements, **where AAA is the front clement and DDD is the rear element.** Observe that the **front and rear elements of the queue are also, respectively, the first and last elements of the list. Suppose an element is deleted from the queue.** Then it must he AAA.

📖 This yields the queue in Fig. 6-15(b), **where BBB is now the front clement.** Next, suppose EEE is added to the queue and then FEF is added to the queue.

📖 Then they must be added at **the rear of the queue, as pictured in Fig. 6-15(c),** Note that FFF is now the rear clement.

📖 Now suppose another element is deleted from the queue; then it must he BBB, to yield the queue in Fig. 6-15(d). And soon. Observe that in such a data structure, EEE will he deleted before FFF because it has been placed in the queue before FFF. However, EEE will have to wait until CCC and DDI) are deleted.

Fig.. 6-15

Question: Define deque and priority queue with example. Exam-2016
Answer:
  Deque:
        A double-ended queue (dequeue, often abbreviated to deque) is an abstract data type that generalizes a queue, for which elements can be added to or removed from either the front (head) or back (tail).
Or
        Double-ended queue A deque (pronounced either "deck" or "deque") is a linear list in which elements call be added or removed at either end but not in the middle. The term deque is a contraction of the name double ended queue.
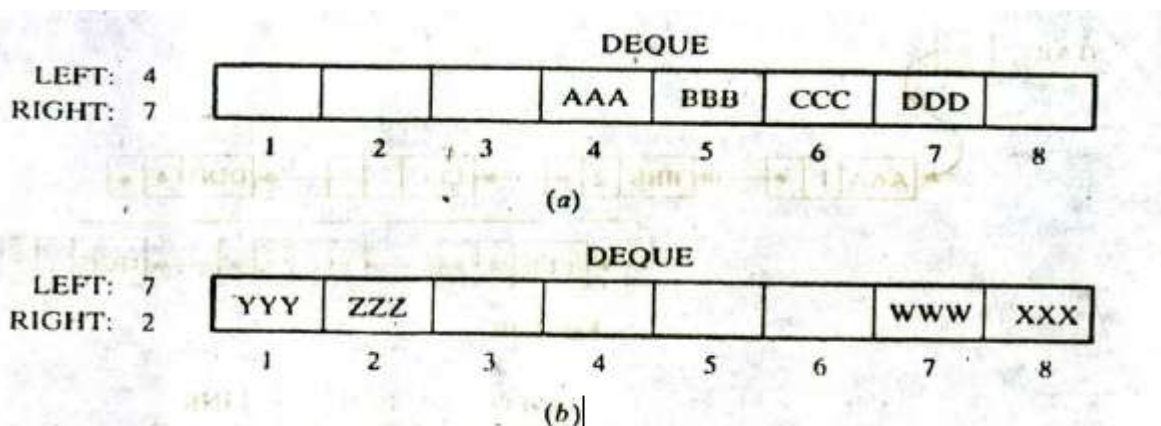
Fig. 6-18

1TY QUEUES

<u>Priority Queue:</u>

A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

Or

Priority queue is a collection of elements such that each clement has been assigned **if** and such that the order in which elements arc deleted and processed comes from the following rules:

(1) An element of higher priority is processed before any clement of lower priority.
(2) Two elements with the same priority arc processed according to the order in which they were added to the queue.

A prototype of a priority queue is a timesharing system: programs of high priority are processed first, programs with the same priority form a standard' queue. There are various ways of maintaining a priority queue in memory. We discuss two of them here: One uses a one-way list, and the other uses multiple queues. The ease or difficulty in adding elements to deleting them from a priority queue clearly depends on the representation that one chooses.

<u>Figure 6-19</u>

Shows a Schematic diagram of a priority queue with 7 elements. The diagram does not tell us whether BBB was added to the list before or after DDD. On the other hand, the diagram does tell us that BBB was inserted before CCC. because BBB and have the same priority number and BBB appears before CCC in the list. Figure 6-20 show the way the priority queue may appear in memory using linear arrays INFO PRN and LINK.
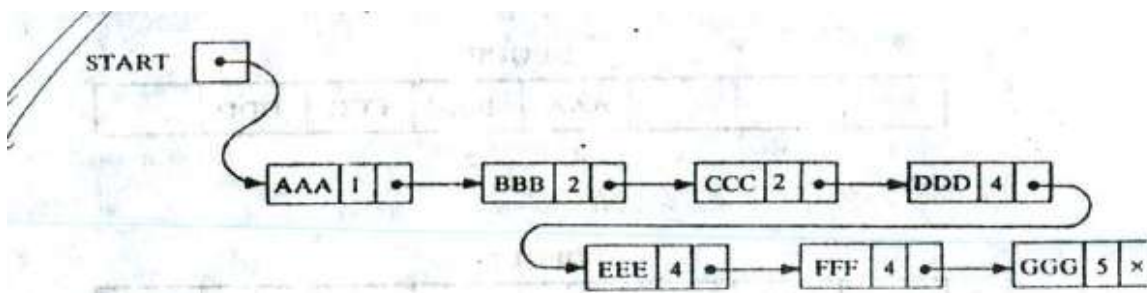

Fig. 6-19

	INFO	PRN	LINK
1	BBB	2	6
2			7
3	DDD	4	4
4	EEE	4	9
5	AAA	1	1
6	CCC	2	3
7			10
8	GGG	5	0
9	FFF	4	8
10			11
11			12
12			0

START 5
AVAIL 2

Fig. 6-20

Question: What is priority queue? Why is it important? Exam-2015,2013

Priority Queue:

A priority queue is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

Important of Priority Queue:

   📖 Priority queues provide extra flexibility over sorting, which is required because jobs often enter the system at arbitrary intervals. It is much more cost-effective to insert a new job into a priority queue than to re-sort everything.

   📖 The need to perform certain jobs may vanish before they are executed, meaning that they must be removed from the queue.

   📖 A prototype of a priority queue is a timesharing system: programs of high priority are processed first,

   📖 The main property of the one-way list representation of a priority queue is that is the clement in the queue that should be processed first always appears at the beginning of the one-way list.

   📖 It is a very simple matter to delete and process an clement from our priority queuc. The outline of the algorithm follow

Question: What is overflow and underflow? How can you handle them? Exam-2014

Stack overflow:

Sometimes new data are to be inserted into a data structure but there is no available space, i.e., the free-storage list is empty. This situation is usually called overflow.

Or

Stack overflow happens when we try to push one more item onto our stack than it can actually hold the stack usually can only hold so much stuff. Typically, we allocate (set aside) where the stack is going to be in memory and how big it can get. So, when we stick too much stuff there or try to remove nothing, we will generate a stack overflow condition or stack underflow condition, respectively.

Handle Overflow:

The programmer may handle overflow by printing the message OVERFLOW. In such a case, the programmer may (hen modify the program by adding space to the underlying arrays. Observe that overflow will occur with our linked lists when AVAIL = NULL and there is an insertion.

Stack underflow

The term underflow refers to the situation where one wants to delete data from a Data structure that is empty.

Or

Stack underflow happens when we try to pop (remove) an item from the stack, when nothing is actually there to remove. This will raise an alarm of sorts in the computer, because we told it to do something that cannot be done.

Handle Underflow:

The programmer may handle underflow by printing the message Underflow observe that underflow will occur with our linked lists when start null and There is a deletion,

Question: What is garbage collection? When does it place? Exam-2016
Question: What is garbage collection and compaction? Exam-2014

Suppose some memory space becomes reusable because a node is deleted From a list or an entire list is deleted from a program. Clearly, we want the space to be available for future use. One way to bring this about is to immediately reinsert the space into the free-storage list. This is what we will do when we implement linked lists by means of linear arrays. However, this method may be' too time-consuming for the operating system of a computer, which may choose an alternative method, as follows.

Garbage Collection

The operating system of a computer may periodically collect all the deleted space on to the free-storage list. Any technique which does this collection is called Onto the garbage collection.

When it place:

Garbage collection usually takes place in two steps.

First: The computer runs through all lists, tagging those cells which are currently in use, and then the computer runs through the memory

Second: collecting all untagged space onto the free-storage list. The garbage collection may take place when there is only some minimum amount of space or no space at all left in the free-storage list, or when the CPU is idle and has time to do the collection.

Generally speaking, the garbage collection is invisible to the programmer.

Compaction:

Data compaction is the reduction of the number of data elements, bandwidth, cost, and time for the generation, transmission, and storage of data without loss of information by eliminating unnecessary redundancy, removing irrelevancy, or using special coding.
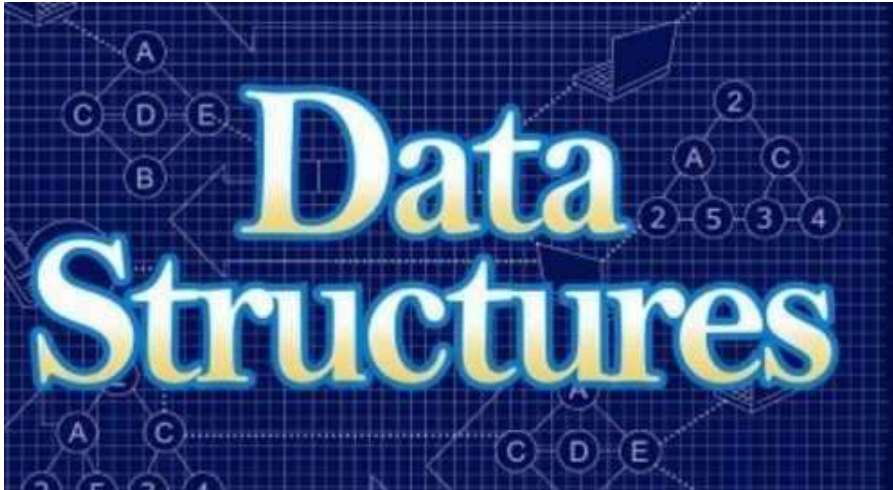
Important Question

Question: What is recursion? Explain the use of recursion. Exam-2016
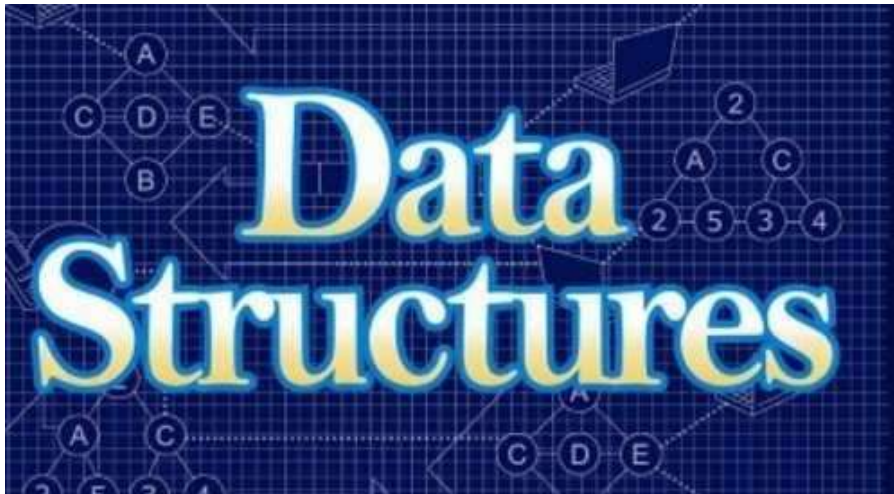**Answer:**
Recursion:
            Recursion is an approach in which a function calls itself with an argument. Upon reaching a termination condition, the control returns to the calling function.
Or
            A more common way to use **recursion** in programming is to base the **recursion** on a **recursive data structure**. A **data structure** is **recursive** if we can define it in terms of itself. For example, we can give a **recursive** definition of a string: A string is either empty, or it is a character followed by a string.

