

Data Structure and Algorithm

[Syllabus: BPSC CS: Arrays: Representation and operations. Sparse and dense matrices: Concept and operation. Stacks and queues: Concept, structures and basic operations. Quick-sort and Polish notation: Applications of stack. Recursion: Concept and applications. Linked lists: Representation and various operations. Trees: Binary trees, traversing binary trees. Binary search trees: Various operations. Binary heaps: Heap sort. Huffman's algorithm. Graphs: Representations and operations. Spanning trees, shortest path and topological sorting. Internal sorting: Insertion sort, selection sort, merge-sort, radix sort, Basic hashing techniques. Algorithm and complexity: Asymptotic notations. Basic algorithm techniques and analysis: Divide and conquer, dynamic programming, greedy method, branch and bound, string matching, computational geometric problems, graph algorithms, spanning trees, shortest paths, max-flow problem, searching algorithms. Techniques for analysis of algorithms, approximation algorithms, parallel algorithms.

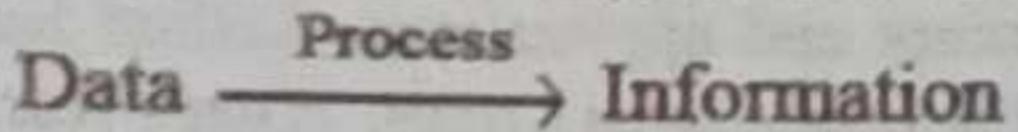
NTRCA CS: Array, Linked lists: Stacks, Queues, Recursion: Trees: Graphs, Analysis of algorithm, design of algorithm, mathematical foundation of algorithm, asymptotic notations, summations, recurrences, sets etc. Divide and Conquer Algorithms: The Greedy method: Dynamic programming: Basic Traversal & Search technique: Backtracking: Branch and Bound.

NTRCA ICT: Introduction - Data Structures and Complexity of Algorithms, Time Space Tradeoff, Searching Techniques- Linear and Binary Searching; Sorting and Recursion - Discussion of Common Sorting Techniques: Insertion Sort, Selection Sort, Bubble Sort, Quick Sort, Merge Sort, Radix Sort; Factorial and Tower of Hanoi Problem; Linked Lists - Abstract Data Types, List ADTs, and Linked Lists: Singly, Two Way and Circular Linked Lists; Stacks and Queues - Stacks and Queues and their Implementation Strategies; Prefix, Infix and Postfix Expressions, their Transformation and Evaluation Algorithms; Hashing - Hash Indices and Hash Functions, Static and Dynamic Hashing, Collisions in Hash Indices and Collision Resolving Techniques; Trees - Tree Concepts, Binary Tree, BST, Heaps, Heap Sort, Huffman Encoding Technique, AVL Tree, B Tree and B+ Tree; Graphs - Graph Terminologies, Representing Graphs, Graph Searching: BFS and DFS, Shortest Path Problems, Minimum Spanning Tree, Minimum Spanning Tree Algorithms, and Topological Sorting; Problem Solving Strategy - Greedy Algorithms, Divide and Conquer Strategy, Dynamic Programming and Backtracking. Introduction - Algorithms, Analyzing & Designing Algorithms, Correctness of Algorithms; Greedy Algorithms - Introduction to Greedy Algorithms, Greedy Choice Property, Greedy vs. Dynamic Programming, Fractional Knapsack Problem, Activity Selection Problem, Huffman Encoding, Task Scheduling Problem, Coin Changing Problem, Kruskal's and Prim's Minimum Spanning Tree Algorithms; Divide and Conquer Algorithms - Introduction to Divide and Conquer Design Technique, Quick Sort, Merge Sort, Proof of Correctness, and Run Time Analysis; Dynamic Programming - Introduction to Dynamic Programming Technique, Principle of Optimality, Optimal Substructure Property, Assembly Line Scheduling, Matrix Chain Multiplication, LCS, Viterbi Algorithm, Bitonic Euclidean Traveling Salesperson Problem and Runtime Analysis; Graph Searching and Shortest Path Problems - Breadth First Search, Depth First Search, Flow Networks, Single Source and All Pair Shortest Path Algorithms]

৩ Data Structure & Algorithm

প্রশ্ন 1. Data & Information কলতে কি বুাৱা? [BPSC-12]
উত্তৰ: Data কলতে কোন নিমিট value বা set of value কে বুাৱা। যা কোন ব্যক্তি বাজ বা কোন বিশ্বের পরিমাণ, বৰ্ণ ইত্যাদি নিৰ্দেশ কৰে। যেমন Male, 34, bitBox ইত্যাদি।

Information: অৰ্থবোধক বা প্ৰক্ৰিয়া কৃত data কে information বলে। যেমন্ত ৫টি গুৰু, 10টি ছাগল ইত্যাদি।



প্রশ্ন 2. Data structure কলতে কি বুাৱা? [National parliament exam-2014,BRTA-2012, BRTA (Senior Computer Operator)-2012, BNPC-2014, BSC-36, বিজ্ঞ মহালয়-2017,সহকৰী প্ৰেমানন্দ]
উত্তৰ: Computer এৰ মাধ্যমে Processing কৰাৰ জন্য Data এৰ Logical এবং Mathematical model হলো data structure.

প্রশ্ন 3. Data structure এৰ প্ৰকাৰভেদ ?[BSC-36, BSC-37, BNPC-2014]

- উত্তৰ: Data structure কে অধানত দুইভাবে ভাগ কৰা যাব।
- Linear data structure:** যে data structure এ data গুলো ক্ৰমানুসৰে বা linear ভাৱে সংগঠিত হয় এবং একটি data item এৰ পৰ আৱেকটি থাকে তাকে Linear data structure বলে।
 - Non linear data structure:** যে data structure এ data গুলো ক্ৰমানুসৰে বা linear ভাৱে সংগঠিত হয় না তাকে Non-Linear data structure বলে।

প্রশ্ন 4. Linear data structure-এৰ কিছু উদাহৰণ লিখ। [National parliament exam-2014]

- উত্তৰ: Linear data structure-এৰ কিছু উদাহৰণ:
- Array
 - Pointer
 - Stack
 - Queue
 - Linked list etc.

প্রশ্ন 5. Non-linear data structure-এৰ কিছু উদাহৰণ লিখ।
উত্তৰ: (i) Tree (ii) Graph etc.

প্রশ্ন 6. Field, Record, File কাকে বলে? [BPSC-14]

উত্তৰ: Field: একজাতীয় ডাটাৰ সমষ্টি কে Field বলে।
Record: দুই বা ততোধিক Field এৰ সমষ্টিকে বলা হয় record।
File: দুই বা ততোধিক record এৰ সমষ্টিকে তৈৰী হয় একটি File থাকে data file বলে।

প্রশ্ন 7. Data structure এৰ operation কলো বৰ্ণনা কৰ ? [INTRCA-14]

উত্তৰ: Data সমূহ Memory তে represent কৰাৰ জন্য Array, Pointer, Linked list ইত্যাদি ব্যবহাৰ কৰা হয়।

Operation:

- Traversing:** কোন record কে একবাৰ মাত্ৰ Access কৰাৰ কৰাকে Traversing বলে।
- Searching:** একটি নিমিট key value এৰ মাধ্যমে record এৰ location খুজে বেৰ কৰাৰ পক্ষতিকে searching বলে।
- Inserting:** পূৰ্বে তৈৰী কৰা কোন data structure এৰ সাথে নতুন record যোগ কৰাৰ কৌশলকে Inserting বলে।
- Deleting:** Data structure হতে কোন record কে বাদ দেওৱাৰ কৌশলকে deleting বলে।
- Sorting:** data সমূহকে কিছু logical order এ সাজানোৰ প্ৰক্ৰিয়াকে sorting বলে।
- Merging:** দুটি ভিন্ন file এৰ record কে একত্ৰিত কৰাৰ কৌশলকে Merging বলে।

প্রশ্ন 8. Why we study data structure?

উত্তৰ: Study of data structure Includes:

- Logical description of data structure.
- Implementation of data structure.
- Quantitative analysis of data structure

প্রশ্ন 9. Use of data structure/Area/Application/Domain.

উত্তৰ: ব্যবহাৰ দেওয়া হলো-

- Compiler design
- Graphics design.
- Data base implement
- (iv) Numerical analysis.
- Artificial Intelligence
- Statistical Analysis package.

প্রশ্ন 10. What is data warehouse ? why we need data structure ? write advantage and disadvantage of data warehouse? [Combined bank (HBFC, KB)-2018]

Data Warehouse: ডেটা ওয়্যারহাউজ হলো ডেটা ও ইনফৰমেশনেৰ মৌলিক বা লজিক্যাল সংযোগ যা বিভিন্ন অপাৰেশনাল ডেটাবেজ থেকে সংগ্ৰহীত হয়। এটি ব্যবসা বিশ্বেৰ, ব্যবসায় বৃক্ষিক্ষা এবং সিকান্ড এহণ প্ৰক্ৰিয়া সমৰ্থন তৈৰিতে ব্যবহৃত হয়।

Advantage of data warehouse:

- একাধিক সোৰ্সৰ ডাটাকে একটি সিস্টেম ডাটাবেজ এবং ডাটা মডেল একীকৃত কৰে, ডাটা উপলব্ধনেৰ জন্য একটি সিস্টেম কুয়েৰ ইঞ্জিন হৈসেই চলে।
- বিপিটিউড ডাটাকে অৰ্গানাইজ কৰে এবং অপোন্টিউড দূৰ কৰে।
- ট্ৰানজেকশন প্ৰসেছিং সিস্টেমেৰ আইসোলেশন জনিত সমস্যা দূৰ কৰে।
- ডাটা ইঞ্সি রাখে, এমনকি যদি ট্ৰানজেকশন সিস্টেম যদি নাও থাকে।
- ডাটা কোয়ালিটি বৃক্ষি কৰে।
- কোন প্ৰতিষ্ঠানেৰ তথ্যকে কনসিস্টেন্সি উপলব্ধন কৰে।
- অধিক কাৰ্যকৰি সিদ্ধান্ত এহণ কৰে।

Disadvantage of data warehouse:

- Underestimation of resources of data loading
- Hidden problems with source systems
- Required data not captured
- Increased end-user demands
- Data homogenization
- High demand for resources
- Data ownership
- High maintenance
- Long-duration projects
- Complexity of integration

প্রশ্ন 11. Need of Data Structure:[Why Data Structure is important?]

- ডেটা স্ট্ৰাকচাৰ গুৰুত্বপূৰ্ণ কৰণ এটি প্ৰায় প্ৰতিটি প্ৰোগ্ৰাম বা সফটওয়্যাৰ সিস্টেমে ব্যবহৃত হয়।
- এটি ভালো কোড লিখতে, কোডেৰ কাঠামো তৈৰি কৰতে এবং সমস্যাগুলি সমাধান কৰতে সহায়তা কৰে।
- আৱাও ভাল ডিজাইন বা ইণ্পুটেশানেৰ মাধ্যমে ডেটাকে আৱাও সহজে নিয়ন্ত্ৰণ কৰা যাব।
- ডেটা স্ট্ৰাকচাৰ হল কেবলমাত্ৰ একটি ধাৰক যা ডাটাকে স্টোৱ, ম্যানিপুলেট এবং আৱেজ কৰতে ব্যবহৃত হয়। এটি অ্যালগৱিন ব্যাব প্ৰসেস কৰা যেতে পাৰে।

প্রশ্ন 12. Characteristics of a Data Structure?

- Correctness** – Data structure implementation should implement its interface correctly.
- Time Complexity** – রানিং টাইম বা ডেটা স্ট্ৰাকচাৰেৰ অপাৰেশন এক্সিকিউশন টাইম যতটা স্বচ্ছ কৰ মত হতে হবে।
- Space Complexity** – ডেটা স্ট্ৰাকচাৰেৰ অপাৰেশন এৰ জন্য Memory usage যতটা স্বচ্ছ কৰ মত হতে হবে।

প্রশ্ন 13. Explain Nested structure and array of structure with examples? [Multiple Ministry (AP)-2016]

Nested Structure: একটি স্ট্ৰাকচাৰেৰ যথন আৱেকটি স্ট্ৰাকচাৰ মেৰাব হিসেবে থাকে তখন তাকে নেস্টেড স্ট্ৰাকচাৰ বলে।

Example:

```

Structure PersonInfo {
    string name, address, city
};

structure Student {
    int studentID;
    PersonInfo pData;
    short year;
    double gpa;
};
  
```

Array of structure: একটি আৱেকটি ডেটা টাইপেৰ পশ্চাপাশ স্ট্ৰাকচাৰ নিজেও থাকতে পাৰে, একে আৱে স্ট্ৰাকচাৰ বলা হয়।

Example:

```

int main(){
    int n;
    const int size=2;
    struct part{ // specify of structure
        int modelnumber;
        int partnumber;
        float cost;
    };
    part part[size]; //define array of structure
  
```

প্রশ্ন 14. What is algorithm? [ICT Ministry (AP & NE)-2017] Write the criteria of algorithm?

Algorithm: এলগৱিন হচ্ছে কোনো একটি কাজ সম্পূৰ্ণ কৰাৰ জন্য কতৰিং সুনিষিট ও ধাৰাবাহিক ধাপেৰ সমষ্টি। অ্যালগৱিনমেৰ একটি কৰ ও শেষ আছে এবং এৰ ধাপ সংখ্যা অবশ্যই সীমিত হতে হবে।

Criteria of algorithm:

একটি অ্যালগৱিনমেৰ(Algorithm) বৈশিষ্ট্যঃ

- Definiteness:** অ্যালগৱিনম স্পষ্ট এবং ব্যাবহীন হওয়া উচিত। এৰ প্ৰতিটি পদক্ষেপ এবং তাৰে ইনপুট / আউটপুটগুলি স্পষ্ট হওয়া উচিত এবং এৰ একটি সঠিক/নিষিট অৰ্থ থাকা লাগবে।
- কাৰ্যকৰিতাট:** একটি অ্যালগৱিনম তখনই কাৰ্যকৰিতা হিসেবে ধাৰা হবে যখন Algorithm এৰ সকল নিয়ম-নীতি এবং সীমাবদ্ধতা মেনে কৰা হবে।
- ইনপুট:** একটি অ্যালগৱিনমেৰ শূন্য বা আৱাও ভাল-সংজৰিত ইনপুট থাকতে হবে।
- আউটপুট:** একটি অ্যালগৱিনমেৰ এক বা একাধিক আউটপুট থাকে, যা ইনপুটগুলিৰ সাথে একটি নিষিট সম্পৰ্ক রাখে।
- ৰাখতা:** একটি অ্যালগৱিনমেৰ ধাপে ধাপে নিকনিদেশ অনুযায়ী থাকতে হবে, যা কোনও প্ৰোগ্ৰাম কোড থেকে ব্যতী হওয়া প্ৰয়োজন।

প্রশ্ন 15. What is Big-sigma, Big-omega and Big-theta ? [BCS-36, BCS-35, কৰিগিৰি শিক্ষা অধিদল- ২০১৯,-পদেৰ নাম: কৃষি ইণ্ডাস্ট্ৰি কলেজটি, ইলেক্ট্ৰনিক্স ও পাওড়াৰ]

Big- O হচ্ছে একটি অ্যালগৱিন রান কৰতে সৰ্বোচ্চ কত সময় লাগবে (worst case) তাৰ পৱিমাপ, অৰ্থাৎ অ্যালগৱিনমেৰ আপাৰ বাটৰ্ড (upper bound) এই নোটেশনেৰ মাধ্যমে প্ৰকাশ কৰা হয়। $f(n) = O(g(n))$ (read as "f of n is big oh of g of n") iff (if and only if) there exist positive constants c and n_0 such that $f(n) \leq cg(n)$ for all $n \geq n_0$, example, $3n+2 = O(n)$ as $3n+2 \leq 4n$ for all $n \geq 2$

$$10n^2 + 4n + 2 = O(n^2)$$

$$10n^2 + 4n + 2 \leq 11n^2 \text{ for all } n \geq 5$$

$g(n)$ is upperbound on the value of $f(n)$ for all $n, n \geq n_0$

Ex-vi. A [-5:20] is a linear array with base address 2000 & word size 3. Find the Location of A[-2], A[0], A[15] & Number of element.

Solⁿ:

$$\text{Loc } A[-2] = 2000 + 3(-2+5) = 2009 \text{ (Ans.)}$$

$$\text{Loc } A[0] = 2000 + 3(0+5) = 2015 \text{ (Ans.)}$$

$$\text{Loc } A[15] = 2000 + 3(15+5) = 2060 \text{ (Ans.)}$$

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

$$= (20 - (-5) + 1) = 26 \text{ (Ans.)}$$

প্রশ্ন 7. Two dimension array এর যে কোন location এর address দেব করার সূত্র লিখ।

সূজ্ঞ let,

$$\begin{matrix} A & [M \times N] \\ & R \quad C \end{matrix}$$

Column major:

$$\text{Loc } (A[R, C]) = \text{Base } [LA] + w\{(R-1) + M(C-1)\}$$

Row major:

$$\text{Loc } (A[R, C]) = \text{Base } [LA] + w\{N(R-1) + (C-1)\}$$

Example

Ex-i. 25 জন ছাত্রের 4 টি class test এর Result সংরক্ষণের জন্য একটি $[25 \times 4]$ two dimension এর score নামে একটি array তে সংরক্ষণ করা হচ্ছে। Base (score) = 200, w = 4 row ভিত্তিক data সংরক্ষণ উপযোগী array টিকে score [12, 3] এর অবস্থান নির্ণয় কর।

Solⁿ:**Row Major:** We know,

$$\begin{aligned} \text{Loc score}[12, 3] &= \text{Base } [LA] + w\{(C-1) + N(R-1)\} \\ &= 200 + 4\{(3-1) + 4(12-1)\} \\ &= 384 \text{ (Ans.)} \end{aligned}$$

Column Major: Self Task.

Ans: 444

Ex-ii. A two dimensional array A[20, 10] is the represented in memory if BA = 5000, W = 2 find the address of A[3, 4], A[10, 5] A[5, 7]

(a) Row major

(b) Column major

Solⁿ:**(a) Row major:**

$$\begin{aligned} \text{Loc } A[3, 4] &= \text{Base } [A] + w\{(C-1) + N(R-1)\} \\ &= 5000 + 2\{(4-1) + 10(3-1)\} \\ &= 5046 \text{ (Ans.)} \end{aligned}$$

$$\begin{aligned} \text{Loc } A[10, 5] &= 5000 + 2\{(5-1) + 10(10-1)\} \\ &= 5188 \text{ (Ans.)} \end{aligned}$$

$$\begin{aligned} \text{Loc } A[5, 7] &= 5000 + 2\{(7-1) + 10(5-1)\} \\ &= 5092 \text{ (Ans.)} \end{aligned}$$

(b) Column major:

$$\begin{aligned} \text{Loc } A[3, 4] &= \text{Base } [A] + w\{(R-1) + M(C-1)\} \\ &= 5000 + 2\{(3-1) + 20(4-1)\} \end{aligned}$$

$$= 5124 \text{ (Ans.)}$$

$$\begin{aligned} \text{Loc } A[10, 5] &= 5000 + 2\{(10-1) + 20(5-1)\} \\ &= 5178 \text{ (Ans.)} \end{aligned}$$

$$\begin{aligned} \text{Loc } A[5, 7] &= 5000 + 2\{(5-1) + 20(7-1)\} \\ &= 5248 \text{ (Ans.)} \end{aligned}$$

Ex-iii. Consider the linear array $A^{M \times N}$ with $w = 4$ if the address of $A[5, 3] = 5050$. Then find the address of $A[3, 5]$ when the array is arranged in (i) row major order, (ii) column major order.

Solⁿ:**Row major:**

$$\begin{aligned} \text{Loc } [5, 3] &= \text{Base} + w\{N(R-1) + (C-1)\} \\ &\Rightarrow 5050 = \text{Base} + 4\{5(5-1) + (3-1)\} \end{aligned}$$

$$\Rightarrow \text{Base} = 4962 \text{ (Ans.)}$$

$$\begin{aligned} \text{Loc } [3, 5] &= \text{Base} + w\{N(R-1) + (C-1)\} \\ &= 4962 + 4\{3(3-1) + (5-1)\} \\ &= 5018 \text{ Ans:} \end{aligned}$$

Column major:

$$\begin{aligned} \text{Loc } [5, 3] &= \text{Base} + w\{(R-1) + M(C-1)\} \\ 5050 &= \text{Base} + 4\{(5-1) + 8(3-1)\} \end{aligned}$$

$$\text{Base} = 4970 \text{ Ans:}$$

$$\begin{aligned} \text{Loc } [3, 5] &= 4970 + 4\{(3-1) + 8(5-1)\} \\ &= 5106 \text{ Ans:} \end{aligned}$$

Ex-iv. NUB (2:5, - 3:1) how many elements contains this array?

Solⁿ: We know,

$$\begin{aligned} \text{Element} &= (UB_1 - LB_1) \times (LB_2 - UB_2) \\ &= (5-1) \times (2+3) \\ &= 20 \text{ elements Ans:} \end{aligned}$$

প্রশ্ন 9. হয় উপাদান বিশিষ্ট একটি linear array "DATA" তে কিভাবে 246, 59, 435, 142, 85, 157 এই ছয়টি পূর্ণ সংখ্যা (integer) সংরক্ষিত থাকবে, চিহ্নের সাথেয়ে দেখাও।

উভয়	DATA [0]=246	0	246
	DATA [1]=59	1	59
	DATA [2]=435	2	435
	DATA [3]=142	3	142
	DATA [4]=85	4	85
	DATA [5]=157	5	157

প্রশ্ন 10. [52, 50, 27, 66, 82] আরে লিস্ট ব্যবহার করে নিম্নলিখিত অপারেশন সম্পূর্ণ করুন। [বিডিই মডেলার্স-২০১৯
সহকারী প্রয়োজনীয় (নন-ক্লাসার)]

i) Insert data into array list

ii) Remove 27 from array list

iii) Insert 99 into 2nd position

iv) Show array list

উভয়: i) Insert data into array list

Array List:

Value	52	50	27	66	82
Index	0	1	2	3	4

ii) Remove 27 from array list

Value	52	50	27	66	82
Index	0	1	2	3	4

iii) Insert 99 into 2nd position(index 1)

Value	52	99	50	66	82
Index	0	1	2	3	4

iv) Show array list

Value	52	99	50	66	82
Index	0	1	2	3	4

প্রশ্ন 11. Linear array এর কিছু algorithm এর নাম লিখ।

উভয়ের লিনিয়ার আরের কিছু Algorithm নিম্নে দেয়া হল-

- (i) Traversing
- (ii) Inserting
- (iii) Deleting

প্রশ্ন 12. Traversing (অমন করা) এর Algorithm লিখ।

উভয়: Traversing এর Algorithm দ্বাৰা প্রত্যেক নোডের সকল

Address অমন কৰা যাব।

Traversing algorithm:

1. Set k: = LB
2. Loop তক K \leq UB পৰ্যন্ত visit
3. K = K+1
4. শূন্পের সমাপ্তি (যখন K \leq UB মিথ্যা হয়ে যাবে)
5. Exit

LB	0	
	1	
	2	
	3	
	4	
	5	
	UB	6

প্রশ্ন 13. Inserting এর Algorithm লিখ। |প্রাচী কল্যাণ ও
বেদেশিক কর্মসংঘান জনশক্তি কর্মসংঘ ব্যৱো-২০১৮ ইলেক্ট্রোকোর্ট (কম্পিউটার)

উভয়:	9	9	
Inserting Algorithm:	8	8	
মনে কৰি আমরা K তম	7	N=7	G
location এ নতুন ITEM	6	G	F
Insert কৰবো, কিন্তু K তম	5	F	E
location এ আগের ডাটা	4	E	D
থাকার কারণে উপরের ডাটা	3	D	ITEM
গুলোকে shift কৰতে হবে,	2	C	C
1. Set J = N	1	B	B
2. Loop J \geq K হয়	0	A	A

প্র

```

If: A[i]<FirstL
    SecondL=FirstL
    FirstL=a[i];
Else if: a[i]>FirstL & a[i]<SecondL
    SecondL=a[i];
Step 3: Print (SecondL)

```

প্রশ্ন 17. লিনিয়ার আয়েরে মেমোরিতে উপস্থাপন কর (One Dimensional)।

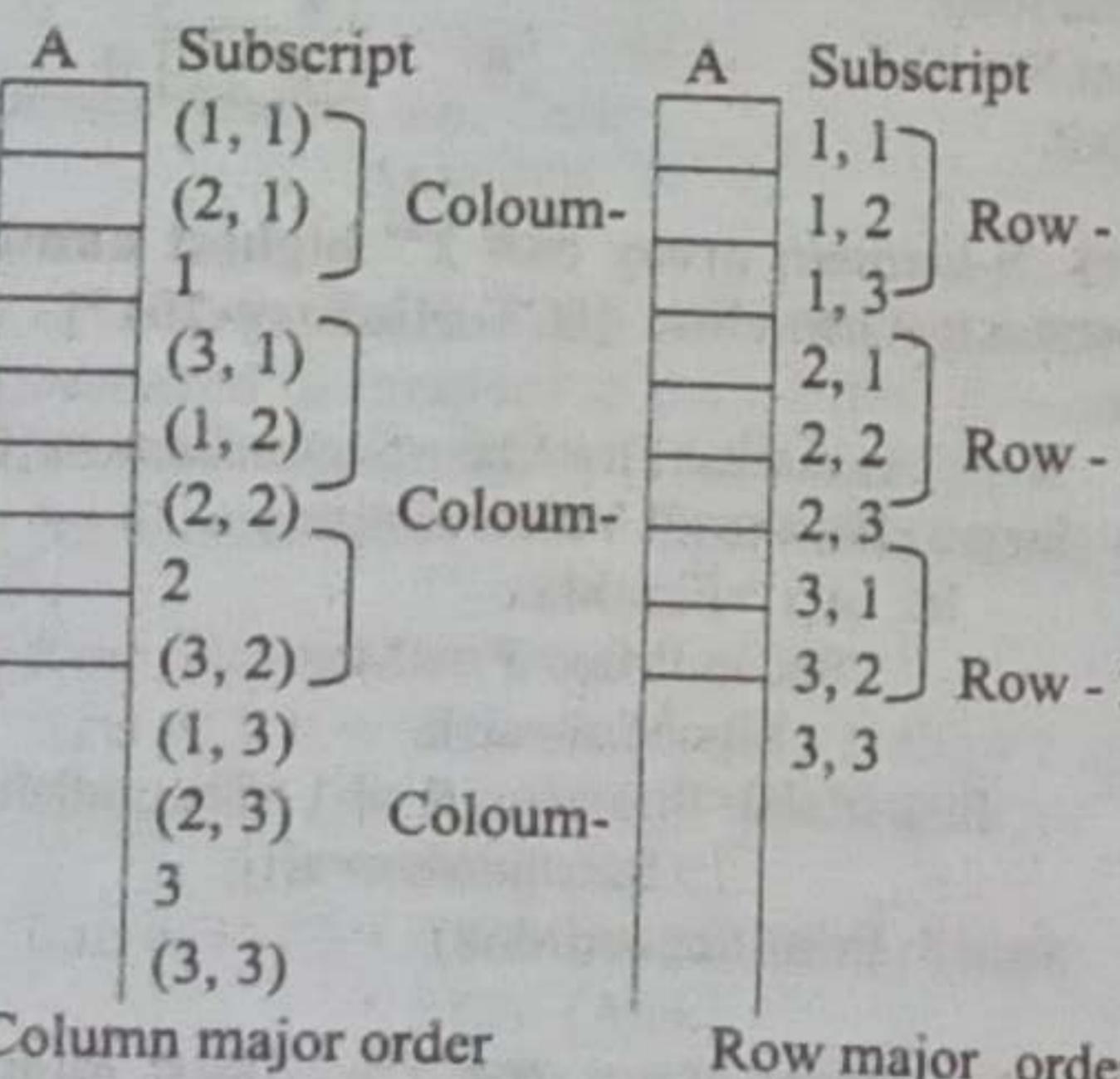
উত্তর: ধরি LA হচ্ছে একটি Linear array। মেমোরিতে সংরক্ষিত আয়ের উপাদান সমূহকে খুব সহজে মেমোরিতে পর্যাপ্ত করা যাব।

একটি আয়ের শুরু ছানের Address (Base address) জানা থাকলে অধিক Address দ্বারা সহজেই উপাদান সমূহকে বের করা যাব।

	element
1000	BA
1001	
1002	
1003	
1004	
.	
n	

প্রশ্ন 18. মেমোরিতে বিমানিক আয়ের উপস্থাপন কীভিত লিখ।
উত্তর: কম্পিউটার মেমোরিতে দুই ভাবে ডাটা উপাদান সংরক্ষণ করা যাব।

- Column major order
- Row major orders



প্রশ্ন 19. Bit box ক্ষেত্রে memory তে কত জায়গা দখল করবে।

উত্তর: 7 byte, প্রতিটি character এর জন্য 1 byte এবং প্রতিটা space এর জন্য 1 byte জায়গা দখল করবে।

প্রশ্ন 20. Memory হিসাব -
উত্তর: 0/1 → এই দুটি হলো bit
4 bit → 1 Nibble
8 bit → 1 byte
16 bit → 2 byte → 1 word
1024 byte → 1 kilo byte
1024 KB → 1 mega byte
1024 MB → 1 Giga byte
1024 TB → 1 Petabyte
1024 PB → 1 Exabyte

প্রশ্ন 21. একটি 12 element লিনিয়ার আয়ের DATA তে যথাক্রমে Integer, float এবং character type ডাটা সংরক্ষণ করতে হবে। প্রতিক ক্ষেত্রে কত byte মেমোরি ব্যবহৃত হবে।

উত্তর: Integer type এর জন্য = $12 \times 2 = 24$ byte
float type এর জন্য = $12 \times 4 = 48$ byte
character type এর জন্য = $12 \times 1 = 12$ byte

প্রশ্ন 22. কোন একটি Array তে 1000000 ডলো উপাদান আছে। লিনিয়ার সার্চ এবং বাইনারি সার্চ Algorithm Complexity নির্ণয় কর এবং উভয়ের মধ্যে তুলনা কর।
Solⁿ: বাইনারী সার্চের ক্ষেত্রে

$$\begin{aligned} n &= 1000000 \\ f(n) &= \text{complexity} \\ f(n) &= \log_2 n = \log_2 1000000 = 20 \end{aligned}$$

লিনিয়ার সার্চের ক্ষেত্রে

$$\text{worst case: } f(n) = n+1 = 1000000+1 = 1000001$$

$$\begin{aligned} \text{Average case: } & f(n) = \frac{n+1}{2} = \frac{1000000+1}{2} = 500000 \end{aligned}$$

সূতরাং দেখা যাচ্ছে লিনিয়ার সার্চ অ্যালগরিদমে দশ লক্ষ উপাদানের মধ্যে নির্দিষ্ট উপাদানকে খুজে কমপক্ষে পাঁচ লক্ষ বার উপাদানের তুলনাক্রমে অন্যদিকে বাইনারী সার্চে 20 বার তুলনার প্রয়োজন হয়।
বাইনারী সার্চের Complexity লিনিয়ার সার্চের Complexity এর তুলনায় অনেক কম হবে।

Self study

প্রশ্ন 1. একটি শিক্ষা প্রতিষ্ঠান 1979 থেকে 2005 সন পর্যন্ত ছাত্র ছাত্রীর পাশের ধারে লিনিয়ার আয়ের মাধ্যমে সংরক্ষণ করে। উক্ত আয়ের element সংখ্যা কত? Ans: 27

প্রশ্ন 2. Bangladesh, Student, DUET, DATA - Structure এ এই শব্দ গুলোর জন্য কত byte জায়গা দখল করবে। Ans: 10 byte, 7 byte, 4 byte.

প্রশ্ন 3. ধরা যাক XXX (-10:10), YYY (1935:1985), ZZZ (35) তিনটি Array প্রতিক আয়ের উপাদান সংখ্যা বের কর।

(i) ধর Base (YYY) = 400 এবং W = 4 হলে YYY[1942], YYY [1977] এবং YYY [1999] এর Address নির্ণয় কর। Ans: 21, 51, 35

প্রশ্ন 4. একটি Linear Array বিবেচনা কর AAA(5:50), BBB(-5:10) এবং CCC(18)। ধর Base (AAA)=30 এবং AAA এর জন্য Memory Cell Words সংখ্যা 4।

- প্রতিক Array Element সংখ্যা বের কর।
- AAA[15], AAA[35] এবং AAA[55] এর Address বের কর।

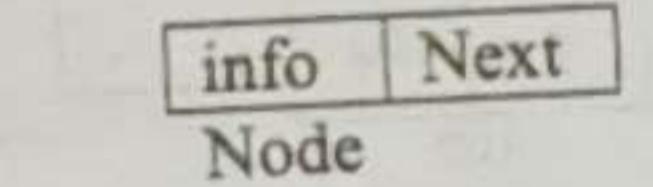
Ans: (a) 46, 16, 18
(b) 70, 150, উক্ত AAA(5:50) Array এর upper bound 50 তাই AAA[55] Loc বিদ্যমান নাই।

Pointer

[প্রোগ্রাম ইন সি পার্ট এ আলোচনা করা আছে]

LINKED LIST

প্রশ্ন 1. Linked list: Linked list হলো একটি linear data structure যা কতগুলো node এর সমন্বয়ে গঠিত। প্রতিটি node এ দুটি field থাকে। একটি INFO field এবং অন্যটি Next pointer field.



প্রশ্ন 2. Linked list এ কোন over flow হয় না?

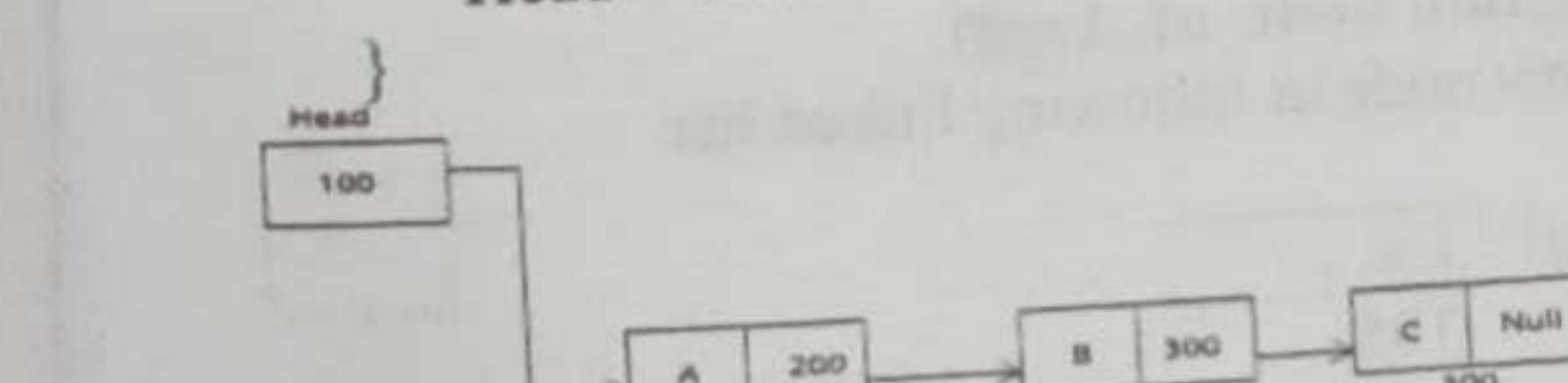
উত্তর: Linked list এ dynamic memory allocation করার কারণে over flow হয় না।

State traversing:

```

if (Head == Null)
    return;
else
    while (Head != Null)
    {
        Head = Head next
    }

```



প্রশ্ন 3. Linked list কত প্রকার ও কি কি? | বিভিন্ন মানদণ্ড- ২০১৭, সহকারী প্রোগ্রাম।

উত্তর: Linked list এর প্রকারভেদ নিম্নরূপ-

- Linear linked list.
- Doubly linked list.
- Circular linked list.
- Header linked list.

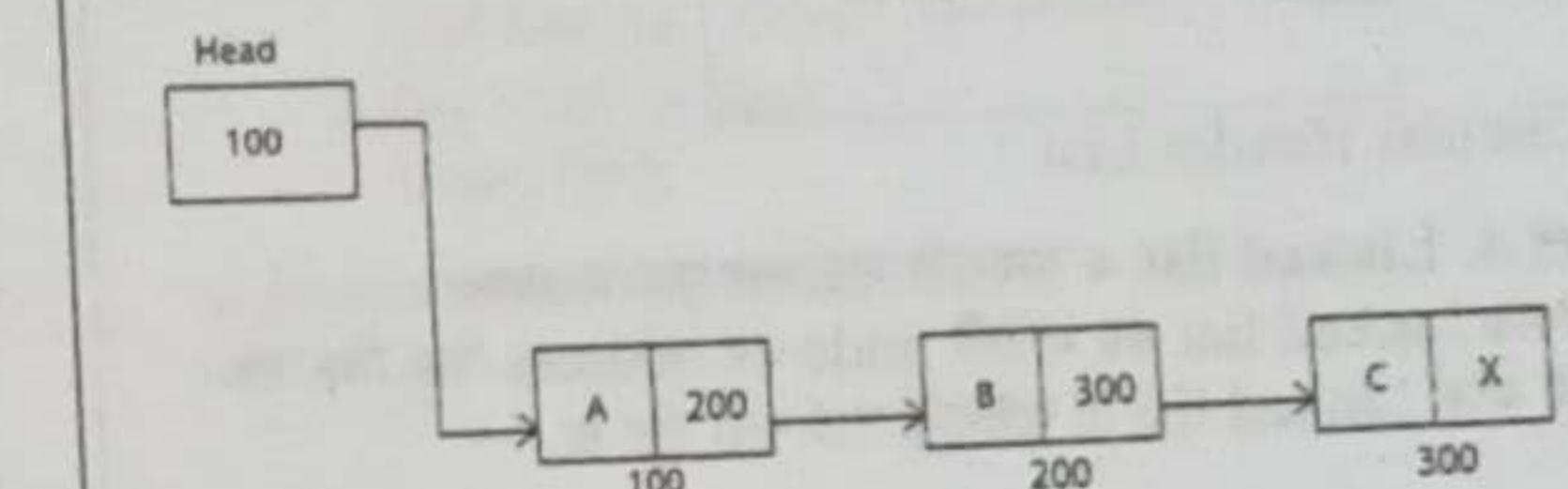
Linear linked list:

Linear linked list হলো one way linked list. যার দুটি অংশ থাকে।

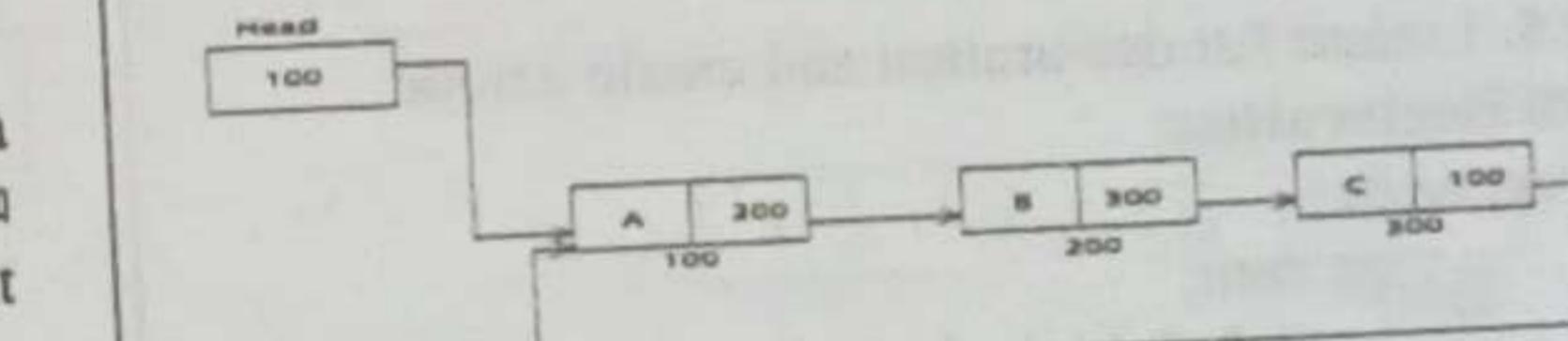
Info/data Field →

--	--

 ← Next Field



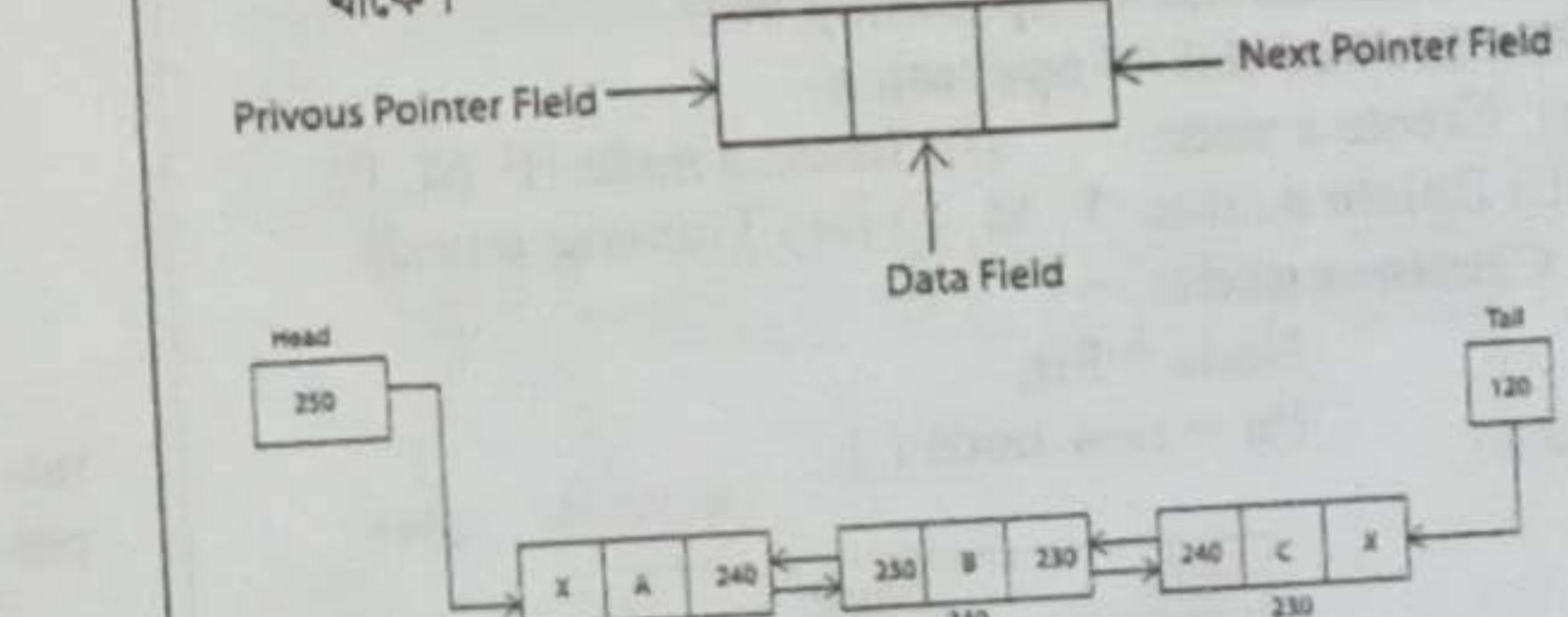
(ii) Circular linked list: Circular linked list মূলত একটি linear linked list। কিন্তু এর last node প্রথম node কে point করে। ফলে কোন Null থাকবে না।



চিত্র: 3 node বিশিষ্ট একটি Circular linked list

(iii) Doubly linked list কি বর্ণনা কর। | বাস্তী কলাম ও বৈদেশিক কলামের জন্য কর্মসূচি কর্মসূচি করা হবে।

Doubly link list হলো two way link list। যার তিনটি অংশ থাকে।



(iv) Header linked list: Header linked list এমন একটি linked list যাতে একটি বিশেষ নোড থাকে। একে Header node বলা হয় এবং ইহা list এর প্রথমে হয়।

(a) Grounded Header: Grounded header list হচ্ছে এমন একটি Header list, যেখানে সর্বশেষ node টি NULL Pointer নির্দেশ কর।

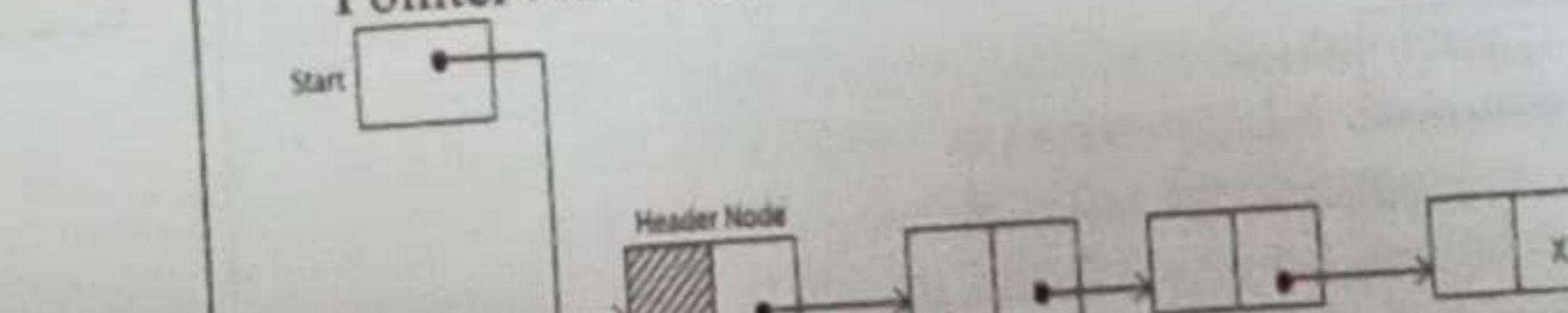
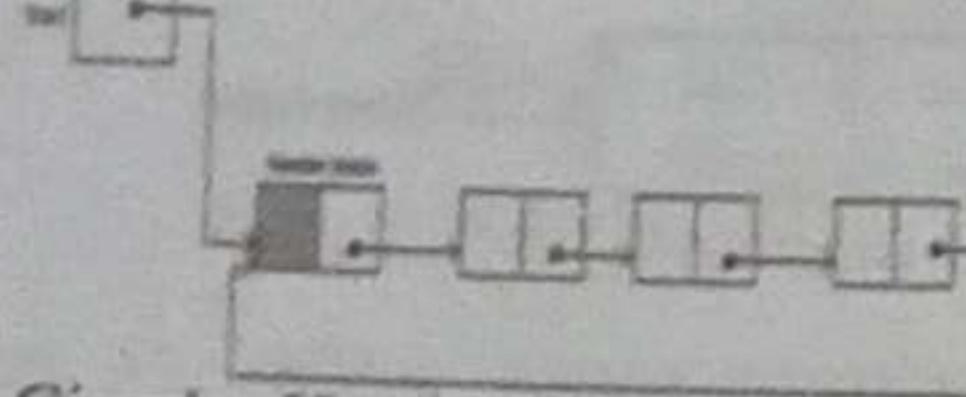


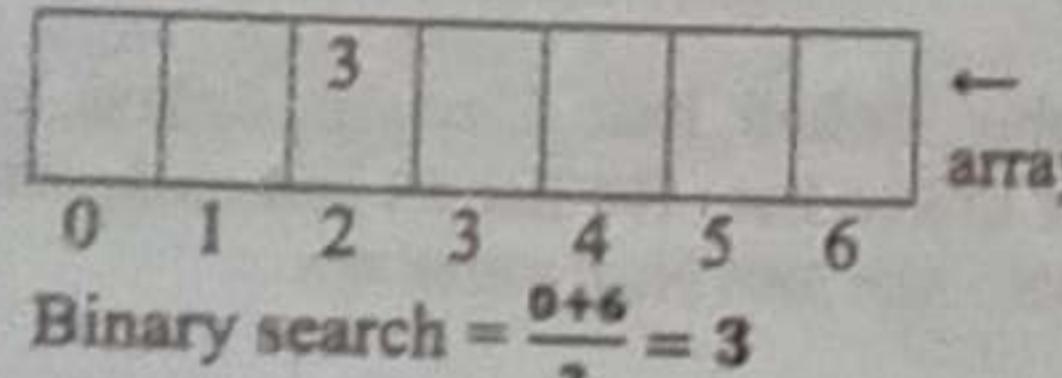
Fig: Graouded Header List

(b) Circular header: Circular header list হচ্ছে এমন একটি header list, যেখানেশের node টি header node এ সিদ্ধে আসে বা মুক্ত থাকে।



Circular Header List

প্র. 4. Linked list এ বাইনারি সার্চ করা যায় না কেন?
উত্তর: Linked list এর প্রতিটি node এর address ভিত্তি হয়।
এইজন্য Linked list এ বাইনারি সার্চ করা যায় না।



প্র. 5. Linked list declaration and create a node
উত্তর: Declaration:

```
Struct node{
    int item;
    node * Link;
}
```

Node Create:

```
Node * Ptr;
Ptr = new node();
```

প্র. 6. Linked list এর operation লিখ।
উত্তর: Linked list এর operation-

- (i) Create a node
- (ii) Insert a node (F, M, L)
- (iii) Delete a node (F, M, L)
- (iv) Traverse a node

i. Create a node:

```
Node * Ptr;
Ptr = new node();
```

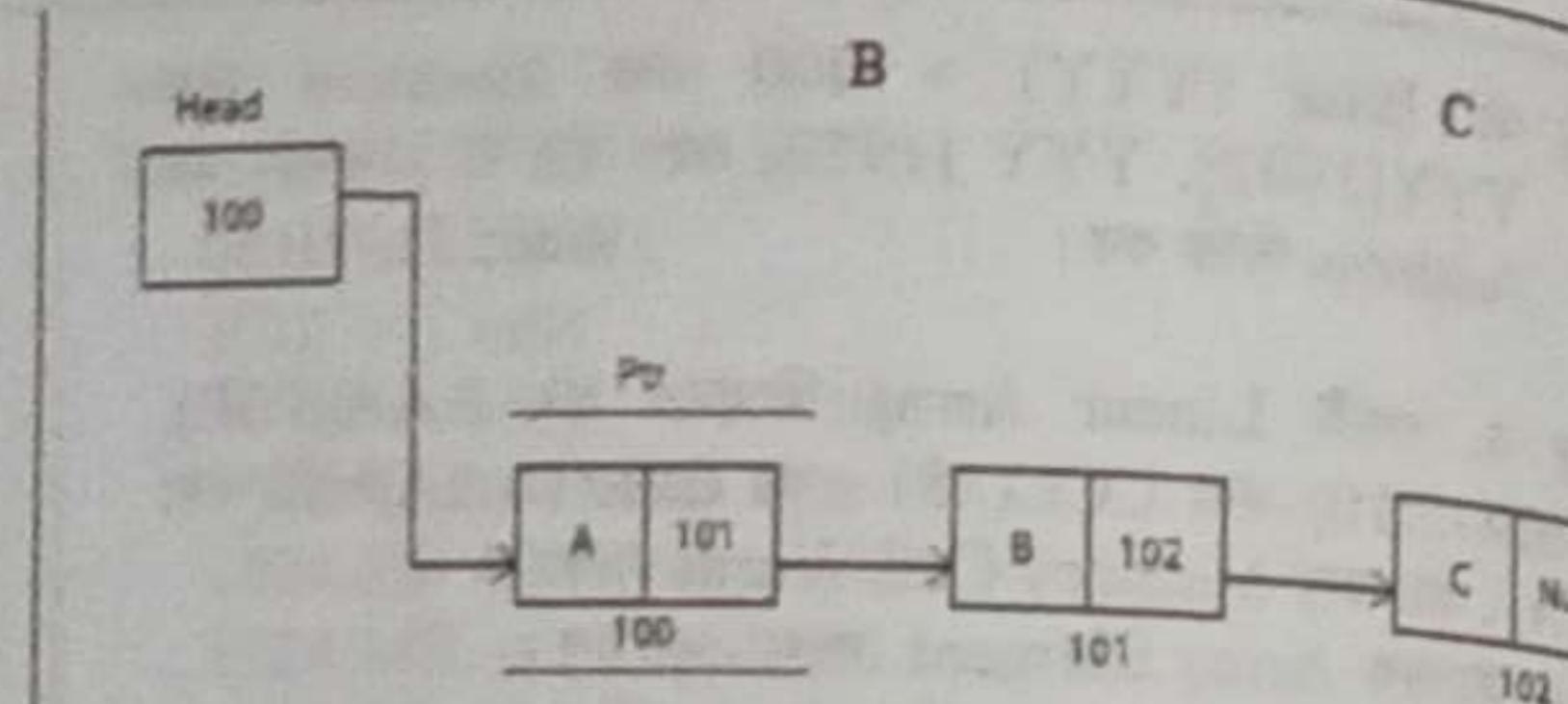
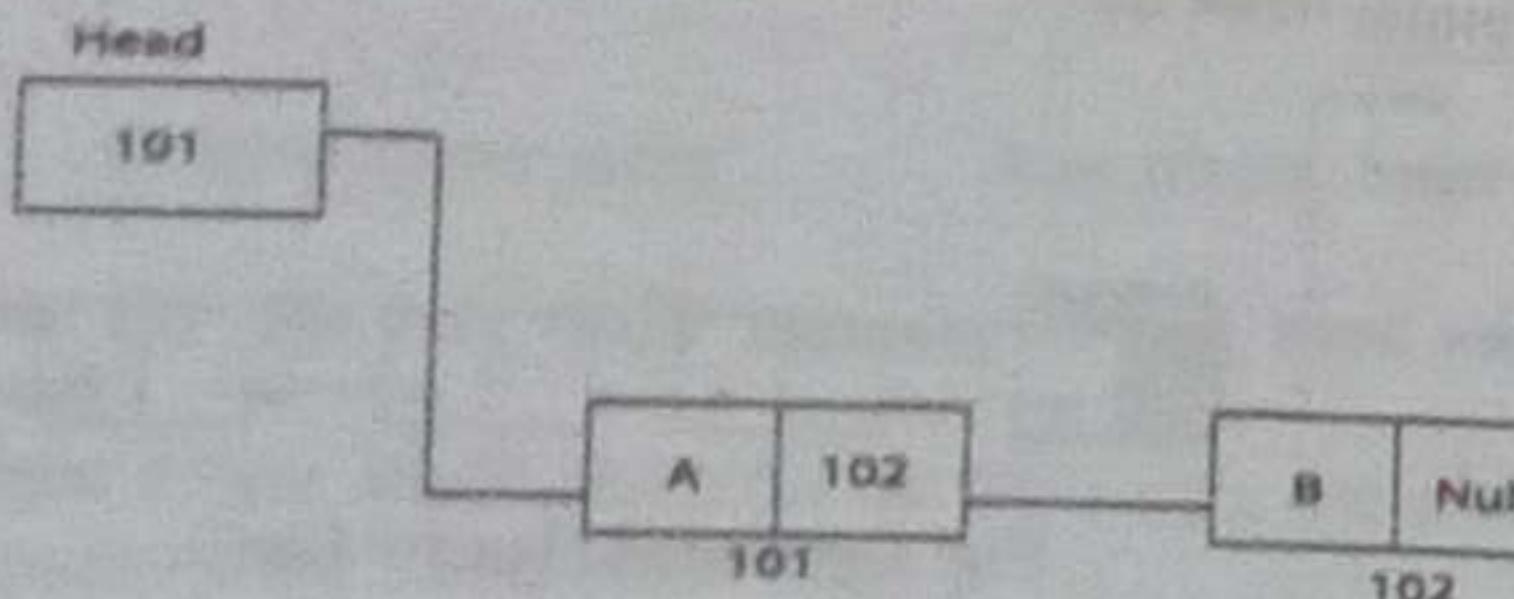


ii. Insert a node: (First, Middle, Last):

a. Insert a node at First:

Insert this node in following linked list

```
Insert A [ ]  
100
```



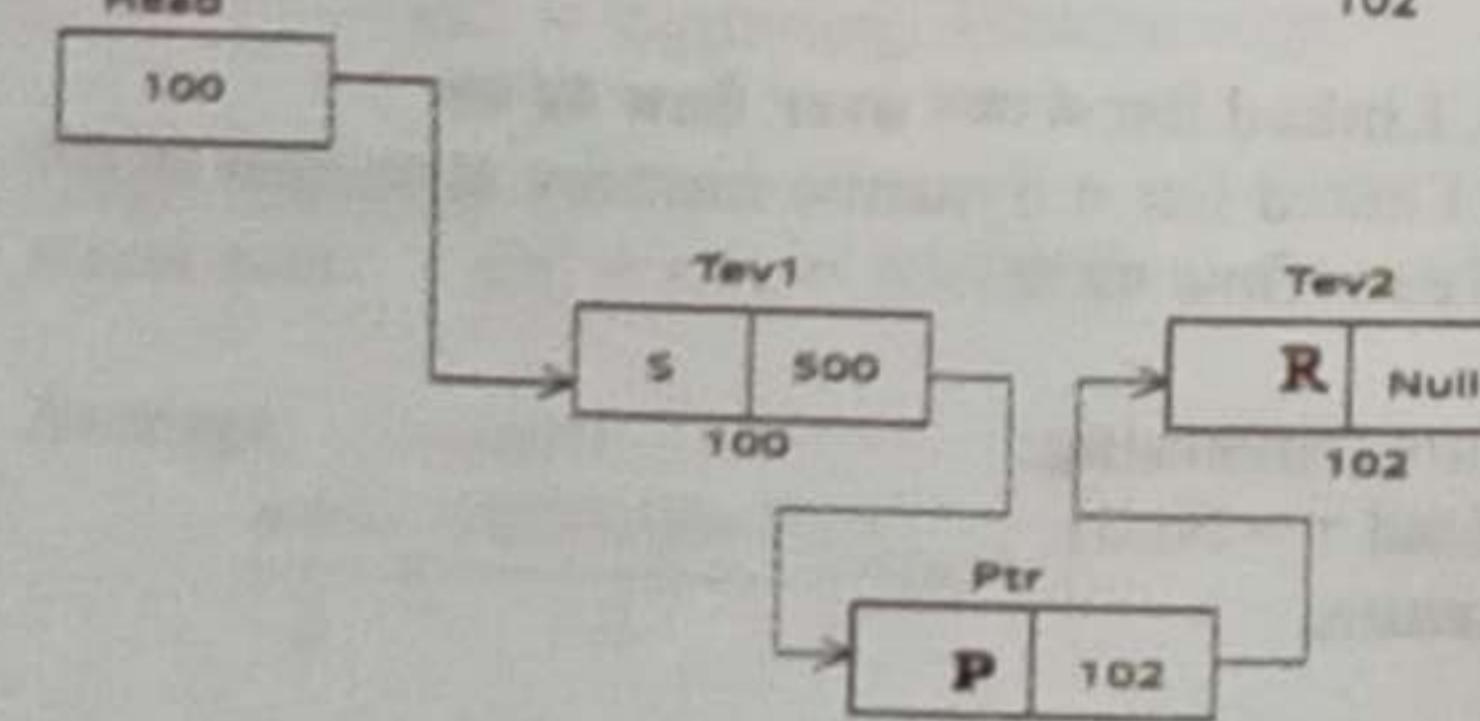
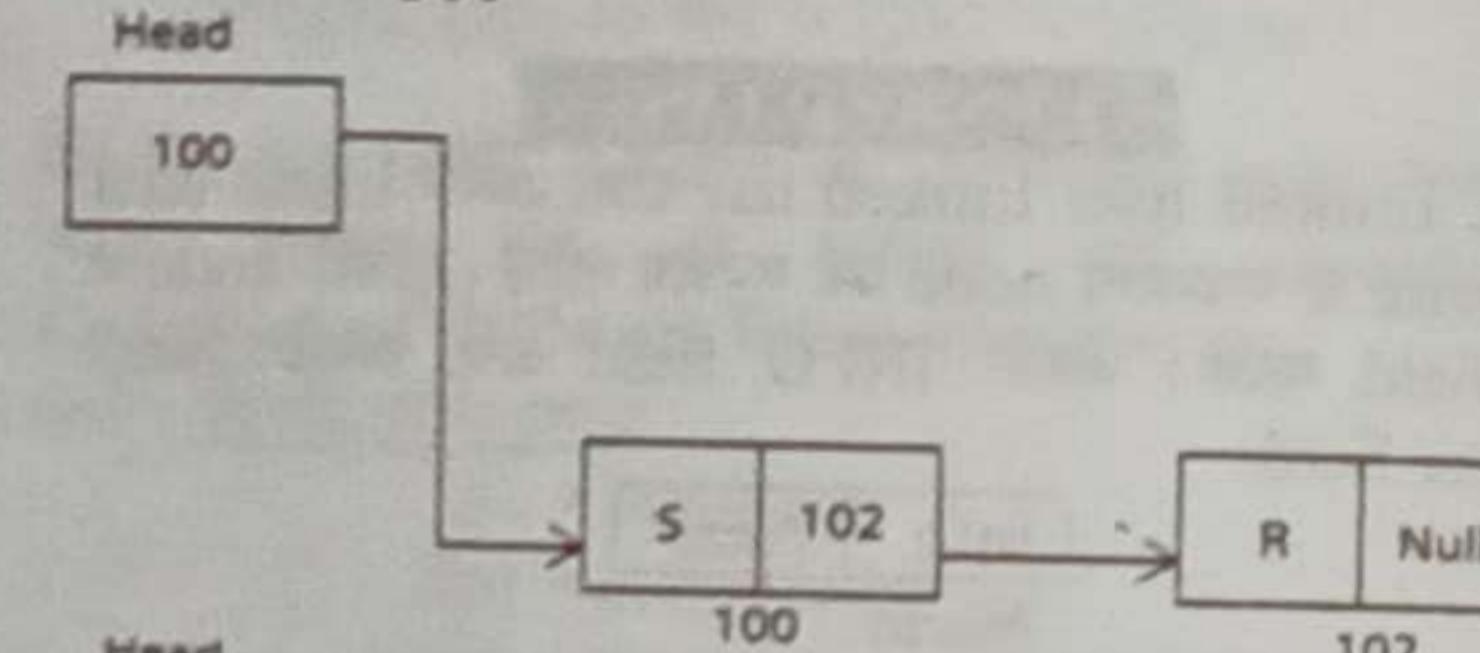
ptr → next = 101

Head = ptr

b. (Insert a node at Middle):

Insert this node in following linked list

```
Insert A [ ]  
500
```



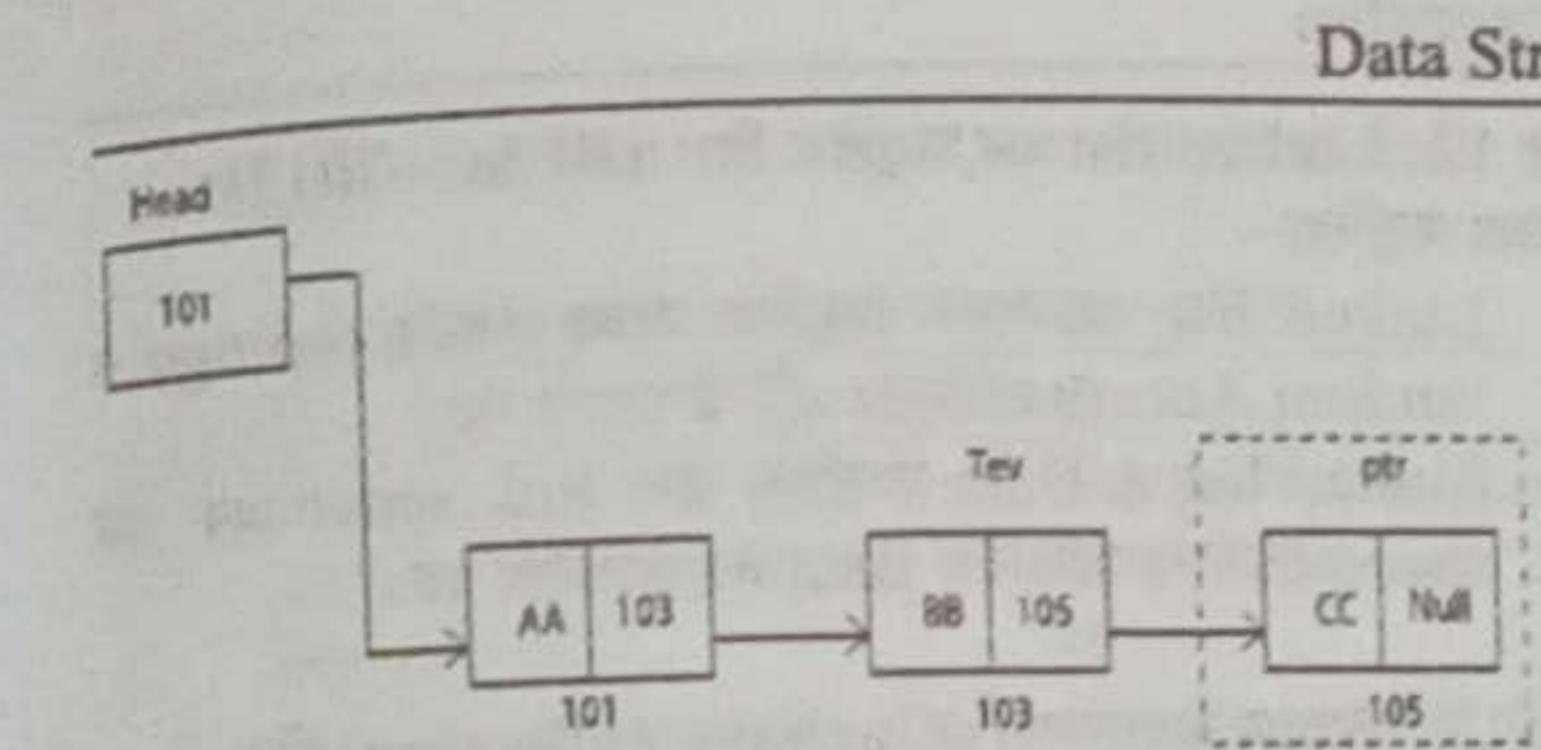
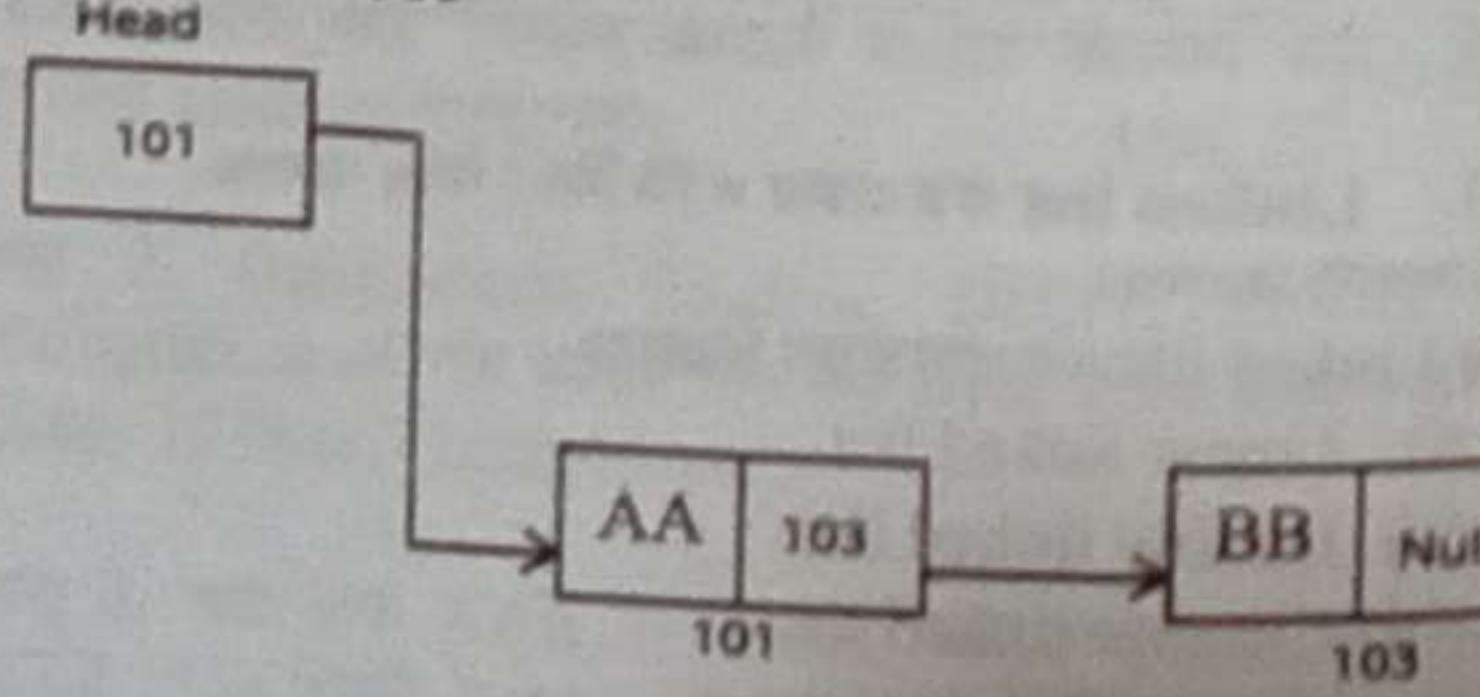
tev → next = Ptr.

ptr → next = tev 2

c. (Insertion node at Last)

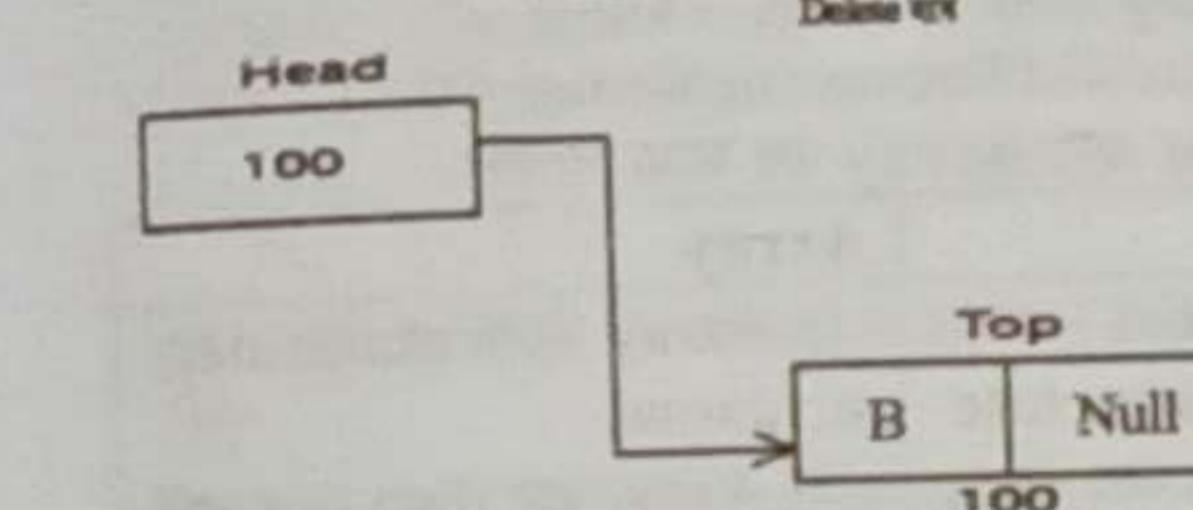
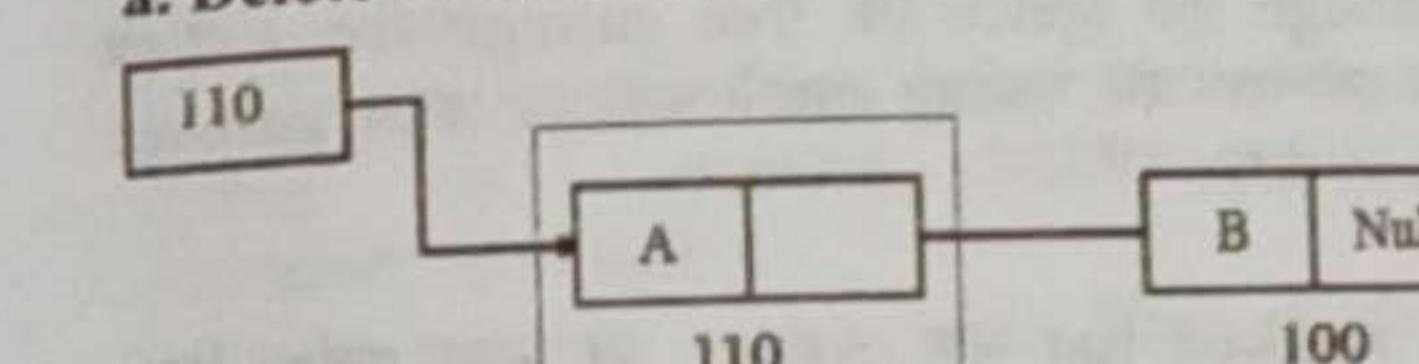
Insert this node in following linked list

```
Insert CC [ ]  
105
```



iii. Delete a node: (First, Middle, Last)

a. Delete a node at First:



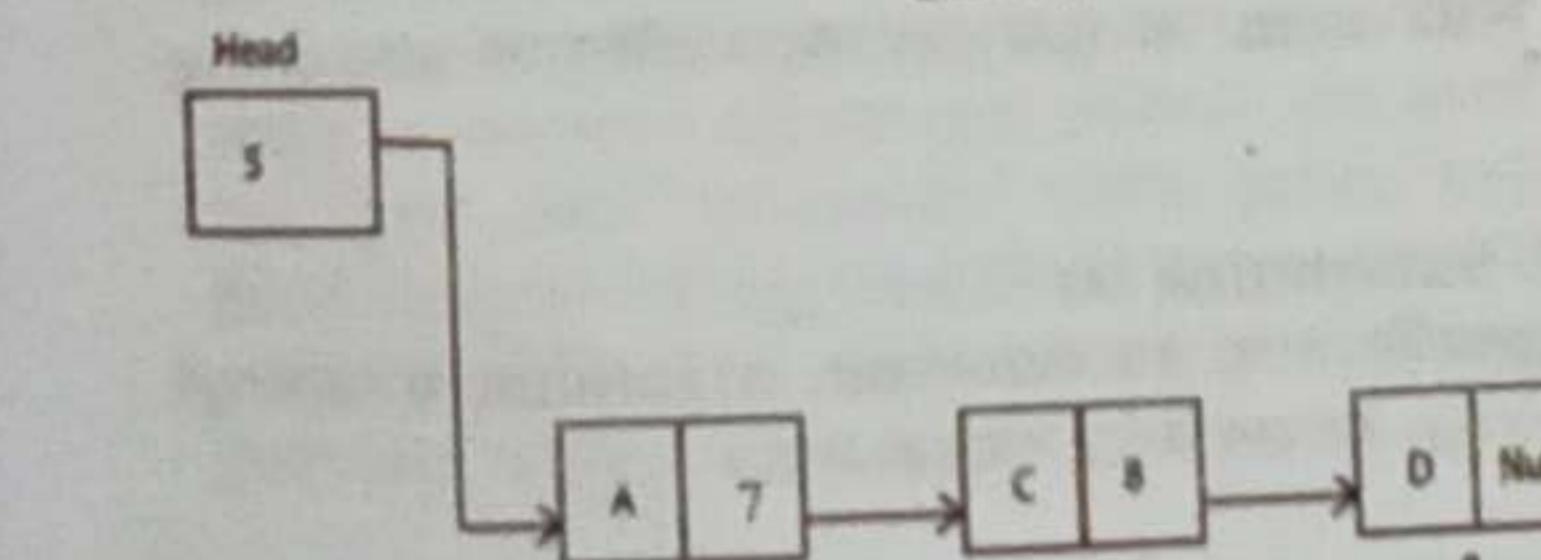
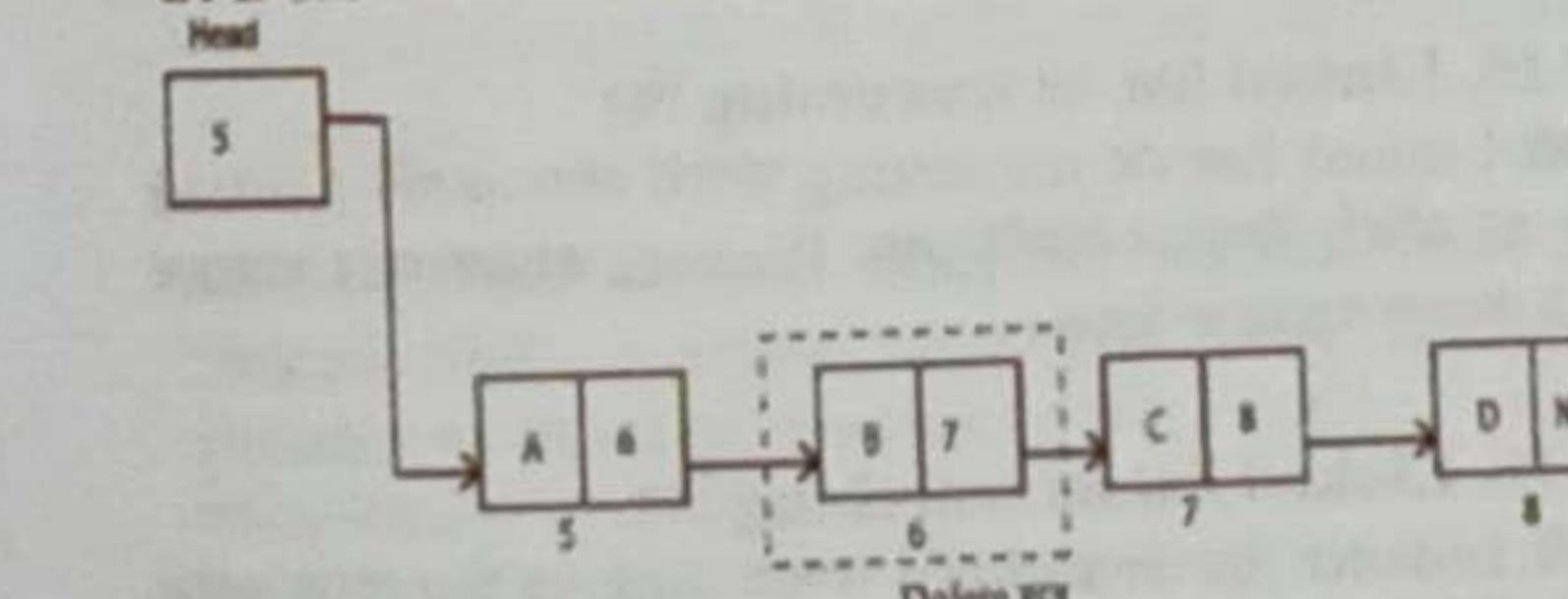
if (Head == null)

print stack null

else

```
Top = Head
Head = Head → next
Free (Top);
```

b. Delete a node at Middle:

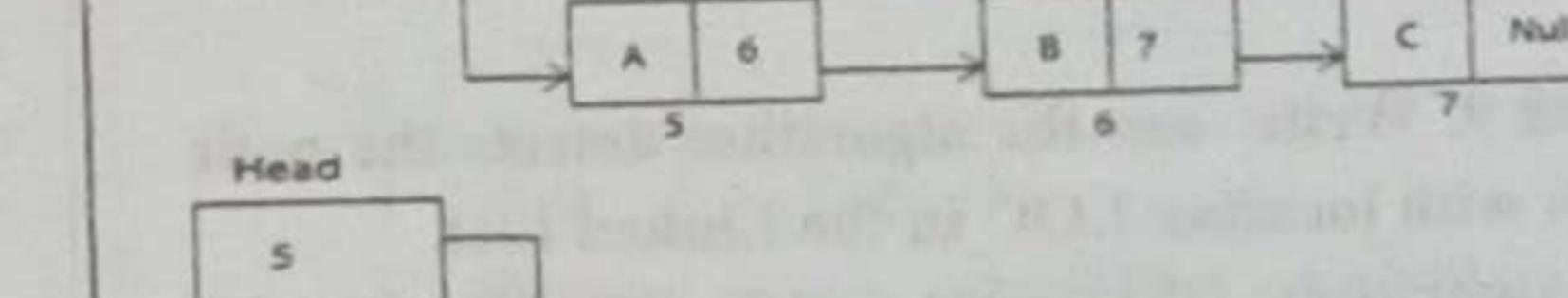
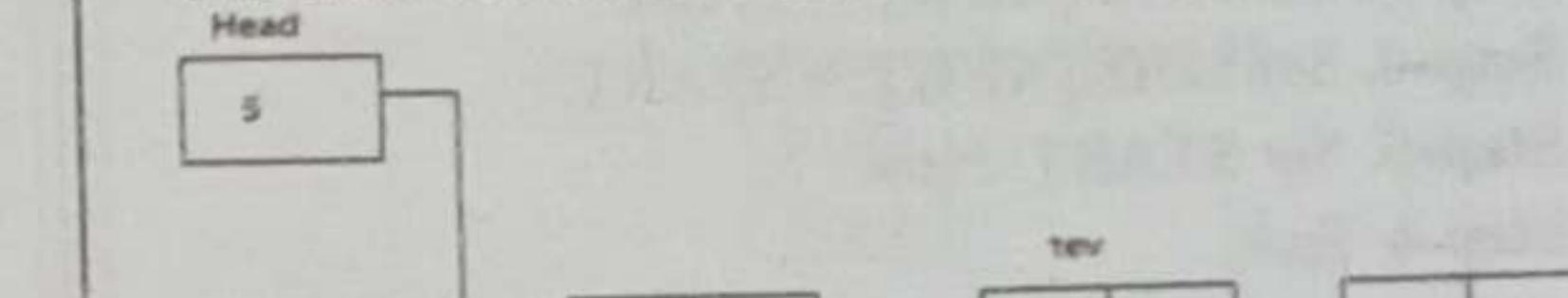


Algorithm:

```
Cin << 6;
While (Ptr != 6) Step 1 = Search element{
    if ptr == element
        tev = tev → next
        Top = element{
```

```
tev = tev → next
if (Ptr == 6)
    Free (Top)
    Ptr = 6;
else
    Not Found
    tev = tev → next
    Free (Ptr);
```

c. Delete a node at Last:



Step 1 : search last element

Step 2 : Top = Ptr

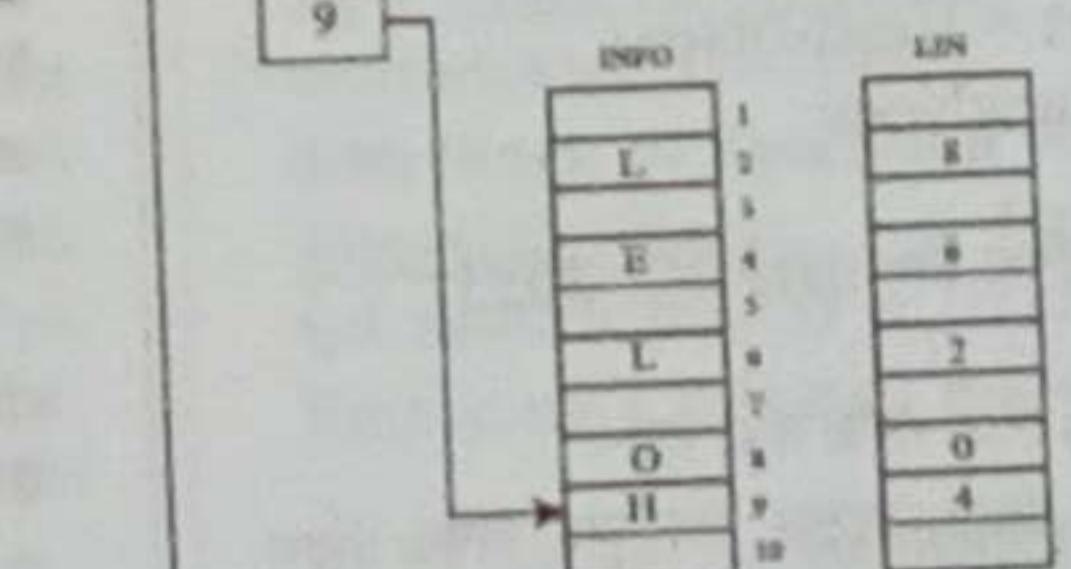
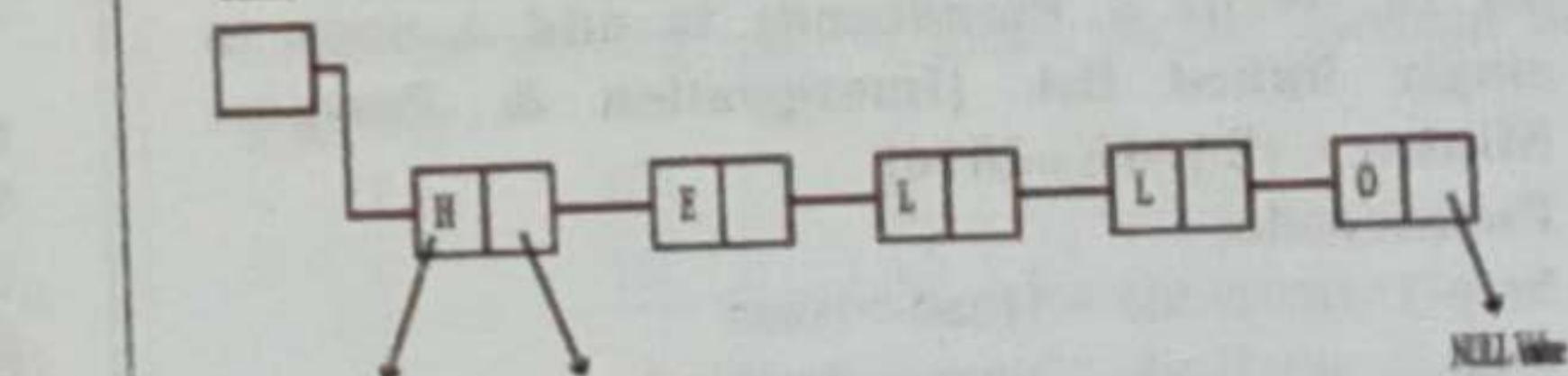
Step 3 : tev → next = null

Step 4 : Free (Top);

প্র. 7. Linked list এর memory representation লিখ।

উত্তর: Linked list এর memory representation-

STRUCT



Here

START = 9 INFO[9] = H is the first character
LINK[9] = 4 INFO[4] = E is the second character
LINK[4] = 6 INFO[6] = L is the second character
LINK[6] = 2 INFO[2] = L is the second character
LINK[2] = 8 INFO[8] = O is the second character
LINK[8] = 0 The NULL value, so the list end here.

Solⁿ:

SL.	Symbol Scanned	Stack	Expression
1	A	(A
2	+	(+	A
3	((+)	A
4	B	(+)	AB
5	*	(+)*	AB
6	C	(+)*	ABC
7	-	(+)-	ABC*
8	((+)-(ABC*
9	D	(+)-(ABC*D
10	/	(+)-(/)	ABC*D
11	E	(+)-(/)	ABC*DE
12	↑	(+)-(/ ↑)	ABC*DE
13	F	(+)-(/ ↑)	ABC*DEF
14)	(+)-()	ABC*DEF/
15	*	(+)-(*)	ABC*DEF/G
16	G	(+)-(*)	ABC*DEF/G* -
17)	(+)	ABC*DEF/G* -
18	*	(+)*	ABC*DEF/G* -
19	H	(+)*	ABC*DEF/G* - H
20)		ABC*DEF/G* - H* +

প্র 18. নিচের Postfix হতে মান নির্ণয় কর। (Normally Using Stack)

$$\begin{aligned} \text{Sol}^n: P = 5, 6, 2, +, *, 12, 4, /, - \\ = 5(6+2) * 12, 4, /, - \\ = 5, 8, *, 12, 4, /, - \\ = 40, 12, 4, /, - \\ = 40, 3, - \\ = 37 \end{aligned}$$

Ans:

SL.No	Symbol Scan	Stack
1.	5	5
2.	6	5, 6
3.	2	5, 6, 2
4.	+	5, 8
5.	*	40
6.	12	40, 12
7.	4	40, 12, 4
8.	/	40, 3
9.	-	37

প্র 19. নিম্নলিখিত infix Expression সমূহকে সমতুল্য Postfix এ কপালৰ কৰ।

- $(A-B) * (D/E)$
- $(A+B\bar{D}) / (E-F)+G$
- $A*(B+D) / E - F * (G+H/K)$

Solⁿ:

$$\begin{aligned} i. (A-B) * (D/E) \\ = [AB -] * [DE/] \\ = AB - DE/* \quad (\text{Ans:}) \end{aligned}$$

$$\begin{aligned} ii. (A+B\bar{D}) / (E-F)+G \\ = (A+[BD]) / [EF -]+G \\ = [ABD+] / [EF -] + G \\ = [ABD+EF -/] + G \\ = ABD+EF -/ G + \quad (\text{Ans:}) \end{aligned}$$

$$\begin{aligned} iii. A*(B+D) / E - F * (G+H/K) \\ = A*[BD+] / E - F * (G+[HK /]) \\ = [ABD + *] / E-F * [GHK /+] \\ = [ABD+E/] - [FGHK/+*] \\ = ABD+E/FGHK/+* - \quad (\text{Ans:}) \end{aligned}$$

প্র 20. $((A+B)*C-(D-E))\uparrow F$ Infix হতে Prefix এ Postfix মান নির্ণয় কৰ। [Bangladesh Bank-2012]

Solⁿ:

Prefix

$$\begin{aligned} ((A+B)*C-(D-E))\uparrow F \\ = ((+AB)*C(-DE))\uparrow F \\ = ((+ABC)-(-DE))\uparrow F \\ = ((-+ABC-DE))\uparrow F \\ = \uparrow - + ABC-DEF \quad (\text{Ans:}) \end{aligned}$$

Postfix

$$\begin{aligned} ((A+B)*C-(D-E))\uparrow F \\ = ((AB+)*C-(DE-))\uparrow F \\ = ((AB+C*)-(DE-))\uparrow F \\ = ((AB+C*DE-))\uparrow F \\ = AB+C*DE-F \quad (\text{Ans:}) \end{aligned}$$

প্র 21. $P = 12, 7, 3, -, /, 2, 1, 5, +, *, +$ Expression হতে (i) Infix Experession এ দেখাও

(ii) মান নির্ণয় কৰ

(iii) Stack এর মাধ্যমে মান নির্ণয় কৰ।

উভয়ে (i) $P = 12, 7, 3, -, /, 2, 1, 5, +, *, +$

$$\begin{aligned} &= 12, [7-3], /, 2, 1, 5, +, *, + \\ &= (12/[7-3]), 2, [1+5], *, + \\ &= (12/[7-3])(2*[1+5]), + \\ &= 12/[7-3] + 2*(1+5) \quad \text{Showed.} \end{aligned}$$

(ii) $P = 12, 7, 3, -, /, 2, 1, 5, +, *, +$

$$\begin{aligned} &= 12, 4, /, 2, 6, *, + \\ &= 3, 12, + \\ &= 15 \quad \text{Ans:} \end{aligned}$$

(iii) Ans:

Sl.No	Symbol Scan	Stack
1.	12	12
2.	7	12, 7
3.	3	12, 7, 3
4.	-	12, 4
5.	/	3
6.	2	3, 2
7.	1	3, 2, 1
8.	5	3, 2, 1, 5
9.	+	3, 2, 6
10.	*	3, 12
11.	+	15

প্র 24. নিচের টিকে একটি Item push এবং POP কৰা ব্যাখ্যা কৰে দেখাও?

AA	BB	CC	DD				
1	2	3	4	5	6	7	8

Top
Maxstk

Solⁿ:

(a) Let stack এ একটি item "EE" push কৰতে হবে।

উভয়ে Push Algorithm হতে পাই-

- যেহেতু Top = 4, go to step - 2

- Top = 4+1 = 5 হবে।

- Stack [Top] = EE

- Return

বর্তমানে stack এর Top = 5 যার উপাদান "EE"

(b) আবার একই stack এর POP (Stack, Item) operation কৰা হলে, POP algorithm হতে পাই

- যেহেতু Top = 4, go to step - 2

- Item = Stack [Top] = DD

- Top = 4 - 1 = 3

- Return

দেখা যাচ্ছে, Stack এর Top = 3 যার উপাদান "CC"

প্র 25.

A	I	T	B			
1	2	3	4	5	6	

উপরের stack টি বিবেচনা কৰ।

(a) কখন stack টি overflow হবে?

(b) কখন B এর আগে T অপসারিত (Delete) হবে?

(c) কখন stack টি underflow হবে?

Solⁿ:

(a) Stack এর size হচ্ছে 6 কাজেই stack টি সর্বোচ্চ 6 টি Data সংযোজন কৰতে হবে, অত্যেক Data সংযোজনের পর stack এর তিন অক্ষণ কৰ এবং Top, maxstk প্রদর্শন কৰ।

Stack: Al, Ar, H, J, R, S

Solⁿ:

Stack এ পর্যায়মে 6 টি Data push কৰার চিন

অপর্যাপ্ত হল-

Top	Maxstk
Al	6

Top	Maxstk
Ar	6

Top	Maxstk
H	6

Top	Maxstk
J	6

Top	Maxstk
R	6

Top	Maxstk
S	6

Top	Maxstk
	6

Top	Maxstk
	6

Top	Maxstk
	6

Top	Maxstk
	6

প্রশ্ন 27. Recursion প্রক্রিয়ার 4! এর মান নির্ণয় কর।

উত্তর: Recursion প্রক্রিয়ার 4! এর মান 9 টি ধরণে সম্পূর্ণ হয়।

$$1. \quad 4! = 4 \cdot 3!$$

$$2. \quad 3! = 3 \cdot 2!$$

$$3. \quad 2! = 2 \cdot 1!$$

$$4. \quad 1! = 1 \cdot 0!$$

$$5. \quad 0! = 1$$

$$6. \quad 1! = 1 \cdot 1 = 1$$

$$7. \quad 2! = 2 \cdot 1 = 2$$

$$8. \quad 3! = 3 \cdot 2 = 6$$

$$9. \quad 4! = 4 \cdot 6 = 24.$$

প্রশ্ন 28. Write are the Applications of Stack? Or

Why do we use stack? [Multiple Ministry(AP)-2017]

Use of stack:

স্ট্যাক একটি রেস্ট্রিকেটেড ভাটা স্ট্রাকচার কারণ এতে সীমিত সংখ্যাক অপারেশন সম্পূর্ণ করা যায়। স্ট্যাক (Stack) ভাটা স্ট্রাকচারে ভাটাগুলো রাখা হয় এমনভাবে যেন নতুন কোনো ভাটা সেখানে রাখতে হলে সবচাইতে উপরে রাখতে হয়, এবং নেয়ার সময়ও সবসময় প্রথমে উপরের ভাটাটাকে নিতে হবে। তারমানে স্ট্যাকে তুমি যাই করো না কেন, সেটা হতে হবে সবচাইতে উপরের ভাটাটাকে। আমরা কোনো ভাটা রাখতে হলে সেটাকে রাখবো আগের গুলোর ওপরে, আর নেয়ার সময়ও নিতে হবে সবার উপরেরটা।

Applications of stack/ Use of stack:

a. Expression Evaluation

b. Expression Conversion

i. Infix to Postfix

ii. Infix to Prefix

iii. Postfix to Infix

iv. Prefix to Infix

c. Backtracking

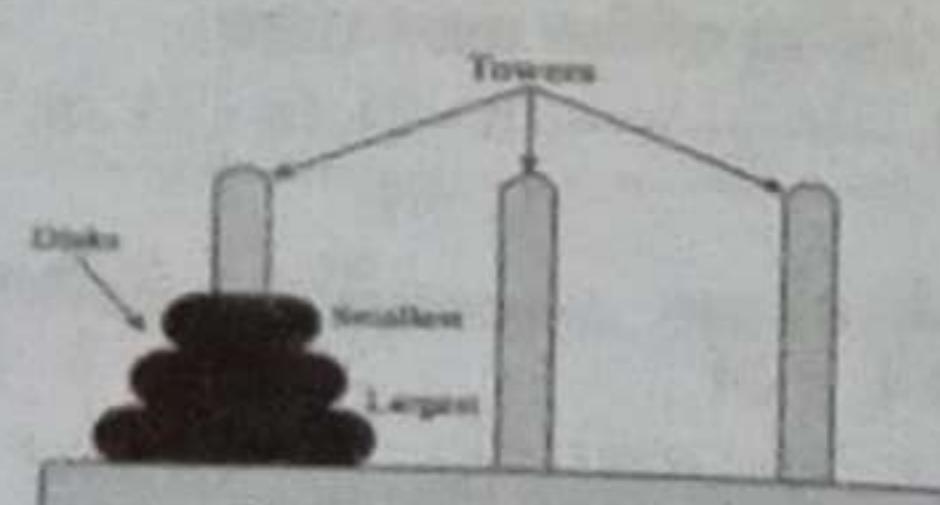
d. Memory Management

e. Recursion Handling

f. Parenthesis Handling

প্রশ্ন 29. What is Tower of Hanoi? What are the rules to be followed by Tower of Hanoi?

টাওহোয়ার অব হানয় হল এক ধরনের বৃক্ষির খেলা। এখেলায় তিনটি দণ্ড থাঢ়াভাবে পাশাপাশি রাখা থাকে। এবং ছেট বড় কিছু ডিক বা চাকতি দণ্ড এবং ডিকের প্রবেশ করান থাকে। এখেলায় প্রথম দণ্ড থেকে তৃতীয় দণ্ডে সবগুলো চাকতি আনতে হয়। তবে কিছু নিয়ম গোলন করতে হয়।



নিয়মগুলো হলো:

1. একসাথে একাধিক চাকতি সরানো যাবে না।
2. কখনই ছেট চাকতির উপর বড় চাকতি রাখা যাবে না।
3. সবসময় উপরের চাকতি ঝালনো যাবে।

n disks এর একটি টাওহোয়ার অফ হ্যানয় পাজলকে সমাধান করতে সমন্বয় $2^n - 1$ স্টেপের প্রয়োজন হবে। উদাহরণ সরঞ্জ তিনি ডিকের একটি পাজলকে সমাধান করতে $2^3 - 1 = 7$ টি স্টেপের প্রয়োজন হবে।

প্রশ্ন 30. Write are the algorithm in Tower of Hanoi ? Algorithm: TOWER(N,BEG,AUX,END)

Step-1: If N=1, then:

- (a) write: BEG => END
- (b) return

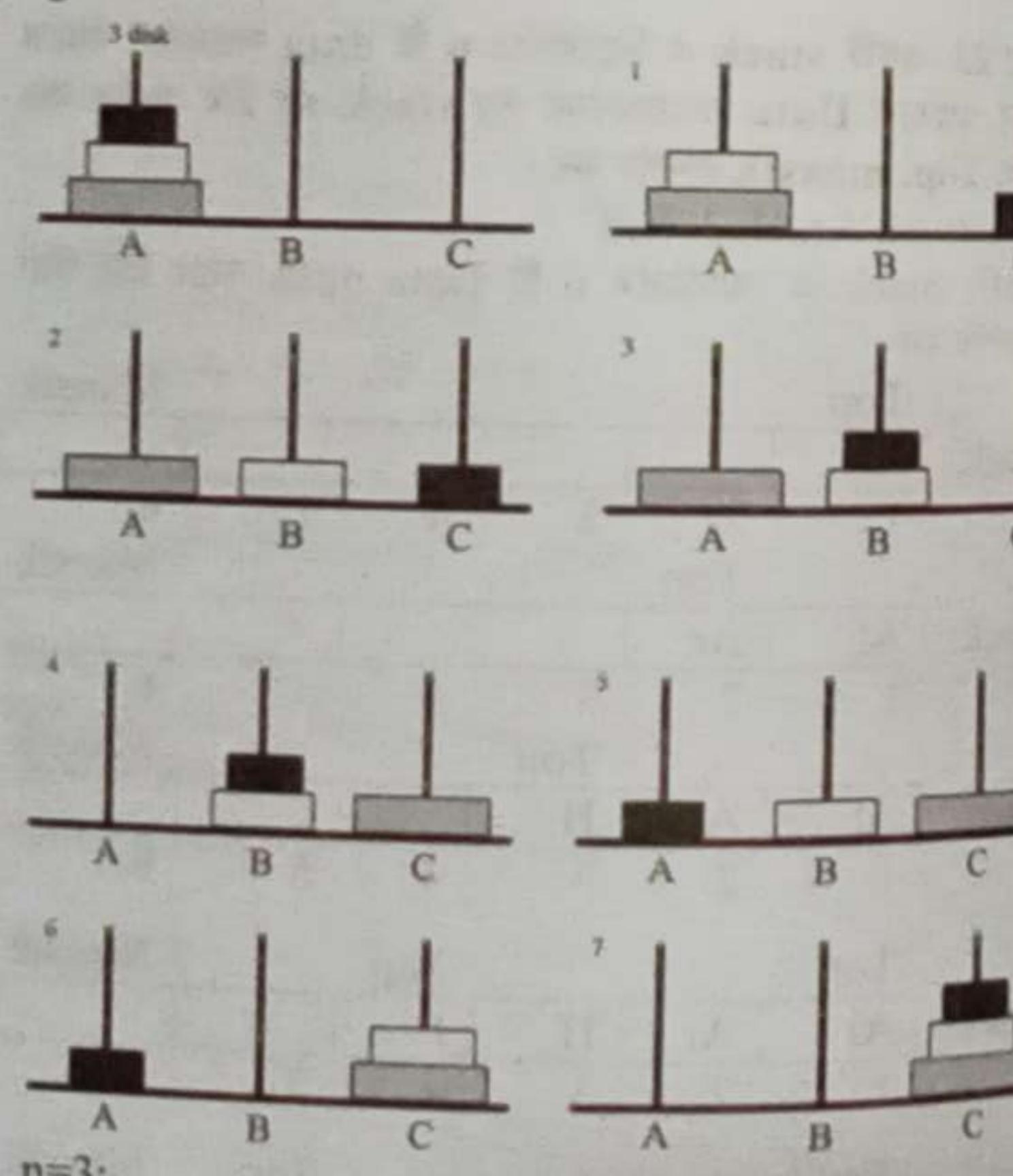
Step-2: Call TOWER (N-1, BEG,END,AUX).

Step-3: write: BEG => END

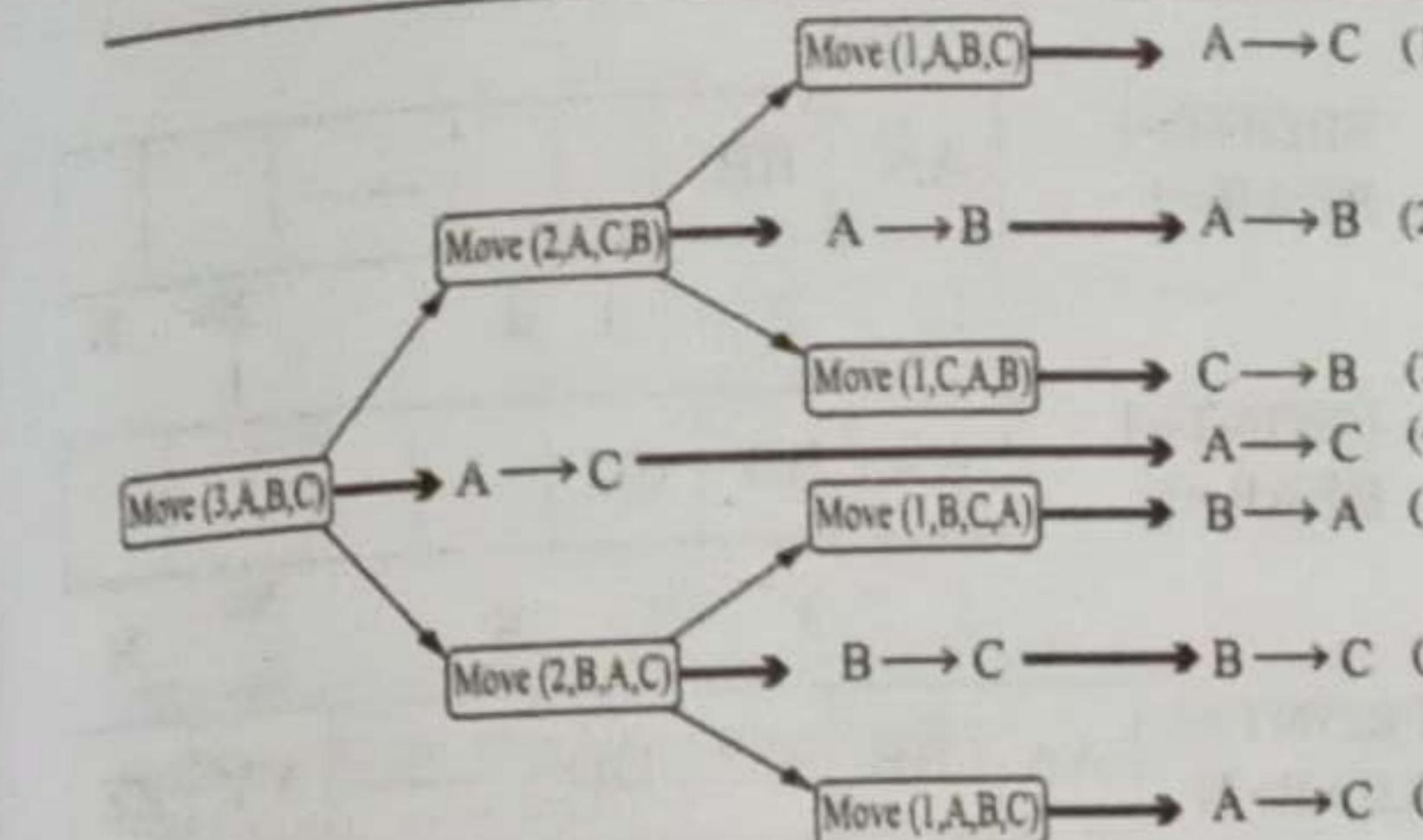
Step-4: Call TOWER (N-1, AUX,BEG,END).

Step-5: return

Example: Suppose three pegs, labelled A,B and C



A=>C, A=>B, C=>B, A=>C, B=>A, B=>C, A=>C



প্রশ্ন 31. Why do we use stack? Differentiate between Stack and Array? [স্ট্যাক কেন ব্যবহার করা হয়? স্ট্যাক এবং আরের মাঝে পার্থক্য লিখ?] [এবাবী কল্পনা ও বৈদেশিক কর্মসূচী জনপ্রিয় কর্মসূচী হ্যানো-২০১৮- ইলেক্ট্রোকের (কলিপটার)]

Use of stack: স্ট্যাক একটি রেস্ট্রিকেটেড ভাটা স্ট্রাকচার কারণ এতে সীমিত সংখ্যাক অপারেশন সম্পূর্ণ করা যায়। স্ট্যাক (Stack) এতে সীমিত সংখ্যাক অপারেশন সম্পূর্ণ করা যায়। স্ট্যাক ভাটা স্ট্রাকচারে ভাটাগুলো রাখা হয় এমনভাবে যেন নতুন কোনো ভাটা সেখানে রাখতে হলে সবচাইতে উপরে রাখতে হয়, এবং নেয়ার সময়ও সবসময় প্রথমে উপরের ভাটাটাকে নিতে হবে। তারমানে স্ট্যাকে তুমি যাই করো না কেন, সেটা হতে হবে সবচাইতে উপরের ভাটাটাকে। আমরা কোনো ভাটা রাখতে হলে সেটাকে রাখবো আগের গুলোর ওপরে, আর নেয়ার সময়ও নিতে হবে সবার উপরেরটা।

মূলত backtracking algorithm এ স্ট্যাক ব্যবহার করা হয়।

স্ট্যাক টেম্পোরারি ভাটা ধারন করে, রিকার্সিভ কাজে, implement functions, parsers, expression evaluation কাজে ও স্ট্যাক ব্যবহার করা হয়।

Array এবং Stack এর মাঝে পার্থক্য নিম্নে দেওয়া হলো:

Array	Stack
Array তে যেকোন position দিয়ে ডাটা enter এবং exit হবে। FILO pattern	স্ট্যাকে একটি শাস্ত দিয়ে ডাটা enter এবং exit হবে।
একই টাইপের ভাটা স্টোর করে।	বিভিন্ন টাইপের ভাটা স্টোর করতে পারে।
ব্যাসিক অপারেশন: insert,delete,merge,modify , traverse,sort	ব্যাসিক অপারেশন: push ,pop,peek
যেকোন পজিশনের ভাটা আরে ইনডেক্সের সাহায্য অ্যাক্সেস করা যাব।	তথ্যময় টপ এর ভাটা রিভ রাইট করা যাব।

Self Study

প্রশ্ন 1. Stack use করে মান নির্ণয় করো

$$(i) 2, 1, 3, +, 5, *, 2, 1, 2, -, 12, /, 6 \quad \text{Ans: } 26$$

$$(ii) 5, 3, +, 2, *, 6, 9, 7, -, /, - \quad \text{Ans: } 13$$

$$(iii) 3, 5, +, 6, 4, -, *, 4, 1, -, 2, 1, + \quad \text{Ans: } 25$$

$$(iv) 3, 1, +, 2, 1, 8, 4, -, 2, *, +, 5, - \quad \text{Ans: } 19$$

প্রশ্ন 2. ধর একটি stack যার মেমরি Cell n = 6 হলে stack নিম্ন পিসিট উপরে সম্পূর্ণ কর।

$$(i) \text{PUSH (STACK, KKK)}$$

- (ii) POP (STACK, ITEM)
- (iii) PUSH (STACK, LLL)
- (iv) PUSH (STACK, SSS)
- (v) POP (STACK, ITEM)
- (vi) PUSH (STACK, TTT)

প্রশ্ন 3. Expression evaluation:

$$(i) 16, 7, 3, -, /, 2, 1, 5, +, *, - \quad \text{Ans: } -8$$

$$(ii) *, +, 5, 6, 3, \quad \text{Ans: } 33$$

$$(iii) 5, 6, 2, +, *, 12, 4, /, - \quad \text{Ans: } 37$$

প্রশ্ন 4. Expression পরিবর্তন কর Opposite two notation.

$$a) (A - B) * (D/E)$$

$$b) (A+B)\uparrow D/(E - F) + G$$

$$c) A * (B+D)/E - F*(G+H/K)$$

$$d) ((A+B)*D)\uparrow (E - F)$$

$$e) (A+B) * C - (D - E) \uparrow (F+G)$$

$$f) 12/(7 - 3)+2*(3+8) - 7$$

$$g) (A+B) * ((C - D) * E+F) \$$$

$$h) 25/(9+3)+7*(13+10)$$

$$i) (A - B) /((D+E) * F)$$

$$j) ((A - B)/D) \uparrow ((E+F) * G)$$

$$k) A+(B+D)/E - F*(G+H+K)$$

$$l) A+(B*C - (D/E\uparrow F)*G) * +$$

$$m) (A\uparrow B+C/D)*(C - D)$$

$$n) (A - B)^* (D+E)$$

$$o) (A+B\uparrow D)/(E - F) * G$$

প্রশ্ন 5. (a) Converting $(A+B)^*C+D/(E+F*G)^*H$ expression into Prefix and Postfix from [36th BCS]

$$\text{Prefix: } (A+B)^*C+D/(E+(F\uparrow G))^*H$$

$$= (+AB)^*C+D/(+E^*FG)^*H$$

$$= (*+ABC)+(/D+E^*FG)^*H$$

$$= (*+ABC)+(*+D/E^*FGH)$$

$$= +*+ABC*/D+E^*FGH$$

$$\text{Postfix: } (A+B)^*C+D/(E+F*G)^*H$$

$$= (AB+)^*C+D/(E+(FG^*))^*H$$

$$= (AB+)^*C+D/(EFG^+)^*H$$

$$= (AB+ C^*) +(DEFG^+/)^*H$$

$$= (AB+ C^*) +(DEFG^++/H^*)$$

$$= AB+ C^*DEFG^++/H^+$$

প্রশ্ন 5. (b) Write prefix and Post fix notation from the statement like $((A+B)^*C-(D-E)\uparrow F)$ [BB(AD)-2016]</p

উত্তর			
SLNO.	FRONT	REAR	STATUS
1. Empty	0	0	-,-,-,-,-
2. Ins (A,B,C)	1	3	A, B, C, -, -
3. Del (A,B)	3	3	-,-, C, -, -
4. Ins (D, E)	3	5	-,-, C, D, E
5. Ins (F, G)	3	2	F, G, C, D, E
6. Del (C)	4	2	F, G, -, D, E
7. Ins (H)	4	3	F, G, H, D, E
8. Del (G)	3	3	-,-, H, -, -
9. Del (H)	0	0	-,-,-,-,-

প্র 10. মনে কর 5 টি element বিশিষ্ট Queue যা initially empty, নিম্নলিখিত operation করো সম্পূর্ণ কর : let Queue টি circular বা linear Queue.