Use of Recursion:

  📖 When are some (relatively) basic (think first year college level CS student) instances when one would use recursion instead of just a loop?
  📖 Space is carved out on the stack for the function's arguments and local variables
  📖 The function's arguments are copied into this new space
  📖 Control jumps to the function
  📖 **The function's code runs**
  📖 The function's result is copied into a return value
  📖 The stack is rewound to its previous position
  📖 Control jumps back to where the function was called

Chapter 4

Links Lists.

Important Question

1. What the advantage Linked List? **Exam-2016**
2. Suppose 10 elements are maintained by array and   another 10 are by Linked List. Which methods take longer time to access 7th element. Justify your answer. **Exam-2016**
3. Briefly discuss inserting and deleting mechanism of an item in the linked list. **Exam-2016**
4. Explain the representation of linked lists in memory. **Exam-2015**
5. List the advantage and disadvantage of Linked list. **Exam-2013**
6. Show with appropriate schematic diagram how insertion and deletion could be carried out into a singly linked list. **Exam-2013**
7. Discuss header linked list. Describe grounded and circular header list. **Exam-2015**
8.
9. What is a linked List? Discuss with example. **Exam-2014**
10. Suppose 20 elements are maintained by array and   another 10 are by Linked List. Which methods take longer time to access 9th element. Justify your answer. **Exam-2014**
11. Briefly discuss inserting mechanism of an item at the beginning, after a given node and at the end. **Exam-2014**
12. Briefly explain two-way linked list. **Exam-2015**
13. What is a two way list? Why is it important? Explain with schematic diagram. **Exam-2014,2016**

Question: What is linked list? What the advantage Linked List? Exam-2016
Question: What is a linked List? Discuss with example. Exam-2014
Question: List the advantage and disadvantage of Linked list. Exam-2013
Answer:
Link List:
      A linked list, or one-way list- is a linear collection of data elements, called nodes, where the linear order is given by means of pointers.

That is, each node is divided into two parts: **the first part contains the information of the clement, and the second part, called the link field** or next pointer field, contains the address of the next node in the list.

Fig. 5-2   Linked list with 6 nodes.
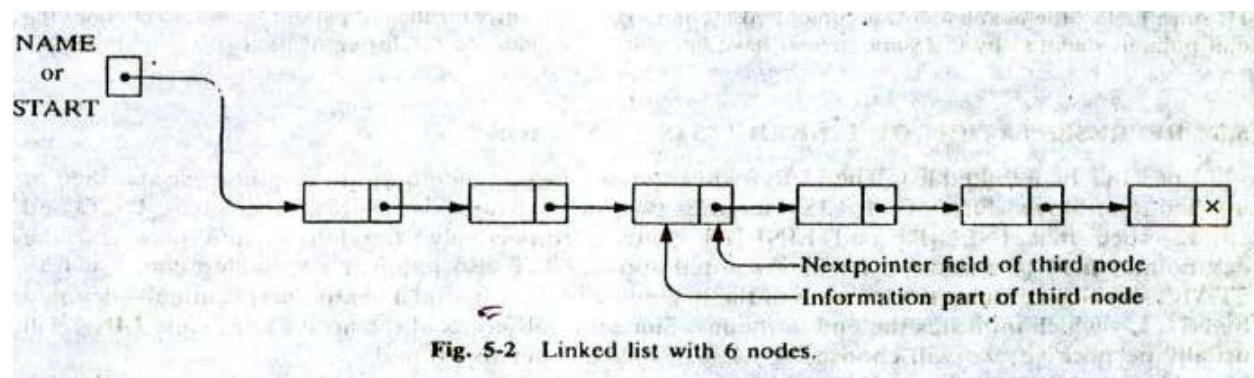
Advantage Linked List:

- Linked List is Dynamic data Structure .
- Linked List can grow and shrink during run time.
- Insertion and Deletion Operations are Easier
- Efficient Memory Utilization ,i.e no need to pre-allocate memory
- Faster Access time, can be expanded in constant time without memory overhead
- Linear Data Structures such as Stack, Queue can be easily implemented using Linked list

Disadvantage of Linked list

Question: Explain the representation of linked lists in memory. Exam-2015
Question: Describe the Traversing a linked-list ?
Answer:
        Let LIST be a linked list in memory stored in linear arrays INFO are LINK with START pointing to the first clement and NULL indicating the end of LIST.

 Suppose we wait to traverse LIST..

    & PTR points to the node that is currently being processed.
    & LINK[PTR] points to the next node to be processed. Thus the assignment

        PTR: = LINK[PTR]
moves the pointer to the next node in the list, as pictured in Fig. 5-8.

Fig. 5-8   PTR := LINK[PTR].
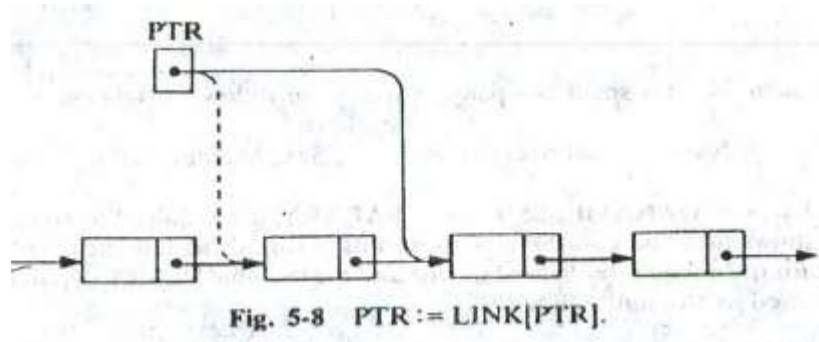
Algorithm of Traversing a Linked list:
        (Traversing a Linked List) Let LIST be a linked list in memory. This algorithm
        traverses LIST, applying an operation PROCESS to each clement of LIST. The
        variable PTR points-to the node currently being processed.
        1. Set PTR := START. [Initializes pointer PTR.J
        2. Repeat Steps 3 and 4 while PTR 0 NULL.
        3. Apply PROCESS to !NF0LPTRI..
        4. Set PTR:=LINK[PTRJ. (PFR now points to the next node.]
   [End of Step 2 loop.]
        Exit.

Question: Briefly discuss inserting mechanism of an item at the beginning, after a given node and at the end. Exam-2014
Question: Briefly discuss inserting and deleting mechanism of an item in the linked list. Exam-2016
Question: Show with appropriate schematic diagram how insertion and deletion could be carried out into a singly linked list. Exam-2013

Inserting Into a Linked list:
            Let LIST be a linked list with successive nodes A and B, as pictured in Fig. 5-14(a). Suppose a node N is to be inserted into the list between nodes A and B. The schematic diagram of such an

(a) Before insertion.

(b) After insertion.
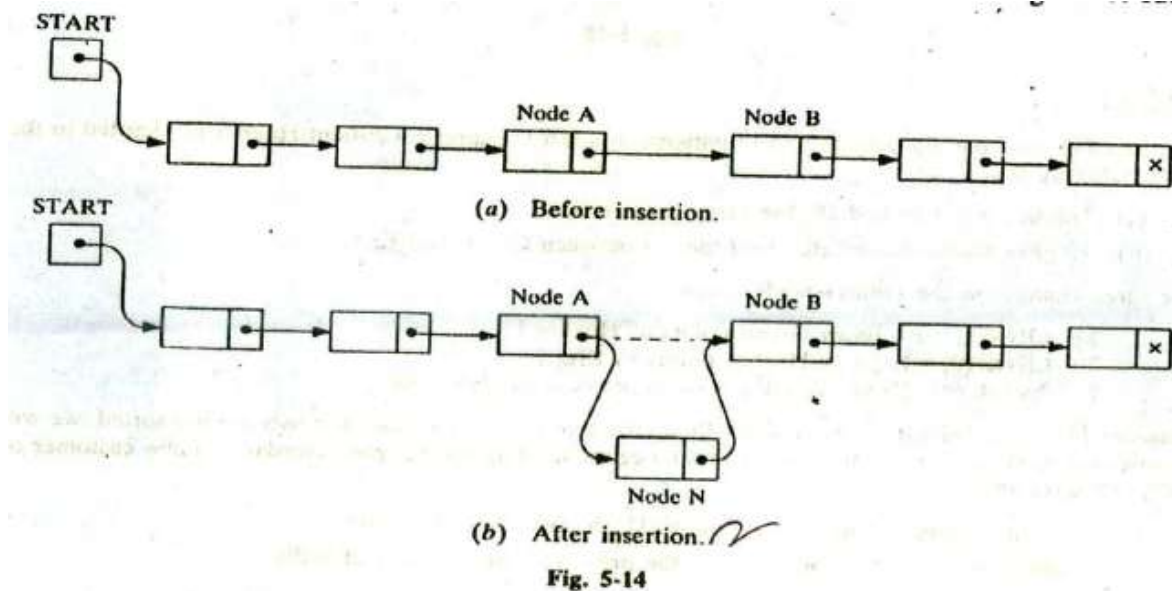
Fig. 5-14

insertion appears in Fig. 5-14(b). That is, node **A now points to the new node N, and node N points to node B**, to which A previously pointed

Question: Discuss header linked list. Describe grounded and circular header list. Exam-2015

Header Linked List:

A header linked list is a linked list which always contains a special node, called the header node, at the beginning of the list.

The following are two kinds of widely used header lists:

   📖 Grounded And
   📖 Circular Header List

(1) **A grounded header** list is a header list where the lost node contains the null pointer. (The term "grounded" comes from the fact that many texts use the electrical ground symbol to indicate the null pointer.)

(2) **A circular header list** is a header list where the last node points bock to the header nodc.

Figure 5-29 contains schematic diagrams of these header lists. Unless otherwise stated or implied, our header lists will always be circular. Accordingly, in such a case, the header nodc also acts as a sentinel indicating the end of the list.