- (i) Initially empty (ii) EN Queue A, B, C
- (iii) DE Queue A (iv) EN Queue D, E
- (v) DE Queue B, C (vi) EN Queue F
- (vii) DE Queue D (viii) EN Queue G, H
- (ix) DE Queue E (x) DE Queue F
- (xi) EN Queue K (xiii) DE Queue G, H, K

Linear Queue:

let, Front = F

Rear = R

(i) initially empty:

1	2	3	4	5	F=0 R=0

(ii) Insert A, B, C:

A	B	C			
1	2	3	4	5	F=1 R=3

(iii) Delete A:

	B	C			
1	2	3	4	5	F=2 R=3

(iv) Insert D, E:

	B	C	D	E	
1	2	3	4	5	F=2 R=5

(v) Delete B, C:

		D	E		
1	2	3	4	5	F=4 R=5

(vi) Insert F:

F		D	E		
1	2	3	4	5	F=4 R=1

(vii) Delete D:

F			E		
1	2	3	4	5	F=5 R=1

(viii) insert G, H:

F	G	H		E	
1	2	3	4	5	F=5 R=3

(ix) Delete E:

F	G	H			
1	2	3	4	5	F=1 R=3

(x) Delete F:

	G	H			
1	2	3	4	5	F=2 R=3

(xi) insert K:

	G	H	K		
1	2	3	4	5	F=2 R=4

(xii) Delete G, H, K:

1	2	3	4	5	F=0 R=0

প্র 11. STACK & Queue এর মধ্যে পার্থক্য লিখ? [BPSC-2017, BRTA-2012 BRTA (S CO)-2012]

STACK

QUEUE

(i) Stack এমন একটি পদ্ধতি যার মধ্যে data সহজে একটির পর আরেকটি সাজানো থাকে।

(ii) STACK LIFO System এ কার্য সম্পন্ন করে।

(iii) Insertion & deletion এর জন্য একটি pointer "TOP" ব্যবহৃত হয়।

(iv) Insert & Delete operation একই প্রক্রিয়া হয়।

(v) Stack এর কোন প্রকার তেই নাই।

Self Study

প্র 1. ধরি একটি queue তে 6 টি memory Cell আছে। যদি queue টি একটি circular array হয় তাহলে অপারেশন করো সম্পূর্ণ কর।

Front = 2 Rear = 4 Queue = -, A, C, D, -, -

(a) F কে সংযোজন কর।

(b) দুটো বর্ণ বিয়োজন কর।

(c) K, L, M সংযোজন কর।

(d) দুটো বর্ণ বিয়োজন কর।

(e) R সংযোজন কর।

(f) দুটো বর্ণ বিয়োজন কর।

(g) S সংযোজন

(h) একটি letter সংযোজন

(i) একটি letter বিয়োজন

(j) একটি letter বিয়োজন

প্র 2. ধরি একটি Dequeue তে 6টি Memory Cell আছে। নিম্নলিখিত অপারেশন করো সম্পূর্ণ কর এবং প্রতিটি ধাপ বর্ণনা কর। LEFT = 2, RIGHT = 4, DEQUE: -, A, C, D, -, -

(a) F কে Right এ সংযোজন।

(b) Left থেকে দুটো বর্ণ বিয়োজন কর।

(c) K, L, M কে Right এ সংযোজন কর।

(d) Left থেকে একটি বর্ণ বিয়োজন কর।

(e) R কে Right এ add কর।

(f) S, T কে Right এ সংযোজন কর।

Ans: Znf step array is full T can not be added.

প্র 3. ধরি, একটি Circular Queue তে 8টি মেমোরি location আছে। নিম্নলিখিত Operation করো সম্পূর্ণ কর।

(i) Initially Empty (Initially Empty)

(ii) ৫, ৭, ১০ এবং ৩ যুক্ত। (সংযোজন) কর।

(iii) ৫, ৭ কে বিয়োজন (Delete) কর।

(iv) ৮, ১২, ১৫, ২০ কে সংযোজন (Insert) কর।

(v) ১০, ৩, ৮ কে বিয়োজন (Delete) কর।

(vi) ১৭, ২৩ কে সংযোজন (Insert) কর।

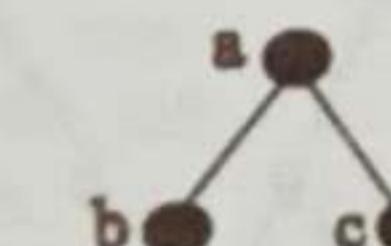
(vii) ১২, ১৫, ২০ কে বিয়োজন (Delete) কর।

৪. TREE

প্র 1. Tree কাকে বলে?

উত্তর: Tree হলো এক ধরনের connected graph যার কোন close পথ থাকবেনা।

Example:



প্র 2. ট্রি এর বৈশিষ্ট্য লিখ।

উত্তর: ট্রি এর বৈশিষ্ট্য কোন নিম্নলিখিত -

• একটি Root node থাকবে, যার level 0।

• n সংখ্যাক নোড বিলিট ট্রিতে বাছ সংখ্যা হবে $(n - 1)$ টি।

• যদি কোন Tree এর level L হয় তবে depth হবে L-1।

• যদি কোন ট্রি এর node সংখ্যা n হয় তবে Internal node

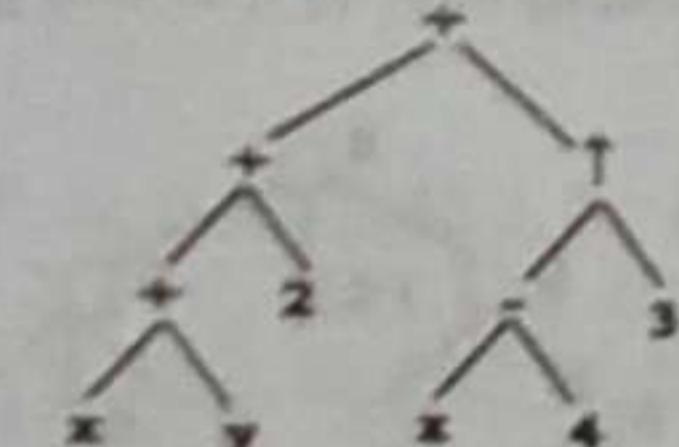
$$\text{সংখ্যা } i = \frac{n-1}{2} \text{ হবে।}$$

• যদি কোন Tree এর Node সংখ্যা n হয় তবে External node বা পাতা সংখ্যা $\frac{n+1}{2}$ হবে।

Solution:

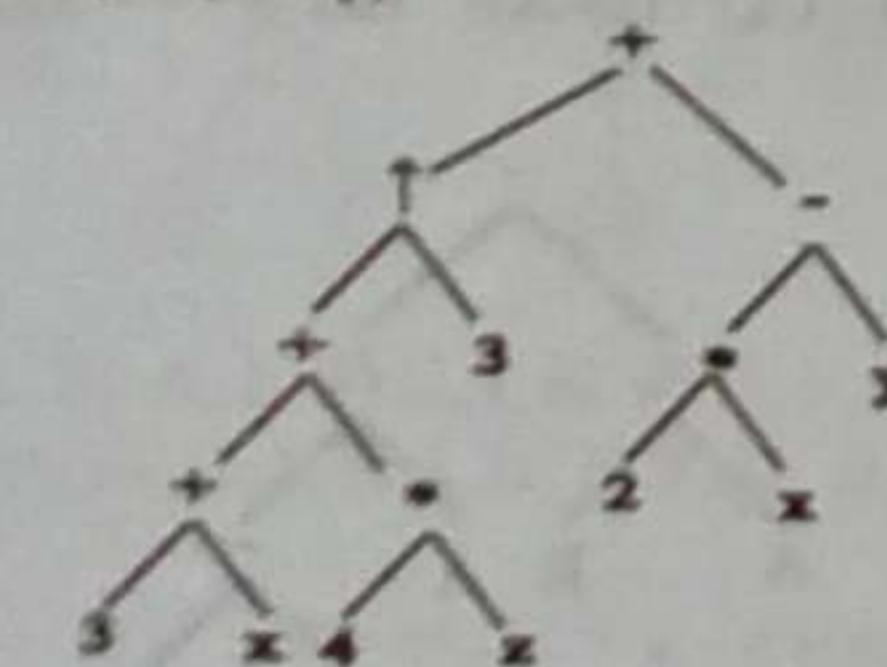
Infix: $(2+3) \uparrow (5-1)$
Postfix: $2, 3, +, 5, 1, -, \uparrow$

(iii) $((x+y)+2) \uparrow ((n-4) \uparrow 3)$

Solution:

Prefix: $+, +, +, x, y, 2, \uparrow, -, n, 4, 3$
Postfix: $x, y, +, 2, +, x, 4, -, 3, \uparrow, +$

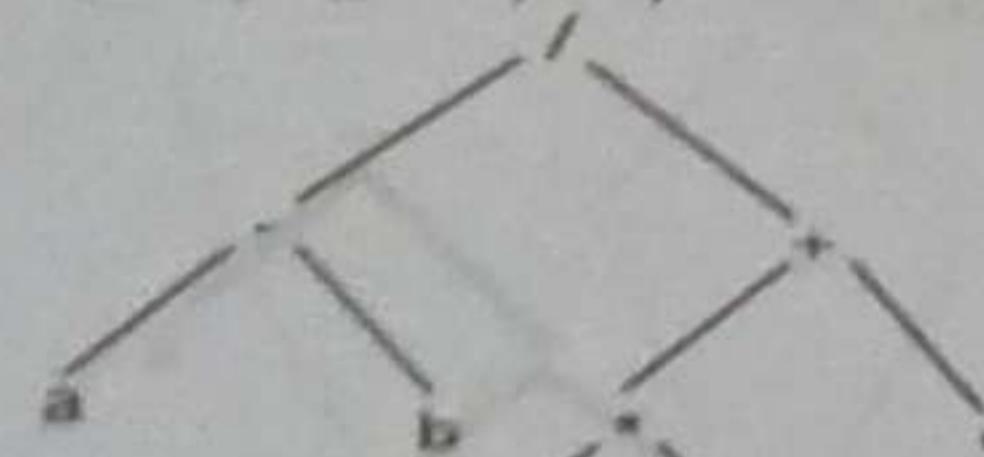
(iv) $(3x+4z)^3+(2x-y)$



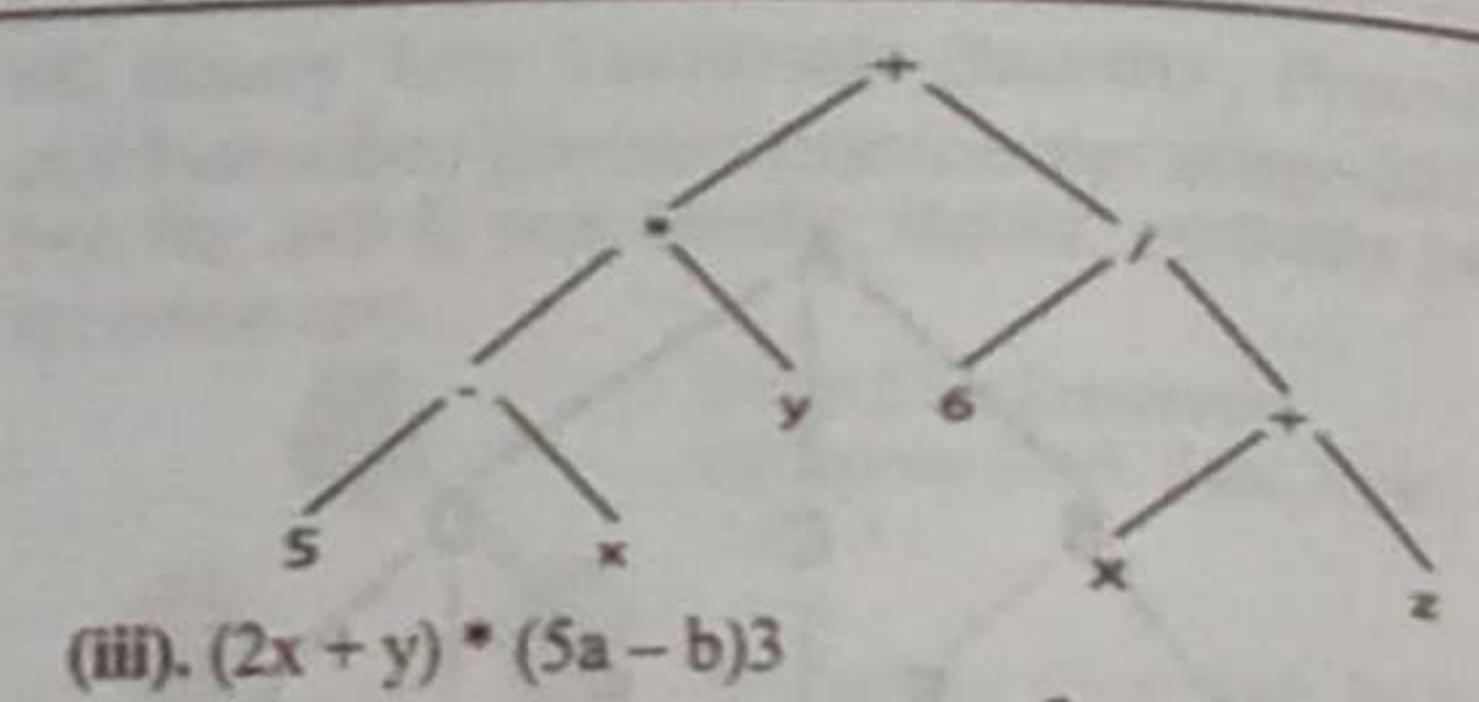
Prefix: $+, \uparrow, +, +, 3, x, *, 4, z, 3, -, *, 2, x, y$
Postfix: $3, x, +, 4, z, *, +, 3, \uparrow, 2, x, *, y, -, +$

Ques 8. Expression একে Binary tree কৈবল্য কর

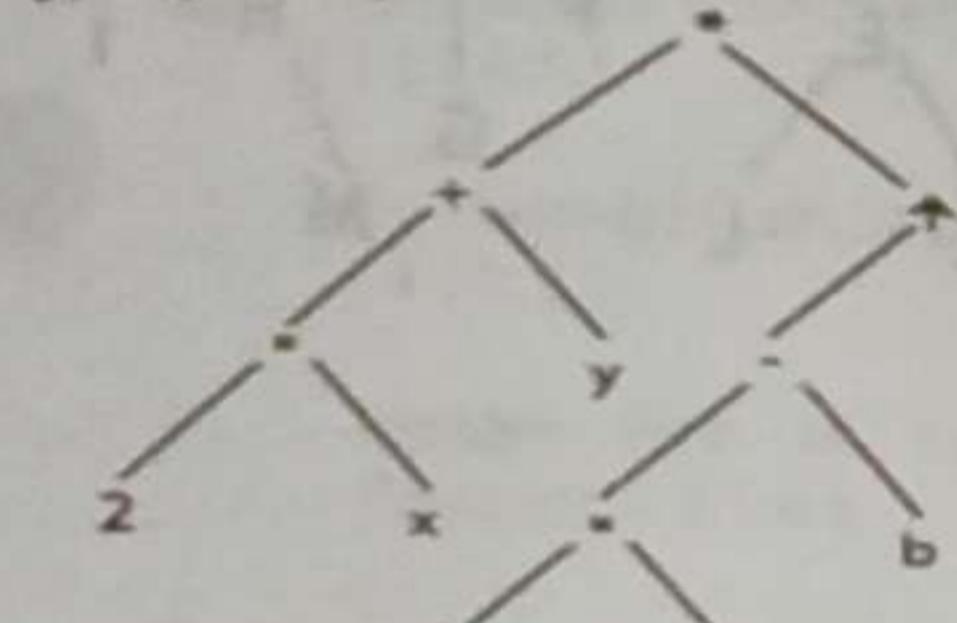
(i). $E = (a - b) / ((c * d) + e)$



(ii) $(5 - x) * y + 6 / (x + z)$
 $= ((5 - x) * y) + (6 / (x + z))$

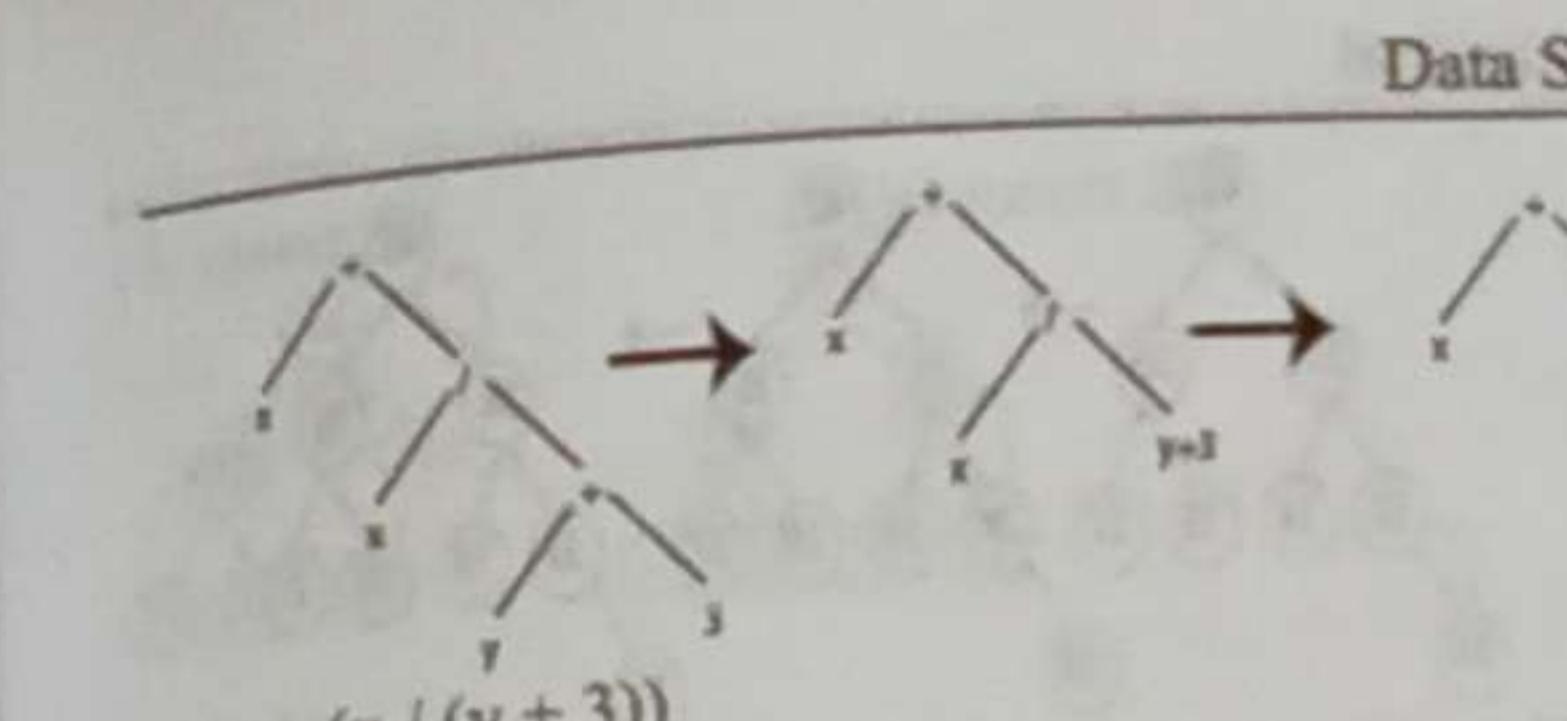
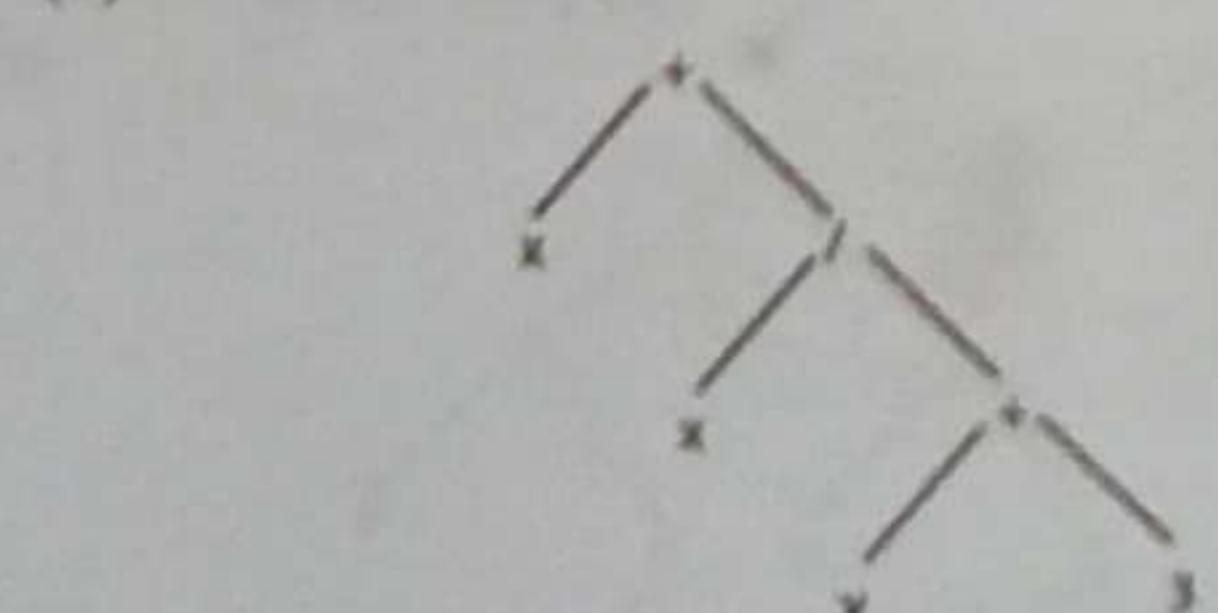
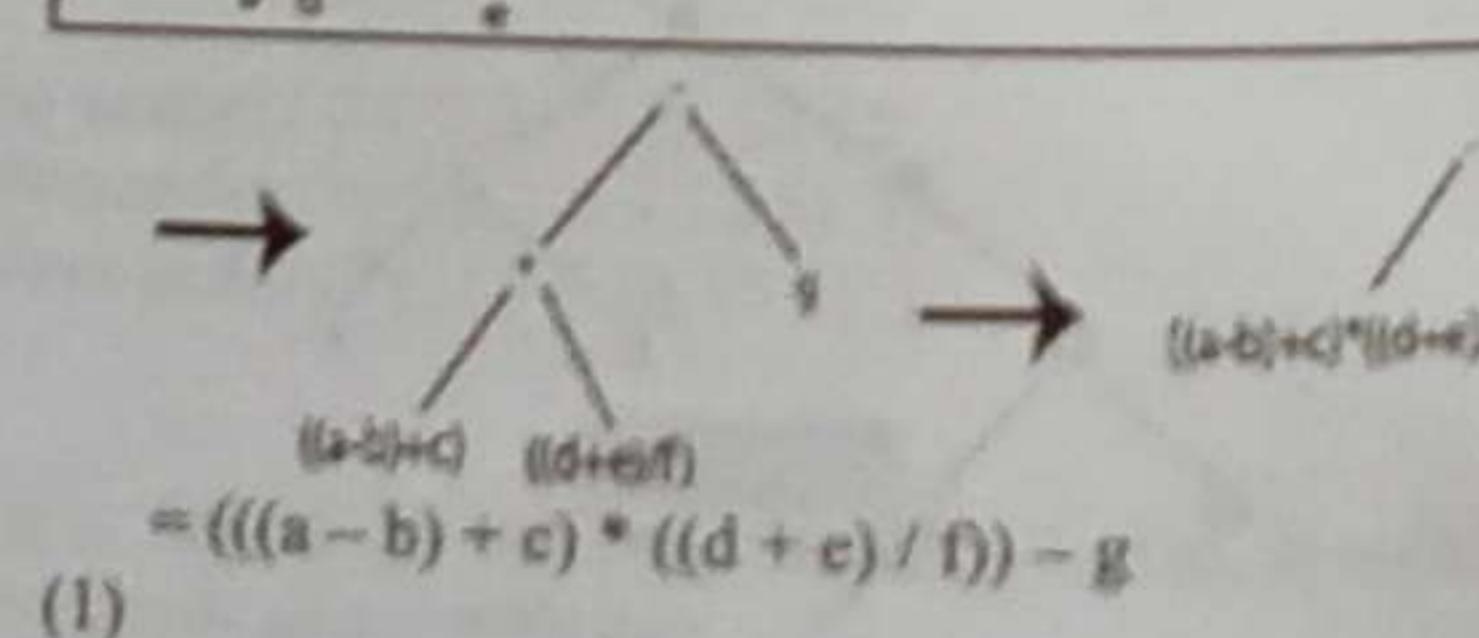
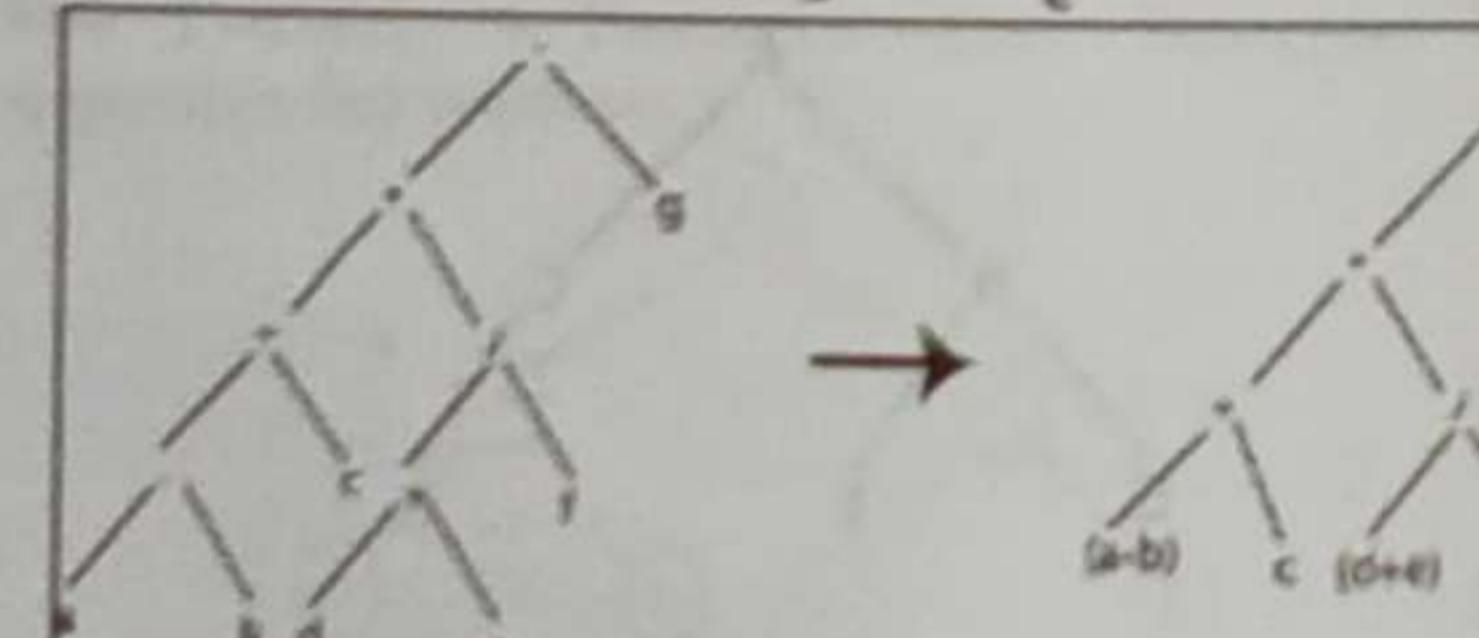
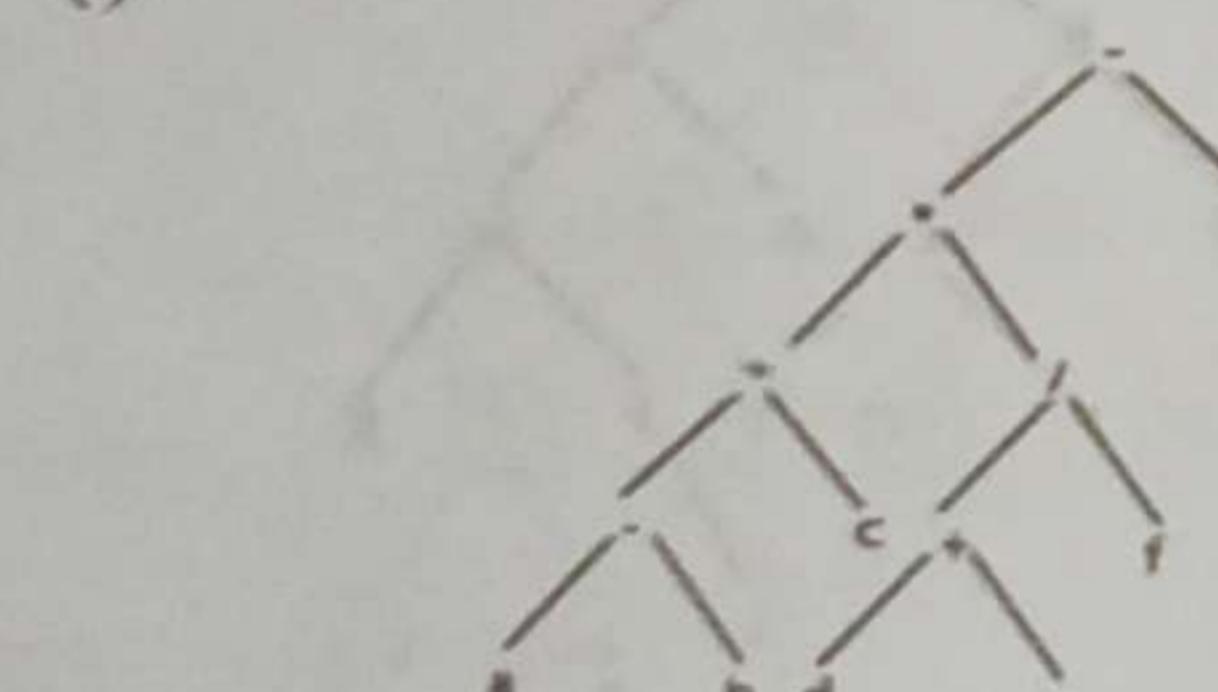


(iii). $(2x + y) * (5a - b)3$



Ques 9. Tree একে Expression

(i).



Ques 6. Expression একে মান কৈবল্য কর।

(i). $+ - * 2 3 5 \uparrow 234$ (ii). $- * 2/843$
 $= + - * 235/2^34$ $= - * 28/43$
 $= + - * 235/84$ $= - * 223$
 $= + - * 235 8/4$ $= - 43$
 $= + - * 2352$ $= 1$ Ans:
 $= + - 652$
 $= + 12$
 $= 3$ Ans:

(iii). $\uparrow - * 33 * 425$ (vi). $521 - 314++*$
 $= \uparrow - * 3385$ $= 51 - 314++*$
 $= \uparrow - 985$ $= 4314++*$
 $= \uparrow 15$ $= 435 + *$
 $= 1$ Ans: $= 48 *$
 $= 32$ Ans:

(iv). $+ - \uparrow 32 \uparrow 23/6 - 42$ (viii). $93/5 + 72 - *$
 $= + - \uparrow 32 \uparrow 23/62$ $= 9/35 + 72 - *$
 $= + - \uparrow 32 \uparrow 233$ $= 35 + 72 - *$
 $= + - \uparrow 3283$ $= 3 + 5 72 - *$
 $= + - 983$ $= 872 - *$
 $= + 13$ $= 85 *$
 $= 4$ Ans: $= 40$ Ans:

(v). $* + 3 + 3 \uparrow 3 + 3 3$
 $= * + 3 + 3 \uparrow 3 6 3$
 $= * + 3 + 3 729 3$
 $= * + 3 732 3$
 $= * 735 3$
 $= 2205$ Ans:

(vi) Expression টি evaluate কৈবল্য: $32 * 2 \uparrow 53 - 84 / -$.
জো সমস্যা আছে তাহলে প্রশ্ন কৈবল্য কর।
উত্তর: $32 * 2 \uparrow 53 - 84 / -$

$$\begin{aligned} &= [3 * 2] 2 \uparrow [5 - 3] [8 / 4] * - \\ &= [6 \uparrow 2] [2 * 2] - = [36 - 4] = 32 \end{aligned}$$

Self Study

Ques 1. What is prefix form and postfix form

$((x+y) \uparrow 2) + ((x-4)/3)$.

Ans: Prefix: $+ \uparrow xy2/-x43$

Postfix: $xy+2 \uparrow x4 - 3 / +$

Ques 2. What is the value of the prefix expression?

$+ - * 235 \uparrow 234$.

Ans: 3

Ques 3. What is the value of the postfix expression?

$723 * - 4 \uparrow 93 / +$.

Ans: 4

Ques 8. Represent the expression $((x+2) \uparrow 3) * (y - (3+x)) - 5$ using a tree. Also find this expression

in (i) Prefix Notation

(ii) Postfix Notation

(iii) Infix Notation

Ques 9. Represent the expressions $(x+xy) + (x/y)$ and $x + ((xy+x)/y)$. Also find these expression in

(i) Prefix Notation

(ii) Postfix Notation

(iii) Infix Notation

Ques 10. Write each expression using infix expression

(i) $+ * + - 53214$

Ans: 8

(ii) $\uparrow + 23 - 51$

Ans: 625

(iii) $* / 93 + * 24 - 76$

Ans: 27

Ques 11. Write the value of each of these postfix expression

(i) $521 - - 314 + + *$

Ans: 32

(ii) $93/5 + 72 - *$

Ans: 40

(iii) $32 * 2 \uparrow 53 - 84 / * -$

Ans: 32

[N.B: যদি কথনে operator \geq operand হয় তাহলে evaluation করব নয়।]

Ques 12. What is the result of expression?

1. $42 * 1 \uparrow 83 - 8, 4, * -$

2. $+ - \uparrow 32 \uparrow 23/6 - 42$

3. $93/5 + 72 - *$

Ques 13. Tree কৈবল্য কর

Inorder : BEHFACDGJ

Postorder : HFEABIGDC

প্র ১১. Postfix and prefix conversion কর-
 $A + (B * C - (D / E) \uparrow F) * G * H.$

প্র ১২. Draw a tree diagram from the following Expression.

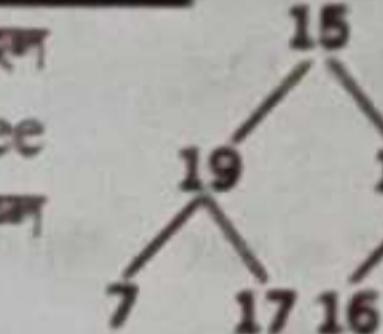
$$E = [a+(b+c)]^*[d-e]/(f+g)]$$

Traverse postorder and preorder

প্র ১৩. Infix Notation এ জপাইব কর।
 $25/(9+3)+7*(13+10)$

২. Heap tree

■ Heap [BCS-36] : Heap হল
একটি complete Binary tree
যেখানে heap Ordering এর সকল
বৈশিষ্ট্য satisfy করে।



Heap দুই প্রকার [তথ্য মোগাদিগ ও অনুক্তি বিভাগ-২০১৯, জাতী এন্ডি সুপারজাইন্স]

(i). Max Heap: যদি/ যদি কোন Root node এর মান child অপেক্ষা বড় অথবা সমান হয় তবে তাকে Max heap বলে। পাশের ম্যাগ্রিপ টিকে ৪২ সবচেয়ে বড় নোডটি রুট এ আছে। তার চাইল্ড নোড ৩৩, ৩৫ নিচে আছে। ৩৩, ৩৫ তারে বামে সেকেন দিকেই থাকতে পারবে। শর্ত হলো রুট অবশ্যই চাইল্ড থেকে বড় হবে।

(ii) Min heap: যদি/ যদি কোন Root node এর মান child অপেক্ষা ছোট অথবা সমান হয় তবে তাকে Min heap বলে।

■ Max Heap ব্যবহার করে Heap sort Algorithm টি
শিখুন। [তথ্য মোগাদিগ ও অনুক্তি বিভাগ-২০১৯, জাতী এন্ডি সুপারজাইন্স]

Max Heap ব্যবহার করে Heap sort Algorithm

Step 1 - Construct a Binary Tree with given list of Elements.

Step 2 - Transform the Binary Tree into Max Heap.
Step 3 - Delete the root element from Max Heap using Heapify method.

Step 4 - Put the deleted element into the Sorted list.
Step 5 - Repeat the same until max Heap becomes empty.

Step 6 - Display the sorted list.

■ Max Heap এ item insert এর নিয়ম:

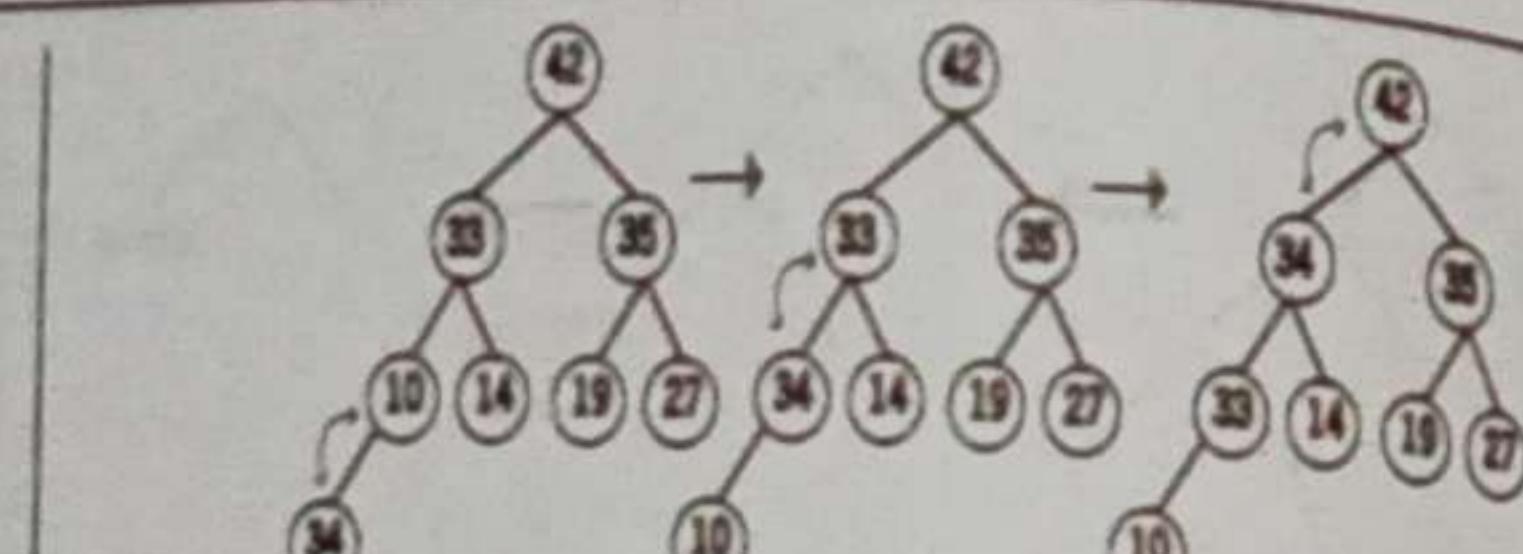
i. Heap- এর শেষে একটি new node create করতে হবে এবং New node এ data insert করতে হবে।

ii. Child node এবং Parent- এর মধ্যে তুলনা করতে হবে।

iii. যদি Child node- এর মান Parent- এর চেয়ে বড় হয় তবে Swap করতে হবে।

iv. 2 এবং 3 নং Step পুনরাবৃত্তি হবে যতক্ষণ পর্যন্ত heap properly দারণ করে।

Ex: Insert 34



■ Max Heap থেকে Root বা যে কোন item delete এর নিয়ম:

i. Root Delete করতে হবে।

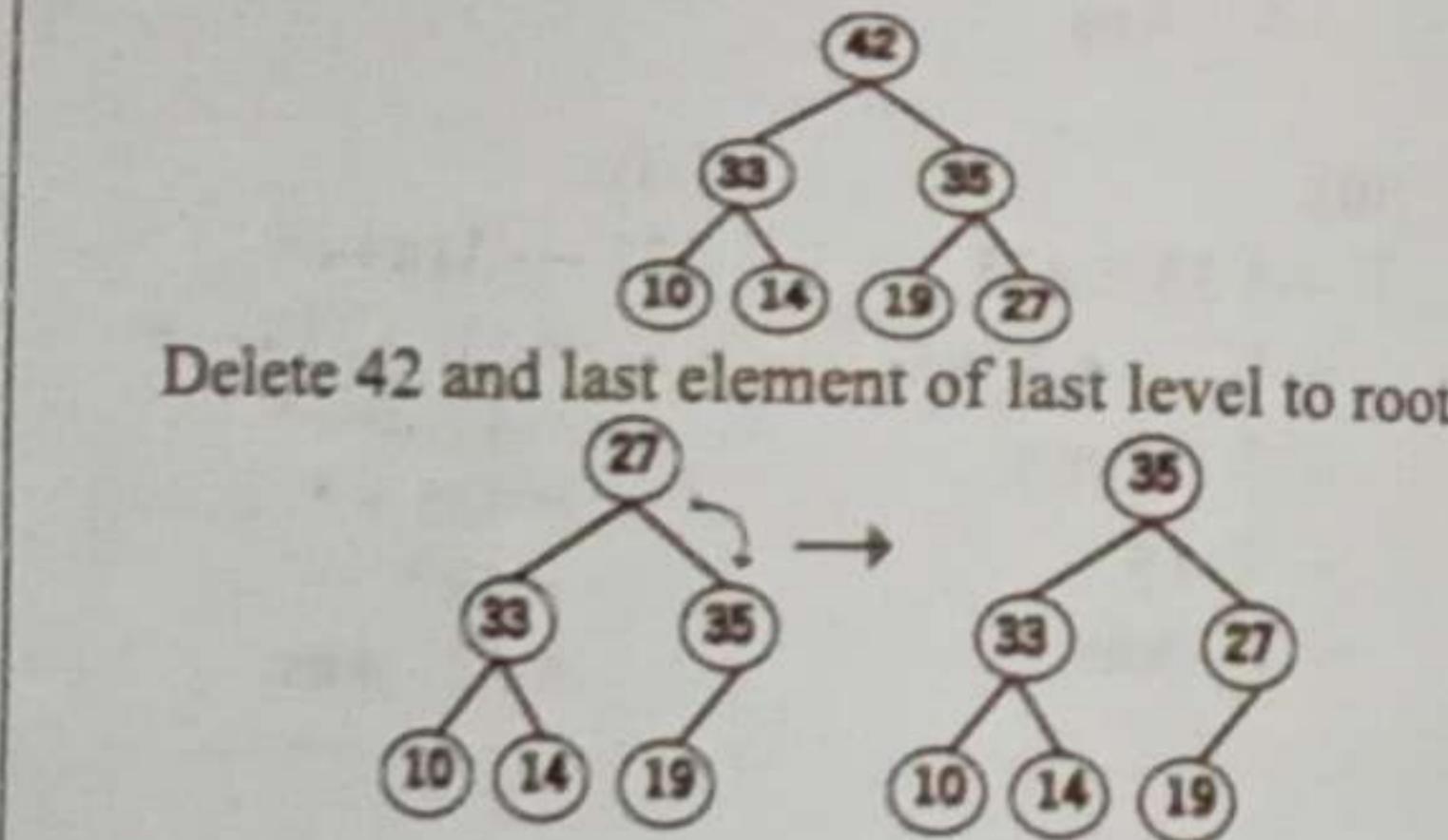
ii. last level- এর last element কে root- এর জাগায় কসাতে হবে।

iii. Child node এবং Parent- এর মধ্যে তুলনা করতে হবে।

iv. যদি Child node- এর মান Parent- এর চেয়ে বড় হয় তবে Swap করতে হবে।

v. 3 এবং 4 নং step পুনরাবৃত্তি হবে যতক্ষণ পর্যন্ত heap properly দারণ করে।

Ex:



■ Heap এর সূবিধা:

1. Heap টৈরী ও সংরক্ষণ করা সহজ কারণ heap একটি Binary tree।

2. Heap কে সংরক্ষণ করতে linked storage- এর প্রয়োজন নেই, সাধারণ Array হলোই চলে।

■ Heap- এর ব্যবহার:

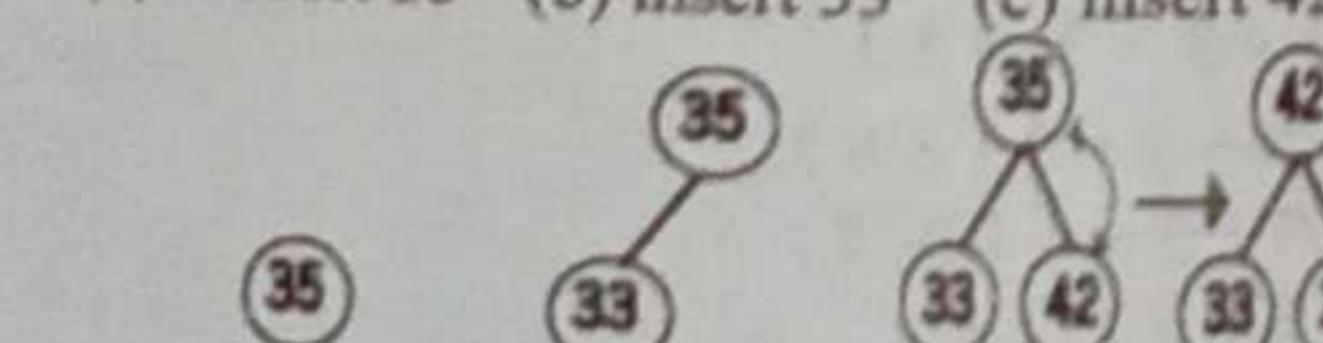
1. স্রোত ব্যৱহাৰ ও সূচনা উপাদান search কৰাৰ জন্য।

2. Advance data structure- এ ব্যবহাৰ কৰা হয়।

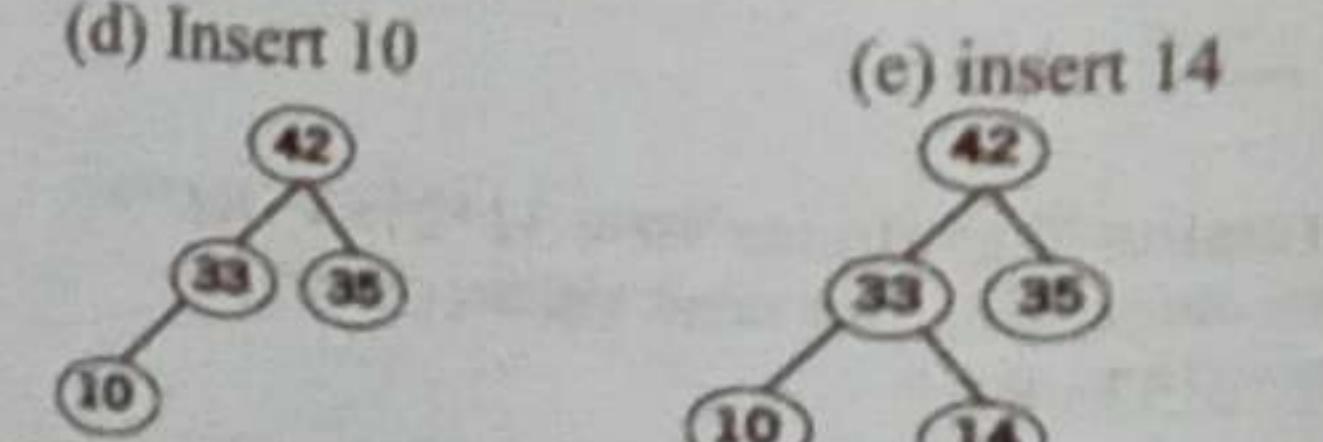
প্র ১. 35, 33, 42, 10, 14, 19, 27, 44, 26, 31 হাবা max Heap গঠন কৰ।

Solution:

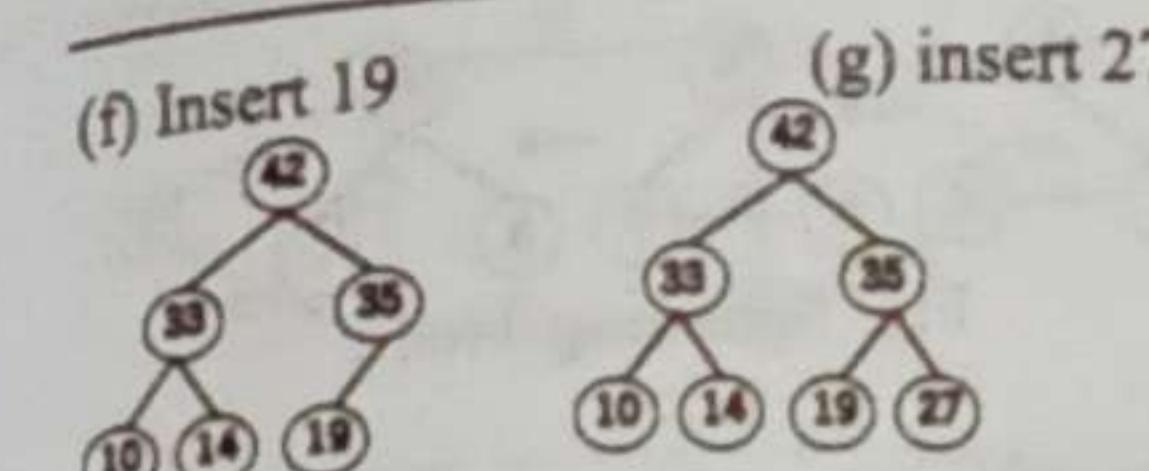
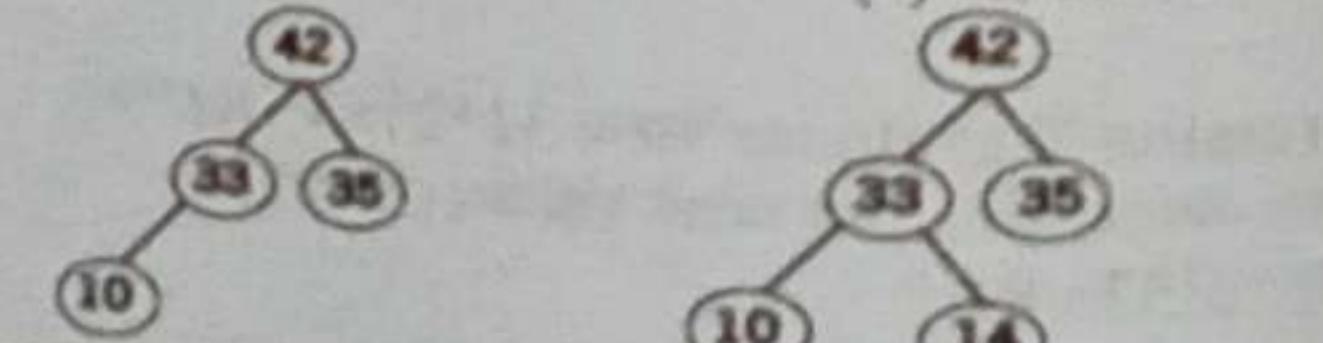
(a) Insert 35 (b) insert 33 (c) insert 42



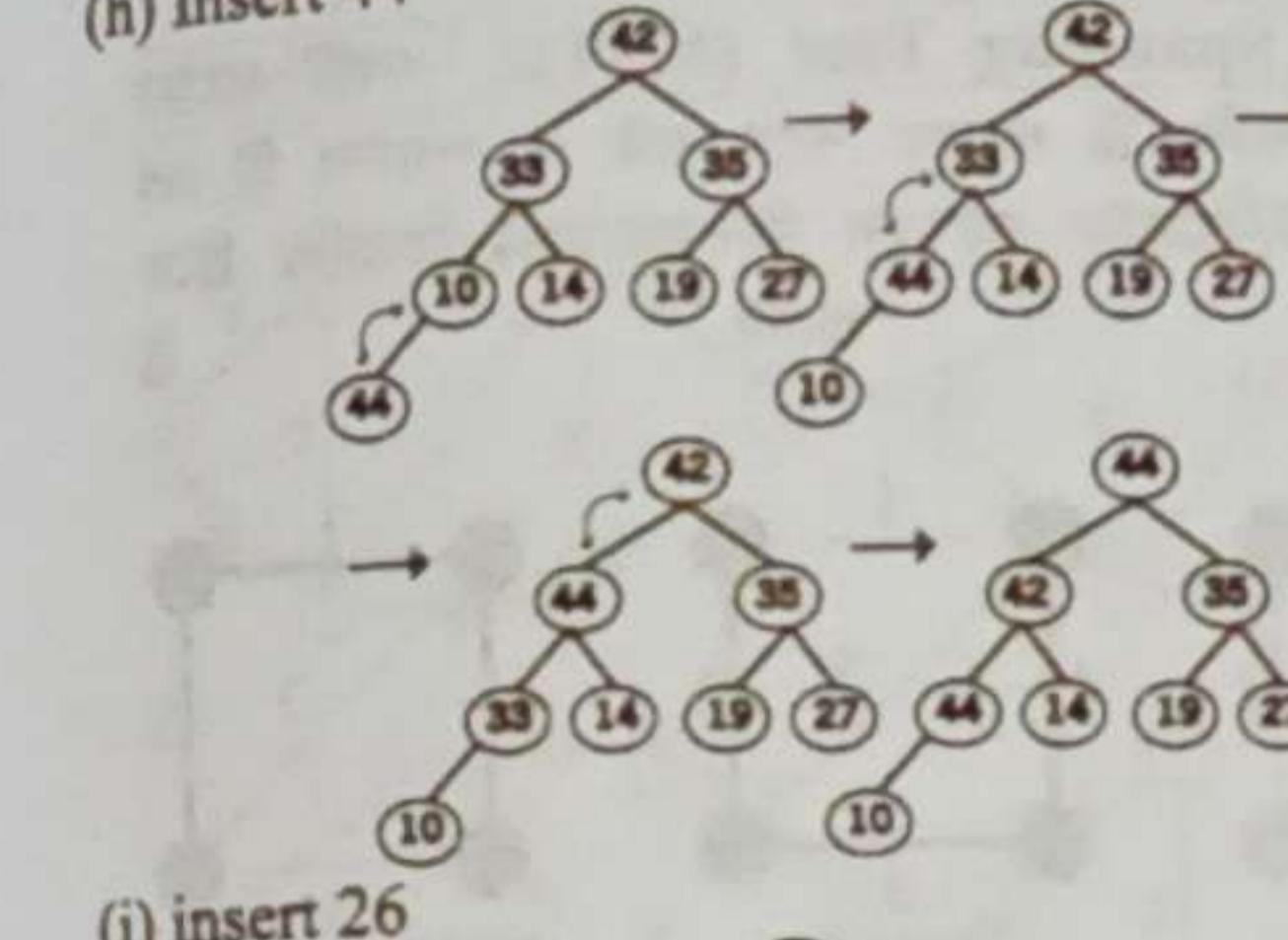
(d) Insert 10



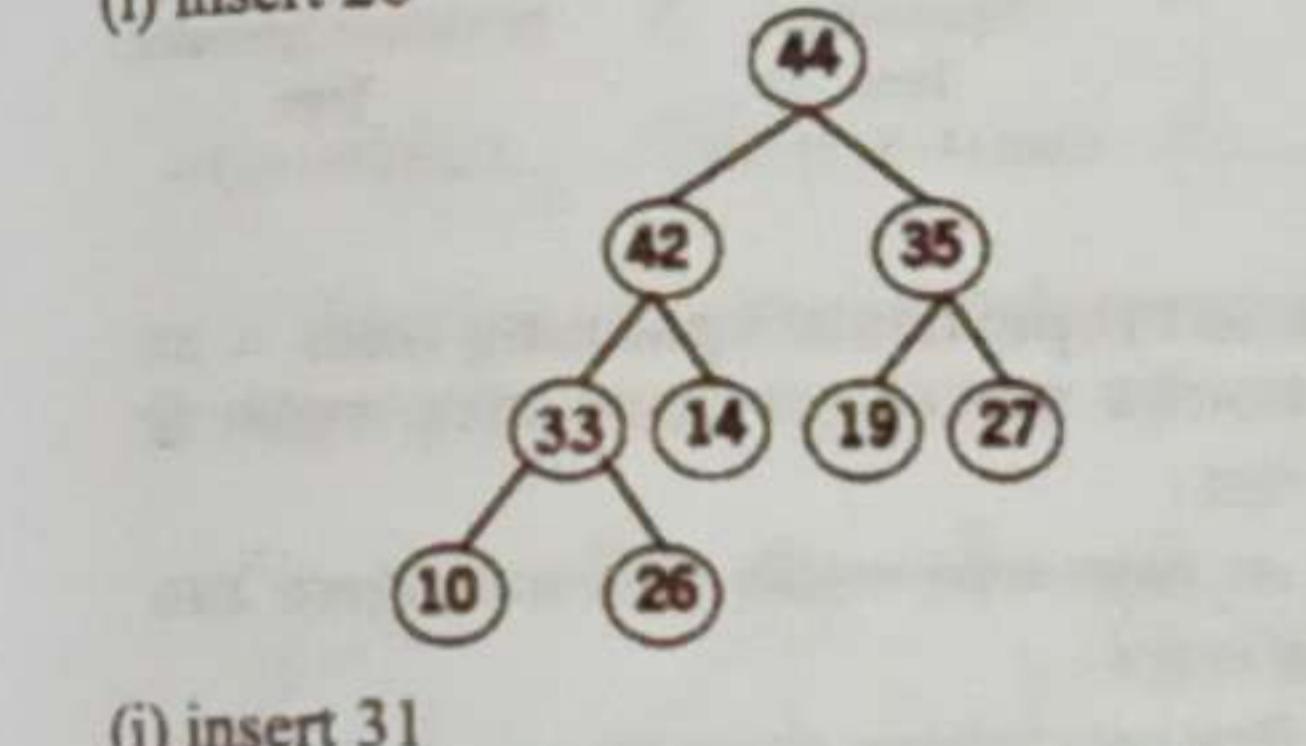
(e) insert 14



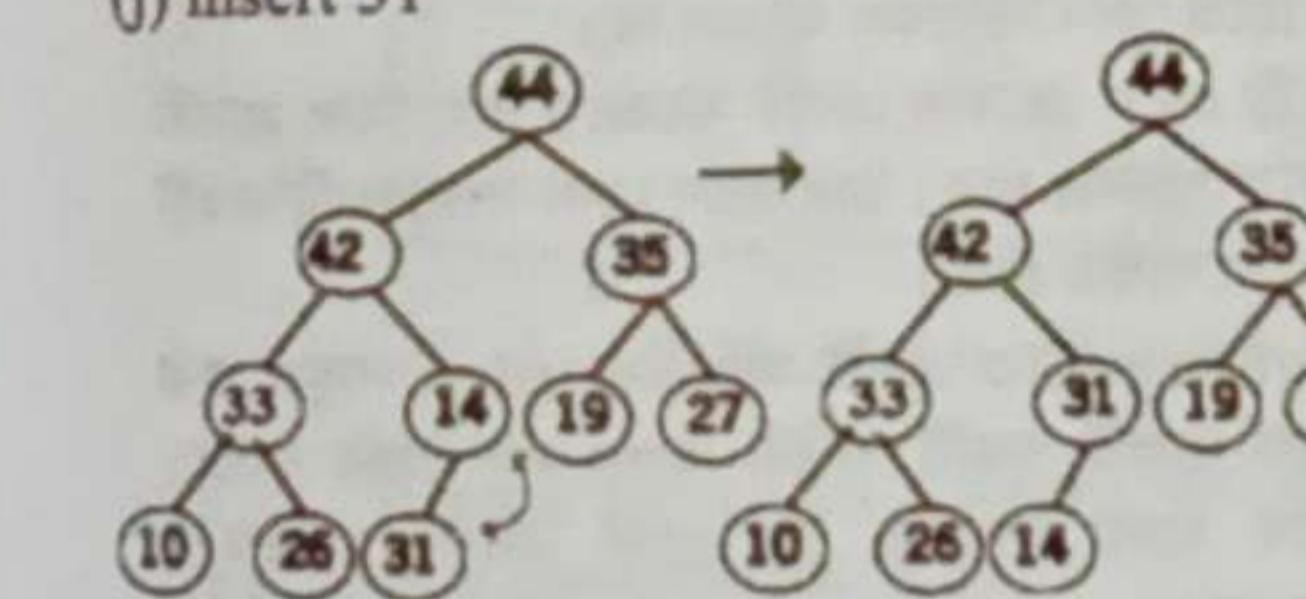
(h) Insert 44



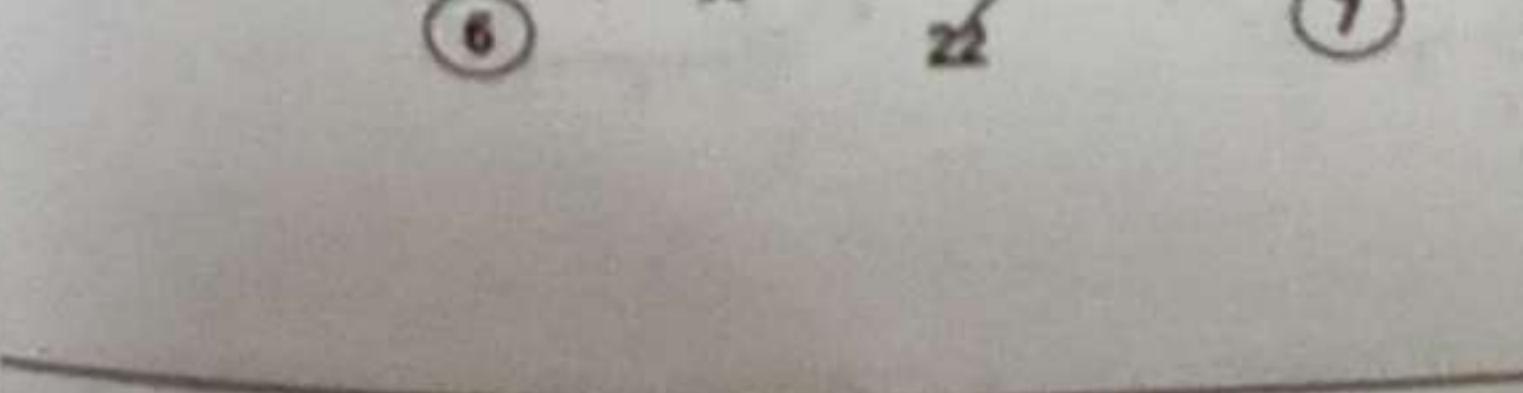
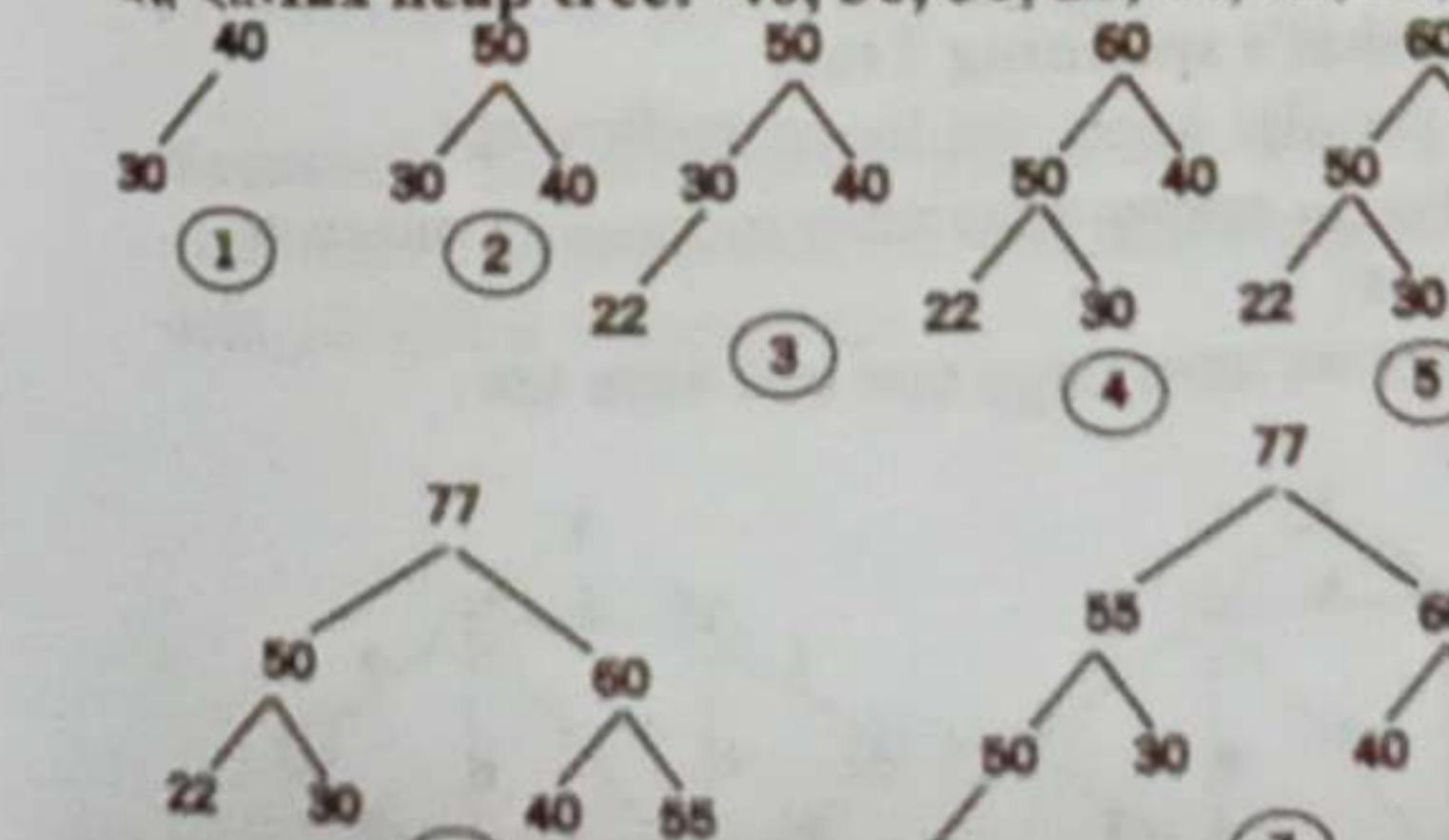
(i) insert 26



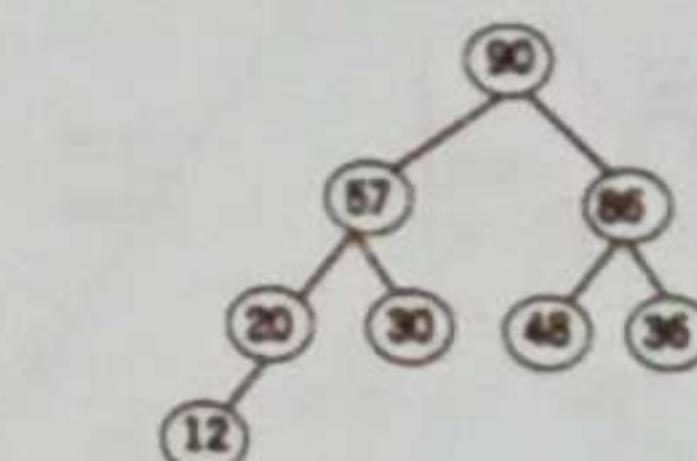
(k) insert 44



প্র ২. Max heap tree: 40, 30, 50, 22, 60, 55, 77, 55



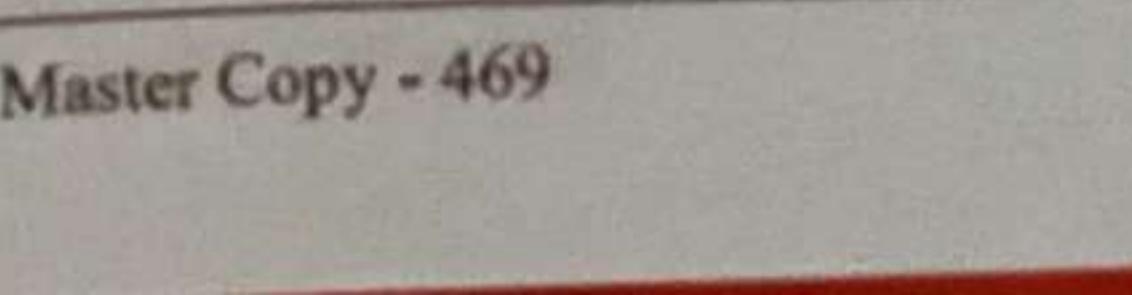
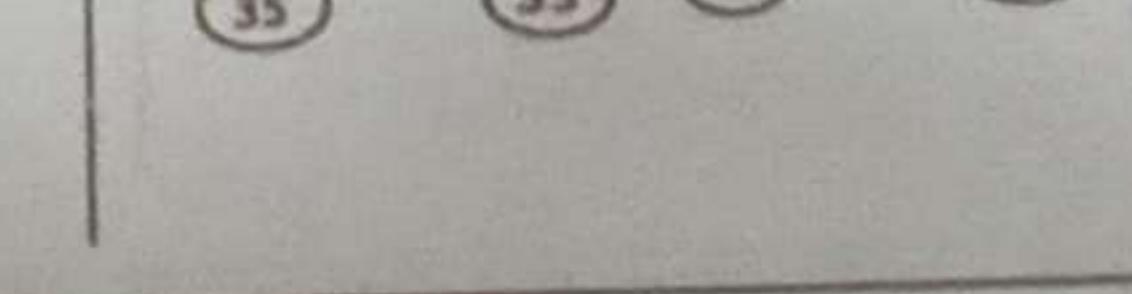
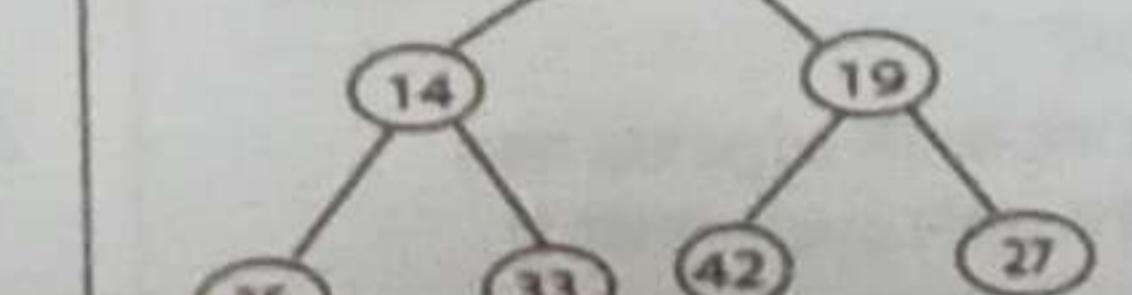
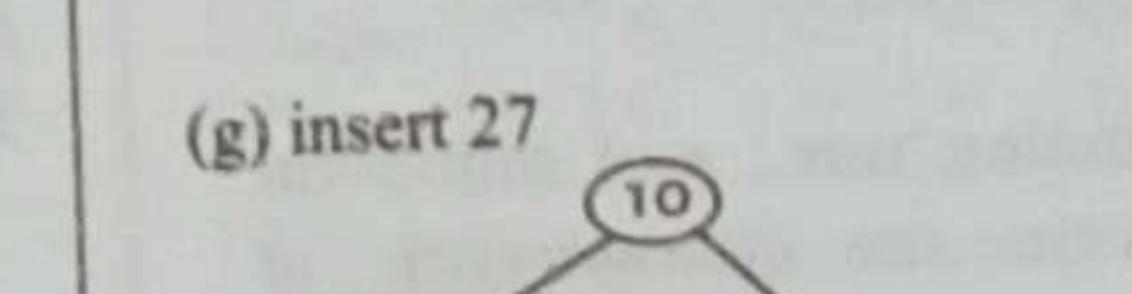
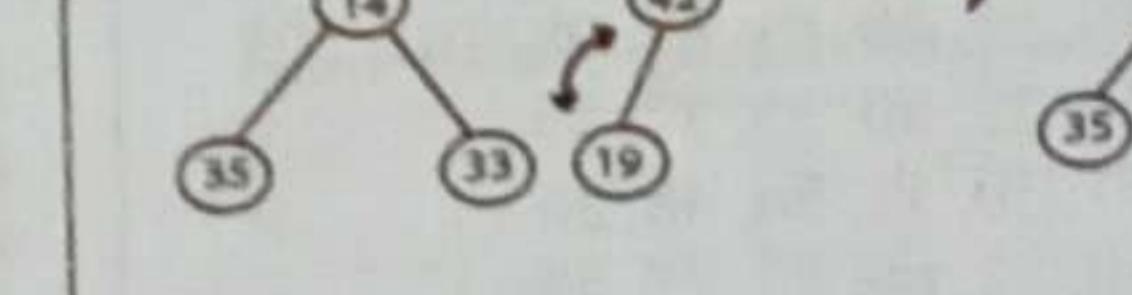
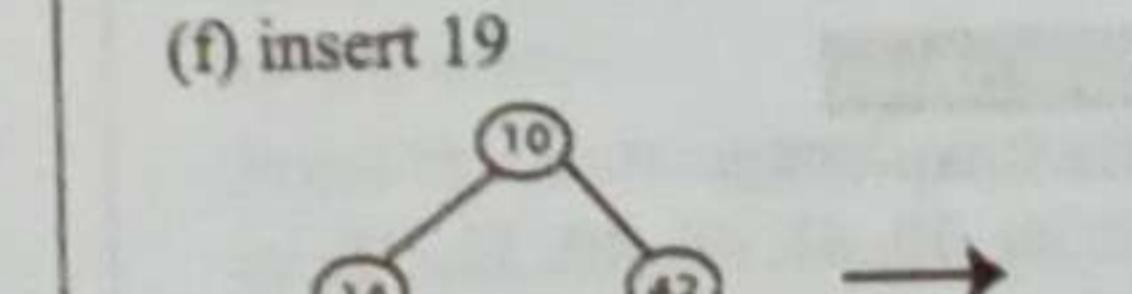
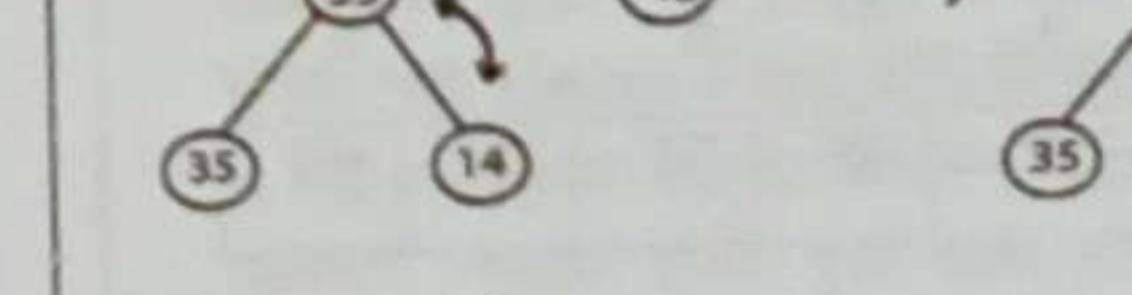
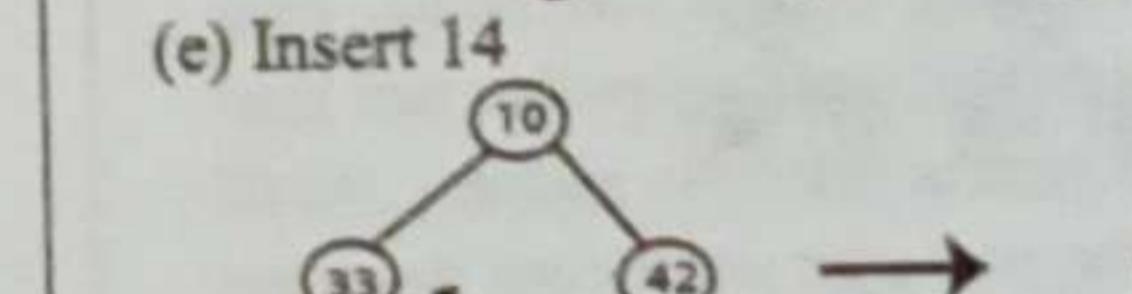
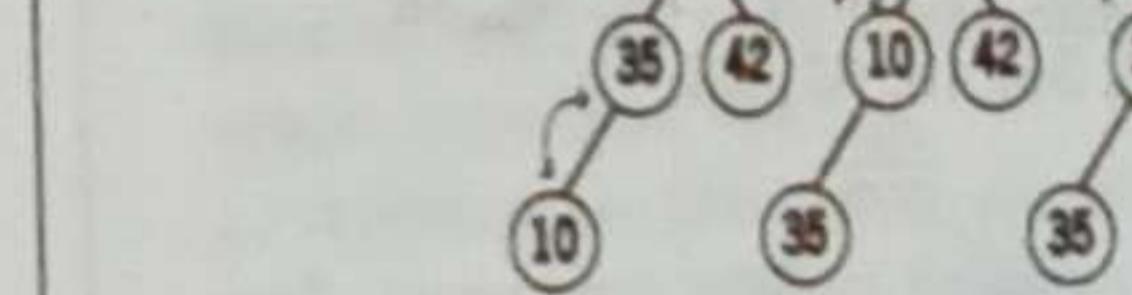
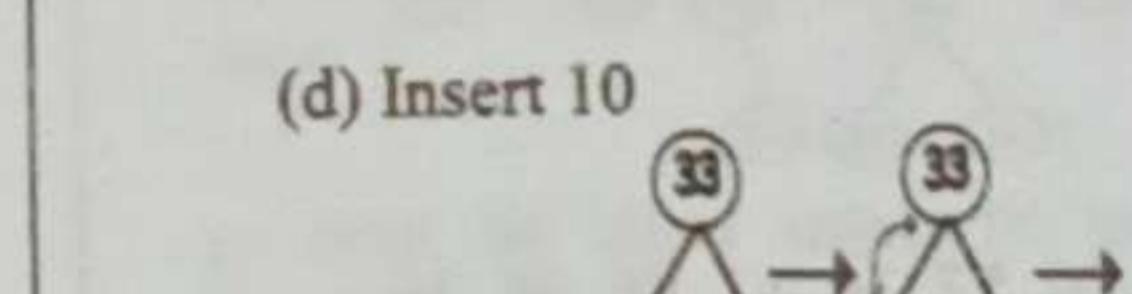
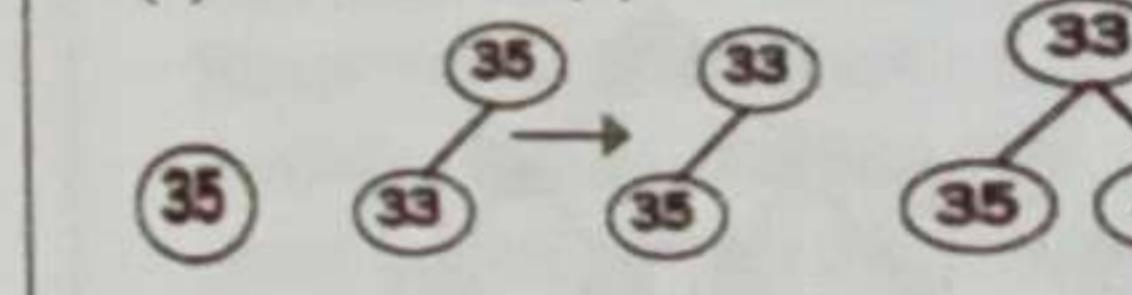
প্র ৩. Max heap tree: 90, 20, 57, 30, 12, 86, 48, 36
Solution:

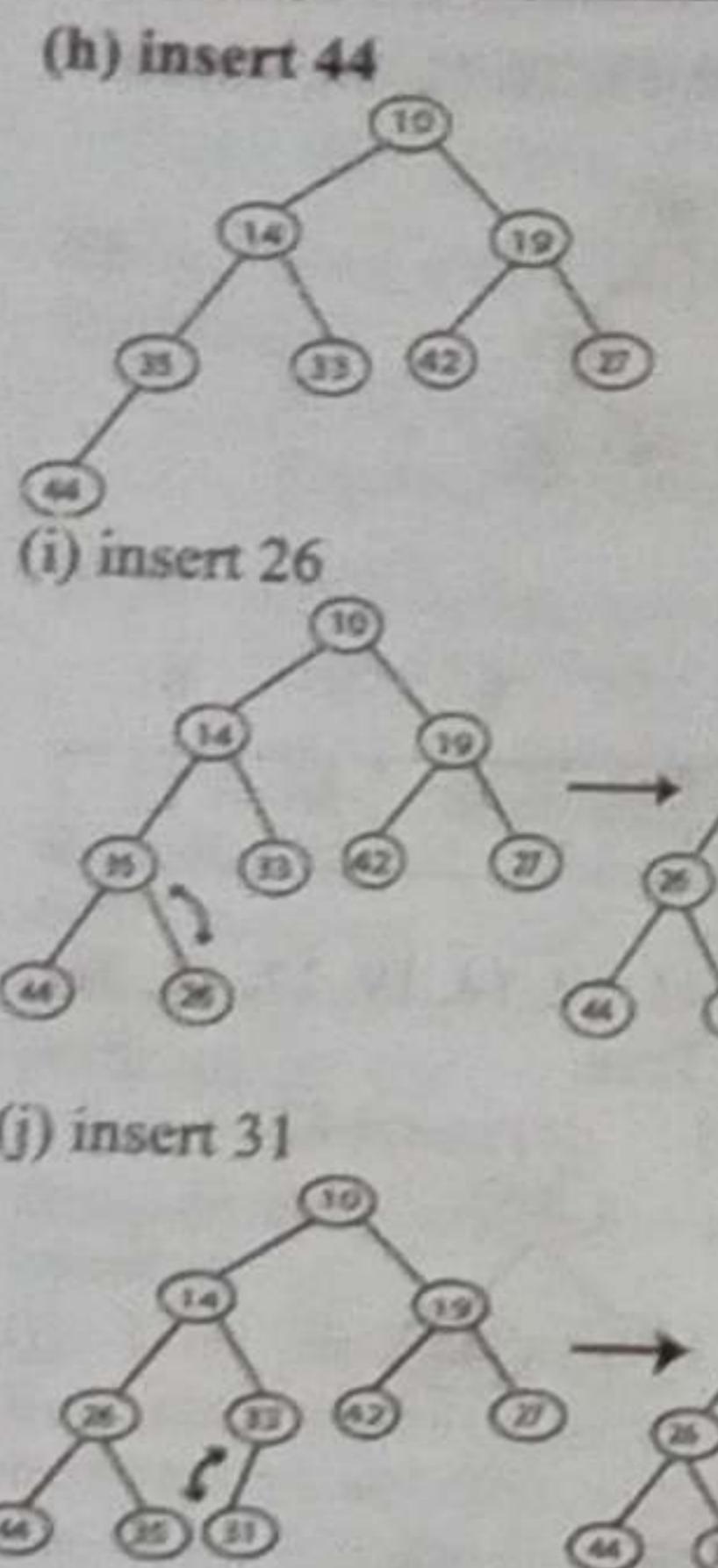


1	2	3	4	5	6	7	8
90	57	86	20	30	48	36	12

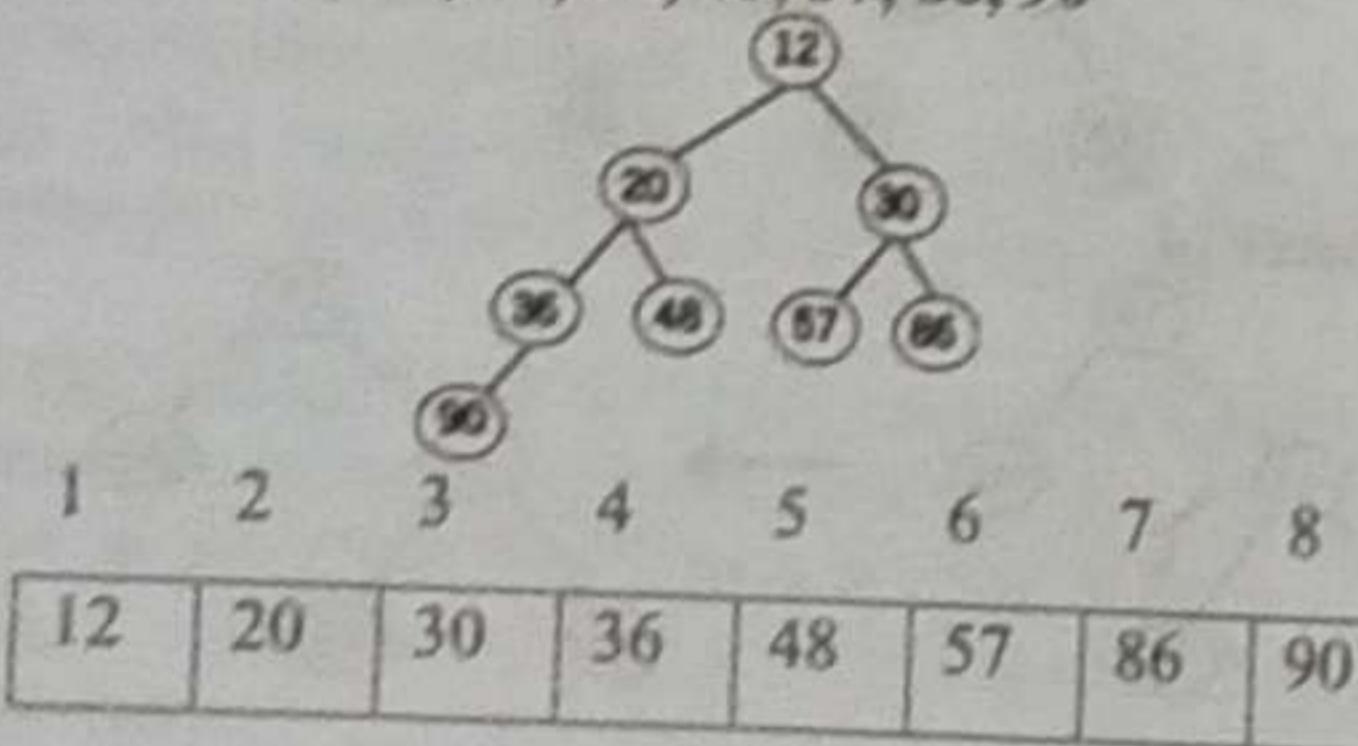
প্র ৪. Given data: 35, 33, 42, 10, 14, 19, 27, 44, 26,
and 31 make MIN Heap Tree.

(a) Insert 35 (b) insert 33 (c) insert 42





প্রশ্ন ১. Min heap: 90, 20, 57, 30, 12, 86, 48, 36
= 12, 20, 30, 36, 48, 57, 86, 90



Self Study

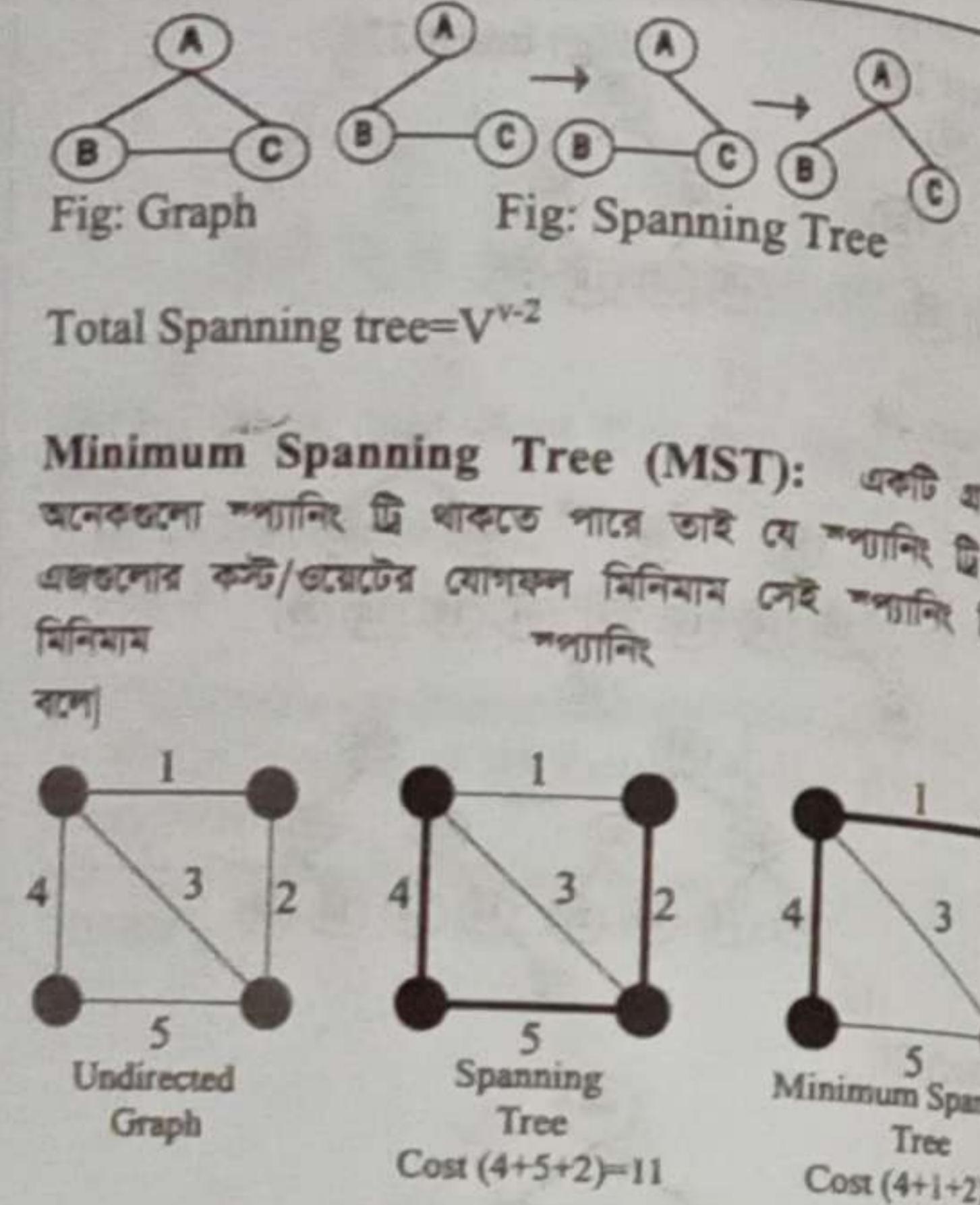
প্রশ্ন ১. Max heap & Min heap দেখাই

- 5, 15, 18, 20, 30, 41, 44, 49, 55, 97
- 28, 30, 50, 55, 60, 61, 62, 98, 101, 205
- 40, 30, 50, 22, 60, 55, 77, 55
- 90, 20, 57, 30, 12, 86, 48, 36
- 44, 30, 50, 22, 60, 55, 77, 53

প্রশ্ন ২. What is spanning tree and minimum spanning tree? Write the characteristic of spanning tree? [BCS-36]

Spanning tree: সবচেয়ে কম সংখ্যক edge দ্বারা সকল Vertex covered করতে হবে। কোন cycle থাকবে না। Disconnected থাকবে না।

Ex:



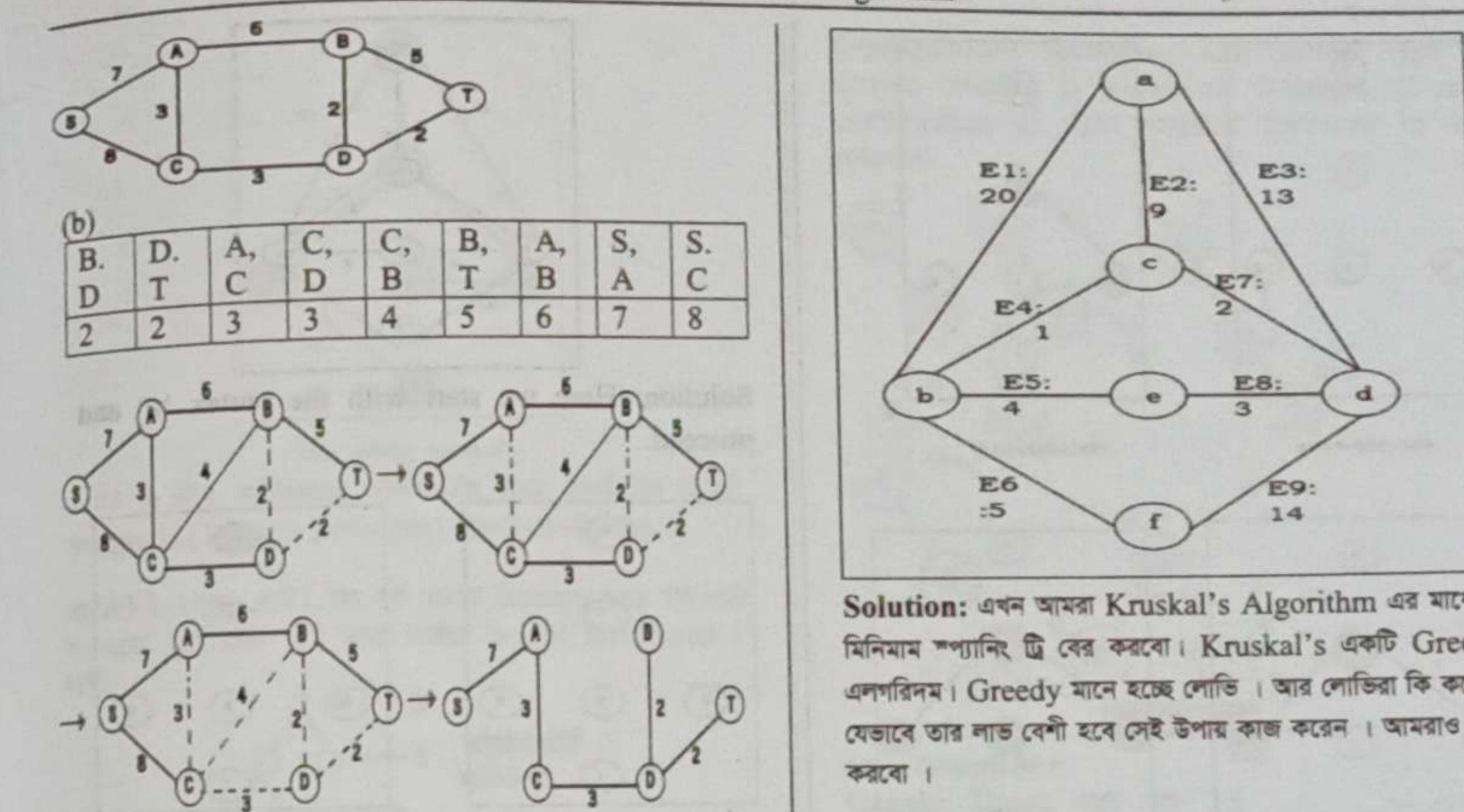
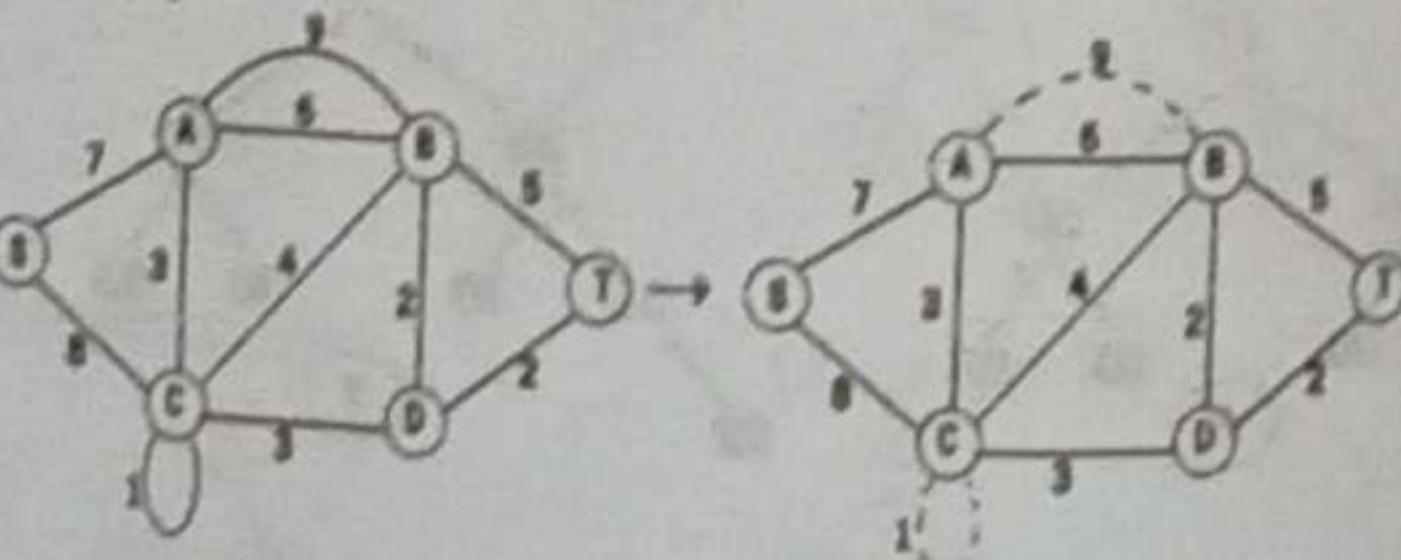
Characteristic/Properties of spanning tree:

- একটি কানেক্টেড গ্রাফ G তে একের অধিক স্প্যানিং ট্ৰি থাকতে পারে।
- গ্রাফ G এর স্বাক্ষৰ সকল স্প্যানিং ট্ৰিতে সমান সংখ্যক ইজে ও ভার্টেক্স থাকবে।
- স্প্যানিং ট্ৰিতে কোন সাইকেল থাকবে না।
- স্প্যানিং ট্ৰি থেকে যেকোন একটি ইজকে বাদ দিলে একটি বিচ্ছিন্ন গ্রাফে পরিণত হবে। তাৰ মানে হলো স্প্যানিং ট্ৰি একটি মিনিমালি কানেক্টেড গ্রাফ।
- যদি spanning ট্ৰিতে একটি অতিরিক্ত edge সংযুক্ত কৰা হয় তাহলে আছিটি একটি সাক্ষীত অথবা লুপ তৈরি কৰবে। তাৰ মানে হলো স্প্যানিং ট্ৰি হলো একটি ম্যাক্সিমালি এসাইক্লিক গ্রাফ।

Kruskal's spanning Tree:

- সকল parallel edges এবং loops বাদ দিতে হবে।
- সকল edge তলোকে তাদেৱ মানেৱ increasing order- এ সাজাতে হবে।
- সবচেয়ে কম মানেৱ edge তলো যোগ কৰতে হবে।

Ex: (a)



Solution: এখন আমোৱা Kruskal's Algorithm এৰ মাধ্যমে মিনিমাম স্প্যানিং ট্ৰি বেৱ কৰবো। Kruskal's একটি Greedy এলগোরিদম। Greedy মানে হচ্ছে লোভি। আৰু লোভিৰা কি কৰেন? যেভাবে তাৰ লাভ বেশী হবে সেই উপায় কাজ কৰেন। আমোৱা তাই কৰবো।

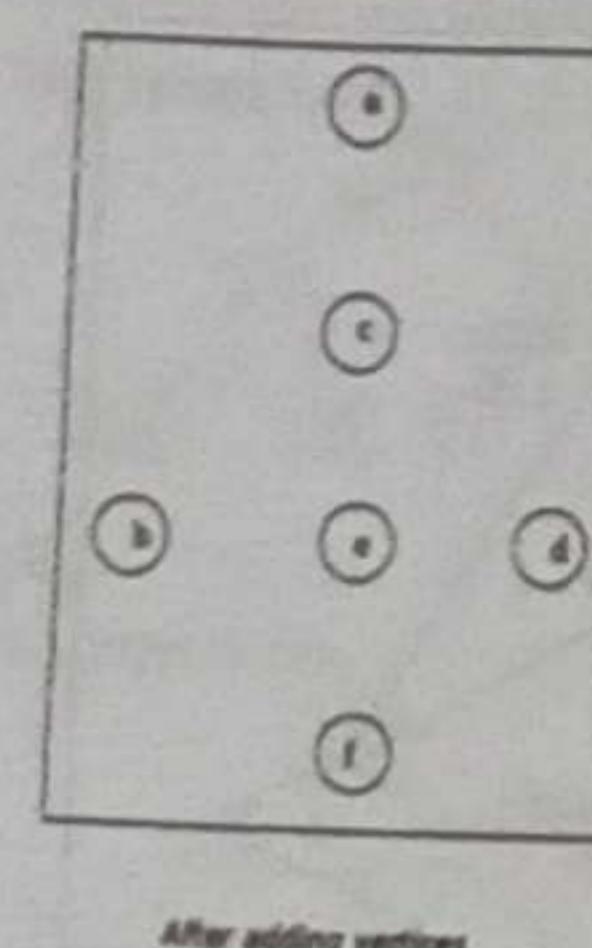
১। প্ৰথমে মূল শাখাৰে এজন্তোকে কস্টেৱ / ওয়েটেৱ ভিত্তিতে ইনক্ৰিঙ্গ অৰ্ডাৱে সেট কৰবো। যাতে কৱে আমোৱা মিনিমাম কস্টেৱ / ওয়েটেৱ এজন্তোকে পেতে পাৰি।

২। তাৰপৰ আমোৱা চেক কৰবো এই মিনিমাম কস্টেৱ / ওয়েটেৱ এজন্তো নেওয়াৰ ফলে আমাদেৱ ট্ৰিতে কোন সাইকেল/লুপ তৈৰি হয়ে যাচ্ছে কিনা। যদি লুপ/সাইকেল তৈৰি হয় তাহলে সেই এজন্তো নিবোনা। অন্যথায় সেই এজন্তো আমাদেৱ মিনিমাম স্প্যানিং ট্ৰি তৈৰি মুক্ত কৰবো। এই লুপ/সাইকেল আছে কিনা সেটা আমোৱা খুব সহজেই চেক কৰতে পাৰি Union Find ডাটা স্ট্ৰাকচাৰ ব্যৱহাৰ কৰে।

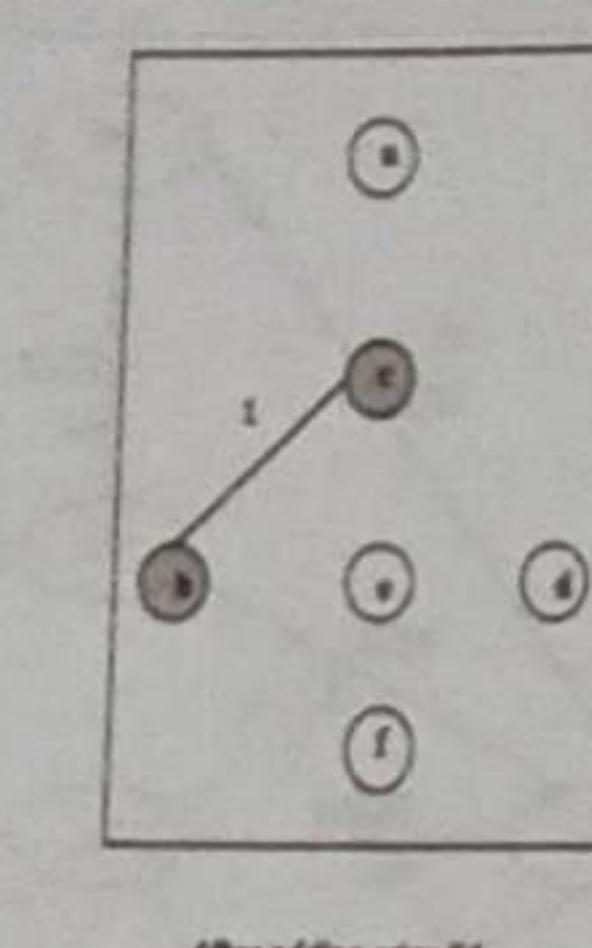
৩। ট্ৰি এৰ সংজ্ঞা অনুসৰে যতক্ষণ না $n-1$ পৰিমাণ এজ আমাদেৱ মিনিমাম স্প্যানিং ট্ৰি তৈৰি মুক্ত হচ্ছে ততক্ষণ আমোৱা একই ভাবে আগামতে থাকবো।

From the above graph we construct the following table and we will rearrange the table in ascending order with respect to Edge weight –

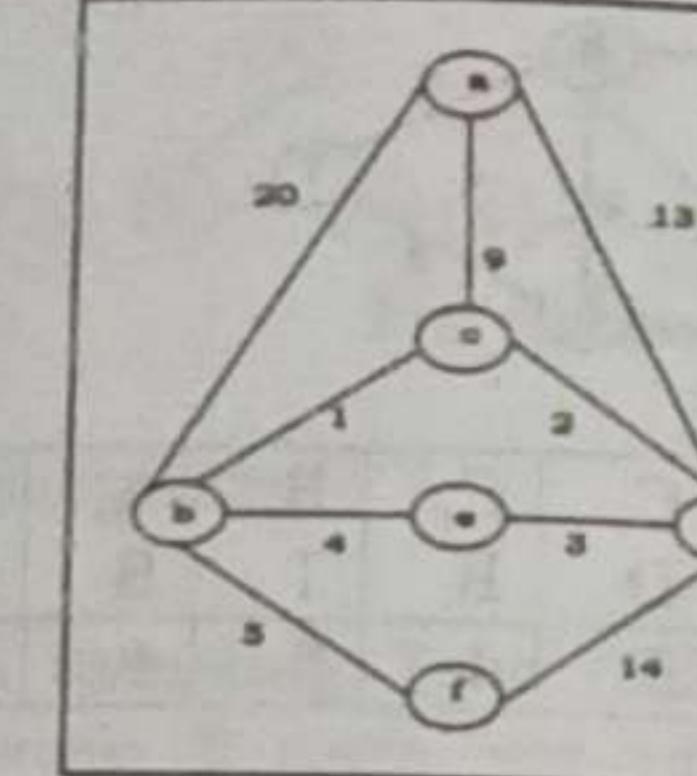
Edge No.	E 4	E 7	E 8	E 5	E 6	E 2	E 3	E 9	E 1
Vertex Pair	(b, c)	(c, d)	(d, e)	(e, f)	(b, f)	(a, b)	(a, d)	(d, f)	(a, b)
Edge Weight	1	2	3	4	5	9	13	14	2



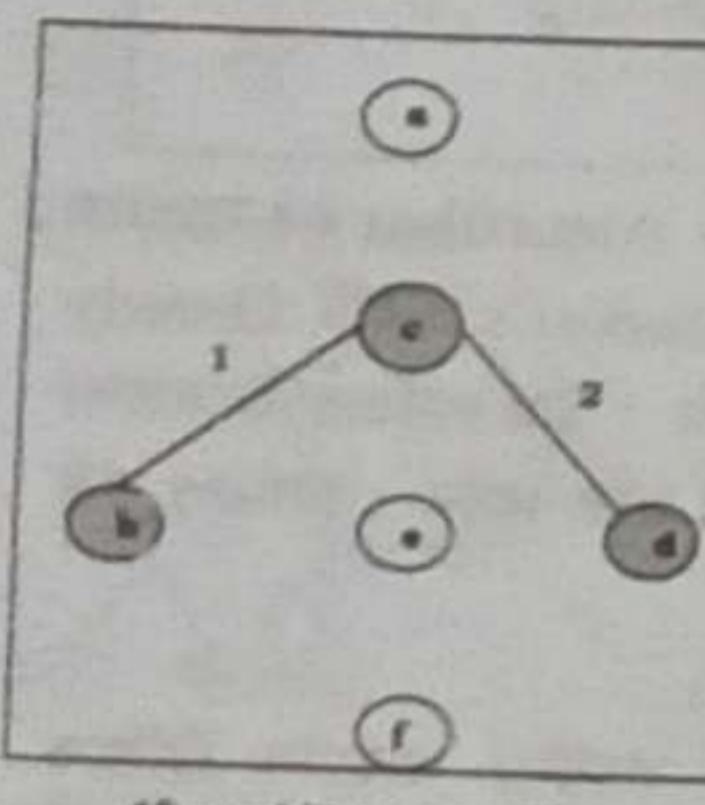
After adding vertices



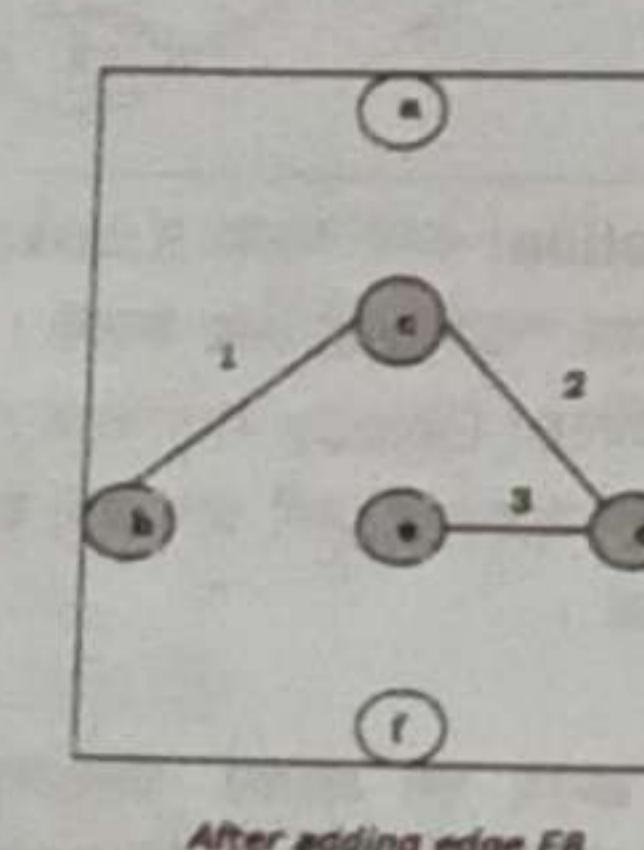
After adding edge E4



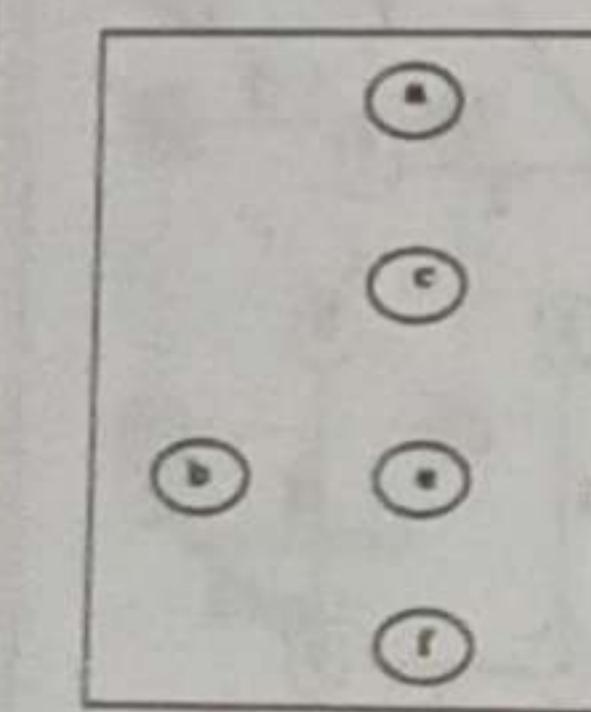
Solution: Here we start with the vertex 'a' and proceed.