Observe that the list pointer START always points to the header node. Accordingly. LINK[START] = NULL indicates that a grounded header list is empty, and LINKISTARTI START indicates that a circular header list is empty

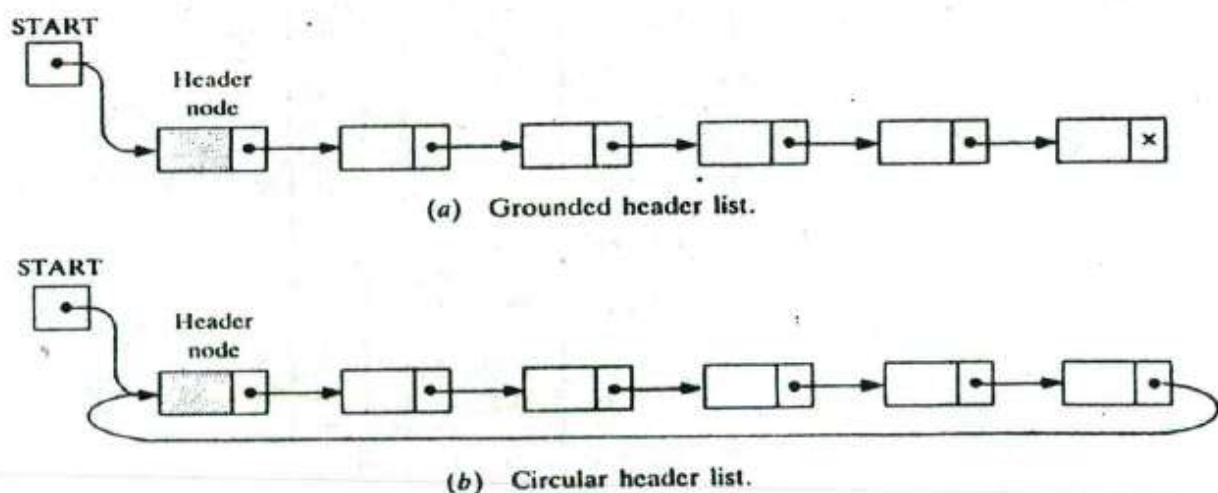(a) Grounded header list.

(b) Circular header list.

Fig. 5-29

A

Question: Briefly explain two-way linked list. Exam-2015
Question: What is a two way list? Why is it important? Explain with schematic diagram. Exam-2014,2016

Tow Way Linked List:
          Two-way list, which can be traversed in two directions: in the usual forward direction from the beginning of the list to the end, or in the backward direction from the end of the list to the beginning.

Furthermore, given the location LOC of a node N in the list, one now has immediate access to both the next node and the preceding node in the list. This means, in particular, that one is able to delete N from the list without traversing any part of the list.

A two-way list is a linear collection of data elements, called nodes, where each node N is divided into three parts:

    (1) An information field INFO which contains the data of N
    (2) A pointer field FORW which contains the location of the next node in the list
    (3) A pointer field BACK which contains the location of the preceding node in the list

The list also requires two list pointer variables: FIRST, which points to the first node in the list, and LAST, which points to the last node in the list.

Explain With Schematic Diagram:
Example:
          Figure 5-33 contains it schematic diagram of such a list. Observe that the null pointer appears in the FORW field of the last node in the list and also in the BACK field of the first node in the list.

Fig. 5-33 Two-way list.
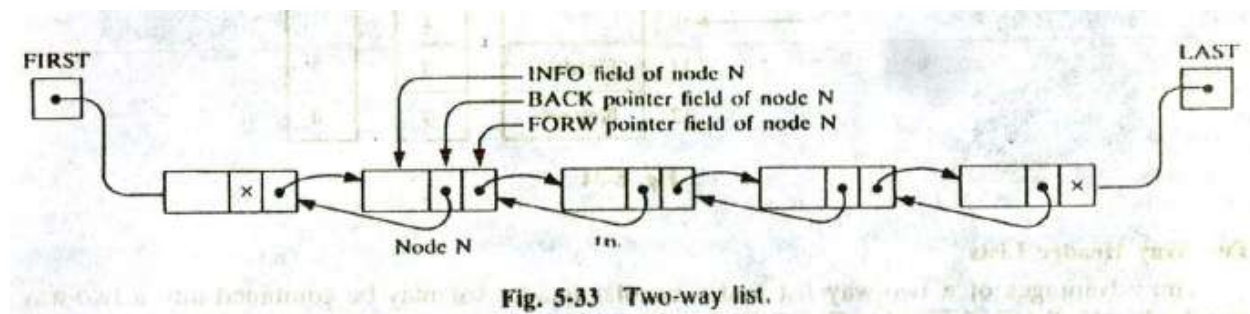
Observe that, using the variable FIRST and the pointer field FORW, we can traverse a two-way list in the forward direction as before. On the other hand, using the variable LAST and the pointer field BACK, we can also traverse the list in the backward direction.

Suppose LOCA and LOCB are the locations, respectively, of nodes A and B in a two-way list. Then the way that the pointers FORW and BACK are defined gives us the following:

    Pointer property: FORW[LOCA]= LOCB if and only if BACK [LOCB] = LOCA

In other words, the statement that node B follows node A is equivalent to the statement that node A precedes node B.

Two-way lists may be maintained in memory by means of linear arrays in the same way as one-way lists except that now we require two pointer arrays, FORW and BACK, instead of one pointer array LINK, and we require two list pointer variables, FIRST and LAST, instead of one list pointer variable START. On the other hand, the list AVAIL of available space in the arrays will still be maintained as a one-way list—using FORW as the pointer field—since we delete and insert nodes. only at the beginning of the AVAIL list.

Important of Doubly Linked List:

    📖 The first and last nodes of a doubly linked list are immediately accessible (i.e., accessible without traversal, and usually called head and tail) and therefore allow traversal of the list from the beginning or end of the list, respectively: e.g., traversing the list from beginning to end, or from end to beginning, in a search of the list for a node with specific data value. Any node of a doubly linked list, once obtained, can be used to begin a new traversal of the list, in either direction (towards beginning or end), from the given node.

    📖 The link fields of a doubly linked list node are often called **next** and **previous** or **forward** and **backward**. The references stored in the link fields are usually implemented as pointers, but (as in any linked data structure) they may also be address offsets or indices into an array where the nodes liv

    📖 Traversal of a doubly linked list can be in either direction. In fact, the direction of traversal can change many times, if desired. **Traversal** is often called **iteration**, but that choice of terminology is unfortunate, for **iteration** has well-defined semantics (e.g., in mathematics) which are not analogous to **traversal**

Question: Suppose 20 elements are maintained by array and another 10 are by Linked List. Which methods take longer time to access 9th element. Justify your answer. Exam-2014

Question: Suppose 10 elements are maintained by array and another 10 are by Linked List. Which methods take longer time to access 7th element. Justify your answer. Exam-2016

Answer:

To Access 7th element from a list,

1.To Access 7th element from Linked list need more time then arrays.

Because in array

If a is array with 10 element a[10];

We can access 7th element by a[7] either insert or delete.

Linked lists are preferable over arrays when:

a) you need constant-time insertions/deletions from the list (such as in real-time computing where time predictability is absolutely critical)

b) you don't know how many items will be in the list. With arrays, you may need to re-declare and copy memory if the array grows too big

c) you don't need random access to any elements

d) you want to be able to insert items in the middle of the list (such as a priority queue)

Arrays are preferable when:

a) you need indexed/random access to elements

b) you know the number of elements in the array ahead of time so that you can allocate the correct amount of memory for the array

c) you need speed when iterating through all the elements in sequence. You can use pointer math on the array to access each element, whereas you need to lookup the node based on the pointer for each element in linked list, which may result in page faults which may result in performance hits.

d) memory is a concern. Filled arrays take up less memory than linked lists. Each element in the array is just the data. Each linked list node requires the data as well as one (or more) pointers to the other elements in the linked list.

Array Lists (like those in .Net) give you the benefits of arrays, but dynamically allocate resources for you so that you don't need to worry too much about list size and you can delete items at any index without any effort or re-shuffling elements around. Performance-wise, arraylists are slower than raw arrays.
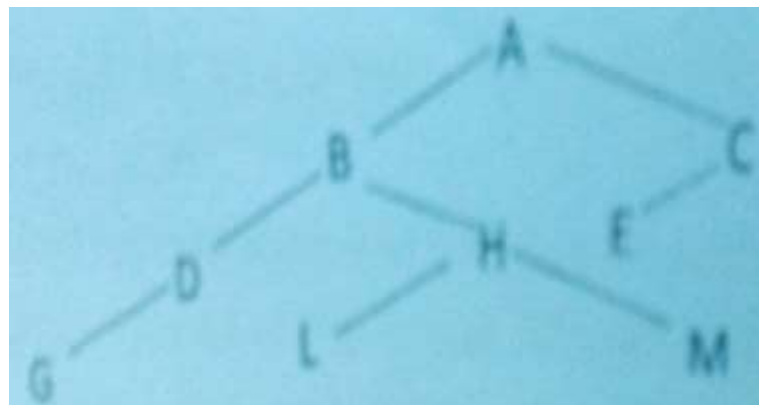
Part-B

Marks-35

Cha pter

5

Trees.
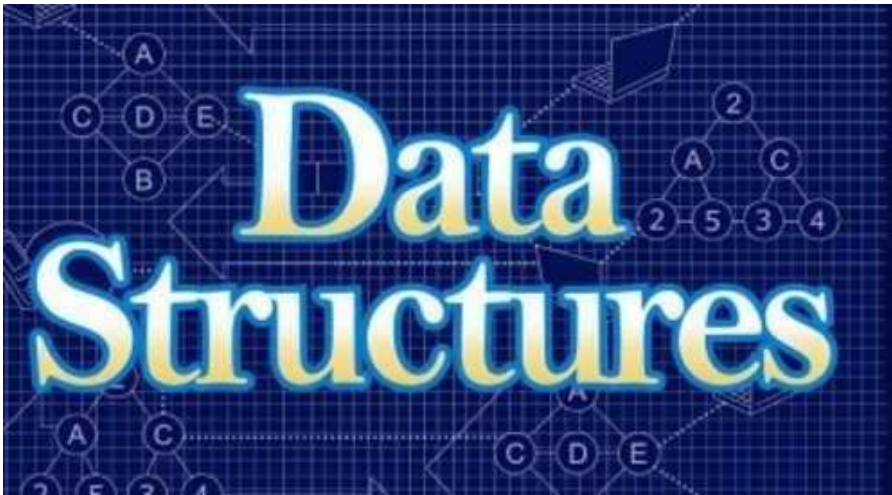
Important Question

1. Simulate the preorder traversal algorithm for the following tree. **Exam-2016**

2. Illustrate similar and copies of a tree with examples. **Exam-2015**
3. What is complete binary tree?What is the parent-child relantionship? **Exam-2015**
4. For the expression:*+a-bc*-de/fgh draw the tree and perform inorder and postorder traversal. **Exam-2015**
5. What is binary search tree? Why binary search tree is important ? **Exam-2016.,2013**
6. What is the difference between maxheap and min heap? **Exam-2016**
7. What is binary search tree? **Exam-2015**
8. Suppose the following six numbers are inserted into an empty binary search tree :33,50,45,52,12,10.Show the tree as each number is inserted into a binary search tree. **Exam-2015** .
9. Suppose the following six numbers are inserted in order into an empty binary seach tree:35,55,45,35,52,10,9 show the tree as each number is inserted into a binary search tree. **Exam-2013**
10. Simulate the maxheap algorithm for following values:77,40,90,65,20,35,95,10,15. **Exam-2015**
11. Simulate the maxheap algorithm for the following values:77,39,95,60,23,39,91,9,12 **Exam-2013**
12. Define heap, leaf and depth of tree. For 5009 nodes, find out the depth of tree. **Exam-2014**

13. Discuss the linked representation of binary tree in memory. **Exam-2014**
14. Simulate the preorder traversal algorithm for the following tree. **Exam-2014**
15. Simulate the preorder traversal algorithm for the following tree. **Exam-2016**

Question: What is binary tree? Discuss with Example?