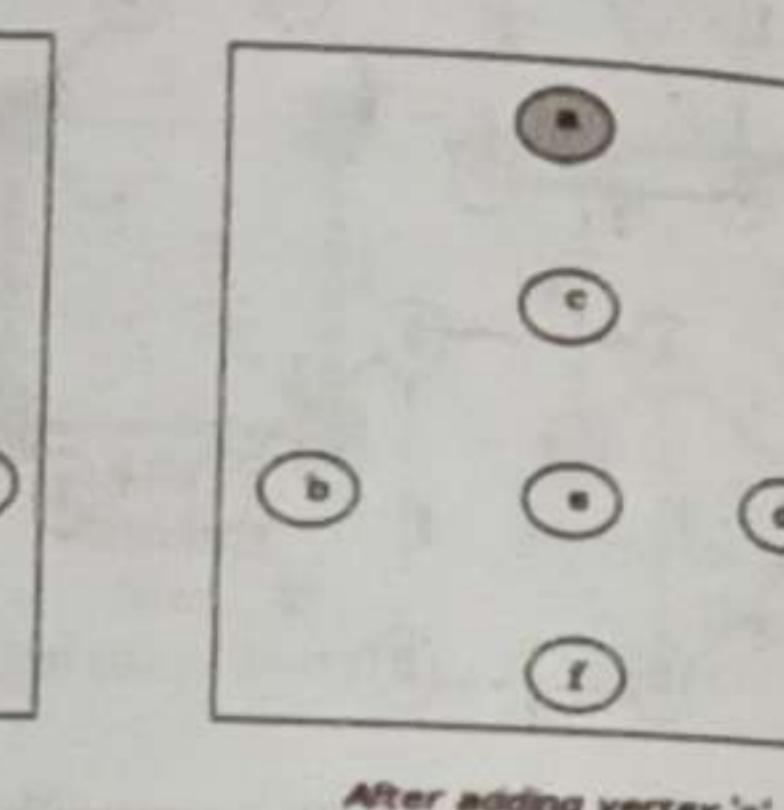
After adding edge E7



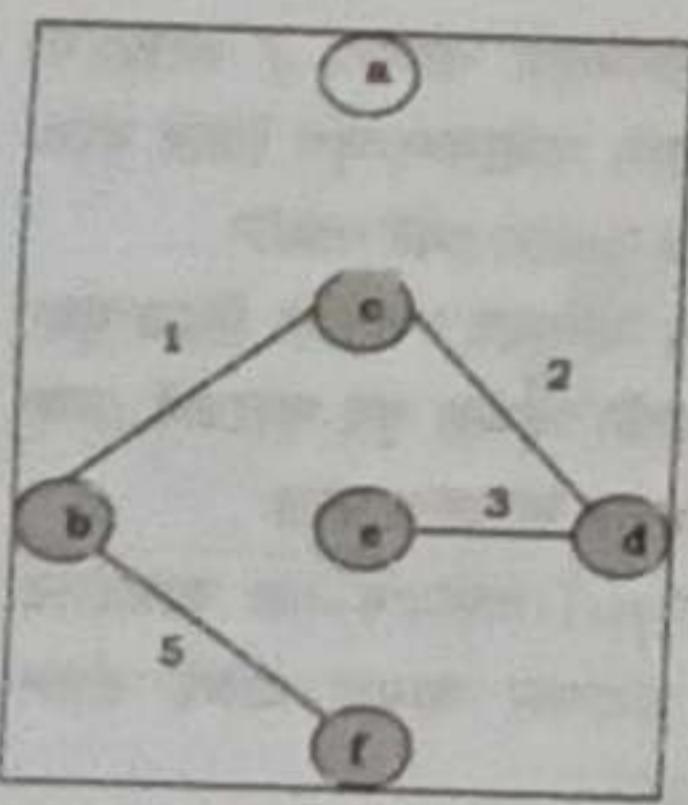
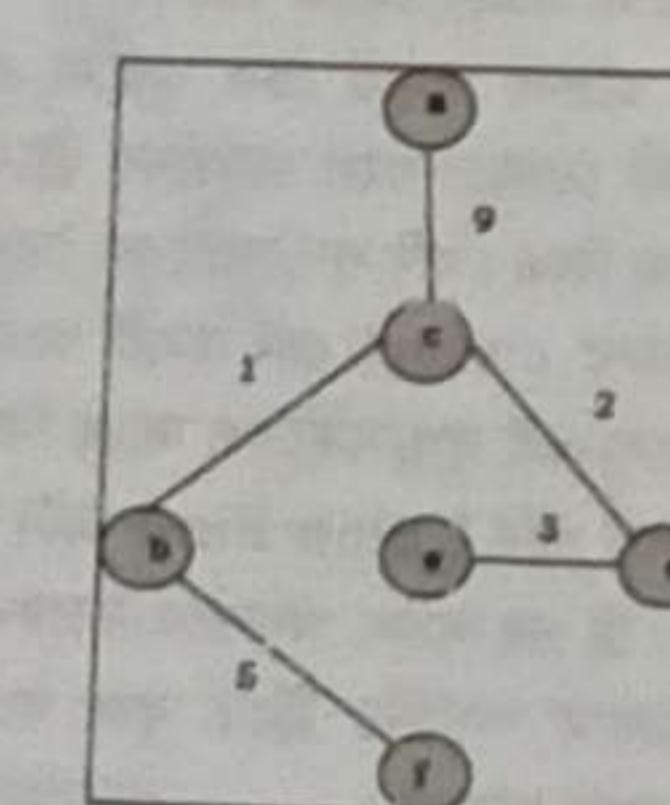
After adding edge E8



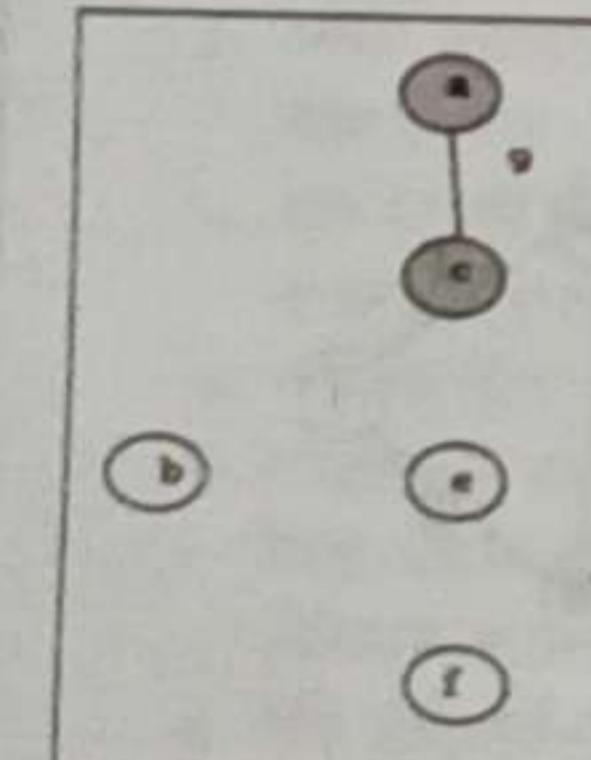
No vertices added



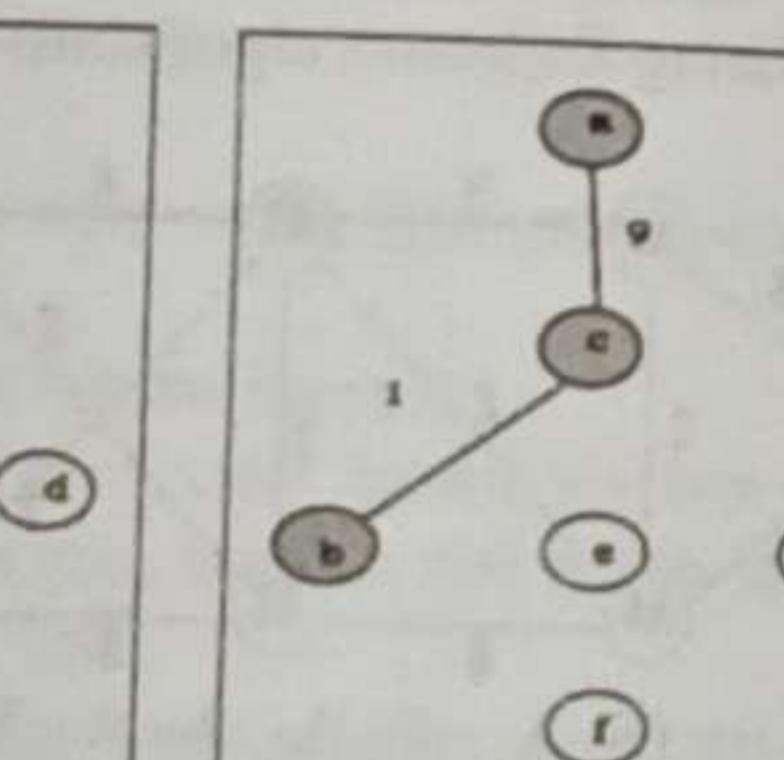
After adding vertex 'a'

After adding edge E6
(don't add E5 since it forms cycle)

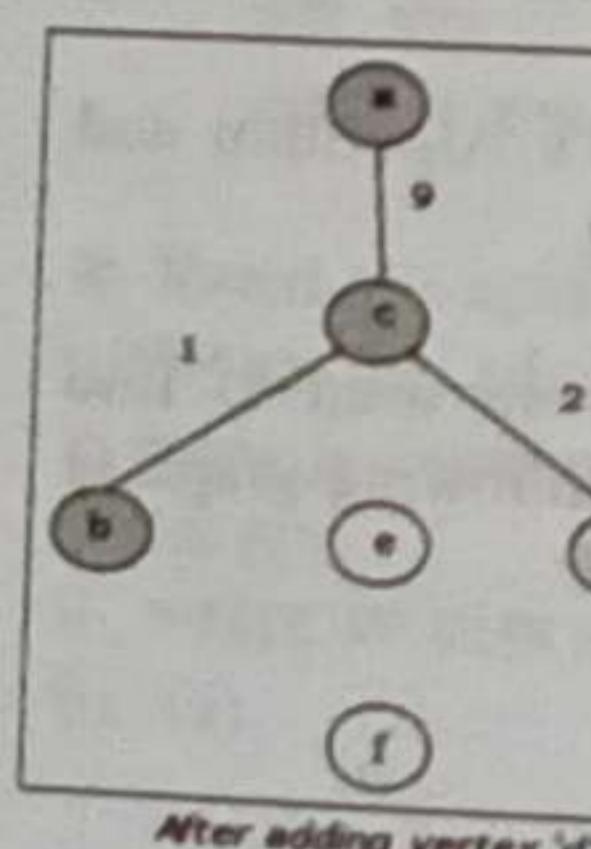
After adding edge E2



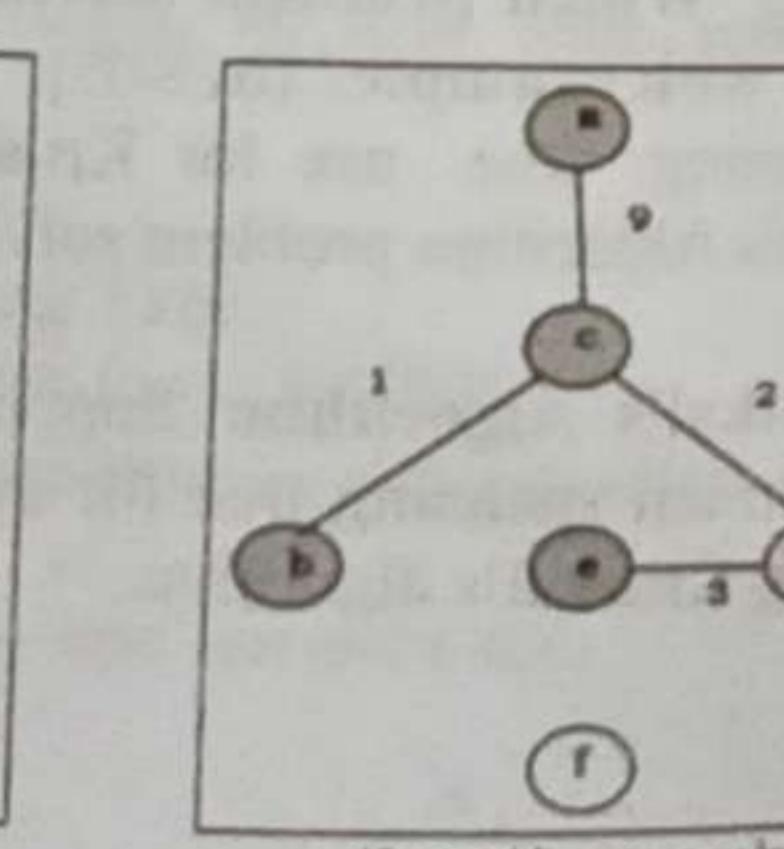
After adding vertex 'c'



After adding vertex 'b'



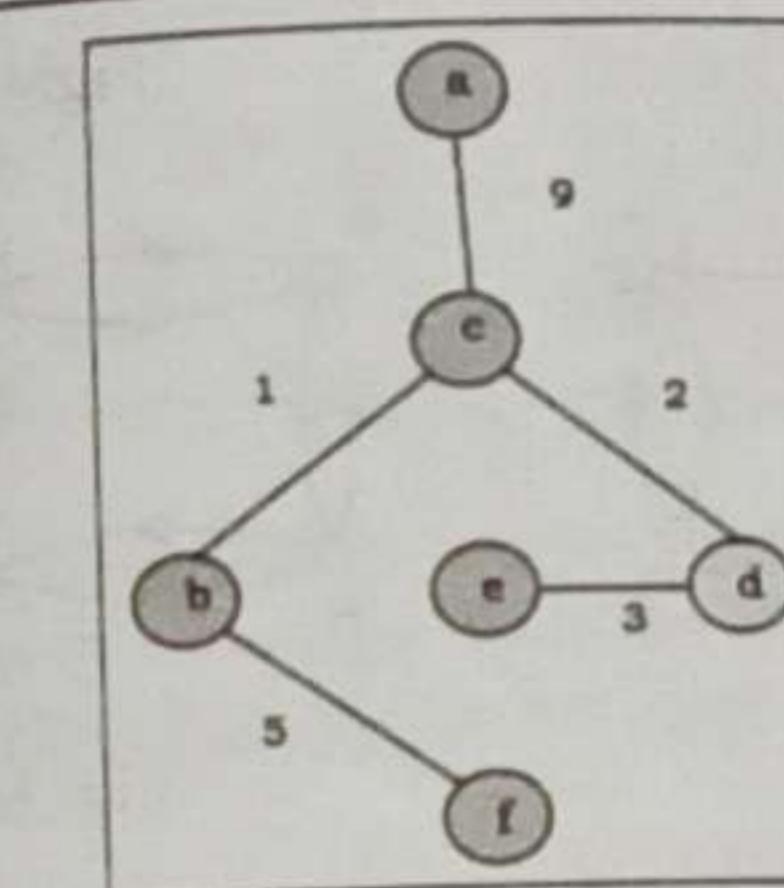
After adding vertex 'd'



After adding vertex 'e'

Since we got all the 5 edges in the last figure, we stop the algorithm and this is the minimal spanning tree and its total weight is $(1+2+3+5+9)=20$ ($1+2+3+5+9=20$).

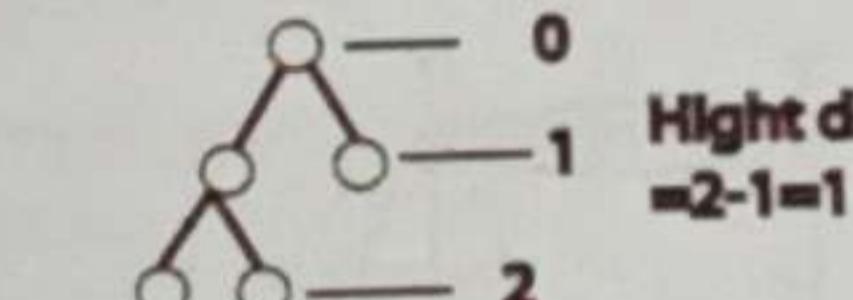
Prim's Algorithm: একটা ভারটেজ থেকে তরু করে এর সাথে কম ঘয়েট ওয়ালা এজ তলো সিরিয়ালি(এলোমেলো নয়) নিয়ে মিনিমাম স্প্যানিং ট্ৰি তৈরি কৰেহেন, এখন আমোৱা প্ৰিমস এলাগিলম ব্যবহাৰ কৰে নিচেৰ খাফ থেকে মিনিমাম স্প্যানিং ট্ৰি বৈৰ কৰাৰো।



After adding vertex 'f'

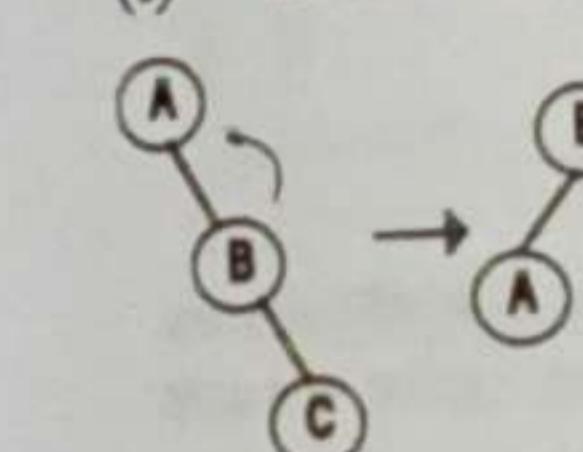
This is the minimal spanning tree and its total weight is $(1+2+3+5+9)=20$ ($1+2+3+5+9=20$).

■ **AVL tree:** AVL হল এক ধৰণেৱ Binary tree যাৰ left এ right Subtree- এৰ মধ্যে সৰোচ height difference 1 হৰে।

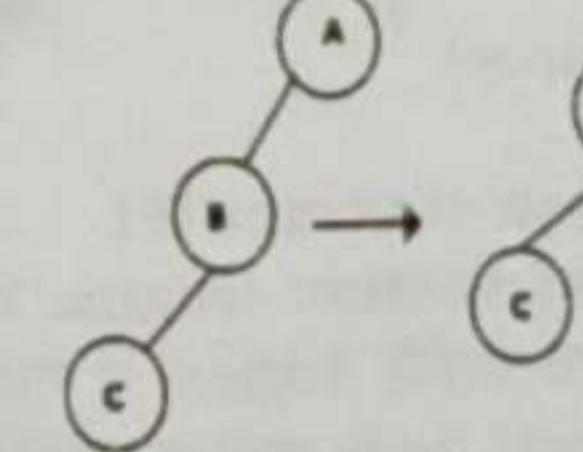


AVL rotation 4 প্ৰকাৰ-

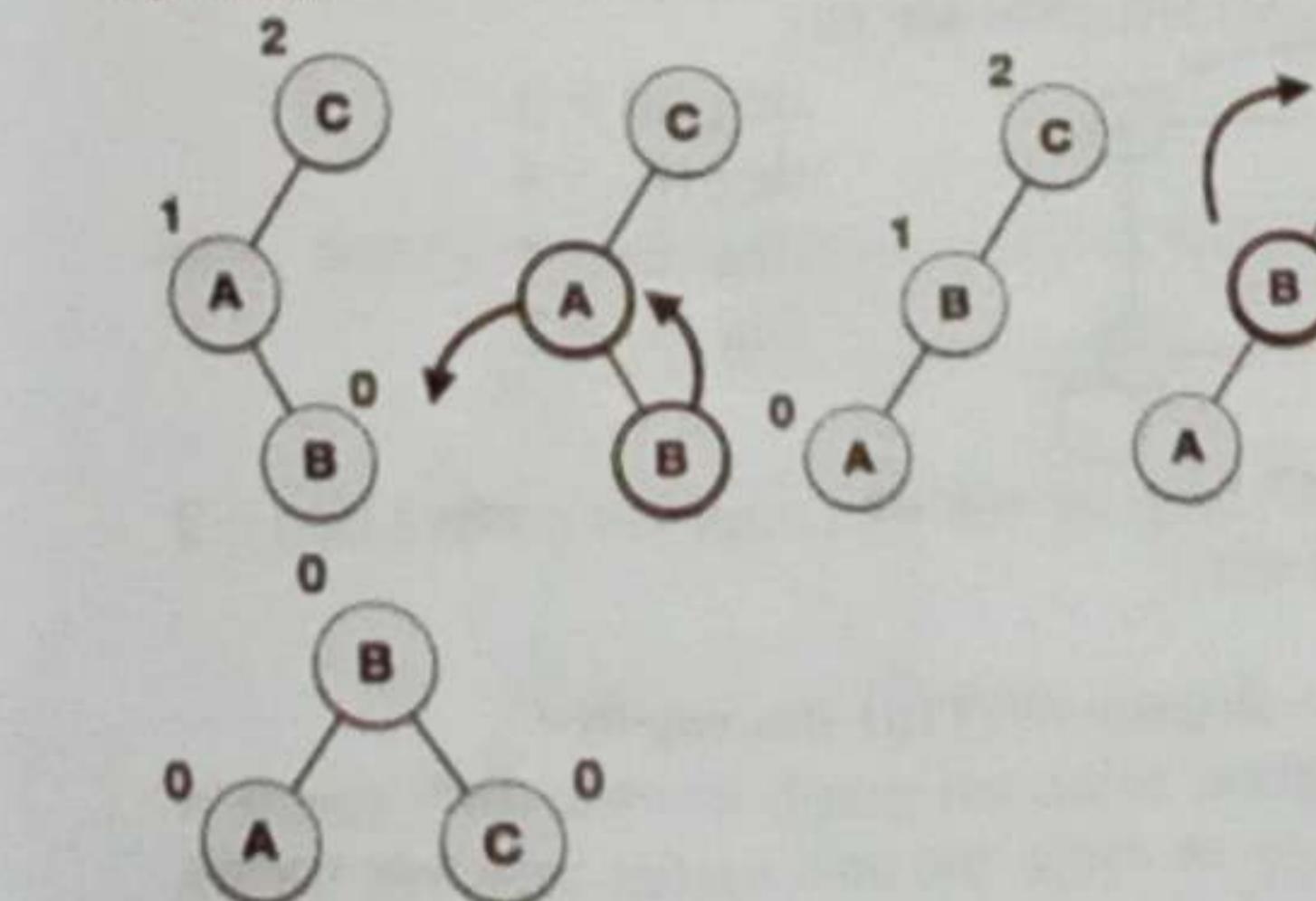
(i) Left rotation:



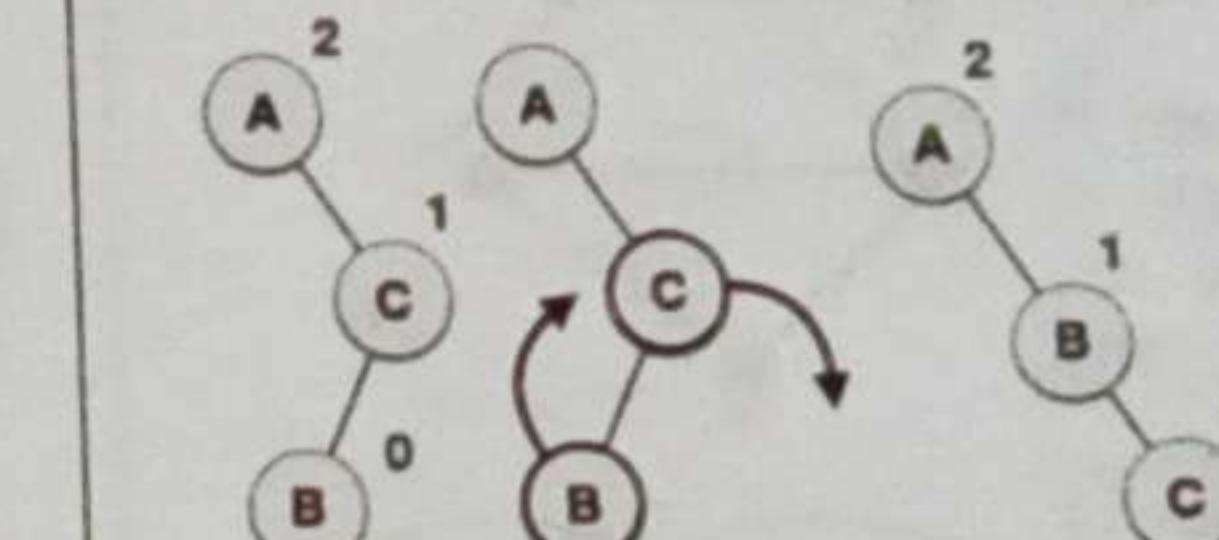
(ii) Right rotation:



(iii) Left-Right Rotation: Let's first check how to perform Left-Right rotation. A left-right rotation is a combination of left rotation followed by right rotation.



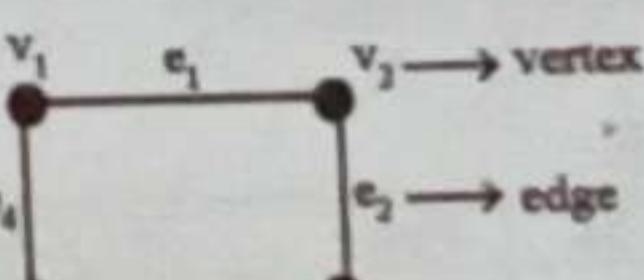
(iv) Right-Left Rotation: The second type of double rotation is Right-Left Rotation. It is a combination of right rotation followed by left rotation.



After adding vertex 'Y'

প্ৰশ্ন 1. Graph কি?

Graph: Graph হলো এক ধৰণেৱ Discrete Structure যাহা Vertices and Edges এৰ সময়ে গঠিত এৰ vertices সমূহ edge বাবা connected থাকে।



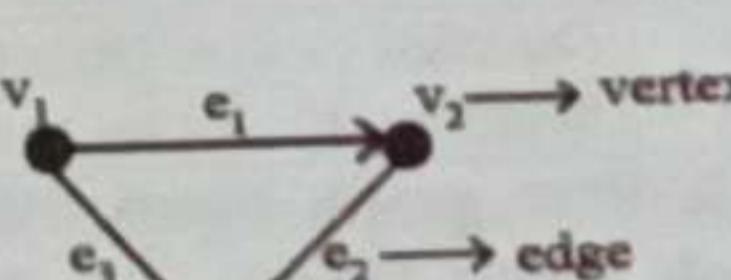
প্ৰশ্ন 2. Graph এৰ প্ৰকাৰতে লিখি?

Graph দুই প্ৰকাৰ : যথা-

- (i). Directed graph
- (ii). Undirected graph.

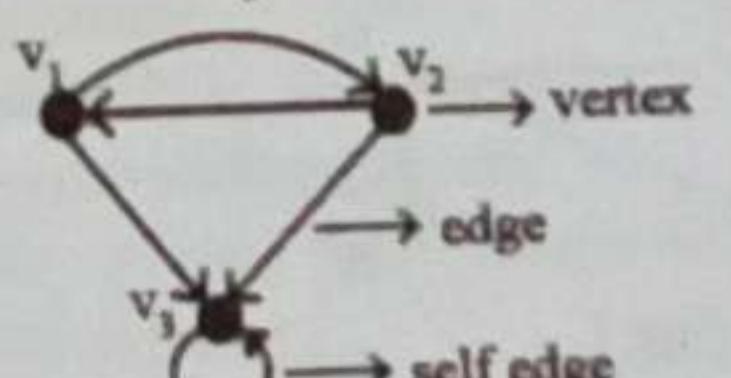
(i). Directed graph

(a) **Directed graph:** যে Graph এৰ direction দেওয়া থাকে তাকে Directed graph বলে।

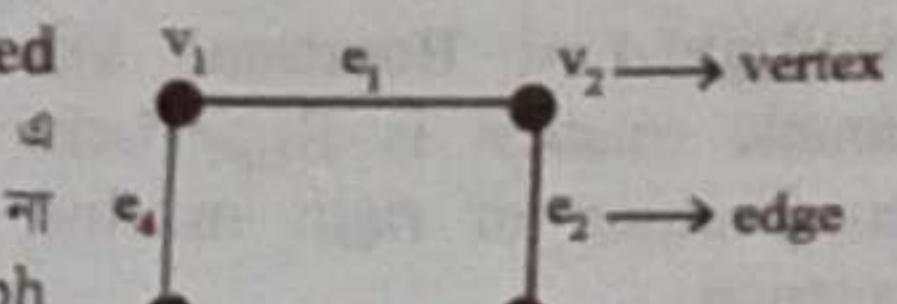


(b) **Directed Multigraph:**

যে directed graph এৰ একটি vertices হতে অন্য vertices এ যাওয়াৰ একাধিক edges থাকে তাকে directed multigraph বলে। এই graph এ self edge থাকতে পাৰে।



(ii). Undirected graph: যে graph এ কোন direction থাকে না তাকে undirected graph বলে।



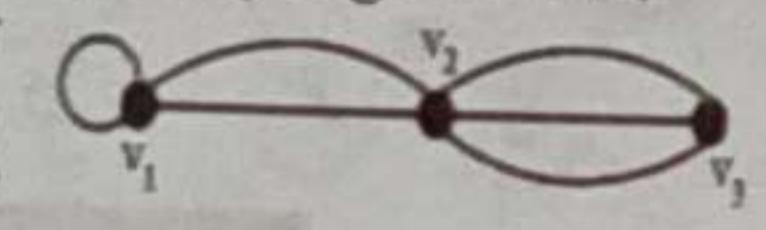
(a) Simple graph: যে undirected graph এ কোন multi edge এবং self loop থাকে না, তাকে simple graph বলে।



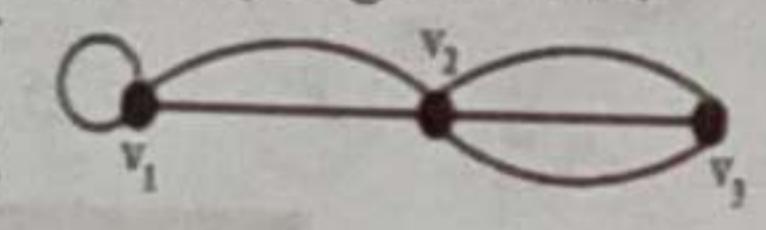
(b) Multigraph: যে undirected graph এ কোন multi edge থাকে কিন্তু কোন self loop থাকে না তাকে multigraph বলে।



[N.B: উক্ত graph এ v_1 হতে v_2 vertex এ যাওয়ার একাধিক (Multi) edge বিদ্যমান।]



(c) Pseudo graph: যে undirected graph এ কোন multi edge এবং self loop বিদ্যমান তাকে Pseudo graph বলে।



NB:

Sl.	Type	Comment
1	Simple Graph	Undirected edges, no parallel edges or loops.
2	Multi Graph	Undirected, no loops but multiedges.
3	Pseudo Graph	Undirected, with loops multiedges.
4	Multigraph	Undirected, No loop but multiedges.
5	Directed graph	Directed, No multiedges but loops.
6	Directed Multigraph	Directed, with multiedges and loops.

প্রশ্ন 3. নিচের graph সমূহকে চিহ্নিত কর এবং undirected graph তাকে Simple graph এ তৈরি করার জন্য কোন edges সমূহকে remove করতে হবে তার set দেবে কর।

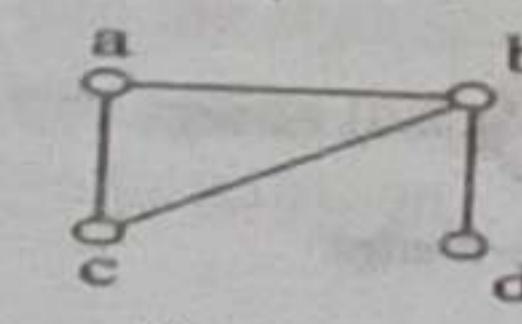


Fig:1

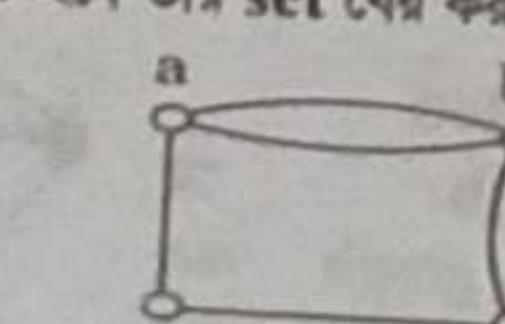


Fig:2

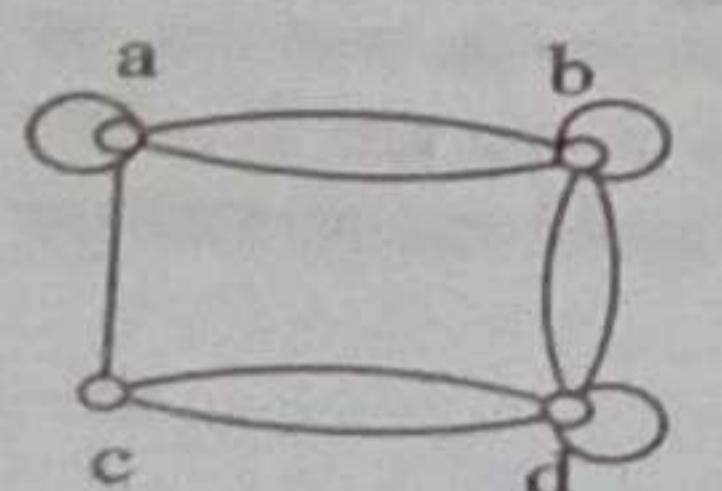


Fig:3

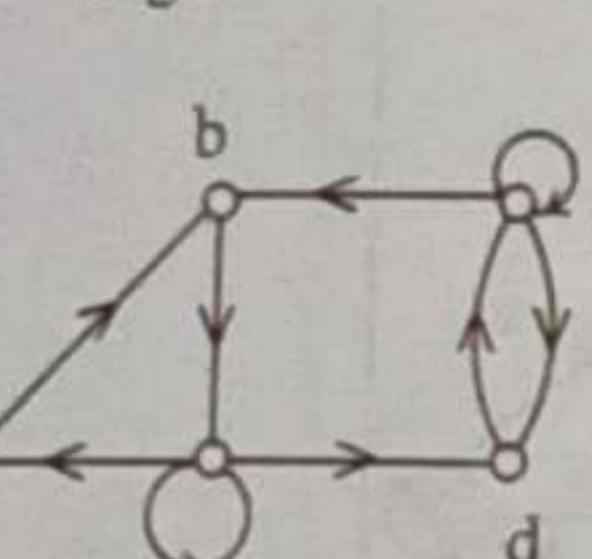


Fig:4

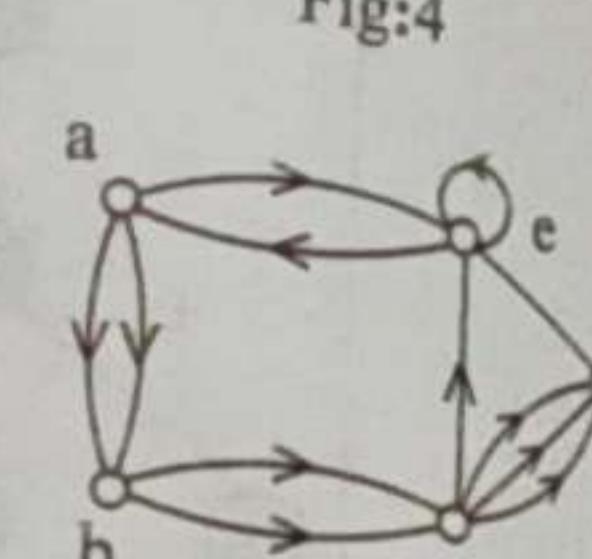
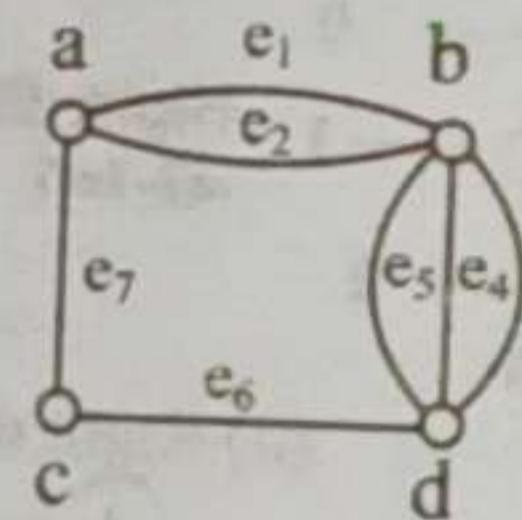


Fig: 5

Ans: 1 simple graph, 2 Multi-graph, 3 pseudo graph, 4 Multi graph, 5 directed multi graph.

সমাধানের টেকনিক

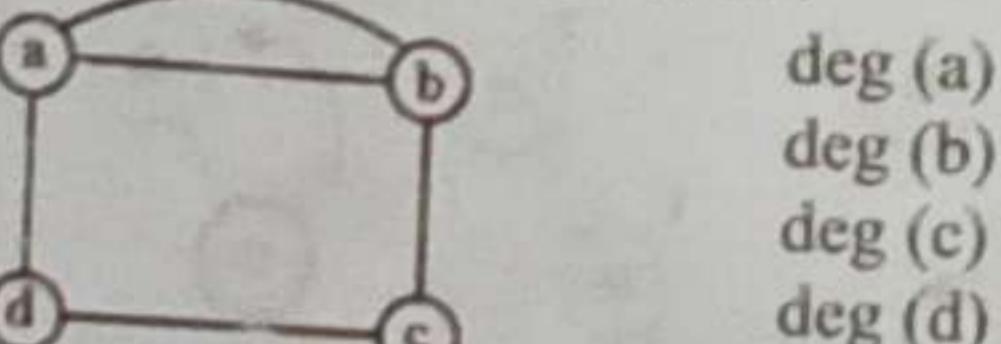
- (i) অথবা edge ডলো নাখারি করতে হবে
- (ii) Self Loop এবং multi edge ডলো remove করতে হবে।

Solⁿ: Fig:2:Ans: {e₂, e₃, e₅}

প্রশ্ন 4. Degree of vertex কি?

■ Degree of vertex: Degree হচ্ছে মাত্রা। একটি vertex থেকে অন্য আর একটি vertex এ যাওয়ার এবং আসার ঘনত্বলো edge থাকে (self loop সহ) তার সমষ্টিকে Degree of vertex বলে।

■ Undirected graph এর ক্ষেত্রে degree of vertex হলো graph এর যে কোন একটি vertex থেকে অন্য আর একটি vertex যাবার ঘনত্বলো পথ (self loop সহ) থাকবে তার সমষ্টিকে বুকায়। একে deg (V) দ্বারা প্রকাশ করা হয়।



deg (a) = 3
deg (b) = 3
deg (c) = 4
deg (d) = 2

[N.B: একটি loop এর অন্য দুই count হবে। অর্থাৎ Loop = 2 গণনা করতে হবে।]

প্রশ্ন 5. In - degree এবং Out-degree কি ?

■ In degree: Directed graph এর ক্ষেত্রে একটি vertex এ ঘনত্বলো edge এর মাধ্যমে অন্য কোন vertex থেকে আসা যায় তার সমষ্টিকে ঐ vertex এর In-degree বলে। একে deg⁻(v) দ্বারা প্রকাশ করা হয়।

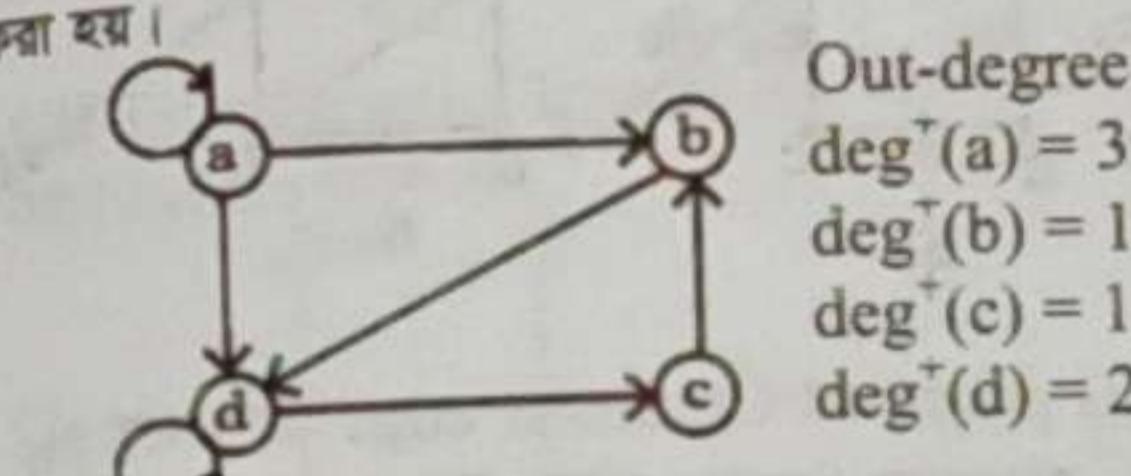
Fig:3

Fig:4

Fig: 5

In-degree
deg⁻(a) = 3
deg⁻(b) = 1
deg⁻(c) = 2
deg⁻(d) = 1

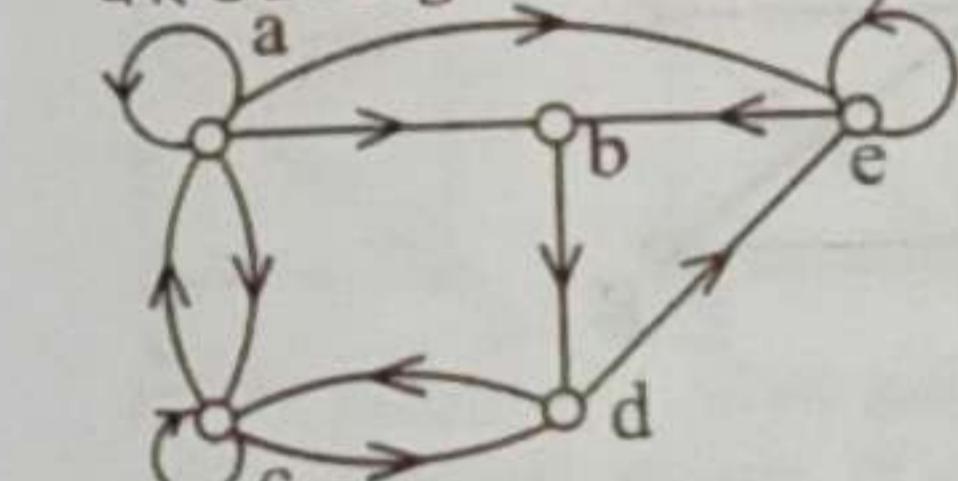
■ Out - degree: Directed graph এর ক্ষেত্রে একটি vertex হতে যতগুলো edge এর মাধ্যমে অন্য vertex এ যাওয়া যায় তার সমষ্টিকে Out-degree বলে। একে deg⁺(v) দ্বারা প্রকাশ করা হয়।



Out-degree
deg⁺(a) = 3
deg⁺(b) = 1
deg⁺(c) = 1
deg⁺(d) = 2

N.B: Directed graph এর ক্ষেত্রে Loop কে Indegree বা Out-degree গণনার সময় 1 ধরতে হবে।

প্রশ্ন 6. নিচের চিত্র পদ্ধতি graph এর vertex এর In-degree এবং Out-degree নির্ণয় কর।

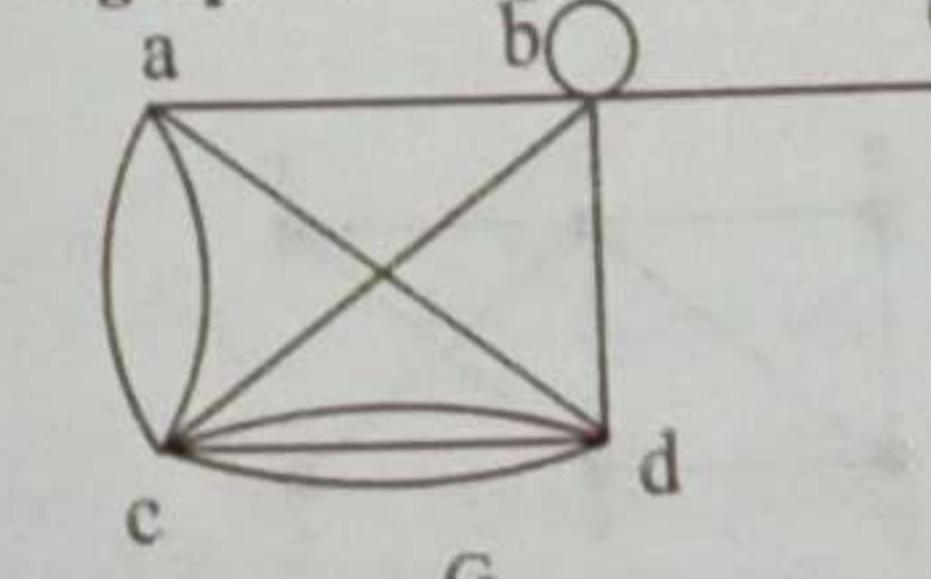


Solution:-

In-degree:-
deg⁻(a) = 2
deg⁻(b) = 2
deg⁻(c) = 3
deg⁻(d) = 2
deg⁻(e) = 3

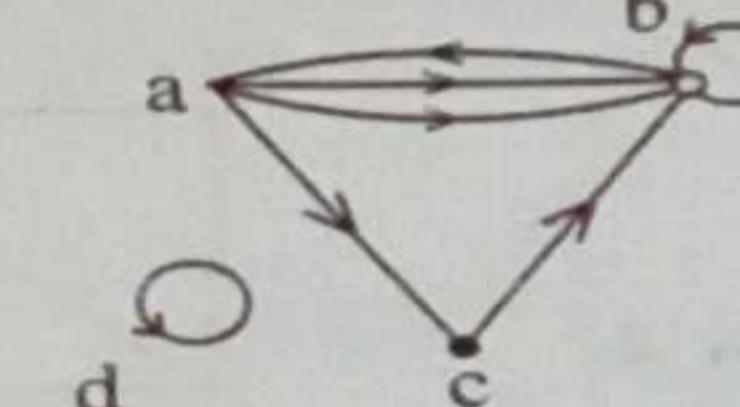
Out degree:-
deg⁺(a) = 4
deg⁺(b) = 1
deg⁺(c) = 3
deg⁺(d) = 2
deg⁺(e) = 2

প্রশ্ন 7. নিচের graph হতে Degrees of vertices বারির কর।

Solⁿ:
চিত্র G তে

deg (a) = 4
deg (b) = 6
deg (c) = 6
deg (d) = 5
deg (e) = 1

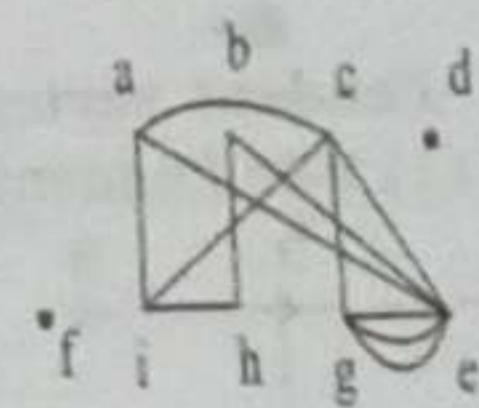
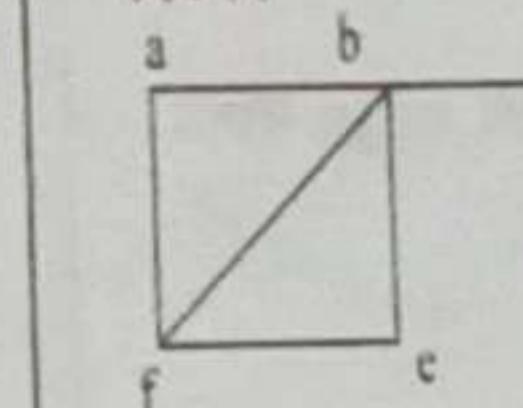
প্রশ্ন 8. নিচের directed graph হতে indegree এবং outdegree বারির কর।

Solⁿ:

In-degree
deg⁻(a) = 1
deg⁻(b) = 4
deg⁻(c) = 1
deg⁻(d) = 1

Out-degree
deg⁺(a) = 3
deg⁺(b) = 2
deg⁺(c) = 1
deg⁺(d) = 1

প্রশ্ন 9. নিচের চিত্র সমূহে degrees of vertices এর যোগফল নির্ণয় কর এবং যাচাই করে দেখাও যে, এই যোগফল সেট edges এর যোগফল

Solⁿ:

Fig(1) :
deg(a) = 2
deg(b) = 4
deg(c) = 1
deg(f) = 3
deg(e) = 2
deg(g) = 0
 $\sum \text{deg}(v) = 12$

Now,
Total Edges = 6
 $2 \times \text{Total Edges} = 12$
 $\sum \text{deg}(v)$

(equal)

Fig (2):
deg(a) = 3
deg(b) = 2
deg(c) = 4
deg(d) = 0
deg(e) = 6
deg(f) = 0
deg(g) = 4
deg(h) = 2
deg(i) = 3
 $\sum \text{deg}(v) = 24$

Now, Total Edges = 12

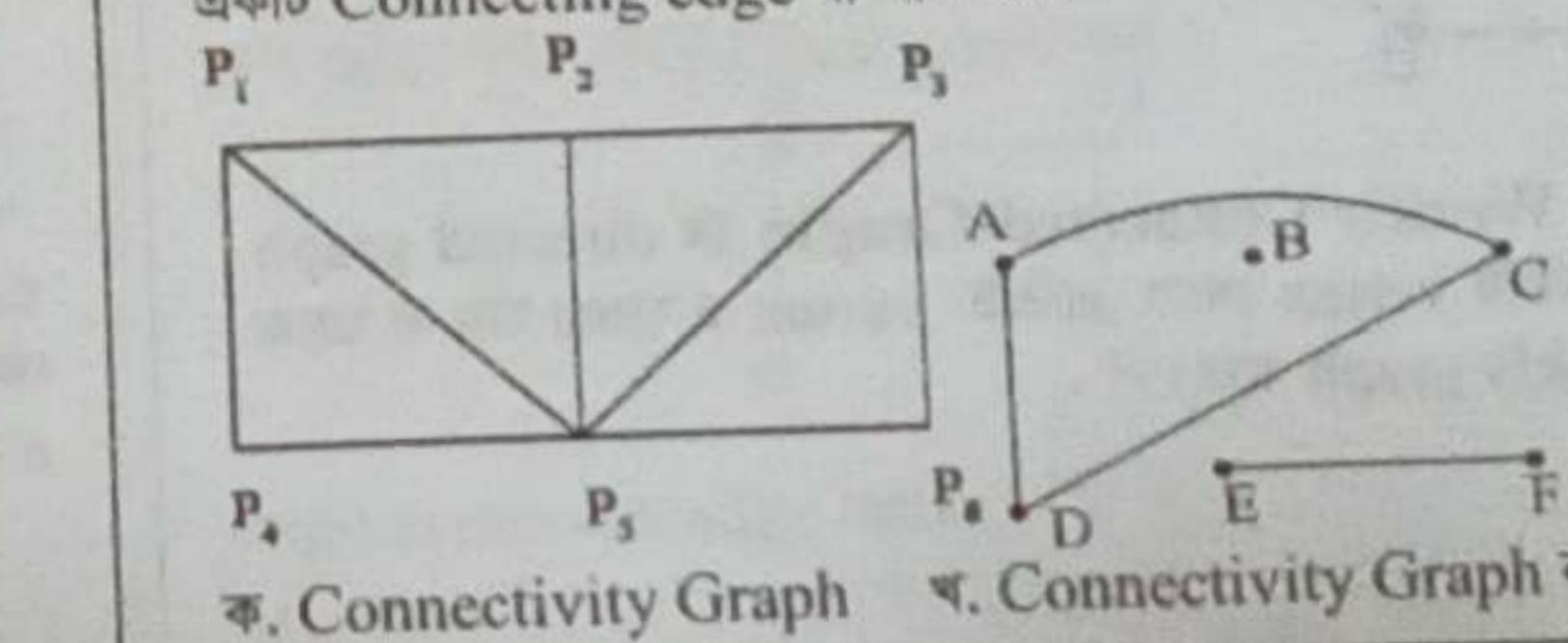
 $2 \times \text{Total Edge} = \sum \text{deg}(v)$

(equal)

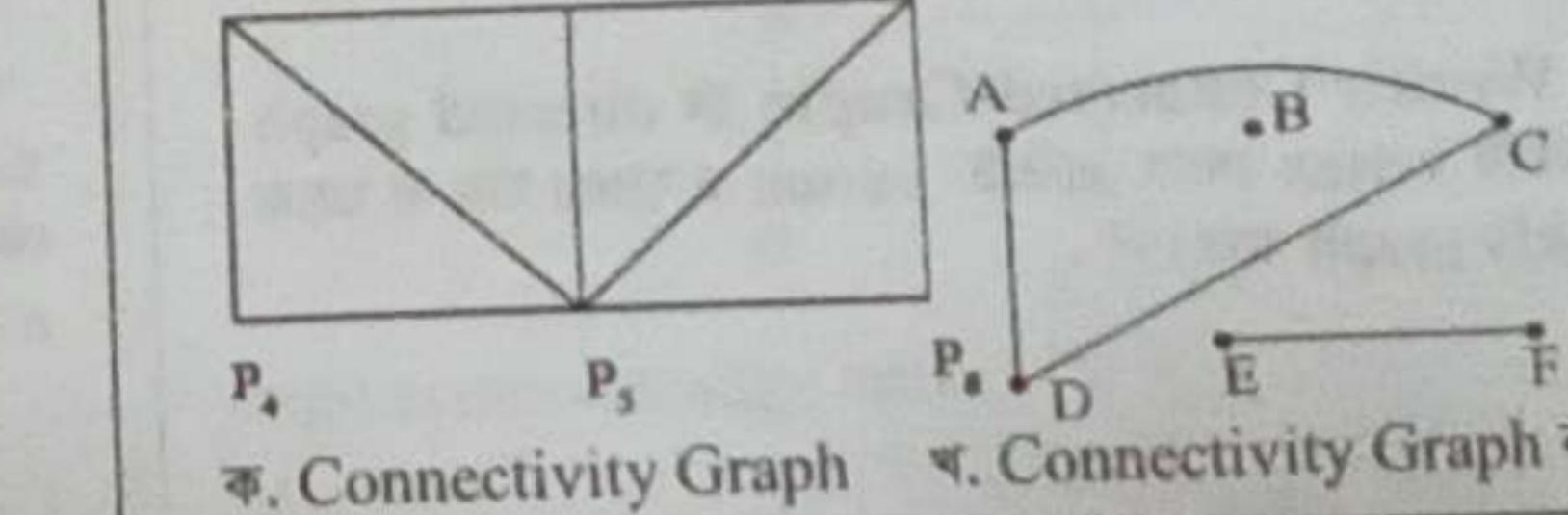
প্রশ্ন 10. বিভিন্ন Graph এর সংজ্ঞা:

i. Connectivity (কানেকটিভি) Graph কি ?

Connectivity Graph: কোন একটি graph এর Connectivity বলতে বুকায়, যে কোন দুইটি vertex এর মধ্যে একটি Connecting edge বা পথ থাকে।

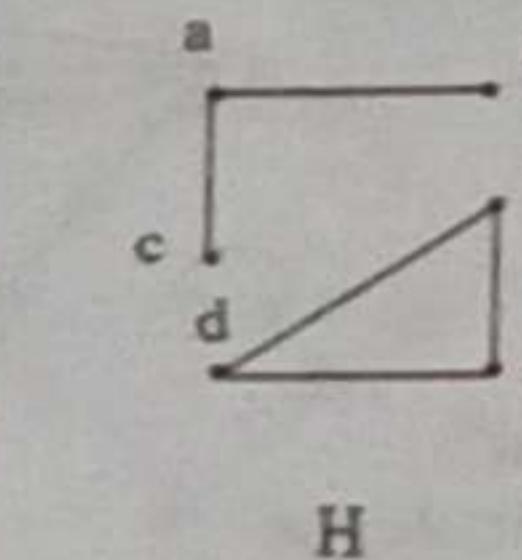
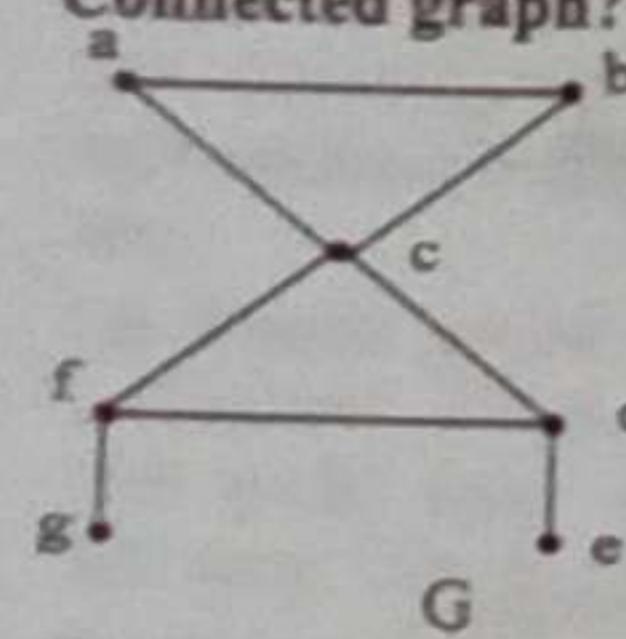


P. Connectivity Graph



q. Connectivity Graph না

ii. নিচের দুটি Graph G এবং H এদের মধ্যে কোনটি Connected graph?

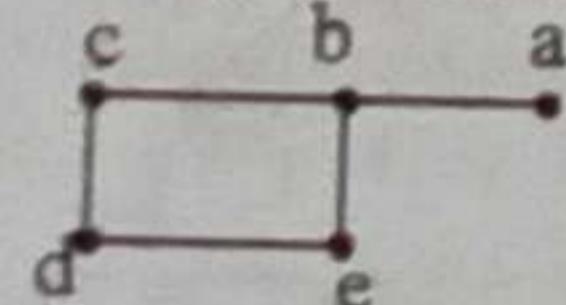


Connected graph: উপরের G graph এ প্রতি দুইটি vertex এ একটি বা অধিক Path বা edge আছে। কিন্তু H graph এ c এবং d, এর মধ্যে কোন Connection বা সম্পোর্ণ edge বা Path নাই। সুতরাং G graph টি Connected এবং H graph টি Connected নহ।

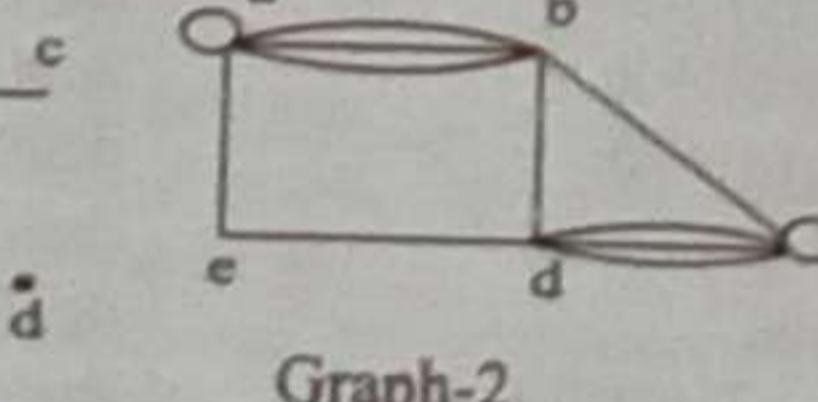
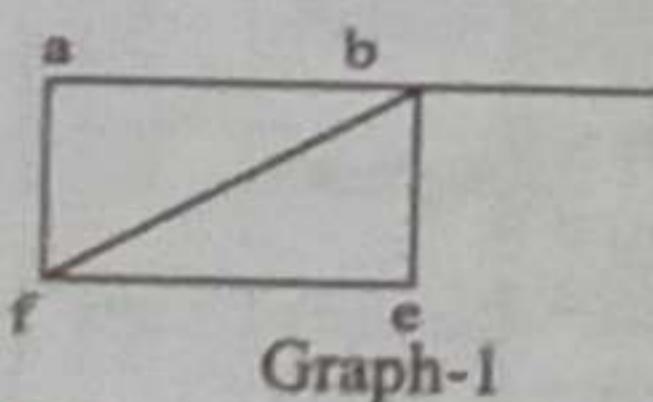
iii. Isolated vertex:- যে vertex এর মাঝে বা edge সংখ্যা ক্ষয় করে তাকে Isolated vertex বলে।

$$\bullet a \quad \deg(a) = 0$$

iv. Pendent vertex: যে vertex এর মাঝে বা edge সংখ্যা এক (1) করে তাকে Pendent vertex বলে।

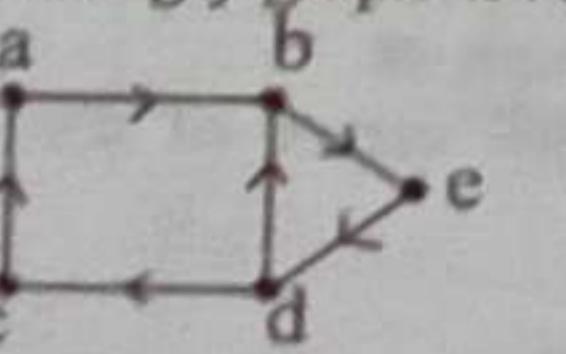


v. Graph হতে Isolated এবং pendent vertices তিক্রিত কর।

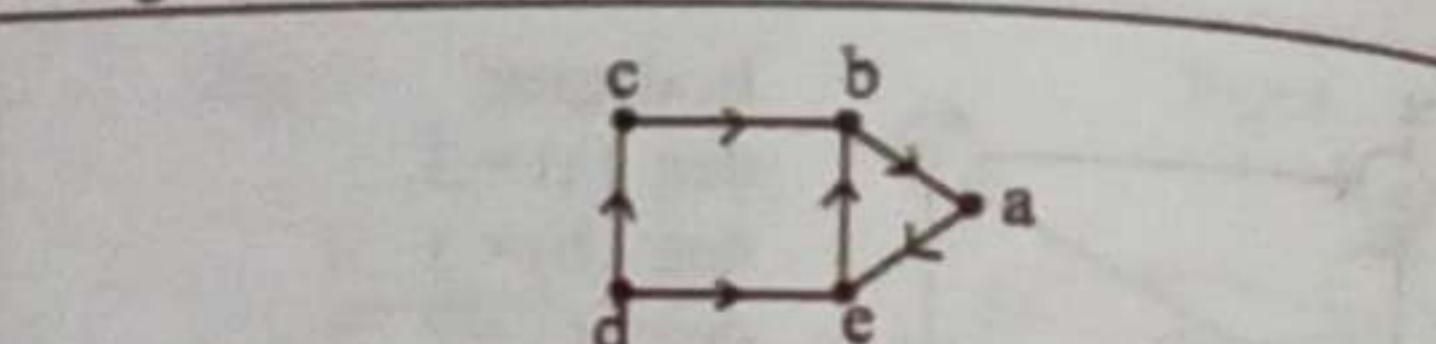


Solⁿ 1: Isolated - d Pendent - c
Solⁿ 2: এই Graph এ কোন Isolated এবং pendent vertices নাই।

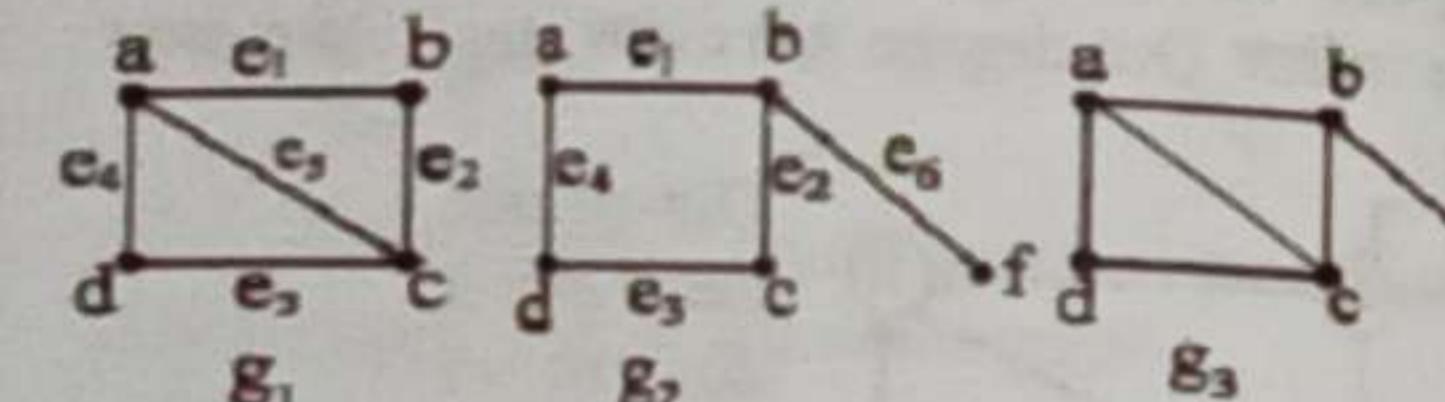
vi. Strongly Connected graph: যে directed graph এর প্রতিটি vertex থেকে প্রতিটি vertex এ যাওয়া যায়, তাকে Strongly graph বলে।



vii. Weakly Connected Graph: যে directed graph এর প্রতিটি vertex থেকে প্রতিটি vertex এ যাওয়া যায় না তাকে Weakly graph বলে।

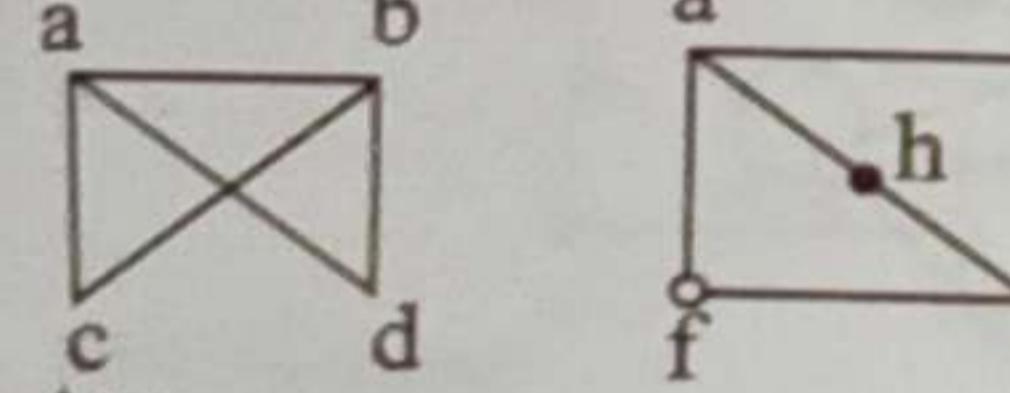


viii. Union of graph: দুই বা ততেওধিক Graph থেকে Common এবং Uncommon vertex এবং edge নিয়ে যে graph তৈরি হয়, তাকে Union graph বলে।

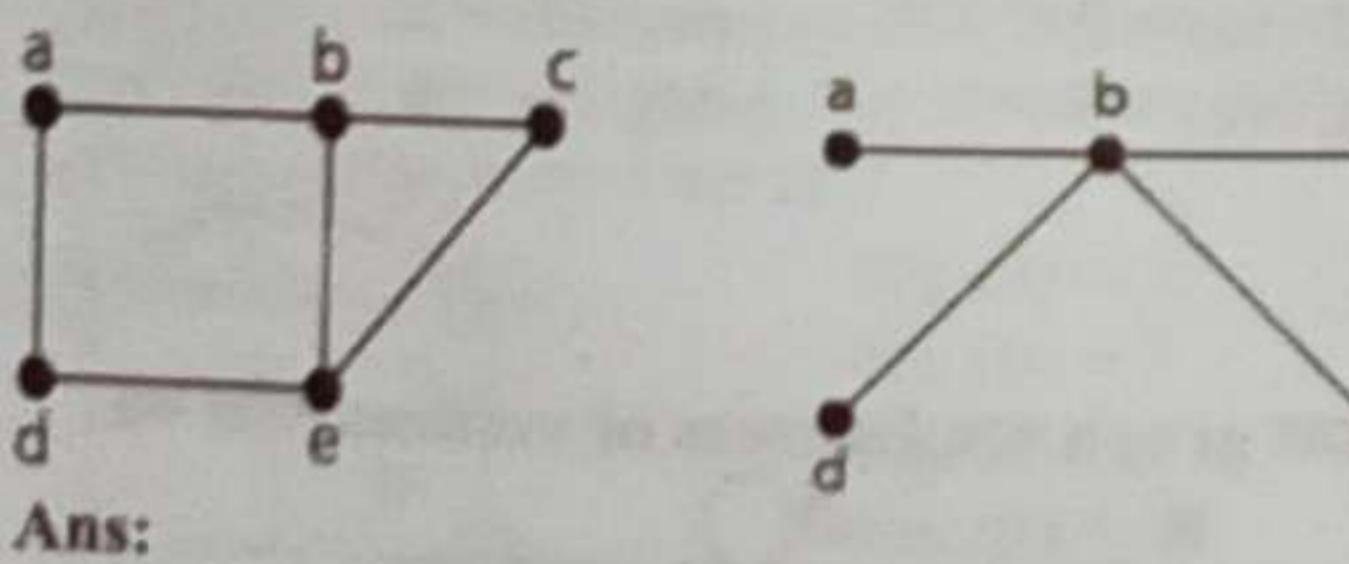
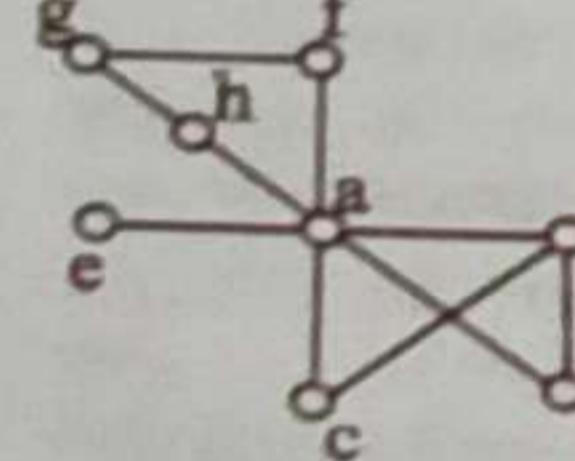


এখানে g₁ এবং g₂ graph এর Common edge তলো হল e₁, e₂, e₃, e₄ এবং Common vertex তলো হল a, b, c, d আর Uncommon edge তলো হল e₅, e₆, এবং Uncommon vertex তলো f।

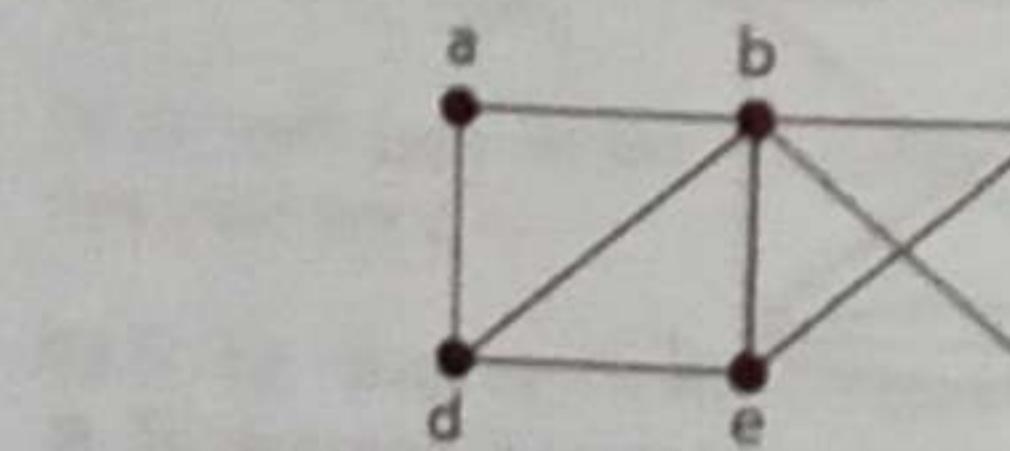
ix. নিচের দুটি graph হতে Union graph তৈরি কর।



Ans:



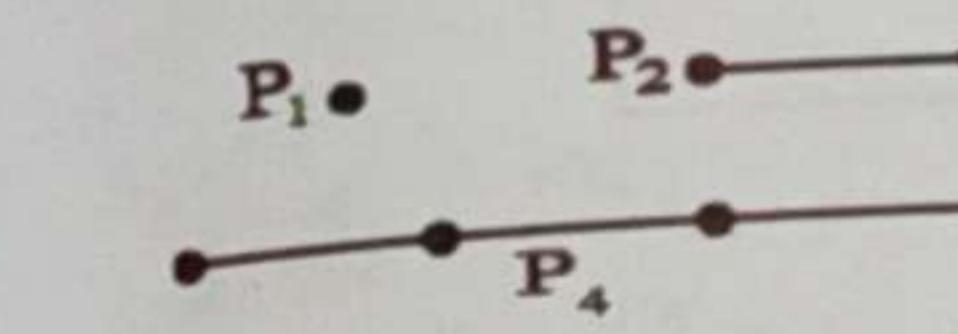
Ans:



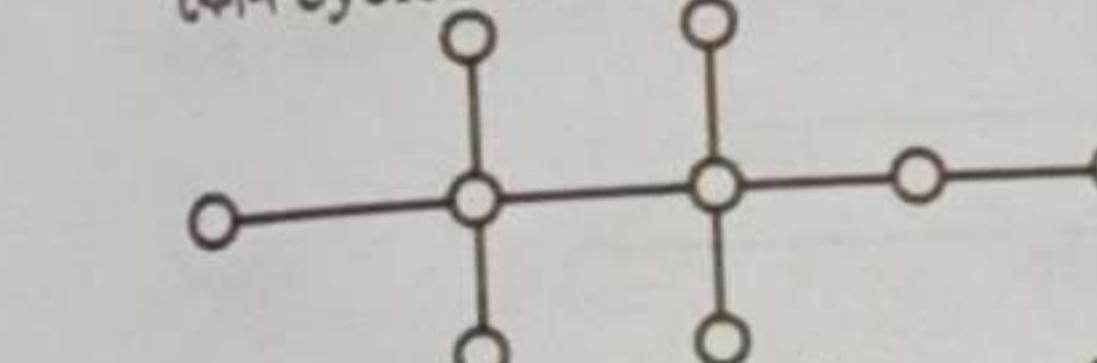
x. Null graph: Null graph এমন এক ধরনের graph যাতে কোন edge থাকে না। একে N_n হিসাবে প্রকাশ করা হয়। এখানে ছোট n হচ্ছে vertex সংখ্যা।

N₁ • N₂ •• N₃ •••

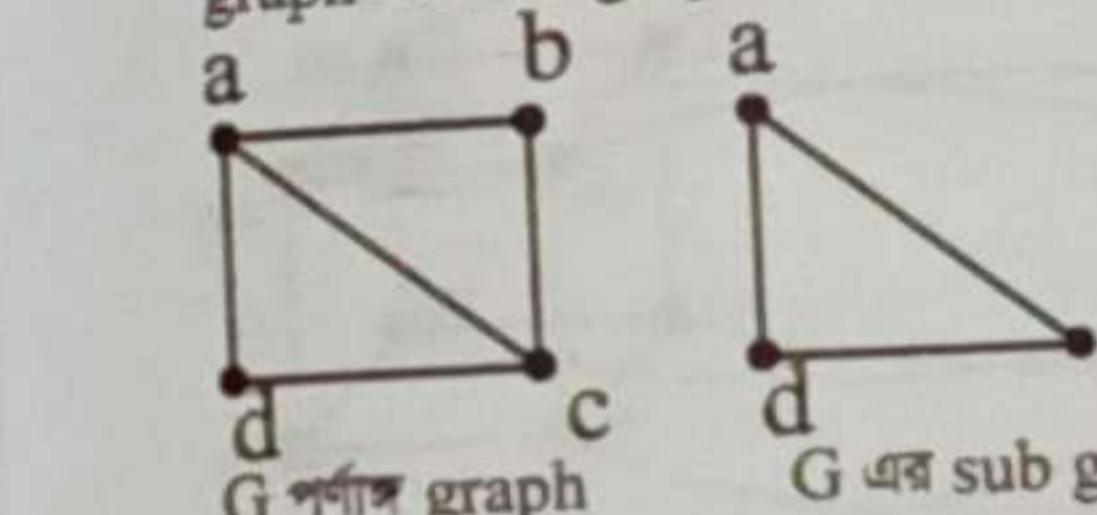
xi. Path graph: Path graph এমন এক ধরনের graph যার Single path থাকে। অর্থাৎ প্রথম vertex এর সাথে 2nd vertex, 2nd vertex এর সাথে 3rd vertex- এ ভাবে পর পর যুক্ত থাকবে। একে P_n হিসাবে প্রকাশ করা হয়। n হল vertex সংখ্যা এবং এর edge সংখ্যা (n - 1)।



xii. Tree Graph: Tree graph এমন এক ধরনের graph যার কোন cycles থাকে না।

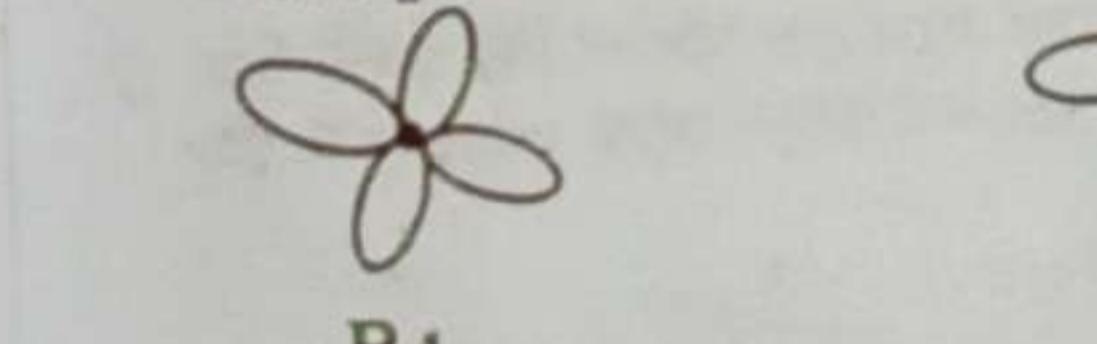


xiii. Sub graph:- যদি কোন একটি পূর্ণাঙ্গ graph এর সাথে অন্য একটি graph মিলে যাব তবে অন্য graph টিকে এই পূর্ণাঙ্গ graph এর Sub graph বলে।



G পূর্ণাঙ্গ graph

G এর sub graph

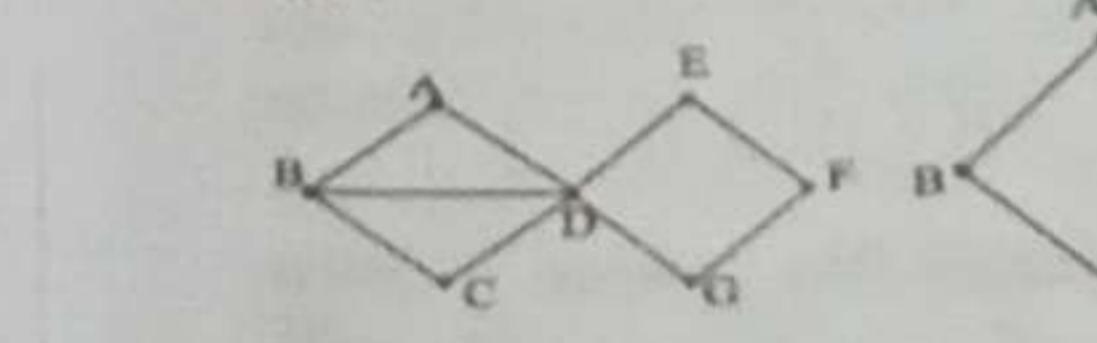


B

B2

B4

xv. Cut Point: যদি কোন Connected Graph এর কোন একটি vertex কে বাদ দেওয়া হয় তবে উক্ত Graph টি দুইটি আলাদা Graph এ পরিণত হয়, তবে উক্ত vertex কে Cut Point বলে।



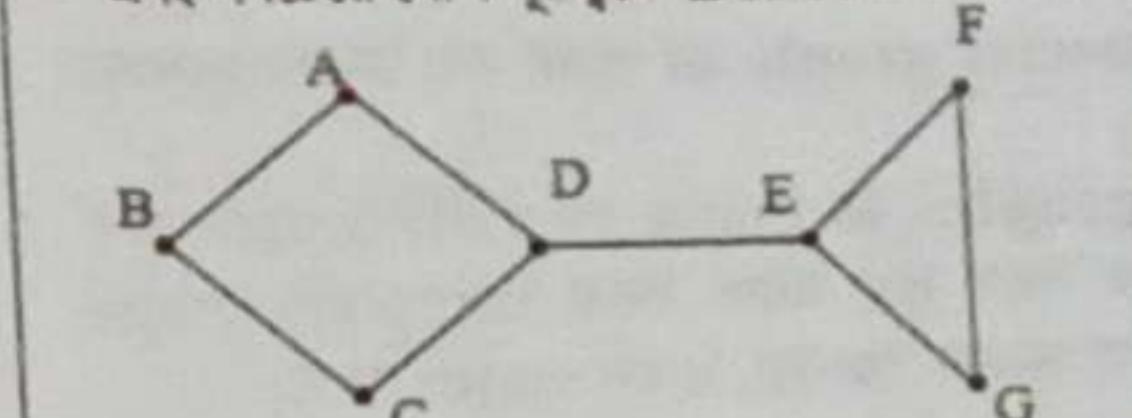
B1

B2

B3

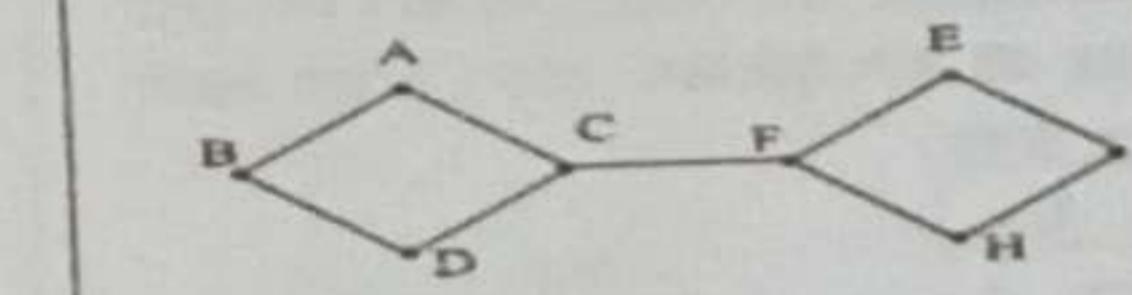
উপরের চিত্রে g₁ একটি Connected graph। g₁ Graph এর vertex D কে যদি বাদ দেওয়া হয় তবে vertex D এর সাথে সংযুক্ত [A,D], [B,D], [C,D], [D, E] এবং [D, G] edge গুলোর কোন অঙ্গস্তু থাকেনা। অর্থাৎ g₁ প্রাক বিজ্ঞিন হচ্ছে g₂ এবং g₃ যাকে পরিণত হয়।

xvi. Distance এবং Diameter: একটি Connected graph এর দুটি vertex এর মধ্যকার সর্বচেয়ে ছোট Path কে Distance এবং সর্বচেয়ে বেশি দূরত্বকে Diameter বলে।



Distance কে d(v₁, v₂) এবং Diameter কে diam(G) হিসাবে প্রকাশ করা হয়। এখানে, G একটি Graph এর v₁ ও v₂ vertex। উপরের Graph এ d(A, B) = 1, d(B, D) = 2, d(B, E) = 3, d(B, F) = 4। এখানে 4 হচ্ছে G graph এর Diameter।

xvii. Bridge (ঁৰজ): যদি কোন Connected graph এর কোন একটি edge কে বাদ দেওয়া হয় তবে উক্ত graph টি ২টি আলাদা Graph এ পরিণত হয়, তবে উক্ত edge কে Bridge বলে।



B1

B2

B3

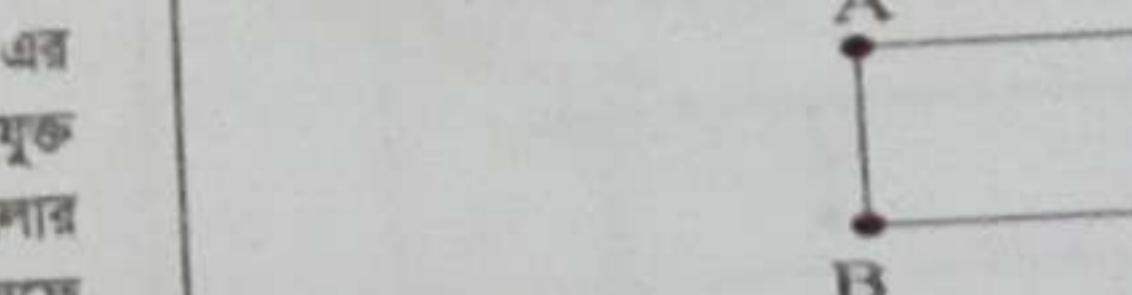
এখানে g₁ একটি Connected graph। g₁ graph থেকে যখন [C, F] edge বাদ দেয়া হয় বা মুছে ফেলা হয় তখন g₁ ও g₂ দুইটি Graph এ ভাগ হয়ে যায়।

xviii. Regular graph: যে graph এর প্রতিটি vertex এর degree একই থাকবে তাকে regular graph বলে।



এখানে প্রতিটি vertex এর degree সমান।

xix. Finite graph: যে graph এর নিম্নিষ্ঠ সংখক vertex এবং edge থাকে তাকে Finite graph বলে।



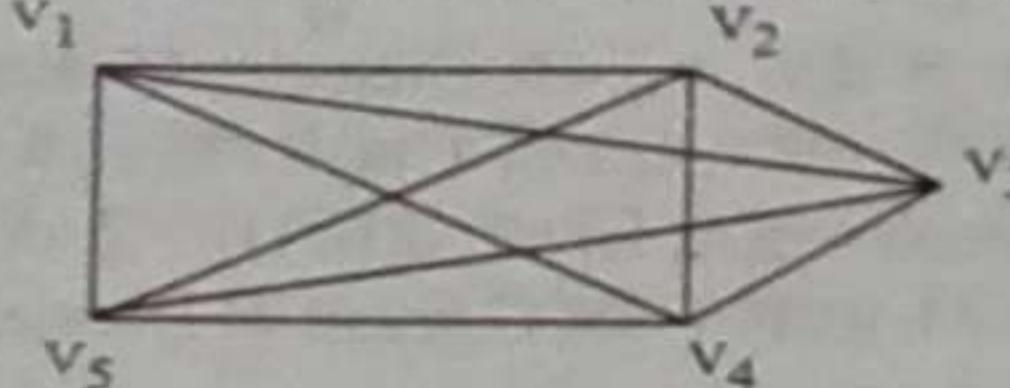
এখানে vertex এবং edge সমান।

xx. Trivial graph:- যে graph এ একটি মাত্র vertex থাকে কিন্তু কোন edge থাকে না তাকে Trivial graph বলে।

A ●

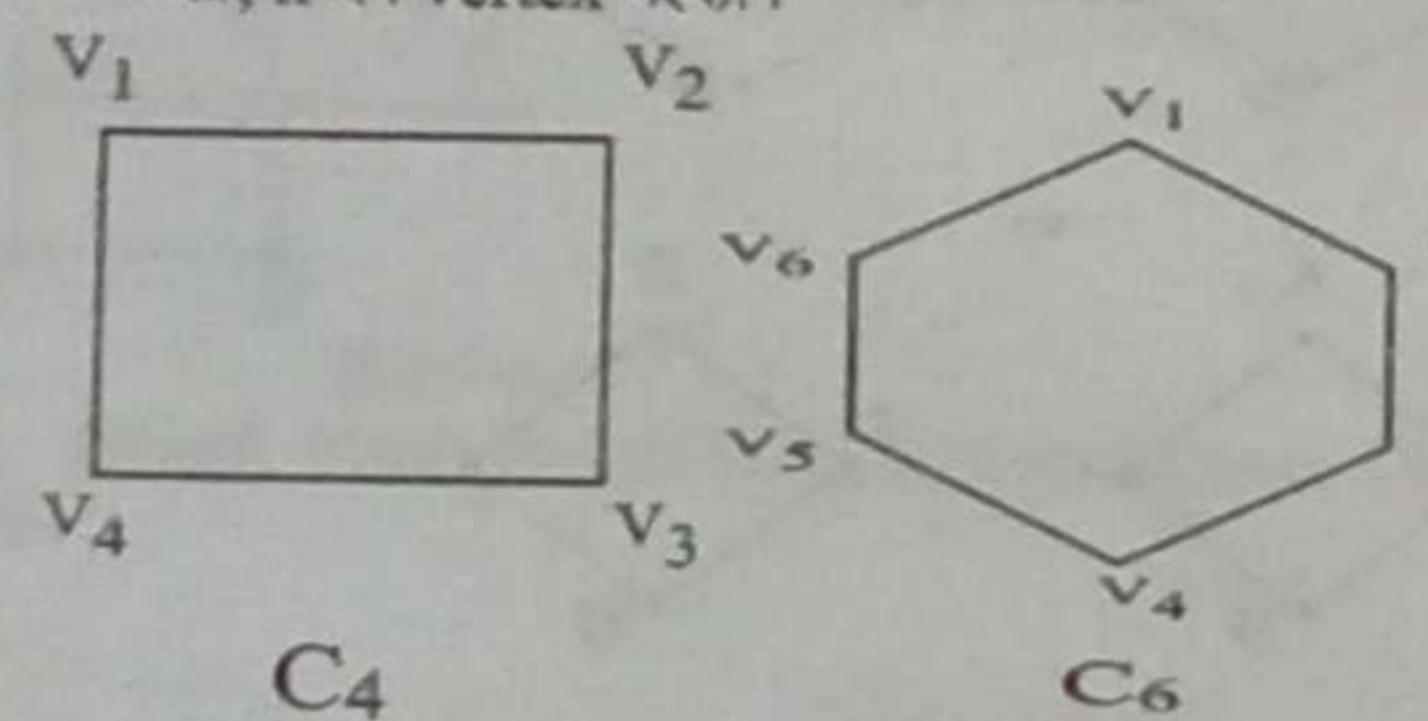
প্র 11. নিচে Special graph এর সম্ভা সহ উদাহরণ দেওয়া হলো।

i. Complete graph:- যে graph এর প্রতিটি vertex অন্য সকল vertex এর সাথে যুক্ত থাকে তাকে complete graph বলে। একে K_n হিসাবে প্রকাশ করা হয়, n হল vertex সংখ্যা।

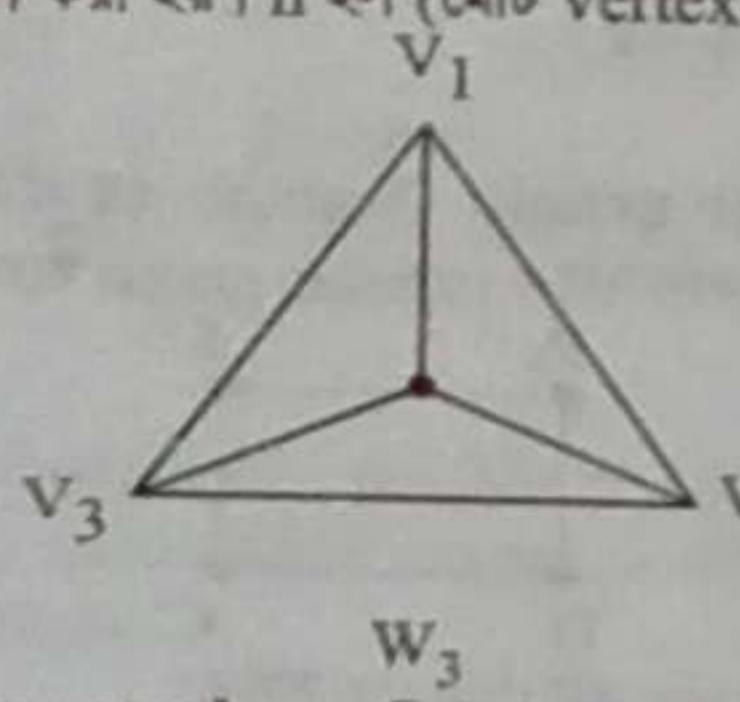


K_5

ii. Cycle graph:- Cycle graph এমন এক ধরণের graph যা ($n \geq 3$) সংখ্যাক vertex দ্বারা গঠিত যেখানে 1st vertex, 2nd vertex এর সাথে, 2nd vertex 3rd vertex এর সাথে এবং last vertex টি 1st vertex এর সাথে যুক্ত হবে। একে C_n হিসাবে প্রকাশ করা হয়, n হল vertex সংখ্যা।



iii. Wheel graph:- যদি Cycle graph এর সাথে একটি vertex যোগ করা হয় তবে তাকে Wheel graph বলে। একে W_n দ্বারা প্রকাশ করা হয়। n হল (মোট vertex-1)।



W_3

iv. n-Cube graph: n-Cube graph এমন এক ধরণের graph যেখানে 2^n সংখ্যাক vertex থাকে যার প্রতিটি vertex কে n bit string আকারে সেবা হয় এবং প্রতি দুটি vertex তখনই Connected হবে যখন তাদের মধ্যে bit পার্শ্বক্য 1 হয়।

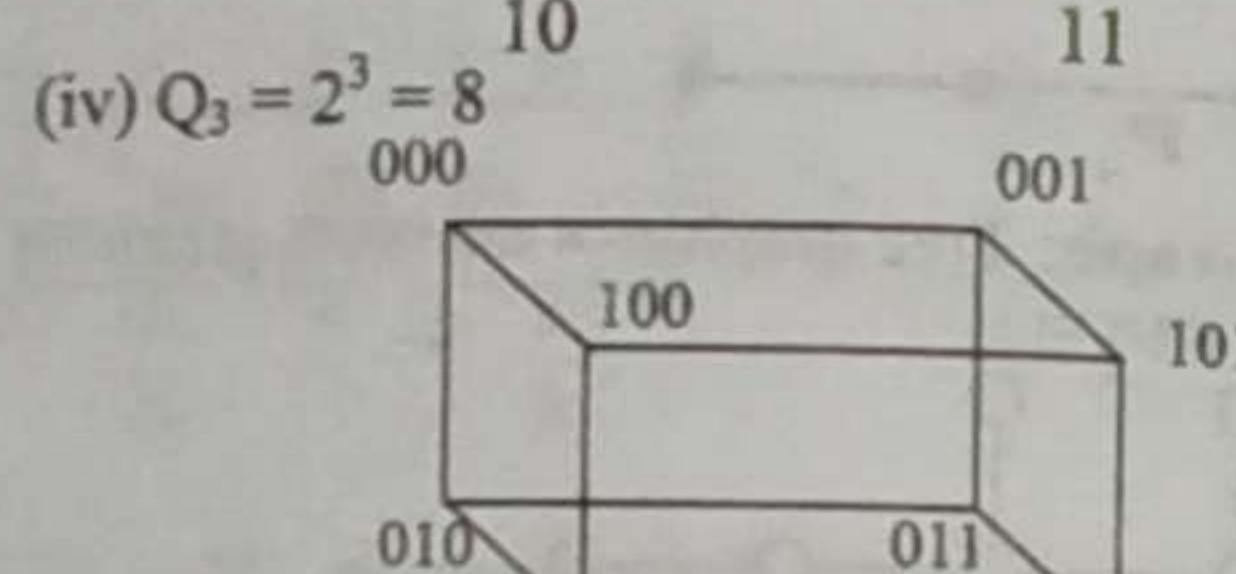
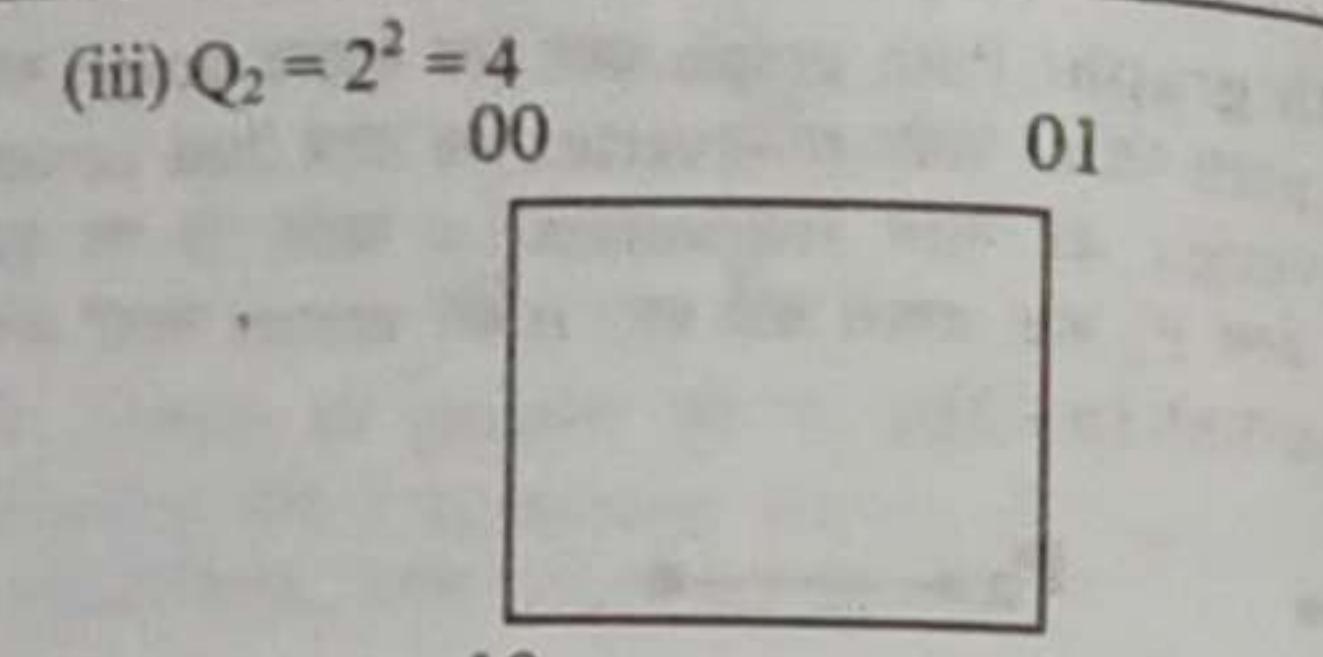
Example:

$$(i) Q_0 = 2^0 = 1$$

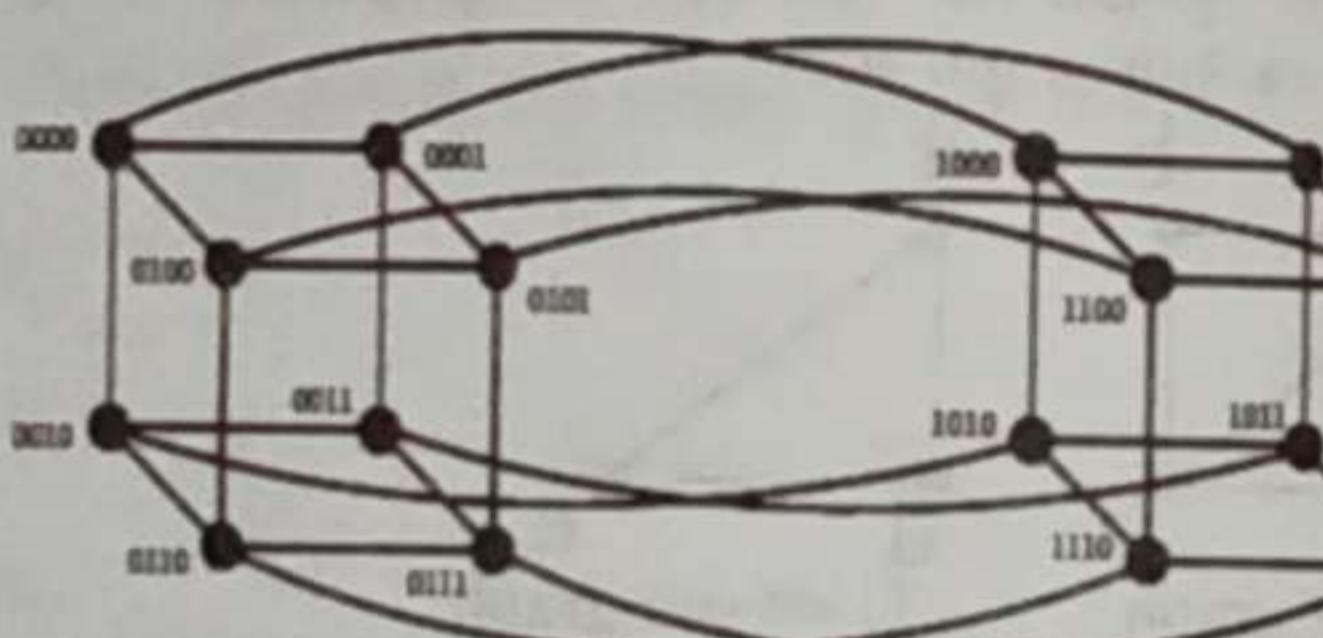
Q

$$(ii) Q_1 = 2^1 = 2$$

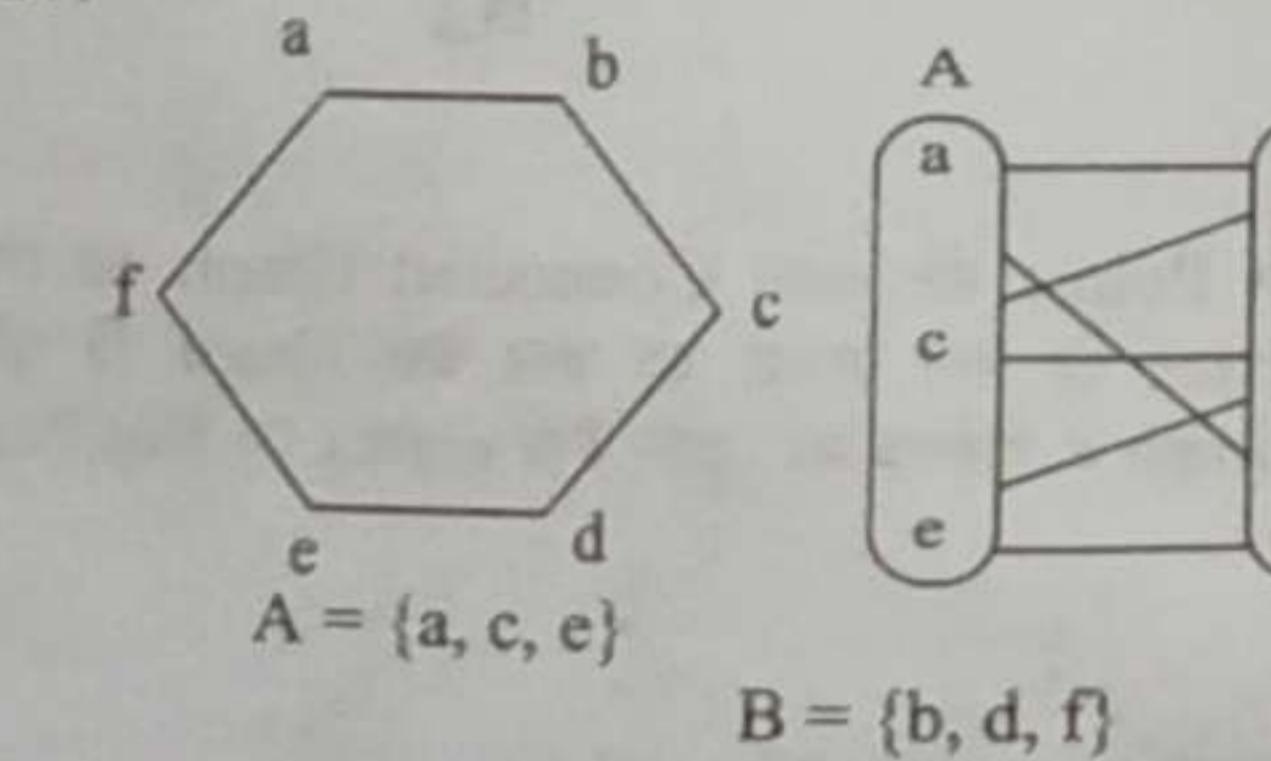
0 I



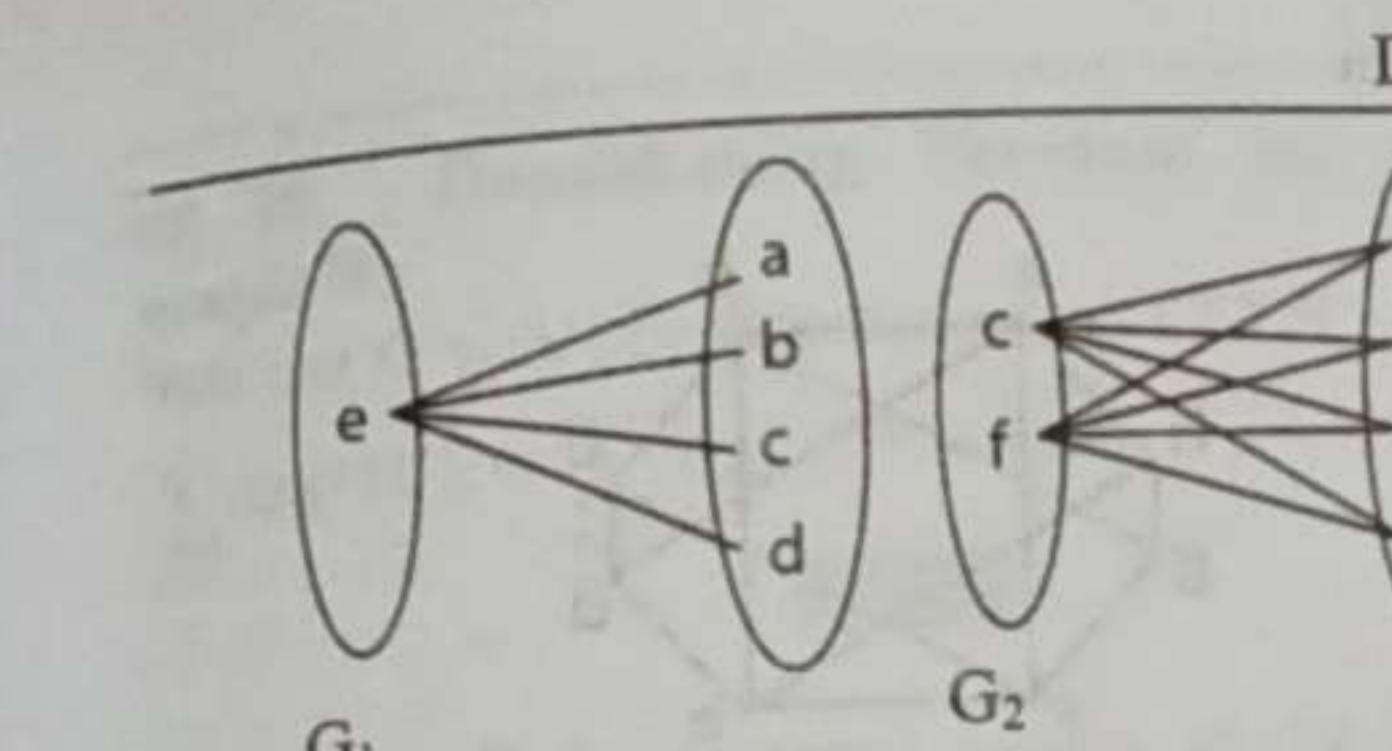
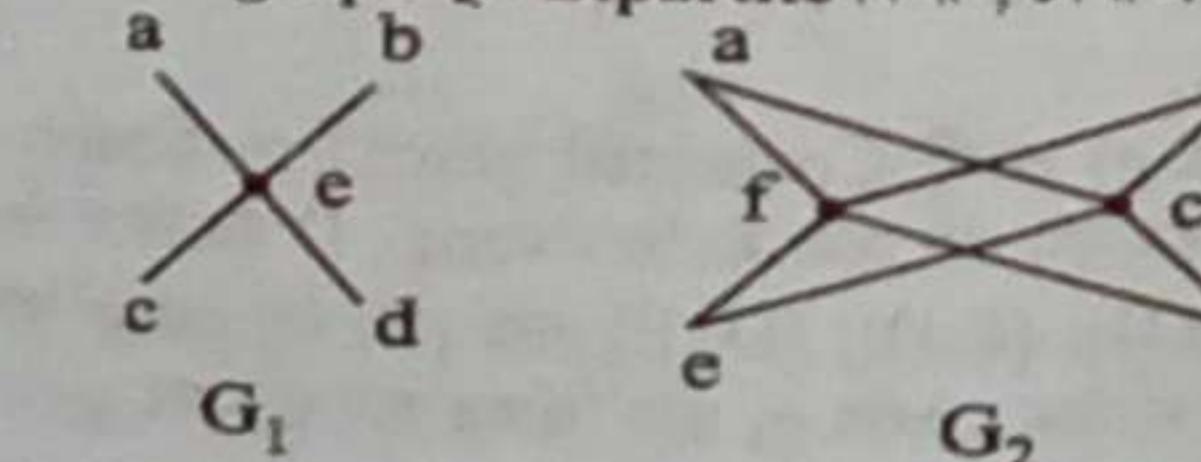
$$(v) Q_4 = 2^4 = 16$$



v. Bipartite graph: একটি graph (G) তখনই bipartite graph বলা যাবে যখন তাদের vertex কে দুটি Set এ এমনভাবে বিভক্ত করা যায় যারা নিজেদের মধ্যে যুক্ত হবে না কিন্তু একটি Set এর উপাদানের সাথে অপর Set এর উপাদানগুলো edge দ্বারা যুক্ত হবে।

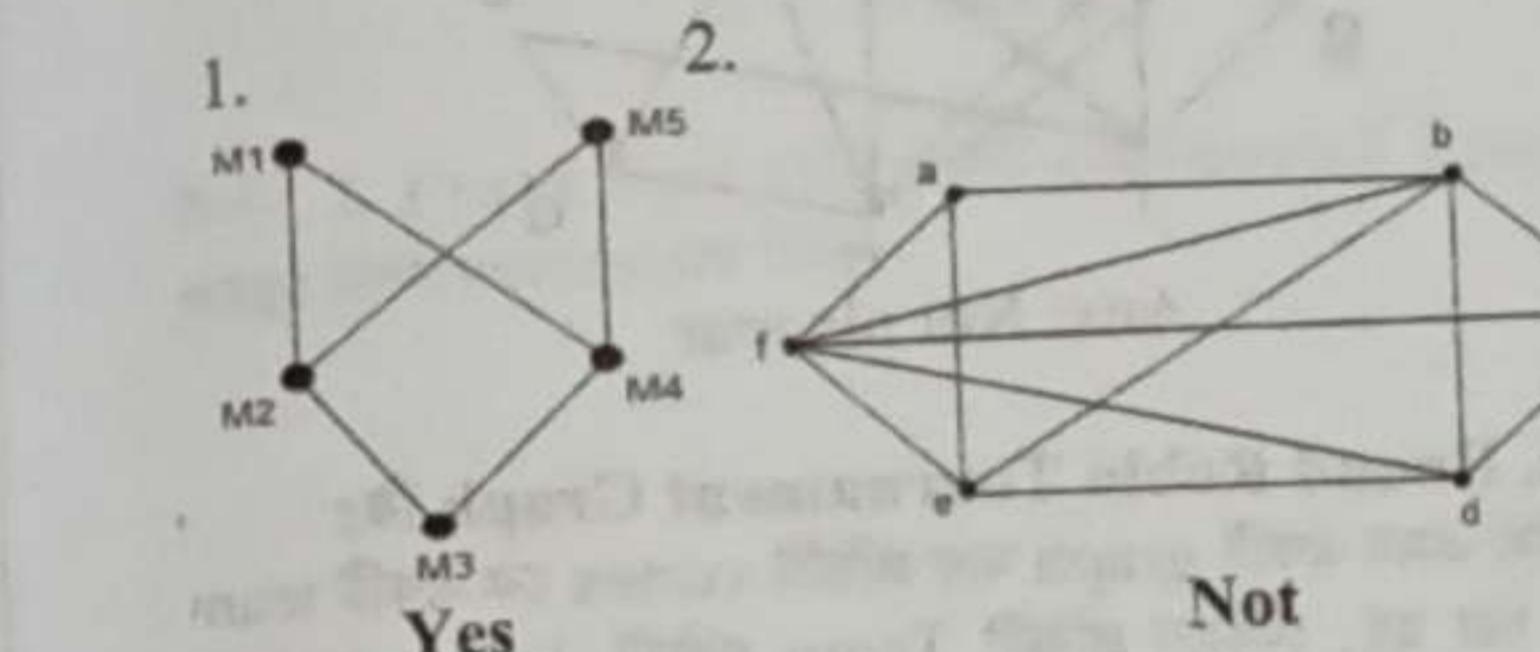


vi. নিচের graph দুটি Bipartite কিনা, দেখো?