Answer:

Binary Trees:

A binary tree T is defined as a finite set of elements, called nodes, such that:

    (a) T is empty (called the null tree or empty tree), or

    (b) T contains a distinguished node R, called the root of T, and the remaining nodes of T forni an ordered pair of disjoint binary trees T and T2.

If T does contain a root R, then the two tree T_1 and T_2 arc called, respectively, the left and right subtree of R. If T_1 is nonempty, then its root is called the left successor of R; similarly, if T_2 is nonempty, then its root is called the right successor of B.

Question: Illustrate similar and copies of a tree with examples. Exam-2015

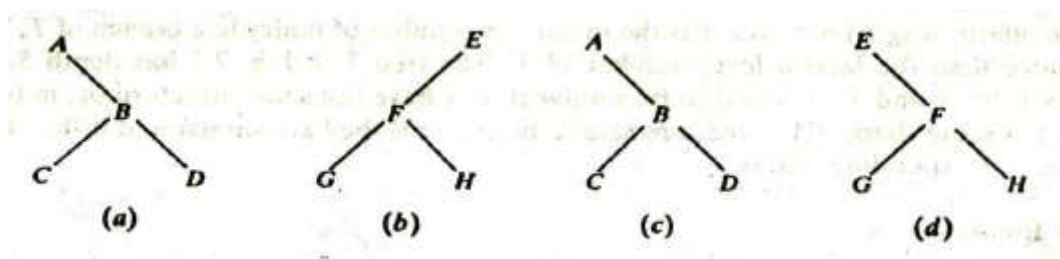**Answer:**

Similar of a Tree:

        The three trees (a), (c) and (a) and (d) are similar
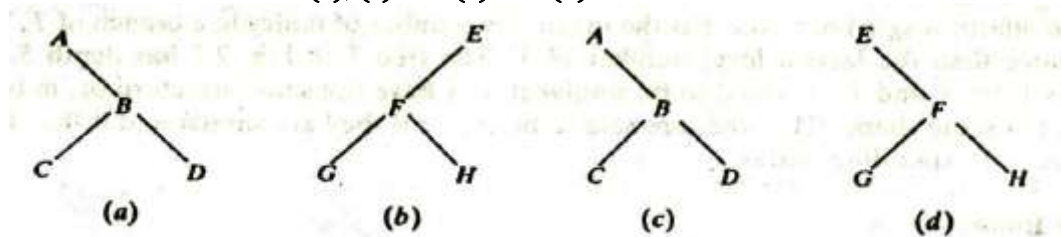
Copy of Tree:

        In particular, the trees (c) are copies since they also have the same data at corresponding nodes

The tree (b) is neither similar nor a copy of the tree (d) because, in a binary tree, we distinguish between a left successor and a tight successor even when there is only one successor.

Question: What is complete binary tree? What is the parent-child relationship? Exam-2015

Complete Binary Tree:

        A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.
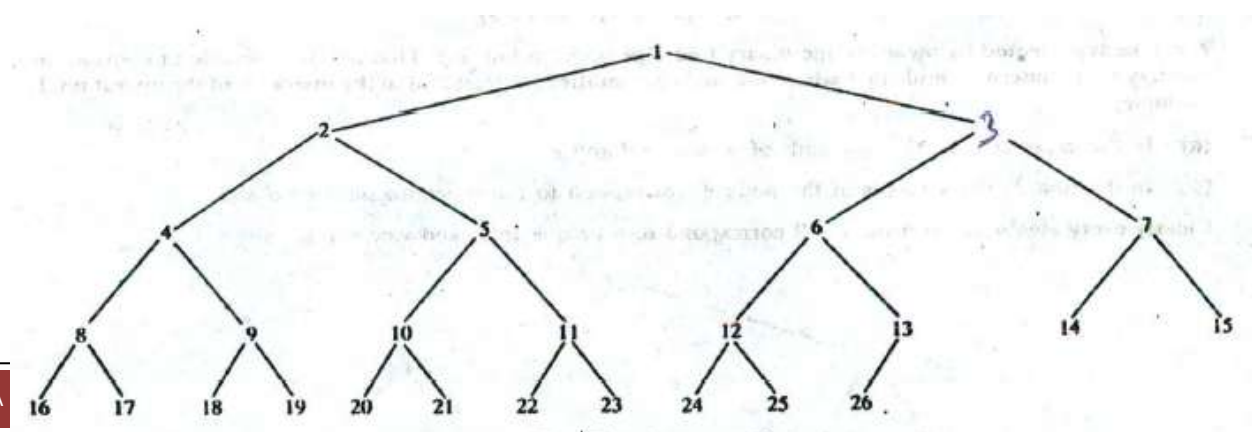

Fig. 7-4 | Complete tree T_{26}.

Parent-Child Relationship :

The term parent-child relationship refers to the unique and enduring bond between a caregiver and his or her child.

To understand the parent-child relationship, we must look at the ways that parents and children interact with one another physically, emotionally, and socially. Think about your parents. How did your relationship with your parents contribute to who you are today, or did it? Many psychologists believe that the relationships between parents and children are very important in determining who we become and how we relate to others and the world.

A heap is an array , where there are parent child relationships, and the index of a child is 2 * parent index, or 2* parent index + 1 , and a child has an order after the parent, in some concrete ordering scheme injected by a client program of a heap. There is importance in maintaining the ordering n variant after the heap is changed. Some (Sedgewick and Wayne), have coined the term "swim" and "sink", where maintenance of the invariant involves a invariant breaking item swimming up above children of lower ordering, and then sinking below any children of higher ordering, as there may be two children , so one can swim above a lower ordered child, and still have another child with higher ordering

Question: For the expression:*+a-bc*-de/fgh   draw the tree and perform inorder and postorder traversal. Exam-2015
Answer:
Question: What is binary search tree? Why binary search tree is important ? Exam-2016.,2013
Question: What is binary search tree? Exam-2015
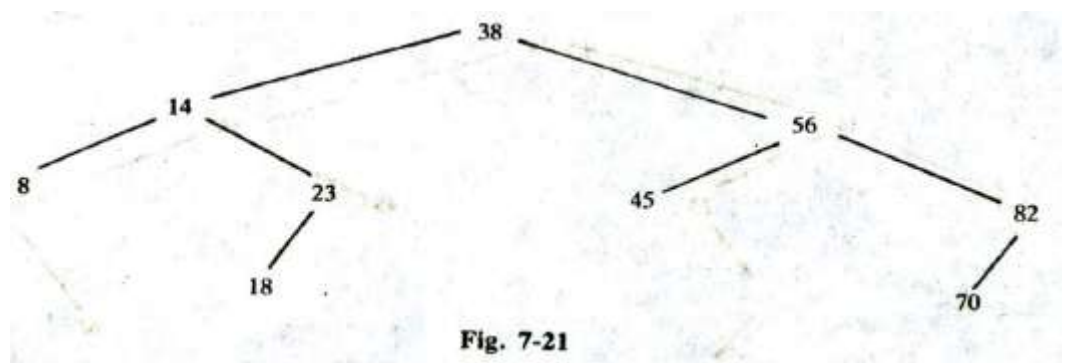**Answer:**

Binary Search Tree:
            The binary tree T is called a binary search tree (or binary sorted tree) if each node N of has the following property:
  📖 The value at N is greater than every value in the left subtree of N and
  📖 N is less than every value in the right subtree of N. (It is not difficult to see that this property guarantees that the inorder traversal of T will yield a sorted listing of the elements of T.)

Example of Binary Search Tree:
            Consider the binary tree T in Fig. 7-21. T is a binary search tree; that is, every node N in T exceeds every number in its left subtree and is less than every number in its right subtree.

Suppose the 23 were replaced by 35. Then T would still be a binary search tree. On the other hand, suppose the 23 were replaced by 40. Then T would not be a binary search tree, since the 38 would not be greater than the 40 in its left subtree.

Fig. 7-21

Question: What is the difference between maxheap and min heap? Exam-2016
Answer:
     There's no real difference between min and max heaps, except how the comparison is interpreted.

Min Heap	Max Heap
Min-Heap − Where the value of the root node is less than or equal to either of its children.	**Max-Heap** − Where the value of the root node is greater than or equal to either of its children

A min-heap supports two operations:
  INSERT(heap, element)
  element REMOVE_MIN(heap)	

Question: Suppose the following six numbers are inserted into an empty binary search tree :33,50,45,52,12,10.Show the tree as each number is inserted into a binary search tree. Exam-2015 .

Question: Suppose the following six numbers are inserted in order into an empty binary search tree:35,55,45,35,52,10,9 show the tree as each number is inserted into a binary search tree. Exam-2013

Question: Define heap, leaf and depth of tree. For 5009 nodes, find out the depth of tree. Exam-2014

<u>Heap of a Tree:</u>

        A heap is a specialized tree-based data structure that satisfies the heap property:

If A is a parent node of B, then the key (the value) of node A is ordered with respect to the key of node B with the same ordering applying across the heap.
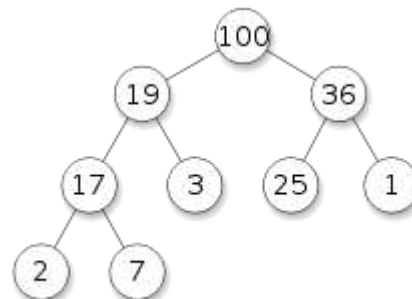
Or

Heap is a special case of balanced binary tree data structure where the root-node key is compared with its children and arranged accordingly. If α has child node β then −

        key(α) ≥ key(β)

As the value of parent is greater than that of child, this property generates Max Heap. Based on this criteria, a heap can be of two types

      📖 **Max Heap**
      📖 **Min Heap**

<u>Leaf of a Tree:</u>

        A terminal node is called a leaf, and a path ending in a leaf is called a branch.