G_1 graph টি হলো bipartite, এখানে vertex গুলোকে দুটি ভাগে বিভক্ত করা যায়। যেমন $\{e\}$ $\{a, b, c, d\}$
 G_2 graph টি হলো bipartite। এখানে vertex গুলোকে দুটি ভাগে বিভক্ত করা যায় $\{c, f\}$ $\{a, d, b, e\}$

vii. Find which graph is bipartite or not:-



2. a, b, c, d, e

3. a, b, c, d, e

4. a, b, c, d, e

5. a, b, c, d, e

6. a, b, c, d, e

7. a, b, c, d, e

8. a, b, c, d, e

9. a, b, c, d, e

10. a, b, c, d, e

11. a, b, c, d, e

12. a, b, c, d, e

13. a, b, c, d, e

14. a, b, c, d, e

15. a, b, c, d, e

16. a, b, c, d, e

17. a, b, c, d, e

18. a, b, c, d, e

19. a, b, c, d, e

20. a, b, c, d, e

21. a, b, c, d, e

22. a, b, c, d, e

23. a, b, c, d, e

24. a, b, c, d, e

25. a, b, c, d, e

26. a, b, c, d, e

27. a, b, c, d, e

28. a, b, c, d, e

29. a, b, c, d, e

30. a, b, c, d, e

31. a, b, c, d, e

32. a, b, c, d, e

33. a, b, c, d, e

34. a, b, c, d, e

35. a, b, c, d, e

36. a, b, c, d, e

37. a, b, c, d, e

38. a, b, c, d, e

39. a, b, c, d, e

40. a, b, c, d, e

41. a, b, c, d, e

42. a, b, c, d, e

43. a, b, c, d, e

44. a, b, c, d, e

45. a, b, c, d, e

46. a, b, c, d, e

47. a, b, c, d, e

48. a, b, c, d, e

49. a, b, c, d, e

50. a, b, c, d, e

51. a, b, c, d, e

52. a, b, c, d, e

53. a, b, c, d, e

54. a, b, c, d, e

55. a, b, c, d, e

56. a, b, c, d, e

57. a, b, c, d, e

58. a, b, c, d, e

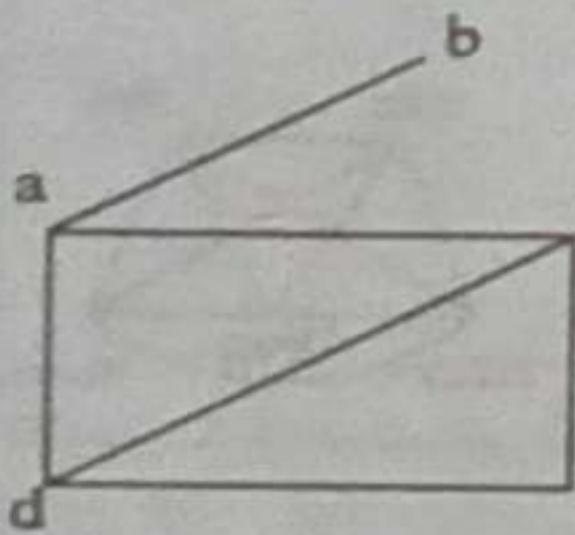
59. a, b, c, d, e

- प्रश्न २०. निचेरे Graph तले अकलन कर, Color कर एवं Chromatic Number देवे कर
 (i) $K_{2,3}$ (ii) $K_{2,5}$ (iii) K_3 (iv) $K_{5,6}$ (v) K_8 , (vi) W_9 ,
 (vii) C_9 , (viii) C_{10} , (ix) W_{10}
 Ans: i) 2, ii) 2, iii) 3, iv) 2, v) 8, vi) 4, vii) 3, viii)
 2, ix) 3

■ Graph के Matrix एवं प्रत्यक्षरकमतः

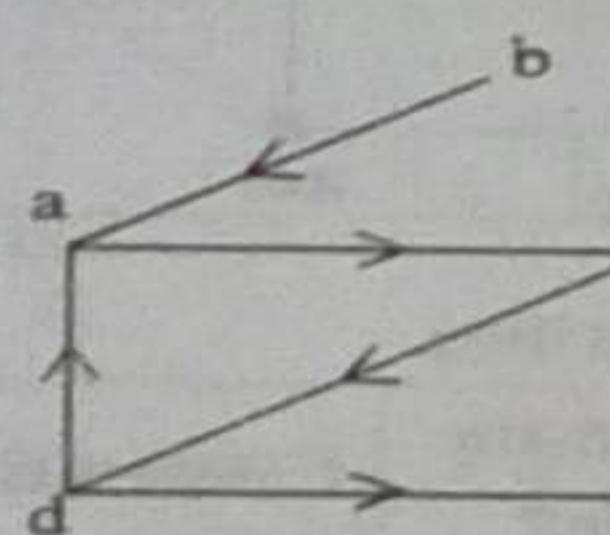
- a) Adjacency list.
- b) Adjacency Matrix.
- c) Incidence Matrix.

a) Adjacency list: कोन graph ए एकटि vertex ए graph एवं कठजले vertex एवं संयुक्त आहे ता ये list एवं याध्यमे उपलब्धन करा हय ताके Adjacency list बले।



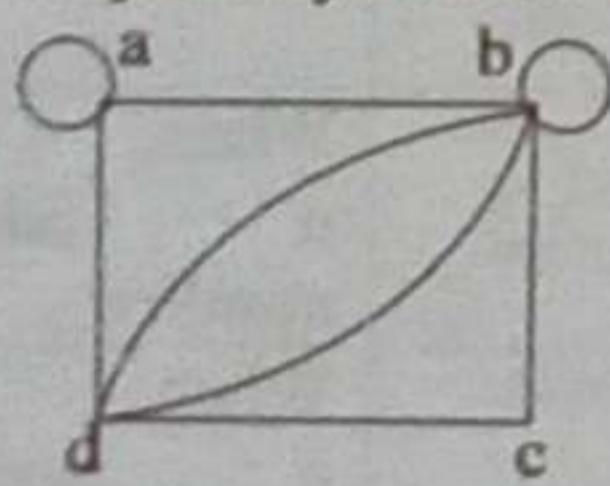
vertex	Adj vertices
a	b,c,d
b	a
c	a,d,e
d	a,c,e
e	c,d

[N.B: Directed graph ए फ्रेंडे एकटि vertex हते अन्य vertex ए याओर यतजले direction आहे।]



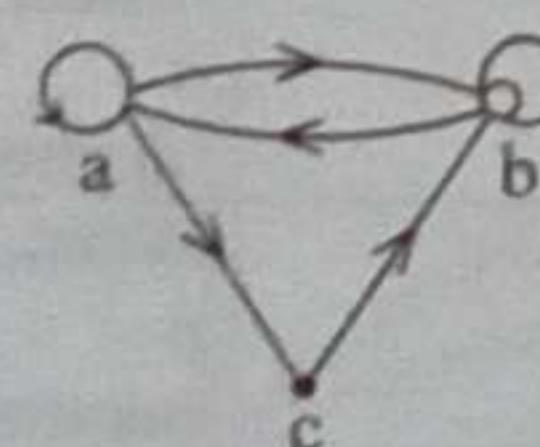
vertex	Adj vertices
A	C
B	A
C	D
D	a,e
E	C

b) Adjacency Matrix: कोन graph ए एकटि vertex देवे कोन vertex ए याओर योट यतजले Path आहे तार समष्टि ये Matrix एवं याध्यमे उपलब्धन करा हय ताके Adjacency Matrix बले।



	a	b	c	d
a	2	1	0	1
b	1	2	1	2
c	0	1	0	1
d	1	2	1	0

Graph

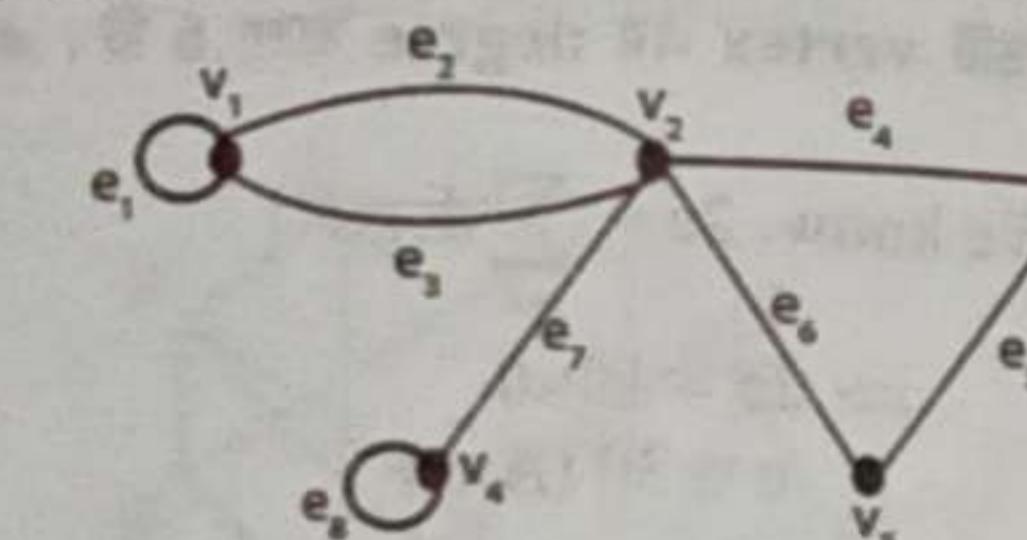


Matrix

	a	b	c
a	1	1	1
b	1	1	0
c	0	1	0

[N.B: Directed graph ए तधु मात्र याओर Path Count हवे।]

c) Incident Matrix: कोन graph ए एकटि edge देवे कोन vertex के संयुक्त कराहे ता ये Matrix एवं याध्यमे उपलब्धन करा हय ताके Incident Matrix बले। एक्सें vertex समूह row आकारे एवं edge समूह column आकारे बसवे। Vertex समूहे Connected edge के (1) एवं उपलब्धन करा हय।

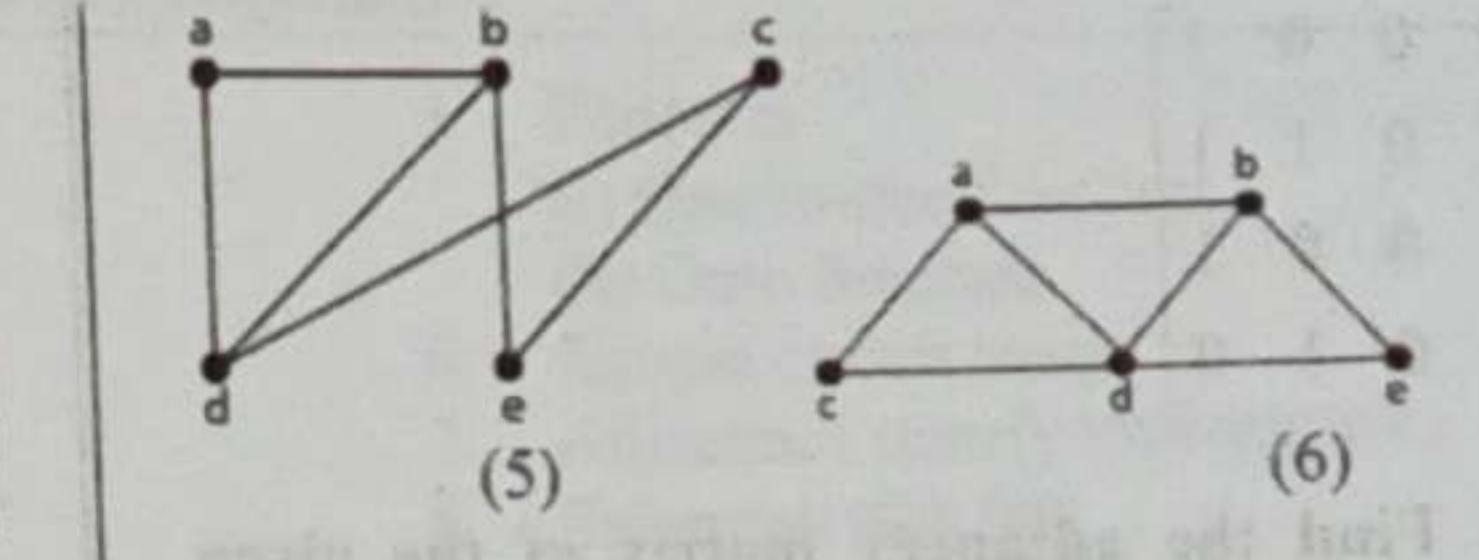


Graph	Matrix
	$\begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$

Matrix

	a	b	c	d	E
a	0	1	0	1	0
b	1	0	1	0	1
c	0	1	1	0	0
d	1	0	0	0	1
e	0	0	1	0	1

vertex	Adj vertices
a	b, d
b	a, d, e
c	b, c
d	a, b, c
e	b, c



1) ग्राफके कीভाबे उपलब्धन करा याय ? उदाहरणसह व्याख्या करन।

[विडियो लेसन्स-२०१९-संस्करणी श्रेणीमार्ग]

उत्तर: ग्राफके साधारण दृष्टि उपाये उपलब्धन करा याय।

❖ Adjacency Matrix

❖ Adjacency List.

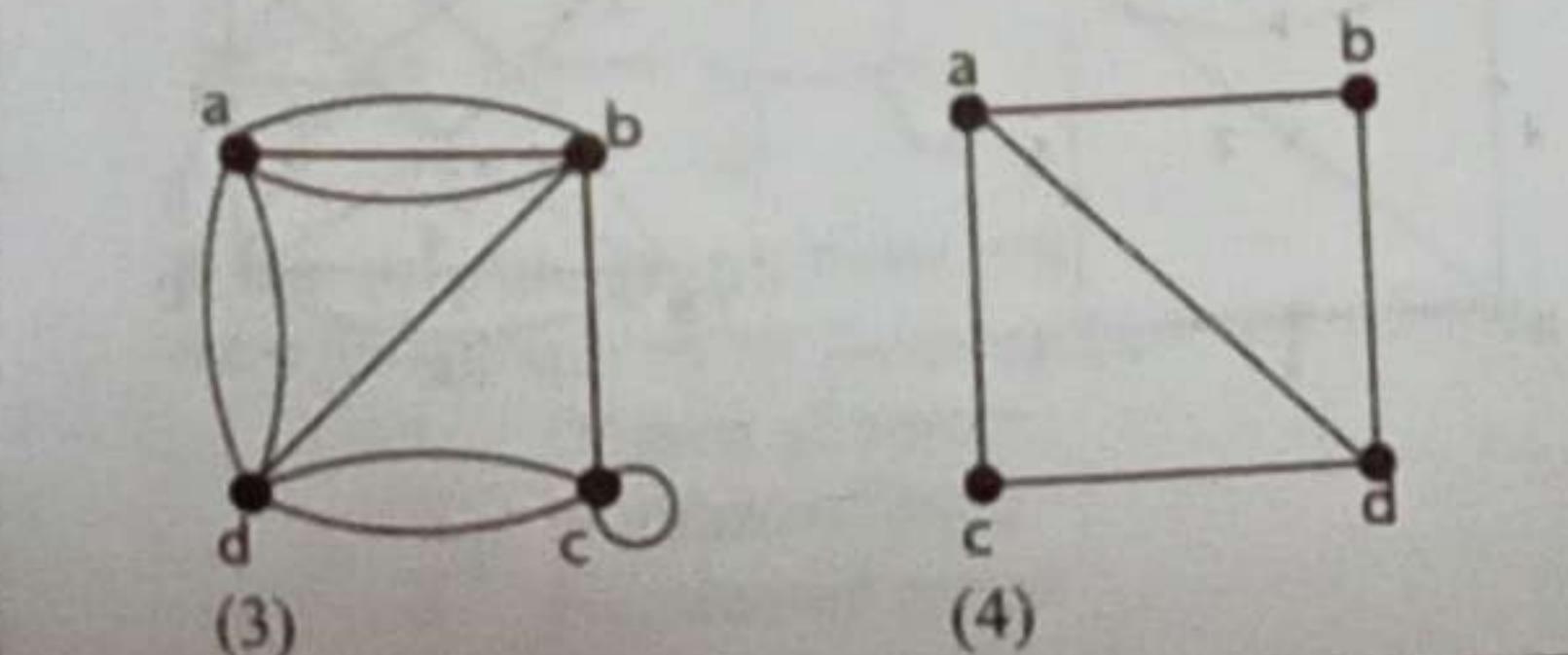
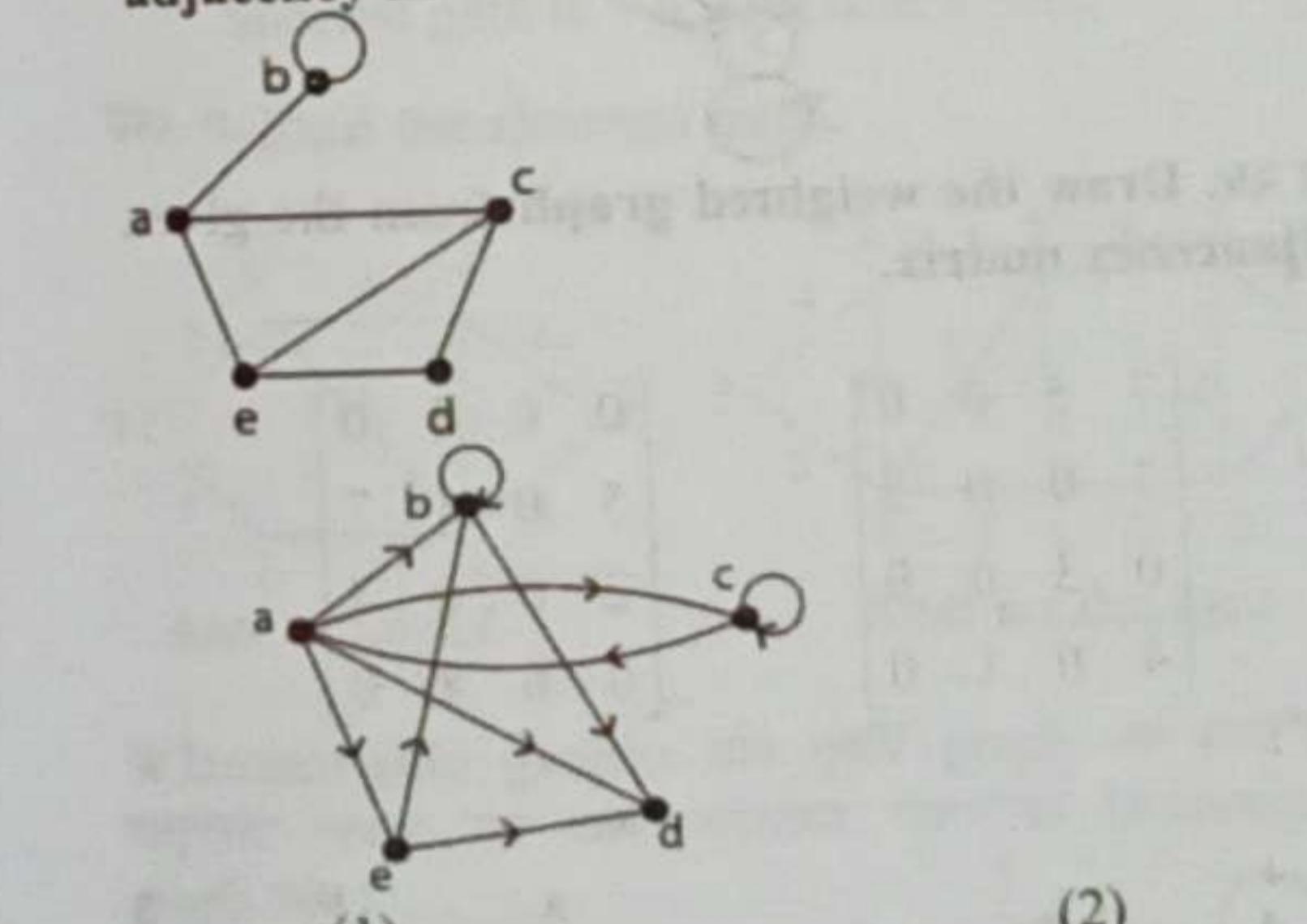
उदाहरणसह व्याख्या करा हलो:

Graph	Matrix	List																																										
	$\begin{matrix} A & B & C & D & E & F \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$	<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th> </tr> </thead> <tbody> <tr> <td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td> </tr> <tr> <td>B</td><td>A</td><td>D</td><td>E</td><td>F</td><td></td> </tr> <tr> <td>C</td><td></td><td>A</td><td>D</td><td>F</td><td></td> </tr> <tr> <td>D</td><td></td><td></td><td>B</td><td>C</td><td></td> </tr> <tr> <td>E</td><td></td><td></td><td></td><td>F</td><td></td> </tr> <tr> <td>F</td><td></td><td></td><td></td><td></td><td>D</td> </tr> </tbody> </table>	A	B	C	D	E	F	A	B	C	D	E	F	B	A	D	E	F		C		A	D	F		D			B	C		E				F		F					D
A	B	C	D	E	F																																							
A	B	C	D	E	F																																							
B	A	D	E	F																																								
C		A	D	F																																								
D			B	C																																								
E				F																																								
F					D																																							

F=D

Self Study

उत्ता- १. निचेरे graph तले adjacency Matrix एवं adjacency list देवे कर

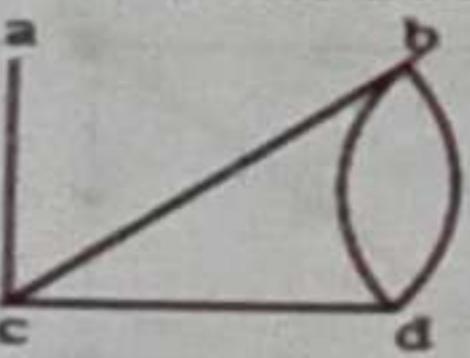


उत्ता- २. Draw a directed graph with the given adjacecy matrix.

$$\text{i)} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{ii)} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

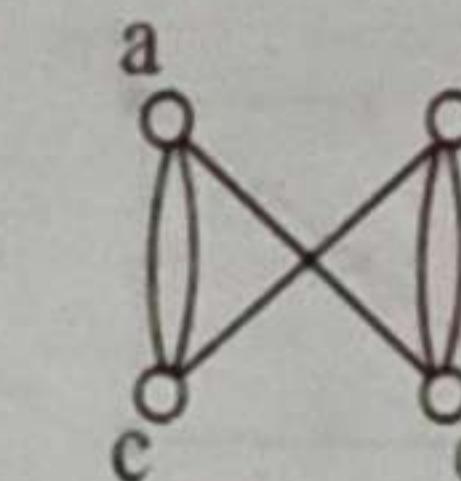
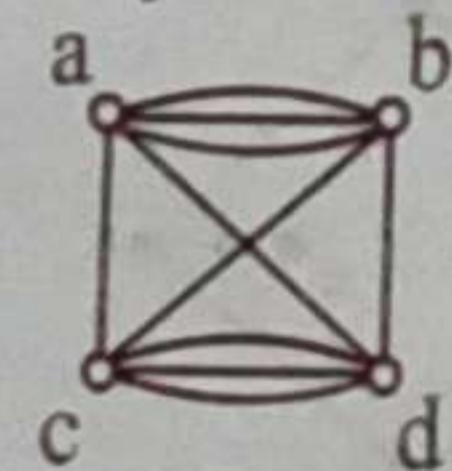
$$\text{iii) } \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

উদা- ৭. Find the adjacecy matrix of the given undirected multigraph



	a	b	c	d
a	0	0	1	0
b	0	0	1	2
c	1	1	0	1
d	0	2	1	0

Self study:

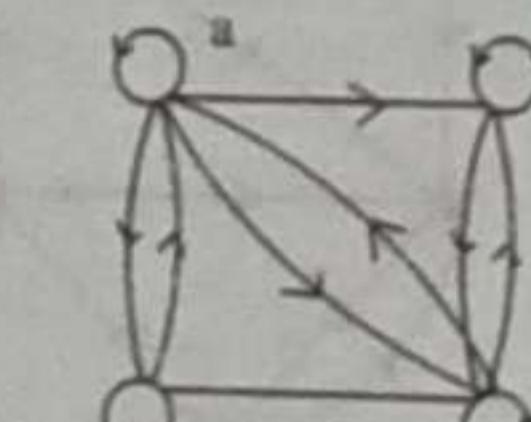
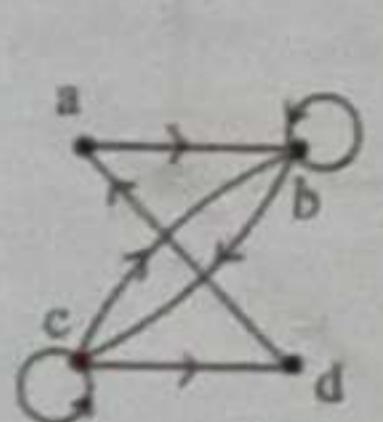


উদা- 8. Draw graph with the given adjacency matrix.

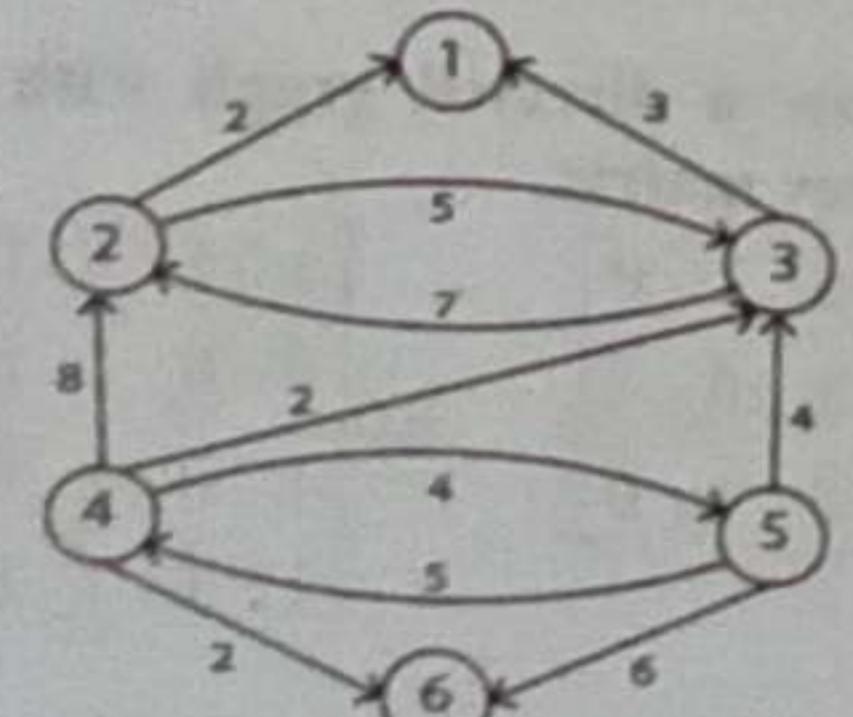
(i)	0	2	0	1
	2	0	3	0
	0	3	0	1
	1	0	1	0

(ii)	0	1	3	0	4
	1	2	1	3	0
	3	1	2	0	1
	0	3	0	0	2
	4	0	1	2	4

উদা- ৯. Find the adjacecy matrix of the given directed multigraph.



প্রশ্ন ২৬. নিম্নলিখিত Graph টিকে একটি Matrix এর সাহায্যে প্রকাশ কর ?



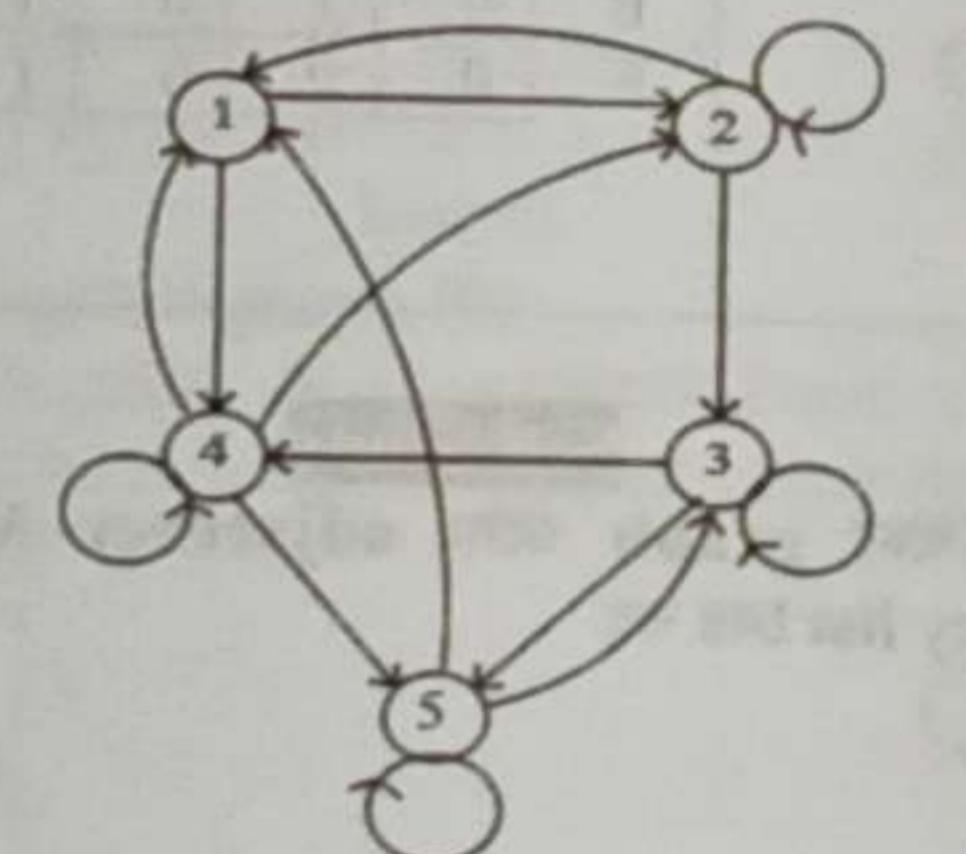
Solⁿ:

1	2	3	4	5	6
1	0	0	0	0	0
2	2	0	5	0	0
3	3	7	0	0	0
4	0	8	2	0	4
5	0	0	4	5	0
6	0	0	0	0	0

প্রশ্ন ২৭. নিম্নে Matrix দিয়ে একটি Graph তৈরি কর, যেখন 1 প্রকাশ করে একটি পথ আছে, 0 প্রকাশ করে পথ নাই।

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

Solⁿ:

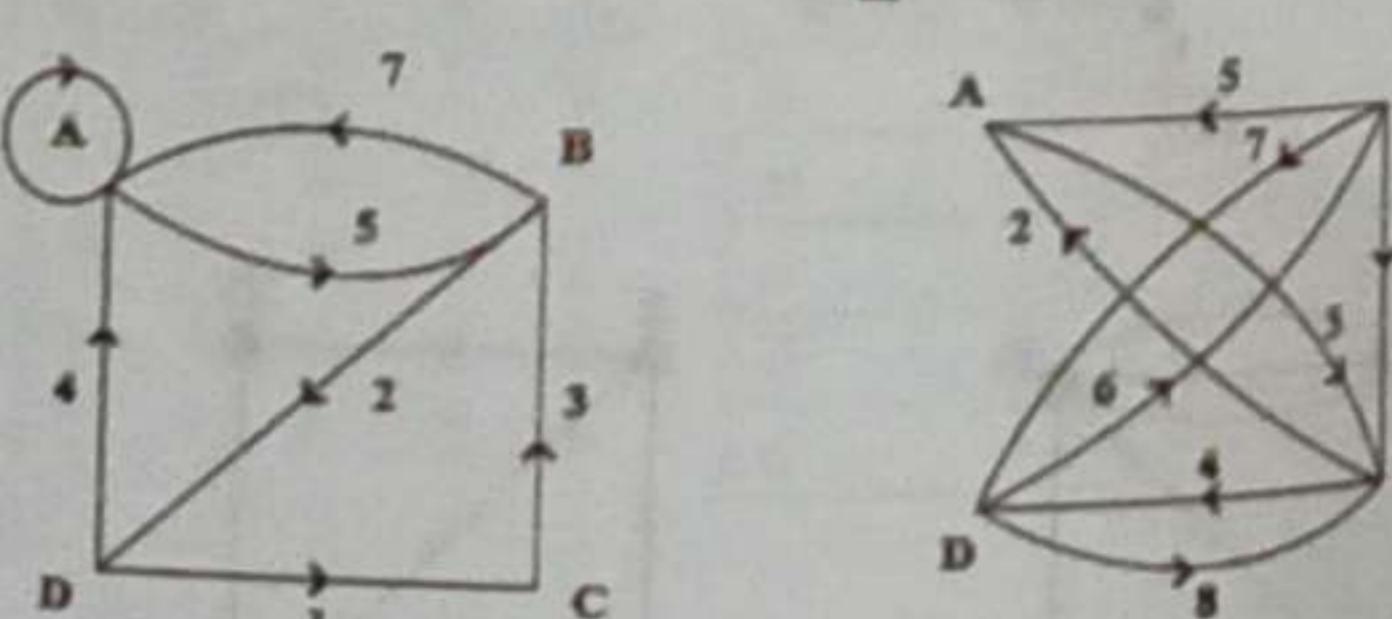


প্রশ্ন ২৮. Draw the weighted graph from the given adjacency matrix.

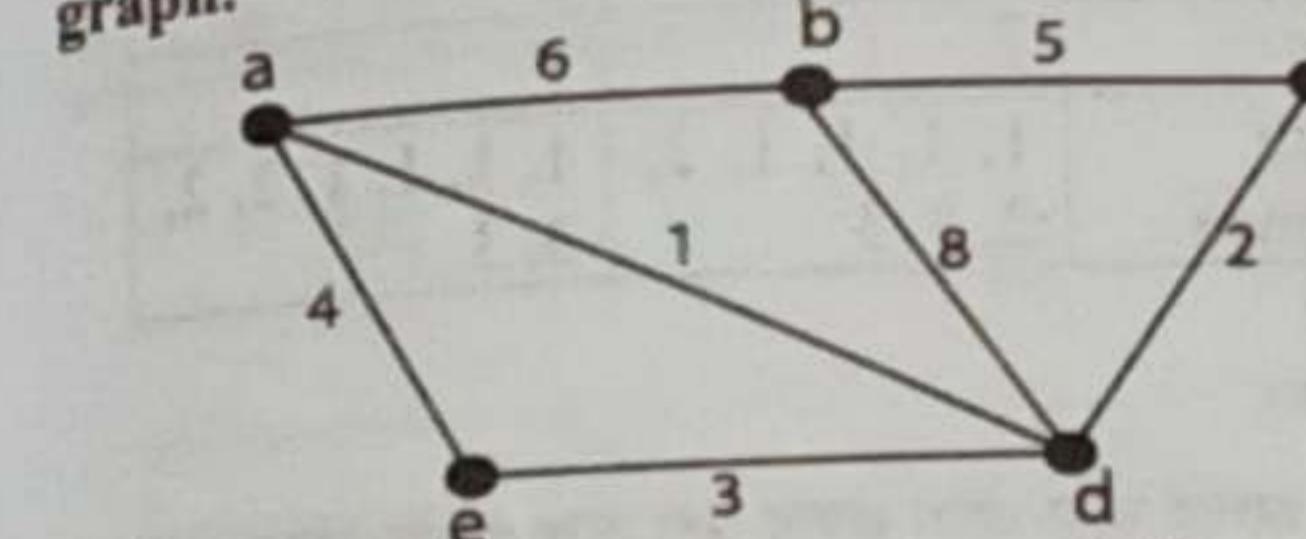
7	5	0	0
7	0	0	2
0	3	0	0
4	0	1	0

0	0	3	0
5	0	1	7
2	0	0	4
0	6	8	0

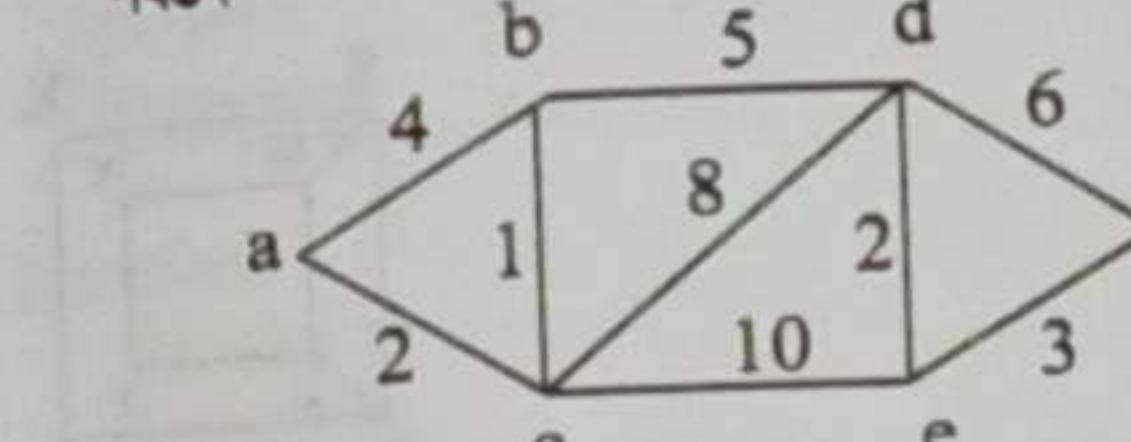
Solⁿ:



উদা- ১০. Find the adjacency matrix of the given graph.



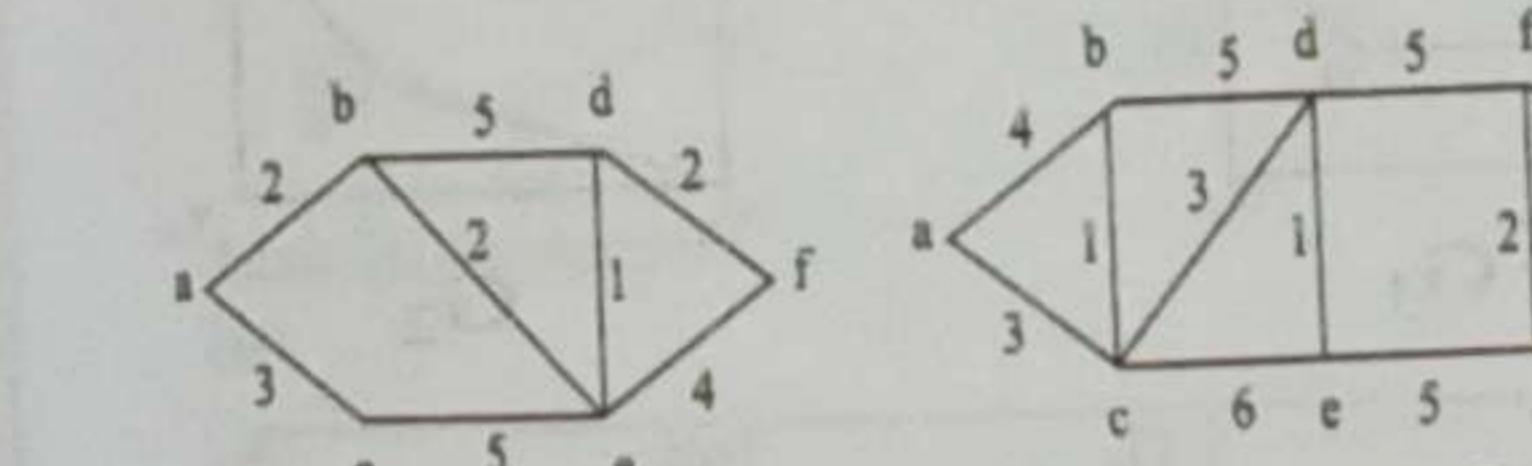
প্রশ্ন ২৯. নিচের graph এর shortest path নিয়ে কোনটা যাবে।



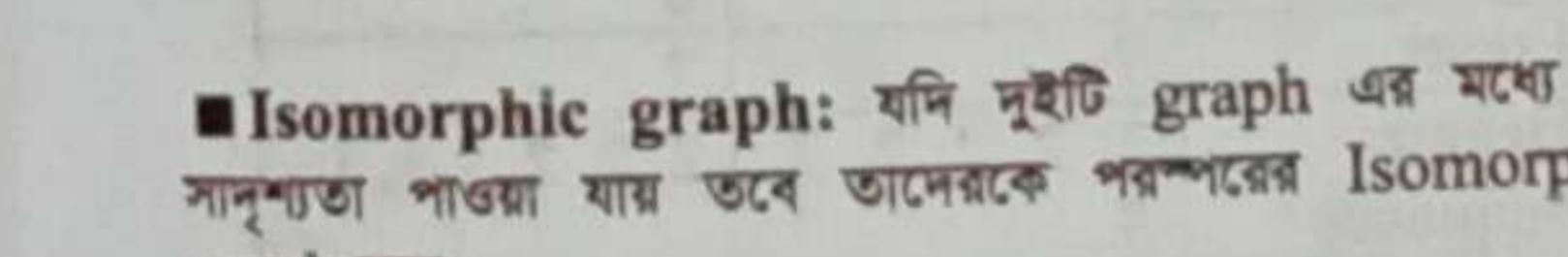
Solⁿ:

- Step-1: a, b = 4
 - a, c = 2
 - a, c, d = 9
 - a, c, b, d = 8
 - a, c, e = 12
 - a, c, b, d, e = 10
 - a, c, d, z = 16
 - a, c, b, d, z = 14
 - a, c, b, d, e, z = 13
- Shortest path is = a, c, b, d, e, z = 13

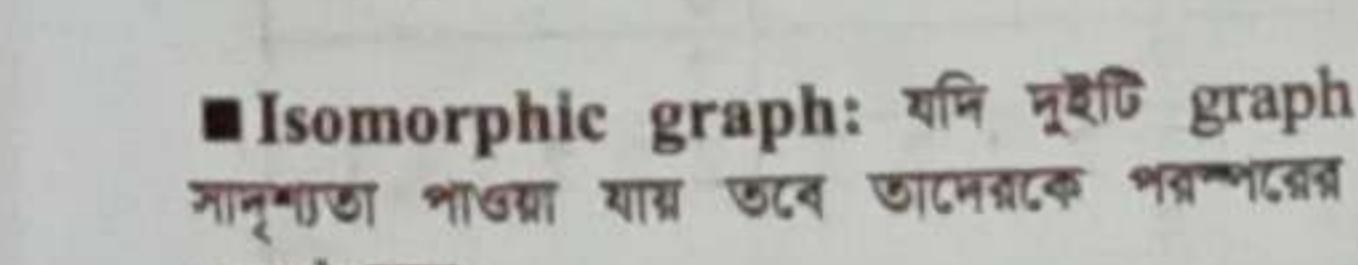
উদা- ১১. Find the shortest path.



Ans: a,b,e,f,g,z

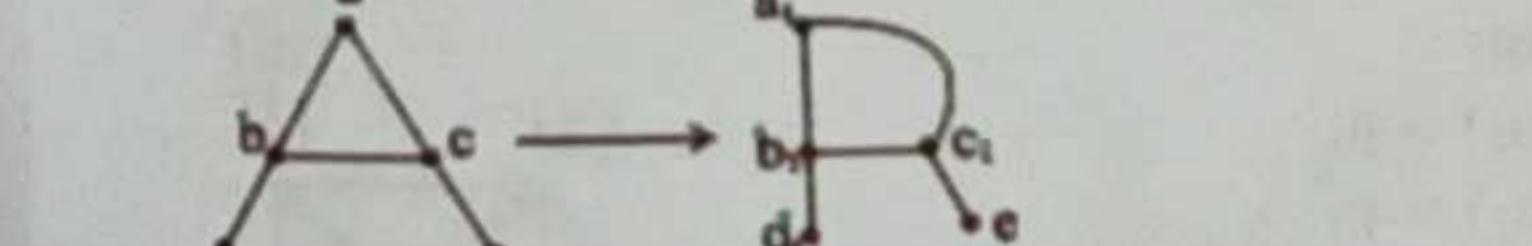


Ans: a,c,d,e,f,g,z



Ans: a,b,e,f,g,z

■ Isomorphic graph: যদি দুইটি graph এর মধ্যে পূর্ণ সামূহিকতা পাওয়া যায় তবে তাদেরকে পরস্পরের Isomorphic graph বলে।



■ Isomorphic graph হওয়ার শর্ত:

Ans: দুইটি graph পরস্পরের Isomorphic হওয়ার শর্তসমূহ

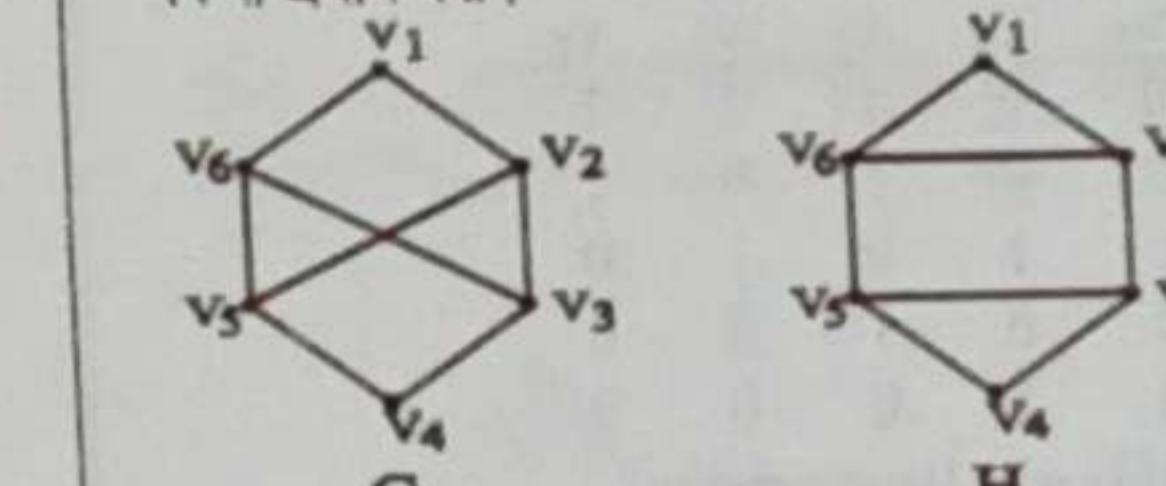
1. Degree of vertex
2. Number of edge
3. Number of vertex
4. Maximum degree

5. Function

- (i) One-to-one function
- (ii) Onto function
6. Simple circuit length
7. Adjacency matrix সমান হতে হবে।
8. Subgraph আছে কিনা

প্রশ্ন ৩০. নিচের graph তালো পরস্পর Isomorphic graph

কিনা প্রমান কর।



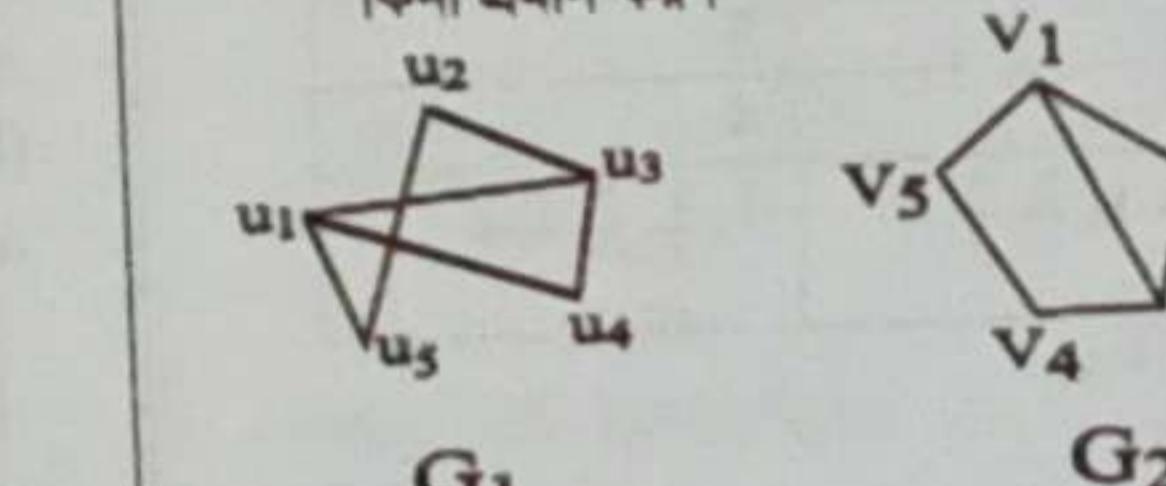
Solution:

	G	H
vertex	6	6
edges	8	8
Degree	2, 2, 3, 3, 3, 3	2, 2, 3, 3, 3, 3
Sequence		
Simple Ckt length	length 3 none	length-3 v1, v2, v6, v1

So, Graph G and H are not Isomorphic.