Or

        An internal node (also known as an inner node, inode for short, or branch node) is any node of a tree that has child nodes. Similarly, an external node (also known as an outer node, leaf node, or terminal node) is any node that does not have child nodes.

<u>Depth of a Tree:</u>

        The depth (or height) of a tree T is the maximum number of nodes in a branch of T.

Or

        The height of a node is the length of the longest downward path to a leaf from that node. The height of the root is the height of the tree. The depth of a node is the length of the path to its root (i.e., its root path).

Question: Simulate the maxheap algorithm for following values:77,40,90,65,20,35,95,10,15. Exam-2015

Question: Simulate the maxheap algorithm for the following values:77,39,95,60,23,39,91,9,12 Exam-2013

Question: Discuss the linked representation of binary tree in memory. Exam-2014

Answer;

Linked Representation of Binary Trees:

Consider it binary tree T. Unless otherwise stated or implied, T will he maintained in memory by means of -,I linked representation which uses three parallel arrays, INFO, LEFT and RIGHT, and a pointer variable ROOT as follows. First of all, each node N of will correspond to a location K such that:

(1) INFO[K] contains the data at the node N.

(2) LEFT[K] contains the location of the left child of node N.

(3) RIGHT[K] contains the location of the right child of node N.

Furthermore. ROOT will contain the location of the root R of T. If any subtree is empty, then the corresponding pointer will contain the null value if the tree T itself is empty, then ROOT will contain
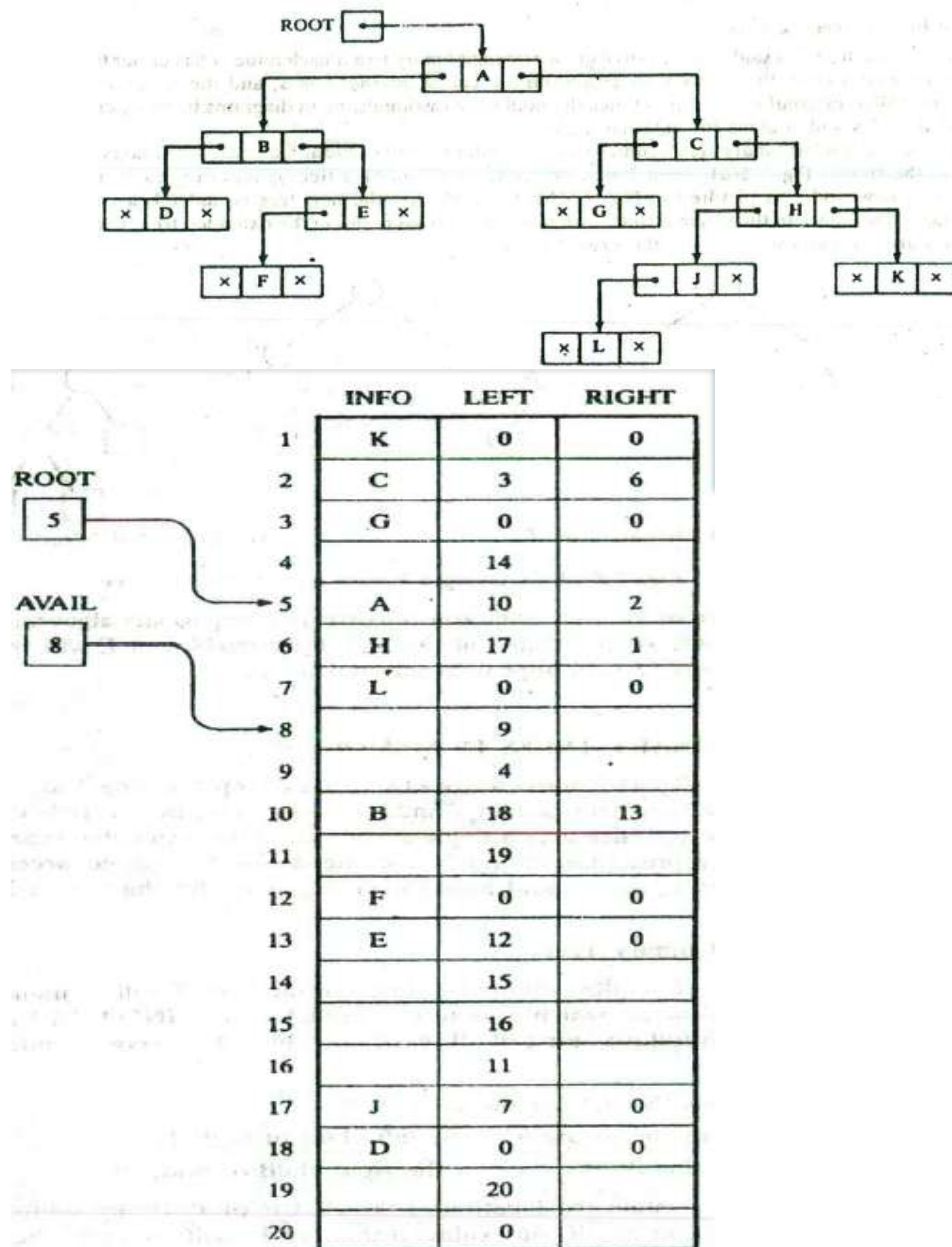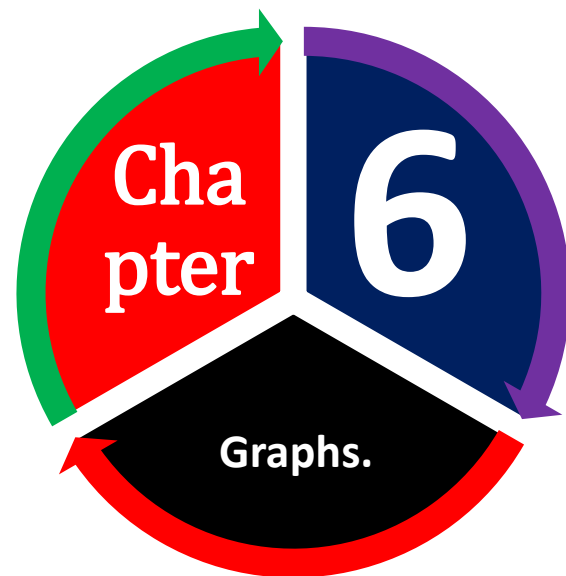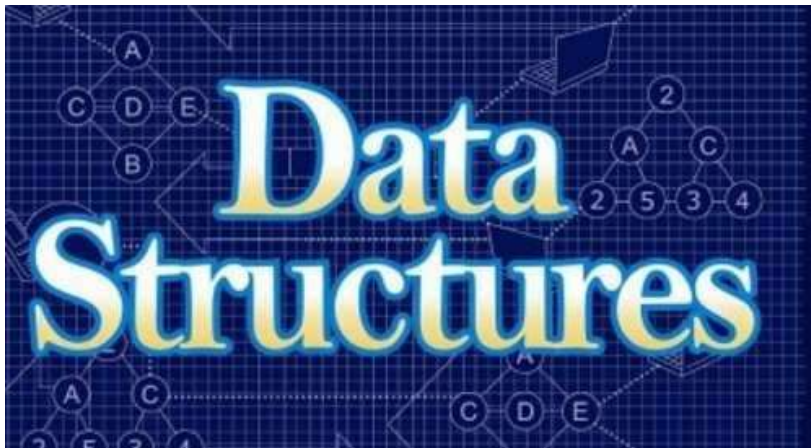the null value.

	INFO	LEFT	RIGHT
1	K	0	0
2	C	3	6
3	G	0	0
4		14	
5	A	10	2
6	H	17	1
7	L	0	0
8		9	
9		4	
10	B	18	13
11		19	
12	F	0	0
13	E	12	0
14		15	
15		16	
16		11	
17	J	7	0
18	D	0	0
19		20	
20		0	

ROOT 5

AVAIL 8

Fig 7.7

Question: Simulate the preorder traversal algorithm for the following tree. Exam-2014,2016

Cha pter **6**

Graphs.

Important Question

1. Define weighted graph and directed graph with example**. Exam-2016**

2. Define the following terms: (i)Degree of a node,(ii)Isolated node,(iii)Path,(iv)Multi Graph. **Exam-2015**

3. Define the following graph terms: (i) Adjacent nodes,(ii) Cycle,(iii) Connected graph, and (iv) Weighted graph. **Exam-2014**

4. Define the following terms: Path,Cycle,Connected Graph and Multi-graph. **Exam-2013**

5. What is Directed Graph? Explain. **Exam-2014**

6. Discuss the sequential Representation of Graph with example. **Exam-2014**

7. Consider the following adjacency matrix below:

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. **Exam-2013**

8. Consider the following adjacency matrix below:

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. **Exam-2014**

9. Consider the following adjacency matrix below:

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. **Exam-2015**

10. Consider the following adjacency matrix below:

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. **Exam-2016**

11. **Use the Warshall's algorithm to find the shortest path matrix of the weighted matrix given below. Exam-2016**

$$\begin{pmatrix} 6 & 8 & 0 & 0 \\ 3 & 0 & 0 & 9 \\ 5 & 8 & 3 & 6 \\ 6 & 2 & 3 & 0 \end{pmatrix}$$

16. Discuss the linked representation of Graph with example **Exam-2015**
17. Discuss the sequential representation of graph with example**. Exam-2016**
18. How many ways a graph can be traversed? What is the significance of the STATUS field? **Exam-2014**
19. Explain the depth-first search technique. **Exam-2014**
20. Consider the adjacency list of the graph G in the following table. Draw the graph and find out the path from A to H with minimum number First Search. **Exam-2015,2014**

21. Consider of the following

Node	Adjacency	Node	Adjacency
A	G,E	E	C
B	C	F	A,B
C	F	G	B,C,E
D	C	H	D

the adjacency list graph G in the table. Find the nodes that are reachable from node C using Depth first search. **Exam-2016**

Node	Adjacency	Node	Adjacency
A	E,G	E	H
B	C	F	A,B
C	F	G	B,C,E
D	C	H	D

Question: What is Graph? Discuss with Example?
Answer:
          A graph C consists of two things:
     (1) A set V of elements called nodes' (or points or vertices)
     (2) A set E of edges such that each edge e in E is identified with a unique (unordered) pair [u, v]
of nodes in V, denoted by e = [u, v].

Sometimes we indicate the parts of a graph by writing G = (V. E).

Suppose e = [u, v]. Then the nodes **u and v are called the endpoints of e, and u and u arc said to be adjacent nodes or neighbors.** The degree of a node u, written deg(u), is the number of edges containing u. If deg(u) = 0—that is, if u does not belong to any edge—then u is called an isolated node.

Question: What is Directed Graph? Explain. Exam-2014
Question: Define weighted graph and directed graph with example. Exam-2016
**Weighted Graph:**
          Definition: A graph having a weight, or number, associated with each edge.
Or
     A graph G is said to be labeled if its edges are assigned data. **In particular, G is said to be weighted graph if each edge e in G is assigned a nonnegative numerical value w(e) called the weight or length of e.**
In such a case, each path P in G is assigned a weight or length which is the sum of the weights of the edges along the path P. If we are given no other information about weights, **we may assume any graph G to be weighted by assigning the weight w(e) = 1 to each edge a in G.**


(d) Weighted graph.
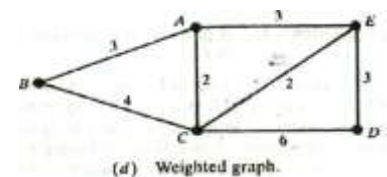
Directed Graph:
          A directed graph G, also called a digraph Or graph, is-the same as a multigraph except that each edge e in G is assigned a direction, or in other words, each edge e is identified wiih an ordered pair (u, v) of nodes in G rather than an unordered pair [u, v].

Suppose G is a directed graph with a directed edge e = (u, v). Then e is also called an arc.
Moreover, the following terminology is used:
          (1) e begins at u and ends at v -
          (2) u is the origin or initial point of e, and u is the destination or terminal point of e.
          (3) u is a predecessor of v, and v is a successor or neighbor of u.
   (4) u is adjacent to v, and v is adjacent to u.
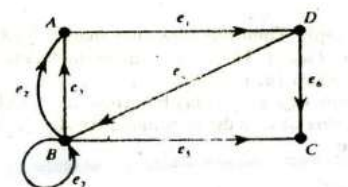
Question: Define the following terms: Path, Cycle, Connected Graph and Multi-graph. Exam-2013
Question: Define the following terms: (i)Degree of a node,(ii)Isolated node,(iii)Path,(iv)Multi Graph. Exam-2015
Degree of node:
          The degree of vertex is number of edges that are connected to the vertex. In figure un directed graph, degree of vertex-1 is 2 and vertex 3 is 3.
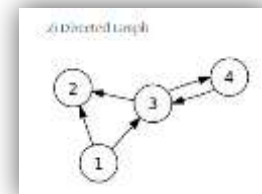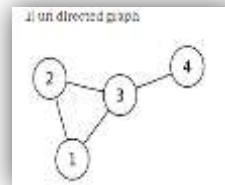
In Degree: This is applicable only for directed graph. This represents the number of edges incoming to a vertex. In above directed graph, degree of 1 is 0 and degree of 2 is 2.

Out degree: This is also applicable only for directed graph. This represents the number of edges outgoing from a vertex. In above directed graph, degree of 1 is 2 and degree of 2 is 0.
Degree of Graph: Sum of degrees of all the vertices of G

Isolated node:
An isolated vertex is a vertex with degree zero

Path:
a path in a graph is a finite or infinite sequence of edges which connect a sequence of vertices which, by most definitions, are all distinct from one another.
In a directed graph, a directed path (sometimes called dipath) is again a sequence of edges (or arcs) which connect a sequence of vertices, but with the added restriction that the edges all be directed in the same direction.

Multi Graph:
A multigraph is a graph which is permitted to have multipleedges (also called parallel edges), that is, edges that have the same end nodes. Thus two vertices may be connected by more than one edge.
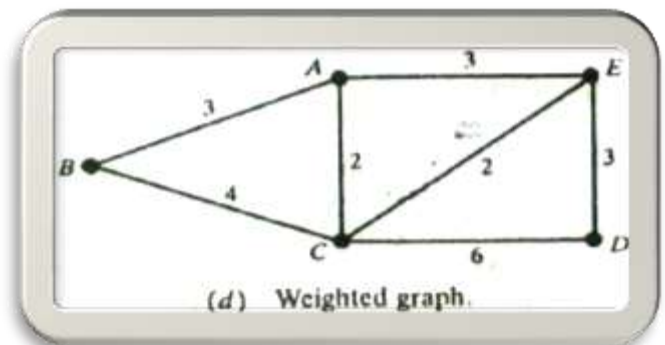
(d) Weighted graph.

Question: Define the following graph terms: (i)
Adjacent nodes,(ii) Cycle,(iii) Connected graph, and (iv) Weighted graph. Exam-2014

Adjacent Nodes:
Suppose **e = [u,v].** Then the nodes u and v are called the endpoints of e, and u and u are said to be adjacent nodes or neighbors.

Cycle:
A cycle graph or circular graph is a graph that consists of a single cycle, or in other words, some number of vertices connected in a closed chain. The cycle graph with n vertices is called Cn.

Or
A cycle is a closed simple path with length 3 or more. A cycle of length k is called a k-cycle.
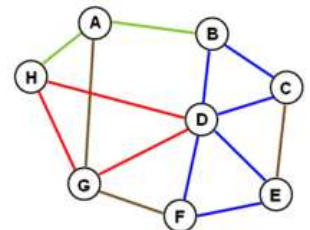
Connected graph:
Graph G is connected if and only if there is a simple path between any two nodes In G.

Question: Discuss the sequential Representation of Graph with example. Exam-2014,2016

<u>Sequential Representation Of Graphs:</u>

There are two standard ways of maintaining a graph G in the memory of a computer.
- 📖 Adjacency Matrix;
- 📖 Path Matrix

<u>Adjacency Matrix:</u>

Suppose **G** is a simple directed graph with in nodes, and suppose the nodes of **G** have been ordered and are called $v_1, v_2, v_3, \ldots \ldots v_n$. Then the adjacency Matrix $A = (a_{ij})$ of the graph G is the $m * n$ in matrix defined as follows:

$$a_{ij} = \begin{cases} 1 \ if \ v_i \ is \ adjacent \ to \ v_j, that \ is \ there \ is \ an \ edge \ (v_i, v_j) \\ 0 \ \ otherwise \end{cases}$$

Such a matrix A, which contains entries of only **0 and 1**, is called a bit matrix or a Boolean matrix.

//…………………………………………………………………………………………………

The adjacency matrix **A of the graph G does depend on the ordering of the nodes of G;** that is, a different ordering of the nodes may result in a different adjacency 'matrix. However, the matrices resulting from two different orderings are closely related in that one can be obtained from the there by simply interchanging rows and columns. Unless otherwise stated, we will assume that the nodes of our graph G have a fixed ordering.

Suppose G is an undirected graph. Then the adjacency matrix A of G will be a symmetric matrix, i.e., one in which $a_{ij} = a_{ji}$ for every i and 1 . This follows from the fact that each undirected edge (u, v) corresponds to the two directed edges (u, u) and (u. u).

The above matrix representation of a graph may he extended to multigraphs. Specifically. if G is a multigraphs, then the adjacency matrix of G is the in x in matrix $A = (a_{ij})$ defined by setting a, equal to the number of edges from $v_i \ to \ V_j$.

<u>Path Matrix:</u>

Let G be a simple directed graph with m nodes, $v_1, v_2, v_3, \ldots \ldots v_n$.The path matrix or reachability matrix of G is the m-square matrix $P = (p_{ij})$ defined as follows:

$$p_{ij} = \begin{cases} 1 \ if \ v_i \ is \ adjacent \ to \ v_j, that \ is \ there \ is \ an \ edge \ (v_i, v_j) \\ 0 \ \ otherwise \end{cases}$$

Suppose there is a path from $v_i \ to \ v_j$, Then there must be a simple path from $v_i \ to \ v_j$, when $v_i \neq v_j$, or there must be a cycle from $v_i \ to \ v_j$, when $v_i = v_j$. Since G has only m nodes, such a simple path must have length $m - 1$ or less, or such a cycle must have length m or less This means that there is a nonzero ij entry in the matrix B_m defined at the end of the preceding subsection.

Question: Consider the following adjacency matrix below:

$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Now find out A^2, A^3, A^4, B^4 and from that make the path matrix and tell whether this is strongly connected or not.

Answer:

Consider the graph G in Fig. 8-3. Suppose the nodes are stored in memory in a linear array DATA as follows:

DATA: X, Y, Z, W

Then we assume that the ordering of the nodes in G is as follows: $v_1 = X, v_2 = Y, v_3 = Z, v_4 = W$.
The adjacency matrix A of G is as follows:

$$A=\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Note that the number of 1's in A is equal to the number of edges in G.
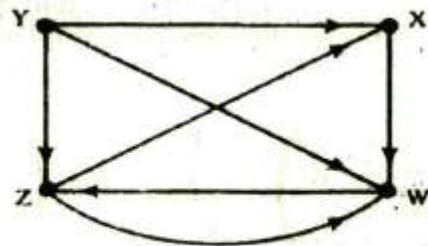

Fig. 8-3

The power of the A^2, A^3, A^4 of the matrix A is $=$

$$A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad A^4 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 2 & 0 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Accordingly, in particular, there is a path of length 2 from v_4 to v_1 , there are two paths of length 3 from v_2 to v_3, and there are three paths of length 4 from v_2 to v_4

Suppose we now define the matrix B, as follows:

$$B_m = A + A^2 + A^3 + \cdots + A^m$$

$$B_4 = \begin{pmatrix} 1 & 0 & 2 & 3 \\ 5 & 0 & 6 & 8 \\ 3 & 0 & 3 & 5 \\ 2 & 0 & 3 & 3 \end{pmatrix} \qquad \text{and} \qquad P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Examining the **matrix P**. we see that the node v_2 is not reachable from any of the other nodes.

   Recall that a directed graph **G is said to be strongly connected if, for any pair of nodes u and v in G, there are both a path from u to v and a path from v to u**. Accordingly, G is strongly connected if and only if **the path matrix P of G has no zero entries**. Thus the graph G in Fig. 8-3 is not strongly Connected .

Question: Consider the following adjacency matrix below:

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

   Now find out A^2, A^3, A^4, B^4 and from that make the path matrix and tell whether this is strongly connected or not. Exam-2013
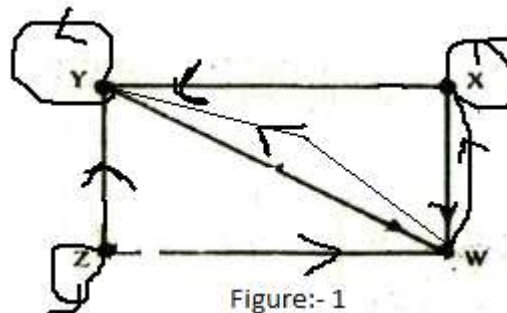**Answer:**

Consider the graph *G* in Fig. 8-3. Suppose the nodes are stored in memory in a linear array DATA as follows:

DATA: X, Y, Z, W

Then we assume that the ordering of the nodes in G is as follows: $v_1 = X, v_2 = Y, v_3 = Z, v_4 = W$. The adjacency matrix *A* of G is as follows:

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Note that the number of 1's in *A* is equal to the number of edges in G.

Figure:- 1

The power of the A^2, A^3, A^4 *of the matrix A is* =

Question: Consider the following adjacency matrix below:

$$\begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. Exam-2014

Question: Consider the following adjacency matrix below:

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. Exam-2015

Question: Consider the following adjacency matrix below:

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Now find out A2,A3,A4,B4 and from that make the path matrix and tell whether this is strongly connected or not. Exam-2016

Question: Use the Warshall's algorithm to find the shortest path matrix of the weighted matrix given below?
Answer:

$$W = \begin{pmatrix} 7 & 5 & 0 & 0 \\ 7 & 0 & 0 & 2 \\ 0 & 3 & 0 & 0 \\ 4 & 0 & 1 & 0 \end{pmatrix}$$

Applying the modified Wirshall's algorithm, we obtain the following matriccs Q0. Q1, Q,, Q3 and Q4 Q. To the right of cach matrix Q,, we show the matrix of paths which correspond to the lengths in the matrix Q.
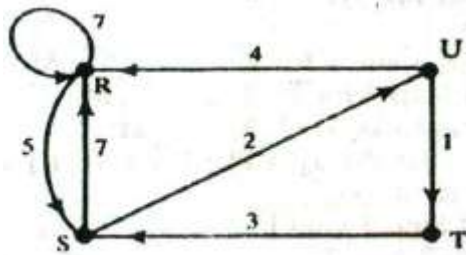

Fig. 8-6

$$Q_0 = \begin{pmatrix} 7 & 5 & \infty & \infty \\ 7, & \infty & \infty & 2 \\ \infty & 3 & \infty & \infty \\ 4 & \infty & 1 & \infty \end{pmatrix} \qquad \begin{pmatrix} RR & RS & - & - \\ SR & - & - & SU \\ - & TS & - & - \\ UR & - & UT & - \end{pmatrix}$$

$$Q_1 = \begin{pmatrix} 7 & 5 & \infty & \infty \\ 7 & 12 & \infty & 2 \\ \infty & 3 & \infty & \infty \\ 4 & ⑨ & 1 & \infty \end{pmatrix} \qquad \begin{pmatrix} RR & RS & - & - \\ SR & SRS & - & SU \\ - & TS & - & - \\ UR & URS & UT & - \end{pmatrix}$$

$$Q_2 = \begin{pmatrix} 7 & 5 & ⑳ & 7 \\ 7 & 12 & \infty & 2 \\ 10 & 3 & \infty & 5 \\ 4 & 9 & 1 & 11 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & - & RSU \\ SR & SRS & - & SU \\ TSR & TS & - & TSU \\ UR & URS & UT & URS \end{pmatrix}$$

$$Q_3 = \begin{pmatrix} 7 & 5 & \infty & 7 \\ 7 & 12 & \infty & 2 \\ 10 & 3 & \infty & 5 \\ 4 & ④ & 1 & 6 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & - & RSU \\ SR & SRS & - & SU \\ TSR & TS & - & TSU \\ UR & UTS & UT & UTSU \end{pmatrix}$$

$$Q_4 = \begin{pmatrix} 7 & 5 & 8 & 7 \\ 7 & 11 & 3 & 2 \\ ⑨ & 3 & 6 & 5 \\ 4 & 4 & 1 & 6 \end{pmatrix} \qquad \begin{pmatrix} RR & RS & RSUT & U \\ SR & SURS & SUT & SU \\ TSUR & TS & TSUT & TSU \\ UR & UTS & UT & UTSU \end{pmatrix}$$

$Q_1[4, 2] = MIN(Q_0[4, 2], Q_0[4, 1] + Q_0[1, 2]) = MIN(\infty, 4 + 5) = 9$

$Q_2[1, 3] = MIN(Q_1[1, 3], Q_1[1, 2] + Q_1[2, 3]) = MIN(\infty, 5 + \infty) = \infty$

$Q_3[4, 2] = MIN(Q_2[4, 2], Q_2[4, 3] + Q_2[3, 2]) = MIN(9, 3 + 1) = 4$

$Q_4[3, 1] = MIN(Q_3[3, 1], Q_3[3, 4] + Q_3[4, 1]) = MIN(10, 5 + 4) = 9$

Question: Use the Warshall's algorithm to find the shortest path matrix of the weighted matrix given below. Exam-2016

$$\begin{pmatrix} 6 & 8 & 0 & 0 \\ 3 & 0 & 0 & 9 \\ 5 & 8 & 3 & 6 \\ 6 & 2 & 3 & 0 \end{pmatrix}$$

Question: Discuss the linked representation of Graph with example Exam-2015
Answer:

Let G be a directed graph with m nodes. G is usually represented in memory by a linked representation, also called an adjacency structure, which is described in this section.

Consider the graph G in Fig. 8-7(a). The table in Fig. 8-7(b) shows each node in G followed by its adjacency list, which is its list of adjacent nodes, also called its successors or neighbors. Figure 8-8 shows a schematic diagram of a linked representation of G in memory. specifically, the linked representation will contain two lists (or tiles), a node list NODE and an edge list EDGE, as follows.
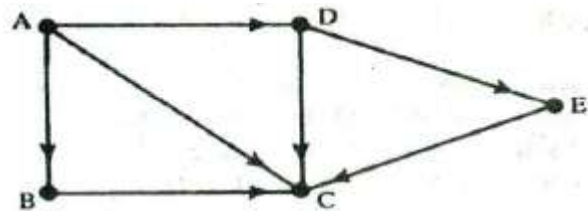
Node	Adjacency List
A	B, C, D
B	C
C	
D	C, E
E	C

(a) Graph G.

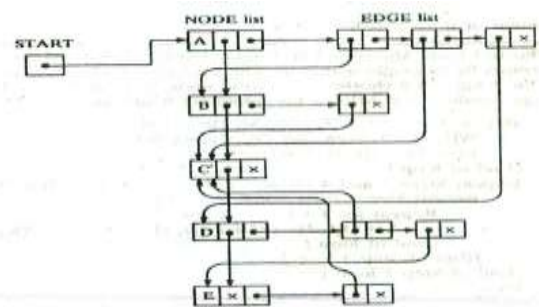(b)   Adjacency lists of G.

Fig. 8-7

Fig. 8-8

(a) Node List.

Each element in the list NODE will correspond to a node in G, and it will be a record of the form:

NODE	NEXT	ADJ	

Here NODE will be the name or key value of the node, NEXT will be a pointer to the next node in the list NODE and ADJ will be a pointer to the first clement in the adjacency list of the node, which is maintained in the list EDGE. The shaded area indicates that there may be other information in the record, such as the indegree INDEG of the node, the outdegree etc.

(b)Edge List:

Each element in the list EDGE will correspond to an edge of G and will be a record of the form.

DEST	LINK	

The field DEST will point to the location in the list NODE of the destination or terminal node of the edge. The field LINK will link together the edges with the same initial node, that is, the nodes in the same adjacency list. The shaded area indicates that there may be other information in the record corresponding to the edge, such as a ficld.EDGE containing the labeled data of the edge when G is a labeled graph, a field WEIGHT containing the weight of the edge when G is weighted graph, and soon. We also need a pointer variable AVAILE for the list of available space in the list EDGE.

Figure 8-9 shows how the graph G in Fig. 8-7(a) may appear in memory. The choice of 10 locations for the list NODE and 12 locations for the list EDGE is arbitrary.
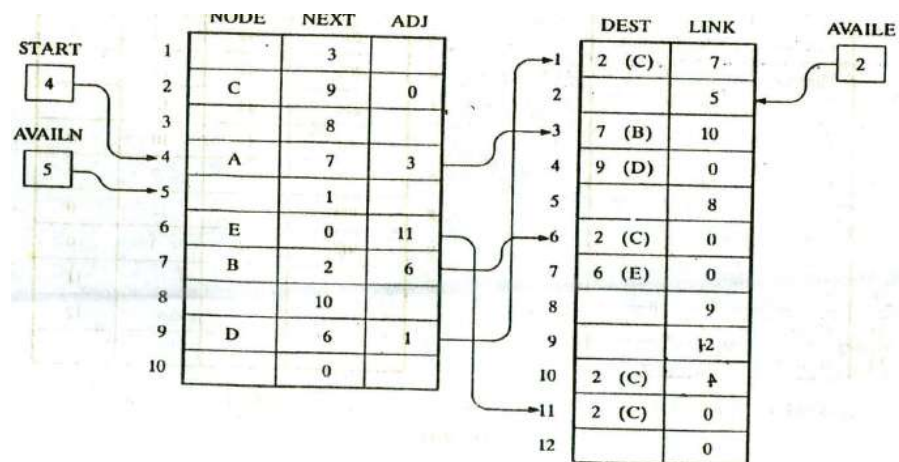
Question: How many ways a Graph can be traversed? What is the significance of the STATUS field? Exam-2014

Question: Explain the depth-first search technique. Exam-2014
<u>Traversing A Graph:</u>
     Many graph algorithms require one to systematically examine the nodes and edges of a graph G. There are two standard ways that this is done.

 📖 **Breadth-First Search(BFS)**
 📖 **Depth-First Search(DFS)**

  The breadth-first search **will use a queue**
  The depth-first search **will use a stack**.
As an auxiliary structure to hold nodes for future processing, and analogously.

During the execution of our algorithms, each node N of G will be in one of three states, called the status of N, as follows:

  STATUS = 1: (Ready state.) The initial State of the node N.
  STATUS = 2: (Waiting state.) The node N is on the queue or stack, waiting to be processed,
  STATUS = 3: (Processed state.) The node N has been processed.
We now discuss the two searches separately.

<u>Breadth-First Search</u>
     In BFS First we examine the starting node A. Then we examine all the neighbors of A. Then we examine all the neighbors of the neighbors of A. And so on. Naturally, we need to keep track of the neighbors of a node, and we need to guarantee that no node is processed more than once. This is accomplished by using a queue to hold nodes that are waiting to be processed, and by using a field STATUS which tells us the current status of any node. The algorithm follows.

Algorithm A: This algorithm executes a breadth first search on a graph 0 beginning at a starting node A.
    1. Initialize all nodes to the ready state (STATUS 1). Put the starting node A in QUEUE and change its status to the waiting state (STATUS = 2).
    3. Repeat Steps 4 and 5 until QUEUE is empty:
    4. Remove the front node N of QUEUE. Process N and change the status of N
     to the processed state (STATUS = 3).

5. Add to the rear of QUEUE all the neighbors of N that arc in the steady state
        (STATUS = 1), and change their status to the waiting state (STATUS = 2).
    [End of Step 3 loop.]
6. Exit.

<u>Depth-First Search:</u>
         The general idea behind a depth-first DFS beginning at a starting node A is as follows. **First we examine the starting node A. Then we examine each node N along a path P which begins at A; that is, we process a neighbor of A. then a neighbor of a neighbor of A. and so on. After coming to a "dead end," that is, to the end of the path P**, we backtrack on P until we can continue along another path ***P′***.. The algorithm follows.

Algorithm : This algorithm executes a depth-first search on a graph G beginning at **it** starting node

        1.Initialize all nodes to the ready state (STATUS = 1).
        2. Push the starting node A onto STACK and change its status to the waiting state (STATUS = 2).
        3. Repeat Steps 4 and 5 until Stack: is empty.
        4.Pop the top node N of STACK. Process N and change its status to the processed state ' (STATUS= 3).
        5.Push onto STACK all the neighborhood of N that are still in the ready state (STATUS =1)   and   change their status to the wailing state (STATUS = 2). End of Step 3 loop.)
        6. Exit.

Question: Consider the adjacency list of the graph G in the following table. Draw the graph and find out the path from A to H with minimum number First Search. Exam-2015,2014

Node	Adjacency	Node	Adjacency
A	G,E	E	C
B	C	F	A,B
C	F	G	B,C,E
D	C	H	D

Question: Consider the adjacency list of the graph G in the following table. Find the nodes that are reachable from node C using Depth first search.   Exam-2016

Node	Adjacency	Node	Adjacency
A	E,G	E	H
B	C	F	A,B
C	F	G	B,C,E
D	C	H	D

Chapter 7

Symbol Tables
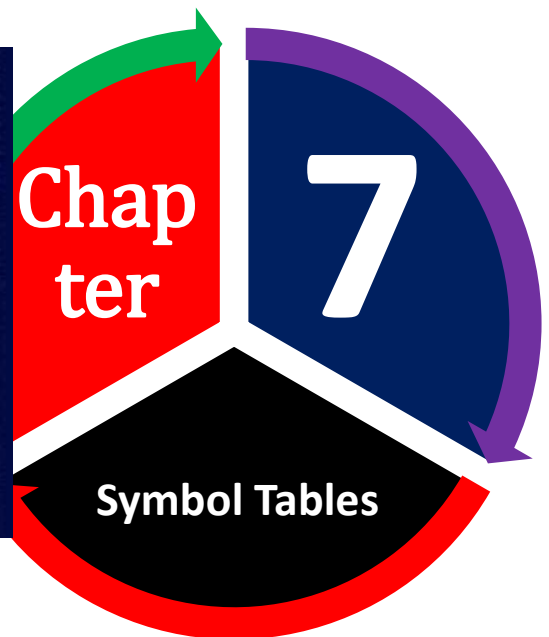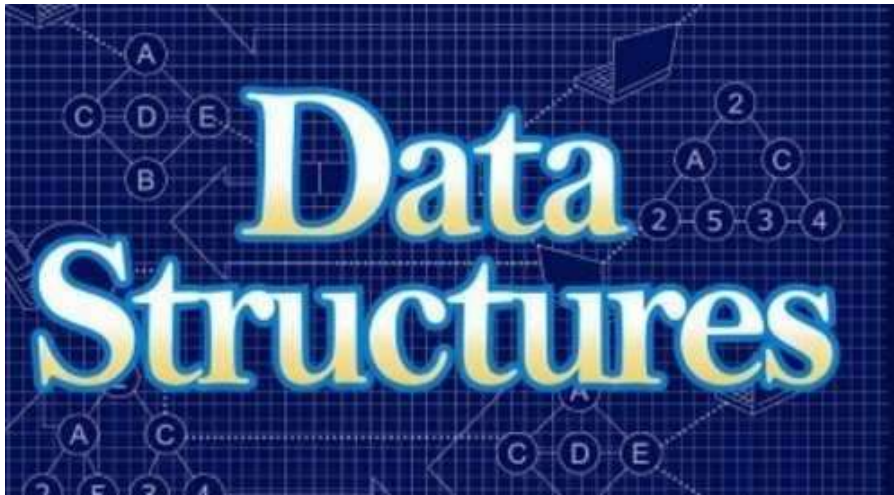
Important Question

1. What are the various hashing techniques ? Give suitable example. **Exam-2013**

2. How will you resolve the collisions, while inserting element into the hash table using separate chaining method? Write the routine for inserting elements into a hash table using the above mentioned technique. **Exam-2013**

Question: What are the various hashing techniques ? Give suitable example. Exam-2013
Answer:

<u>Hashing</u>:
The search time of each algorithm depends on the number n of elements in the collection S of data. This section discusses a searching technique, called hashing or hash addressing, which is essentially independent of the number n.

<u>Various Hashing Techniques</u>:
<u>Firstly:</u>   We assume that there is a file F of n records with a set K of keys which uniquely determine the records in F.

<u>Secondly,</u> we assume that F is maintained in memory by a table T of m memory locations and that L is the set of memory addresses of the locations in T. For notational convenience, we assume that the keys in K and the addresses in L are (decimal) integers. (Analogous methods will work with binary integers or with keys which arc character strings, such as names, since there are standard ways of representing strings by integers.)

(a) !)ic,. ion ,,;ctijoI. Choose .a flu iii her in IalL dUln the n U riibct it of keys iii K. (The number in is usually chosen to be a prime ii umber or a number without small divisors, s ince his frequently minimizes the number of collisions.) The hash function II i ,, defined by 11(k) = k (mod in) or 11(k) k (mud in) 4 1 I le ic k (mod ,n) denotes the remainder when k is divided by in. The second formula is used when we want the hash addresses to range frort I to in ra t her than from 0 to iii –

(b) Mia'suurc mci/run. ihie key A is squared. Then the hash function 11 is defined 1w 11(1<) = I where / is Obtained by deleting 1.1 igi Is from both ends of k ' .We emphasize that the same positions of k 2 must he used for all of the keys.

(b) loIdznç method. the key k is partitioned into a fiLl ill her of parts, k,, k, , where each part, except possibly tire last, has the same number of digits as the required address. Then the parts are added together, ignoring the last carry. That is, ...I-k,where the icar.ling-digit carries. it any, are ignored. Sometimes, for extra ''milling, the even-numbered palls, k,, k 1 ,.., a re each reversed before the addition

(c)

Question: How will you resolve the collisions,while inserting element into the hash table using separate chaining method?Write the routine for inserting elements into a hash table using the above mentioned technique. Exam-2013
**Answer:**

Suppose we want to add *it* record *1?* with key *k* to our tile *F,* but suppose the memory location address */1 1 A)* is alread y occupied. This situation *IS* called *coll,.cwn.* This subsection discusses two general ways of rcsolvi tig collisions. The particular procedure that one chooses depends on many factors. f)uic important factor is the ratio of the number n of keys iii *K* (which is the number of ecords in *F)* to the number *in of* hash addresses in *L.* This rati',, A = *ps/ni.* is called the *load factor.*

First we show that collisions arc almost impossible to avoid. Specifically, suppose it class has 24 students and suppose the table has space for 365 records, One random hush function is to choose the student's birthday as the hash address. Although the load factor A 24 / 365 *7%* is very small, it can be shown that there is a better than fifty-filly chance that two of the students have the same hi t h lay.

The efficiency of it hash function with *it* resolution iroceduie is measured by the average number of *prO/ic's* (key comparisons) needed to hind the location of the record with a given key *k. T*he efficiency depends mainly oii the load factor *A.* Specifically, we are interested in the following two quantities:
  🕮 S(A) = average number of probes for it search
  🕮 U(A) average number of probes for an unsuccessful search
.

Chapter 8

Data Structures

Files
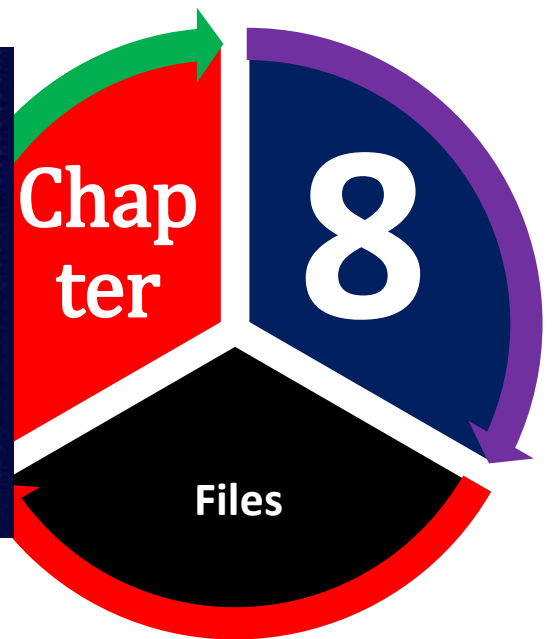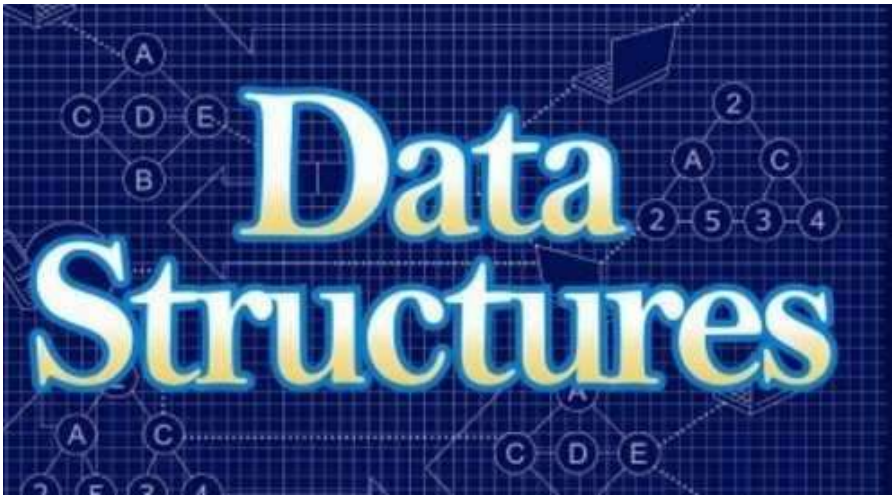
Important Question

About the Authors

Abu Saleh Musa Miah(abid) was born in Rangpur,Bangladesh in 1992. He received the B.Sc. Engineering degree with Honours in Computer science and Engineering in 2014 with FIRST merit Position Engineering first batch from University of Rajshahi ,Bangladesh. He also completed research on the field **COMPUTER VISION** specifically Moving Object Detection with BoofCV tools(java). Currently he is working towards the M.Sc.Engineering degree in the Department of Computer Science And Engineering University of Rajshahi, Bangladesh.

His research interests include **Brain Computer Interfacing**. Sparse signal recovery/compressed sensing, blind source separation, neuroimaging and computational and cognitive neuroscience.

Email: abusalehcse.ru@gmail.com;

https://www.facebook.com/abusalehmusa.miah