প্রশ্ন ৩১. নিচের graph তালো পরস্পর Isomorphic graph

কিনা প্রমান কর।



Solution:

	G ₁	G ₂
vertex		

Adjacency Matrix G_1

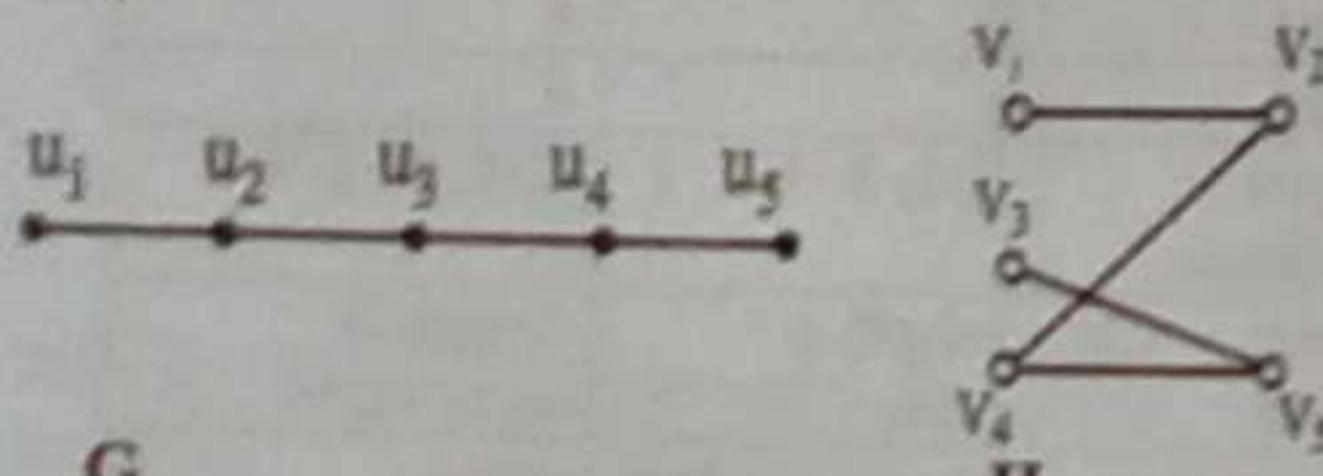
	u_1	u_2	u_3	u_4	u_5
u_1	0	0	1	1	1
u_2	0	0	1	0	1
u_3	1	1	0	1	0
u_4	1	0	1	0	0
u_5	1	1	0	0	0

Adjacency Matrix G_2

	v_1	v_4	v_3	v_2	v_5
v_1	0	0	1	1	1
v_4	0	0	1	0	1
v_3	1	1	0	1	0
v_2	1	0	1	0	0
v_5	1	1	0	0	0

So, Graph G_1 and G_2 are Isomorphic

ex ১২. নিচের pair of graph Isomorphic কির তিহিত কর :

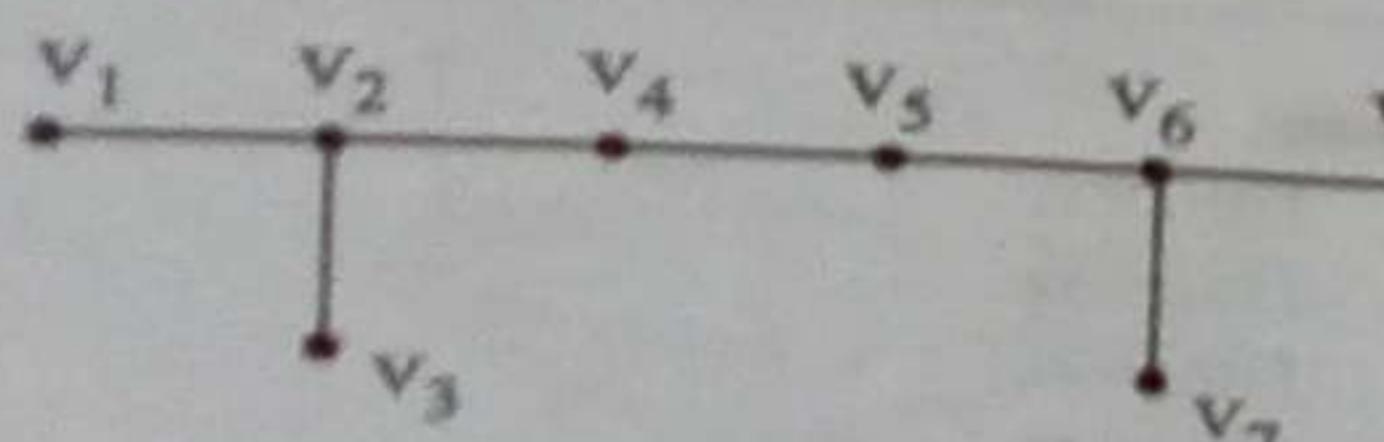
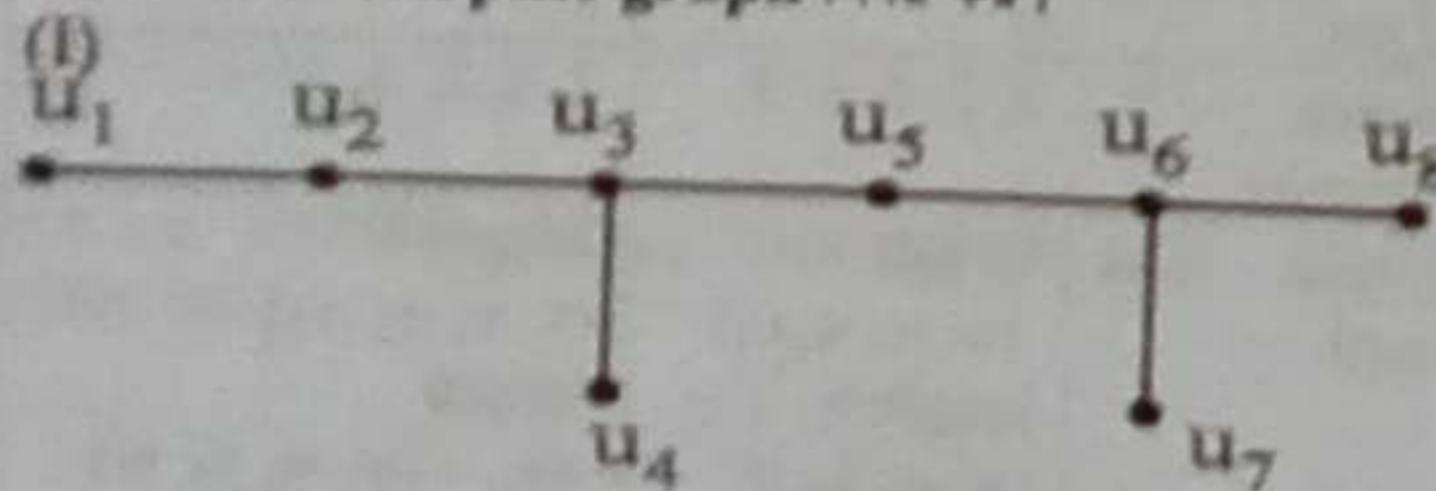
 G

Sol:

	G	H
vertex	5	5
edges	4	4
Degree	1, 1, 2, 2, 2	1, 1, 2, 2, 2
Sequence		

Function: $f(u_1) = v_1$
 $f(u_2) = v_2$
 $f(u_3) = v_3$
 $f(u_4) = v_4$
 $f(u_5) = v_5$ So, graph G এবং H Isomorphic.

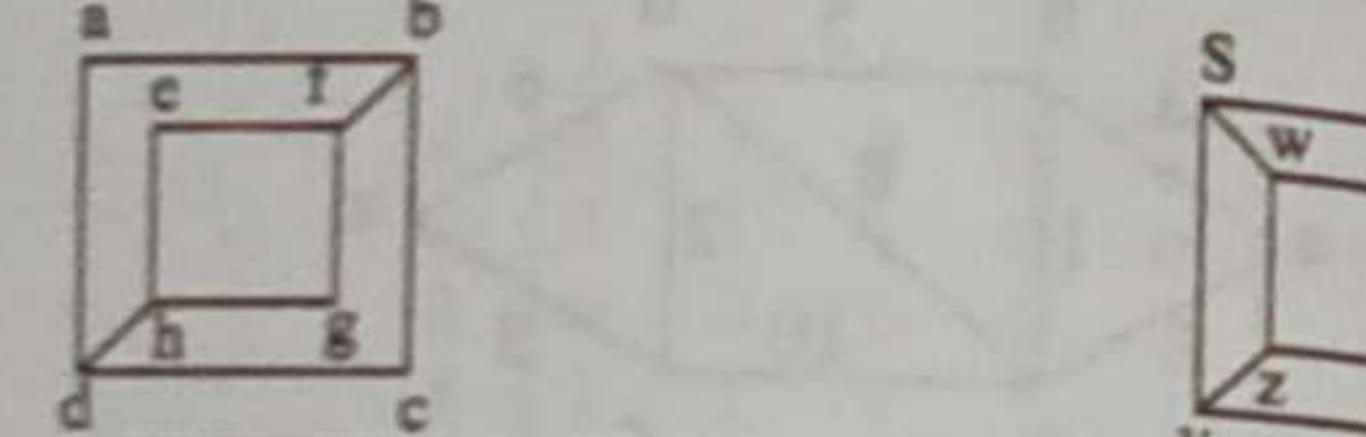
ex ১৩. Isomorphic graph নির্ণয় কর।



	G_1	G_2
Vertex	8	8
Edges	7	7
Degree	1, 1, 1, 1, 2, 2, 3, 3	1, 1, 1, 1, 2, 2, 3, 3
Sequence		

Function:
 $f(u_1) \neq v_1$ Function কর্মসূচী কোন অকার মিল পুরু পাওয়া যাবে না।
 This is not isomorphic graph. (Ans)

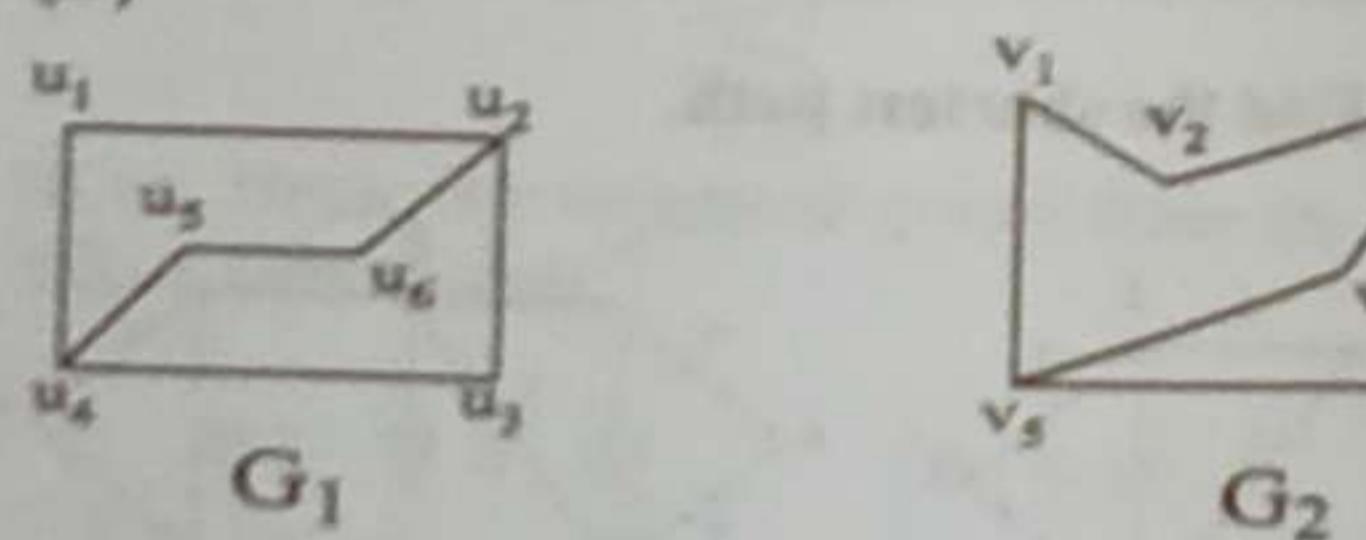
(ii)



	G_1	G_2
vertex	8	8
edges	10	10
Degree	2, 2, 2, 2, 3, 3, 3, 3	2, 2, 2, 2, 3, 3, 3, 3
Sequence		

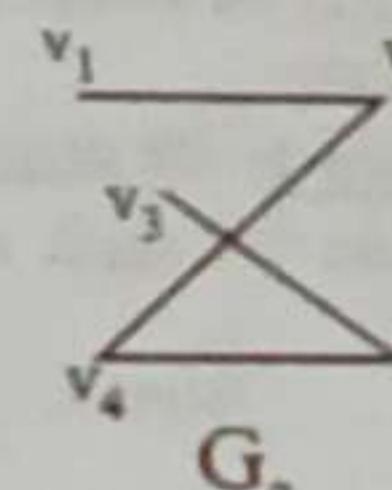
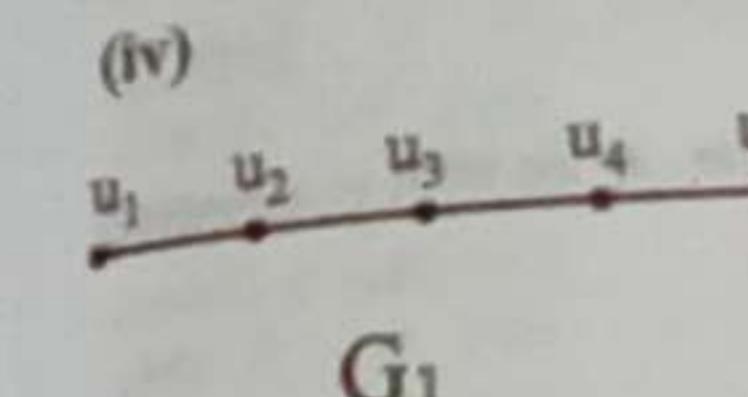
Function:
 $f(a) \neq k$ Function কর্মসূচী one to one function নয় তাই G_1 এবং G_2 isomorphic নয়।

(iii)



	G_1	G_2
vertex	6	6
edges	7	7
Degree	2, 2, 2, 2, 3, 3	2, 2, 2, 2, 3, 3
Sequence		

Function:

 $f(u_1) = v_6$
 $f(u_2) = v_3$
 $f(u_3) = v_4$
 $f(u_4) = v_5$
 $f(u_5) = v_1$
 $f(u_6) = v_2$ সূতরাং G_1 এবং G_2 isomorphic।

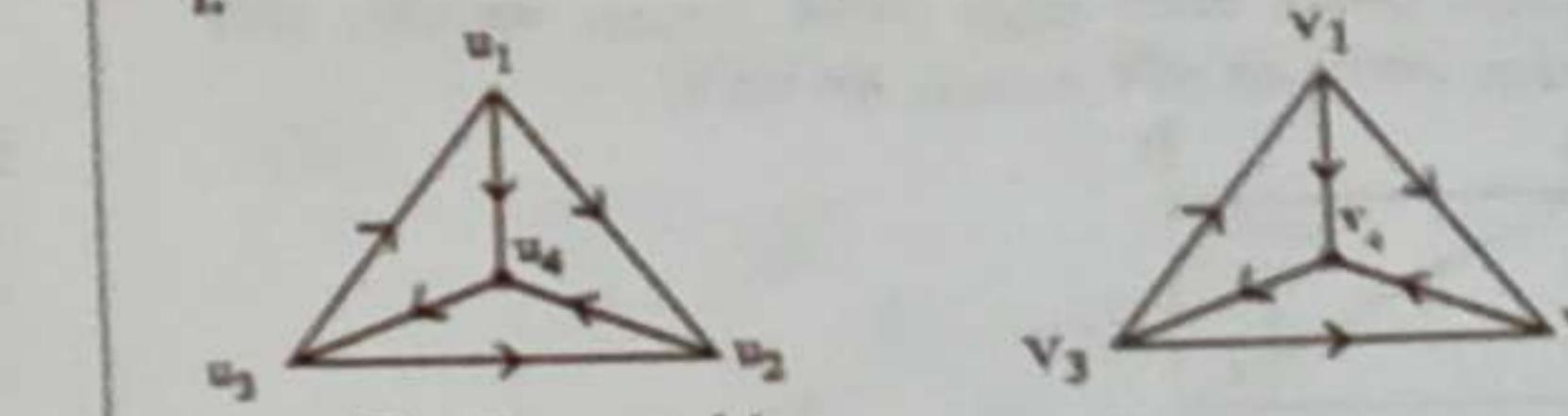
টপ- ২. Are the Simple graphs with the following adjacency matrix isomorphic?

$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$
--------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

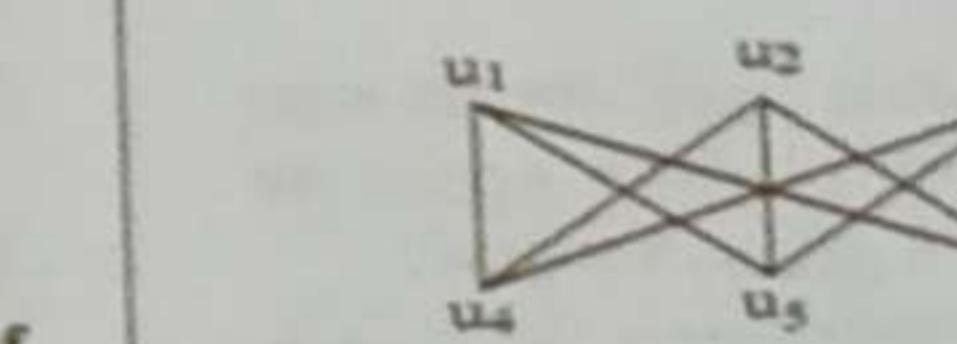
Ans: Not Isomorphic

টপ- ৩. Determine whether the given pair of directed graph is isomorphic.

i.

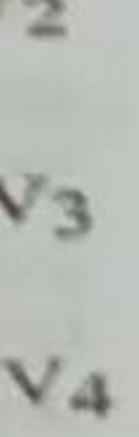


Ans: Yes, Isomorphic



টপ- ৪. Determine whether the given pair of graphs is isomorphic.

i.



ii.



iii.

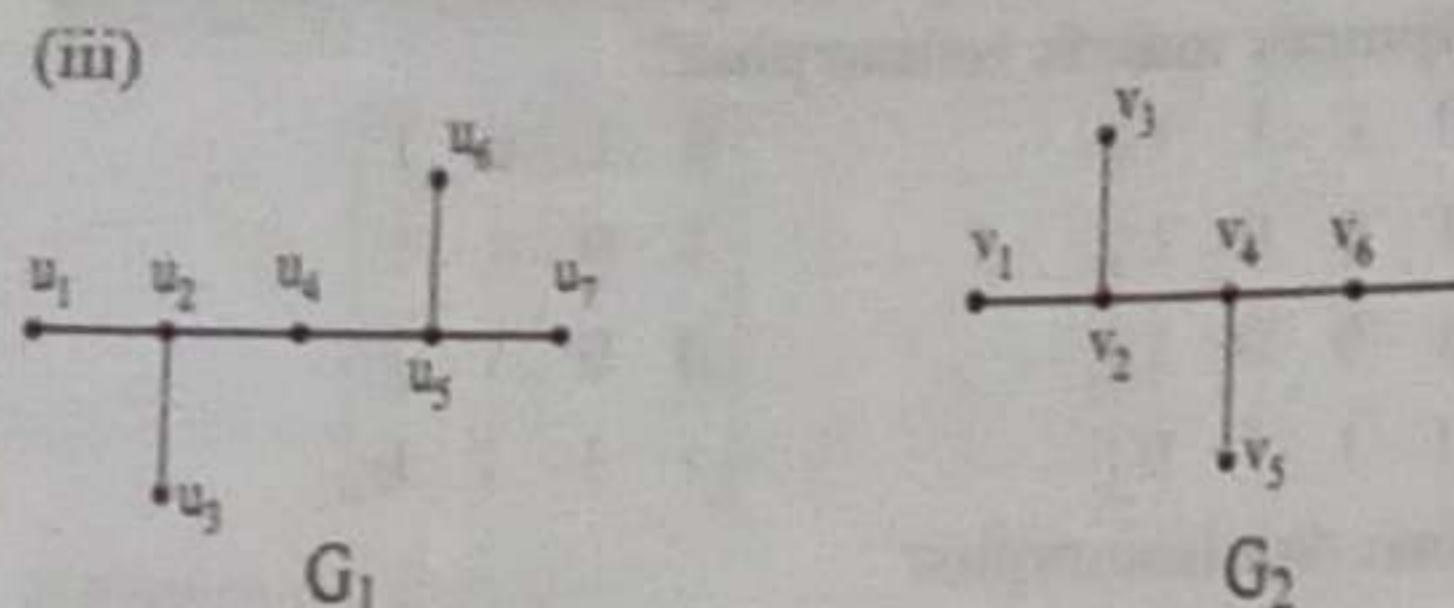


Ans: Isomorphic

iv.

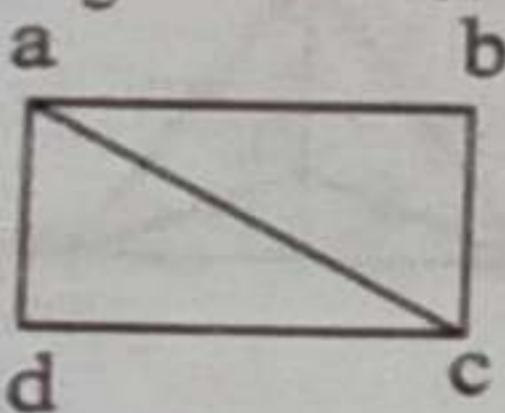


Ans: Isomorphic



Ans: 1 isomorphic , 2-isomorphic , 3 not isomorphic

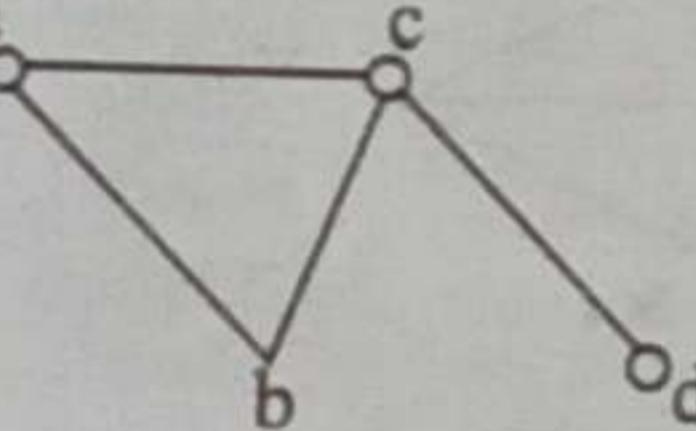
■ Eular graph: Eular graph এমন এক ধরনের Cycle graph দেখানে প্রতিটি edge একবার Access করা যাব। একটি edge একবার এর বেশি Access করা যাব না।



■ Hamilton graph: Hamilton graph এমন এক ধরনের Cycle graph দেখানে প্রতিটি vertex একবার Access করা যাব। একটি vertex একবার এর বেশি Access করা যাব না।

ধৰি, একটি graph $G = (V, E)$ দেখানে vertex (V) = {a, b, c, d} এবং Edges (E) = {a,b}, {a,c}, {b,c}, {c,d} হয়, তাহলে Graph টি অকেন কৰ।

Solution:



Depth First Search (DFS)

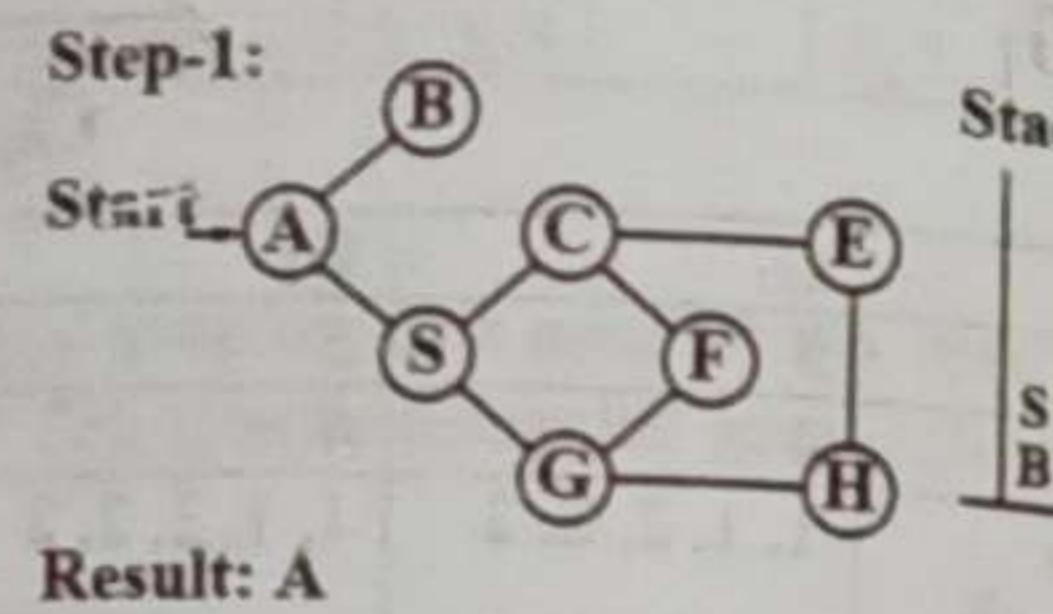
DFS হলো Depth First Search যা একটি tree or Graph এর depth এ search কৰবে। তাৰ অৰ্থ হলো কোন একটি সাইডে search কৰা তাৰ কৰলে সেটা সম্পৰ্ক শৈব কৰে পৰিবৰ্তী side search কৰবে। DFS সম্পৰ্ক কৰাৰ জন্য stack ব্যবহাৰ কৰা হয়। stack LIFO (Last In First Out) পদ্ধতিতে কৰে।

- Rule 1 – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
- Rule 2 – If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
- Rule 3 – Repeat Rule 1 and Rule 2 until the stack is empty.

DFS এৰ উদাহৰণ নিম্নলুপ্ত

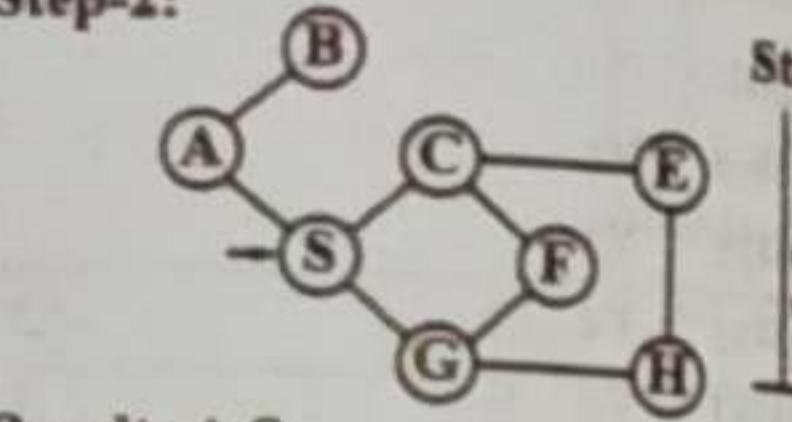
উদাহৰণ:

Step-1: খৰি start node A এবং A এৰ সাথে connected node গুলো stack এ push কৰি।



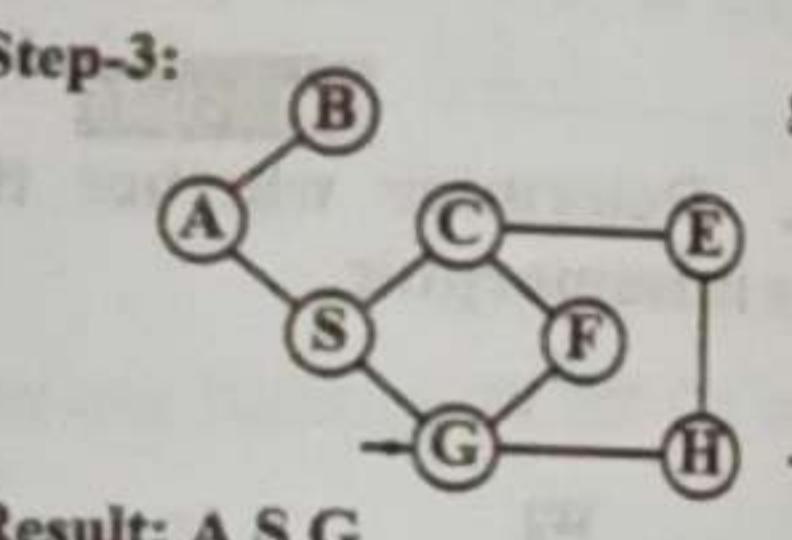
Step-2: S কে

pop কৰে visited result এ add কৰি এবং S এৰ সাথে connected node গুলো stack এ push কৰতে হবে।



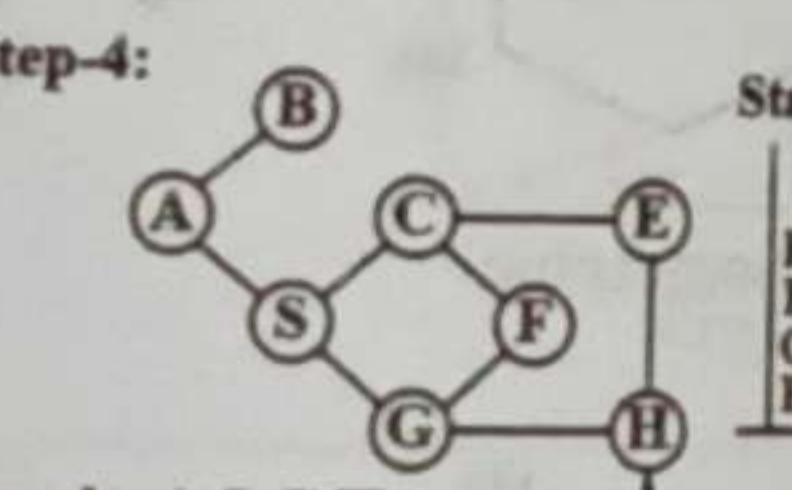
Step-3: G কে

Stack হতে pop কৰে visited result এ add কৰি এবং G এৰ সাথে connected node গুলো stack এ push কৰা হবে।



Step-4: H কে

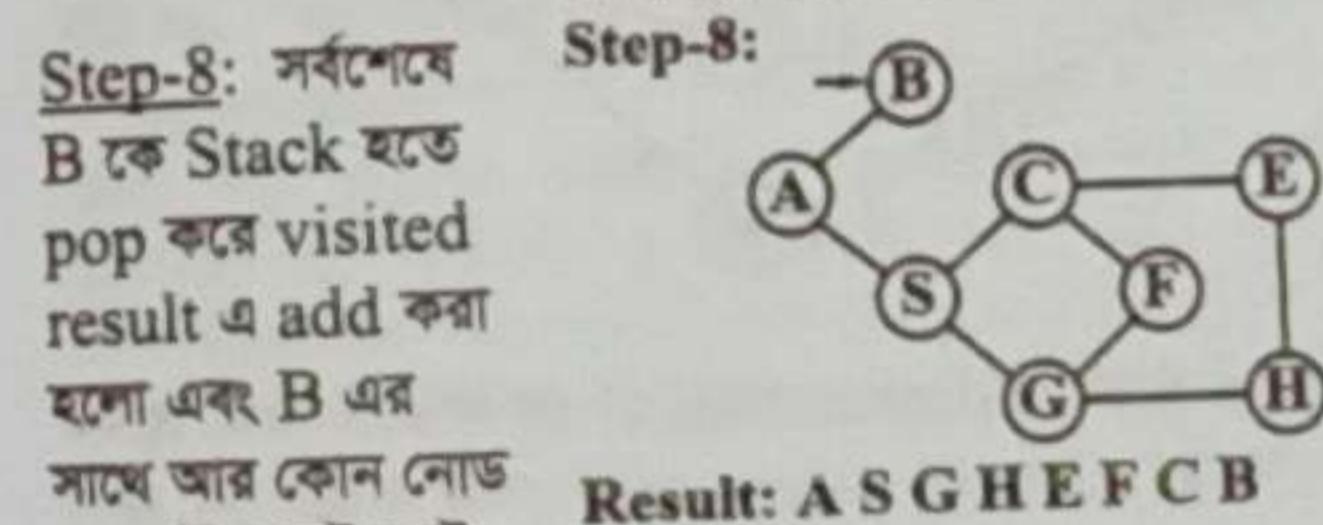
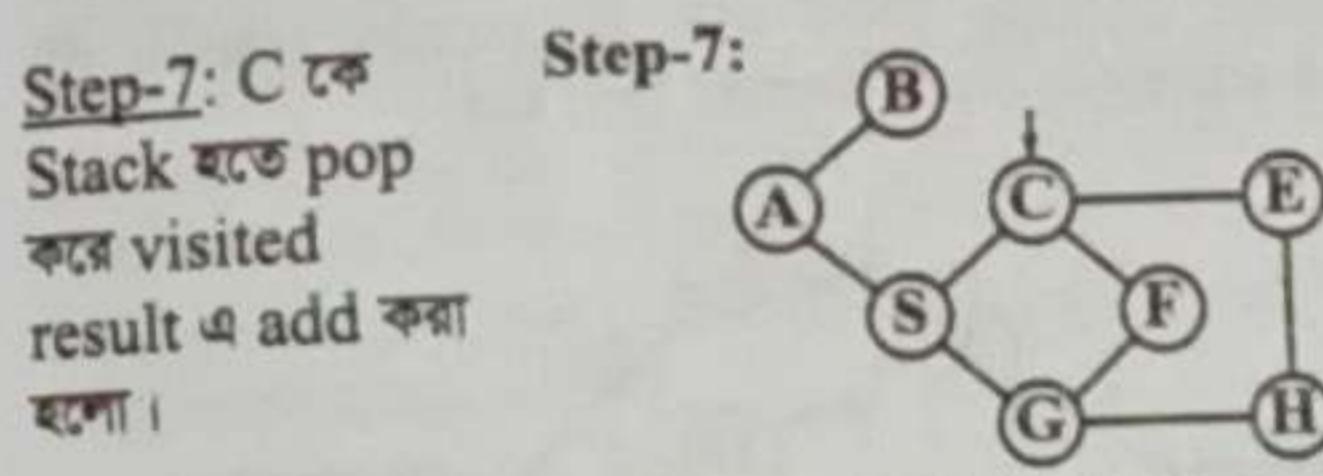
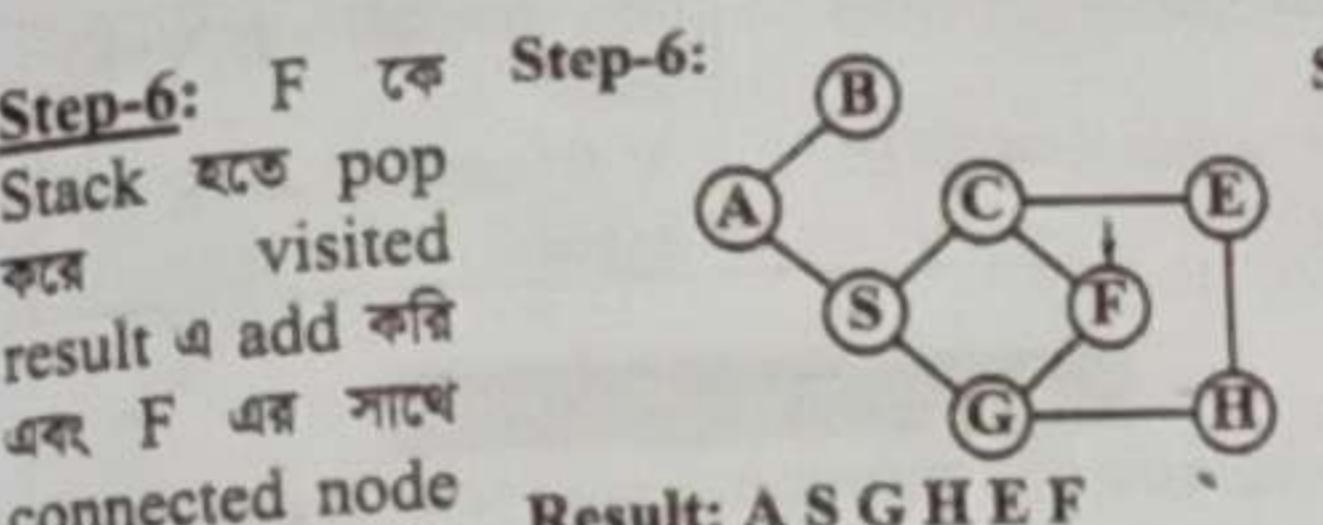
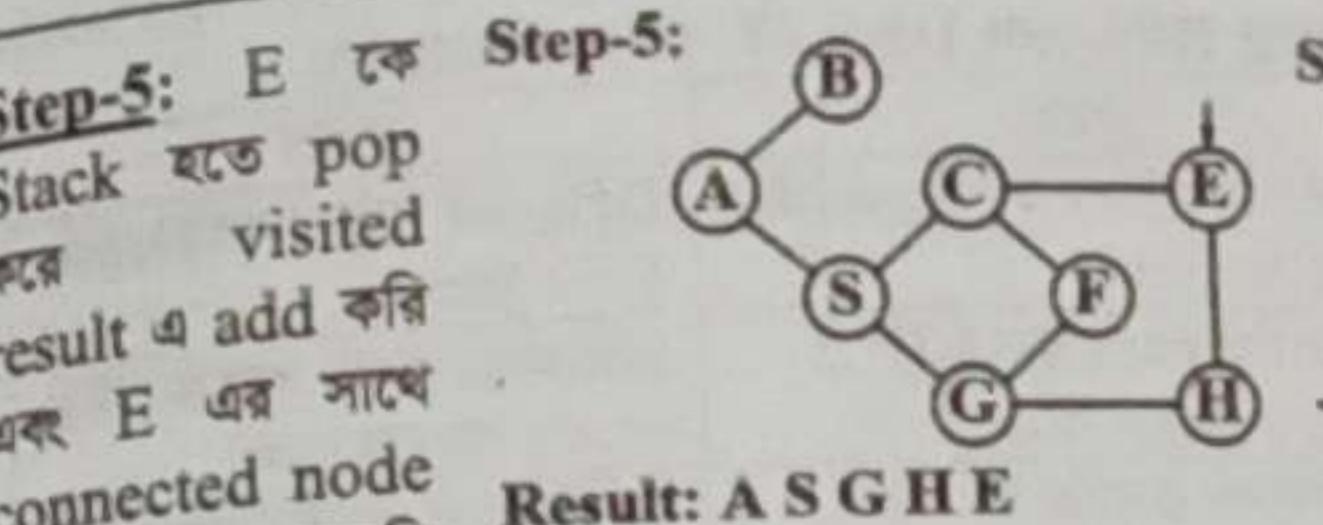
Stack হতে pop কৰে visited result এ add কৰি এবং H এৰ সাথে connected node গুলো stack এ push কৰা হবে।



Step-5: E কে

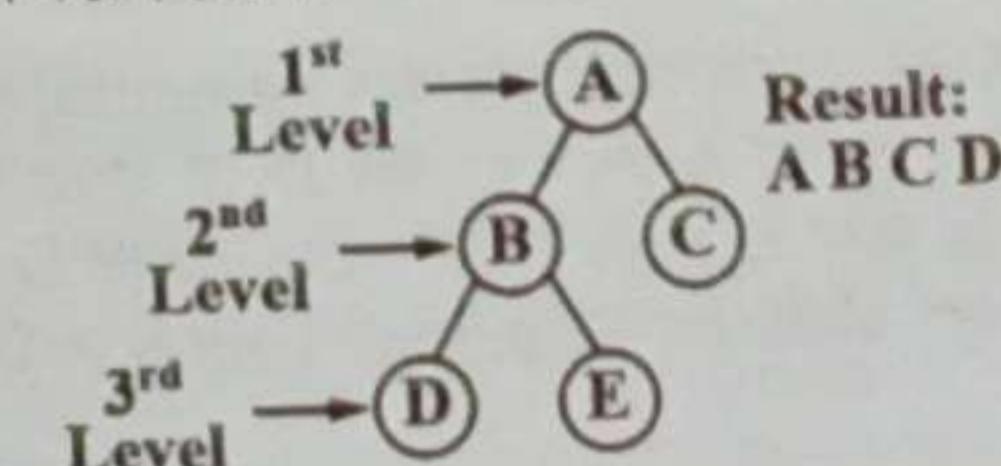
Stack হতে pop কৰে visited result এ add কৰি এবং E এৰ সাথে connected node C যা অলৱেড় stack এ আছে।

Data Structure & Algorithm



BFS

BFS হলো Breadth First Search যা একটি শাফ্ বা প্ৰি এৰ লেভেল ওয়াইজ সার্চ কৰে থাকে। বিএফএস শুধুমাত্ৰ unweighted শাফে কাজ কৰে। যেমন:



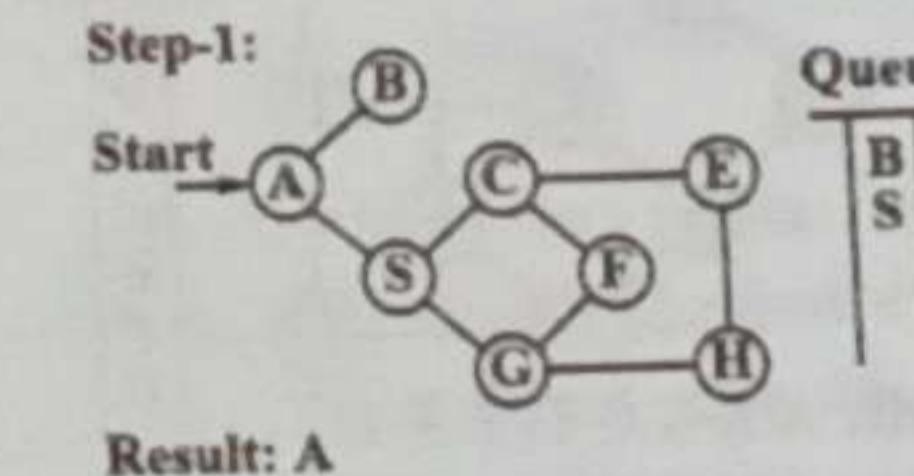
- Rule 1 – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.

- Rule 2 – If no adjacent vertex is found, remove the first vertex from the queue.
- Rule 3 – Repeat Rule 1 and Rule 2 until the queue is empty.

নিচে একটি উদাহৰণের মাধ্যমে দেখানো হলো।

এখানে একটি শাফ দেওয়া আছে। BFS এ ট্ৰাভাৰ্স (Travers) কৰাৰ জন্য আমৰা কিউ (Queue) ব্যবহাৰ কৰবো। এখানে কোন স্টাৰ্টিং নোড (Starting Node) দেওয়া নেই তাই আমৰা অ্যালফাৰেটিকালি A থেকে ট্ৰাভাৰ্স কৰা তাৰ কৰবো।

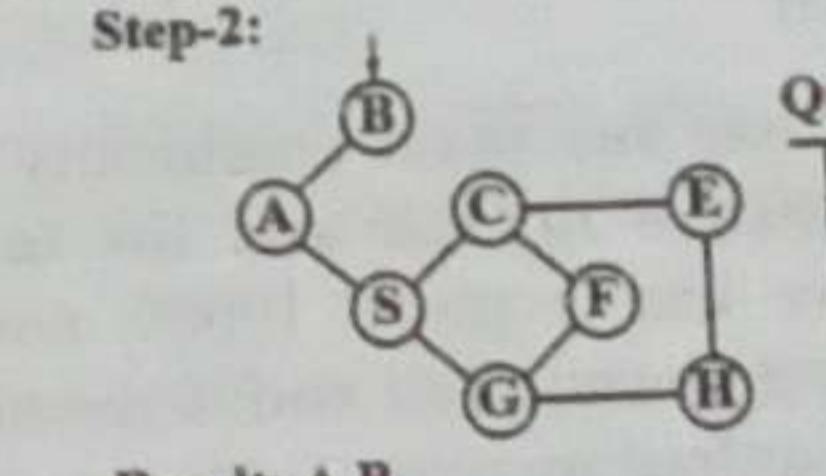
ধৰ-১:



যখন A থেকে তাৰ কৰছি তখন A টা ভিজিটেড রেজাল্টে চলে আসব। এবং A এৰ সাথে কানেক্টেড নোড তলো কিউতে প্ৰবেশ কৰবে।

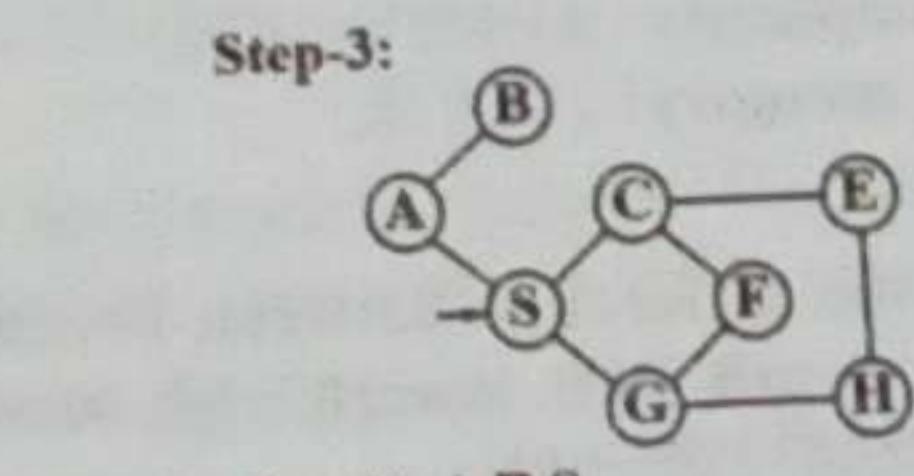
আমৰা জানি, কিউতে যে নোড টি আগে প্ৰবেশ বা IN হয় সেটাই আগে আউট বা OUT হয়, অৰ্থাৎ First in First Out. এবার আমৰা পৰিবৰ্তী নোড ভিজিট (Visit) কৰবো কিউতে যে নোড আগে প্ৰবেশ বা IN কৰেছে।

ধৰ-২: কিউতে প্ৰথমে B আছে তাই ভিজিট হলো এবং এৰ সাথে রিলেটেড নোড তলো ও কিউতে আসবে কিন্তু এৰ সাথে কানেক্টেড নোড তথ্যমাত্ৰ A যা অলৱেড় ভিজিটেড হয়ে গৈছে। তাই বিভিন্ন বাৰ কিউতে প্ৰবেশ কৰবেন।



Result: A B

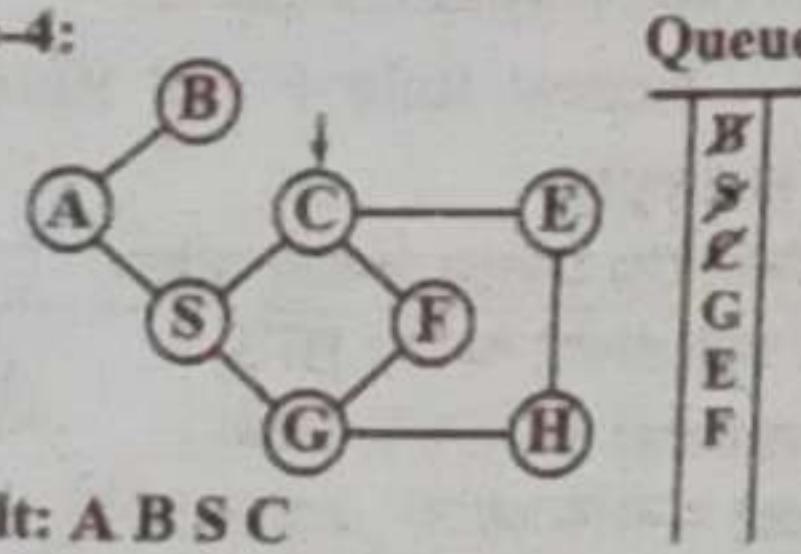
ধৰ-৩: এবাৰ কিউ এৰ পৰিবৰ্তী ভালু S, তাই S ভিজিট হবে এবং এৰ সাথে কানেক্টেড নোড তলো কিউতে এত হবে বা প্ৰবেশ কৰবে। তাই S এৰ সাথে কানেক্টেড E এবং G যা কিউতে এত হয়েছে বা প্ৰবেশ কৰেছে।



Result: A B S

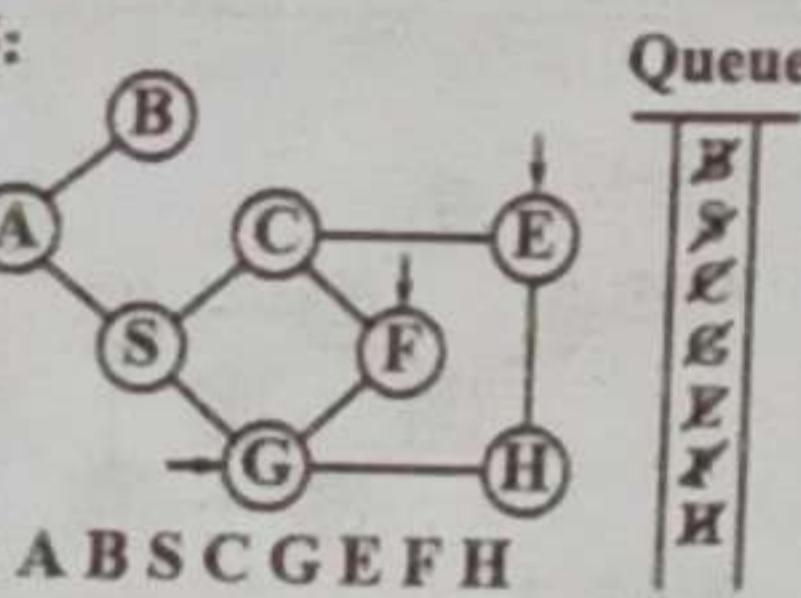
প্রশ্ন-১: এবার C ভিজিট হবে এবং C এর সাথে কানেক্টেড নোড তলো
কিউতে এত হবে বা প্রবেশ করবে।

Step-4:



প্রশ্ন-২: অনুরূপ তাবে পরবর্তী ধাপ তলো ও সম্পূর্ণ হবে যা একটি ধাপে
দেখানো হলো। এটাই ফাইনাল ফলাফল বা রেজাল্ট।

Step-5:



প্রশ্ন ১. Write are the BSF Algorithm? [Ruppur-2018]
Algorithm of BSF:

Step-1: If root.data is equal to search.data then:
return ROOT

Step-2: Else : while DATA not found

If DATA is greater than node.data
then: goto RIGHT subtree
Else: goto LEFT subtree

If DATA found then: return NODE
Step-3: End'while

Step-4: return DATA not found

Step-5: end if

প্রশ্ন ২. Find the time and space complexity of BFS
which has branch 4 branch and the target at
level 5? If cpu can explore 10000 node per
second find the time required and if the memory
1KB find the required memory.[NRCC Exam-2021]

Solution: Branching factor of BFS is $O(b^d)$ where
b is branch and d is the depth of the graph. So the
branching factor is $O(4^5) = 1024$.

Required time $1024/10000 = 102\text{ms}$.

Space Complexity : $O(4^5) = 1024$

Required memory: $1024/1K$.

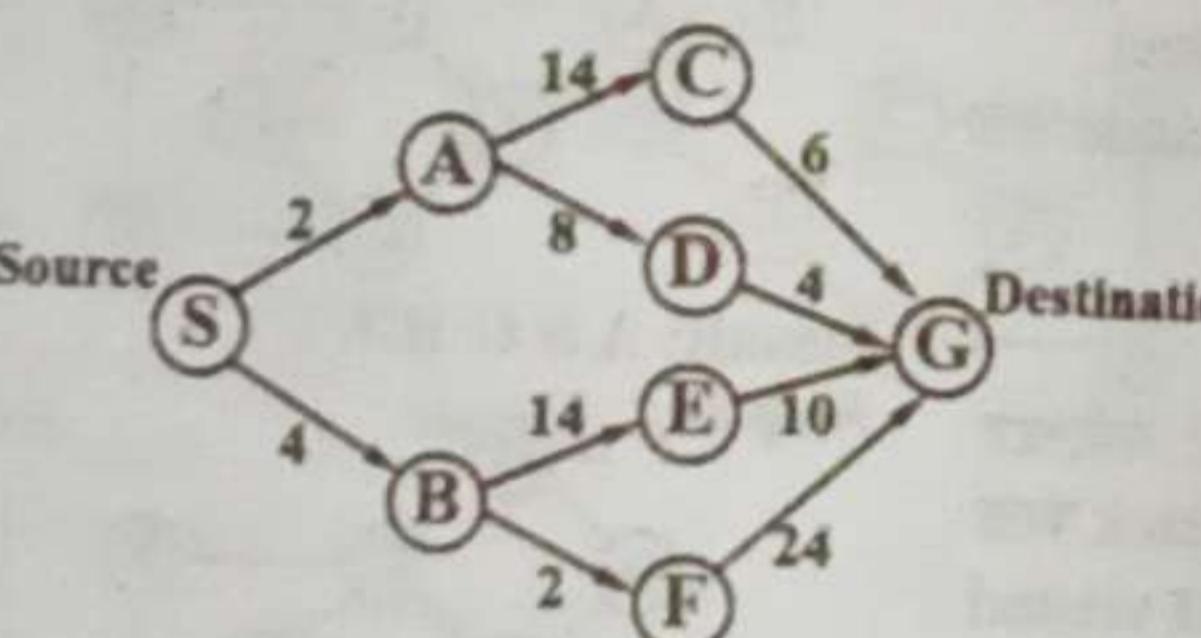
প্রশ্ন ৩. What are the difference between breadth
first search and depth first search? [কৃষি মানবিদ্যার
অধীনে জাতীয় কৃষি প্রশিক্ষণ একাডেমি (নাটা)-২০১৯]

উত্তর: নিচে BFS এবং DFS এর পার্থক্য দেওয়া হল:

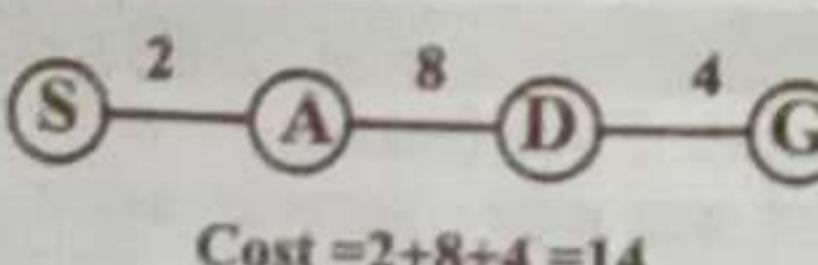
BFS	DFS
BFS, এর পূর্বিক্রম হল Breadth First Search.	DFS, এর পূর্বিক্রম হল Depth First Search.
Shortest path খুঁজতে BFS Queue ব্যবহার করে।	Shortest path খুঁজতে DFS Stack ব্যবহার করে।
ডেস্টিনেশন সোর্সের কাছাকাছি থাকলে BFS better।	ডেস্টিনেশন সোর্সের দিকে দূর থাকলে DFS better।
DFS থেকে slower।	BFS থেকে faster।
টাইম কমপ্লেক্সিটি = $O(V+E)$ যেখানে	টাইম কমপ্লেক্সিটি = $O(V+E)$ যেখানে
V হল vertices এবং E হল edges.	V হল vertices এবং E হল edges.

Greedy search Algorithm

Greedy search হলো minimum cost search algorithm,
যেখানে source থেকে destination এ পৌছাতে সবচেয়ে
মিনিমাম cost এর path বের করবে। যের যাক আমরা airport
থেকে motijheel থেকে চায়, তাহলে আমরা বিভিন্ন way তে যেহে
পারি এর মধ্যে সবচেয়ে minimum way তে motijheel ঘোর
জন্য যে tool choose করতে হবে তা greedy search এ
মাধ্যমে বের করা যাবে। একটি উদাহরণ দেখা যাক:



এখানে Source S থেকে destination G তে যাওয়ার জন্য অনেক
তলো way আছে। Greedy algorithm যেটা করবে তা হলো
minimum cost path find করবে। so lets solve the
problem.



তাই S থেকে A এবং B তে যাওয়া যাবে তবে minimum হলো A
তাই A তে যাবে। তারপর A থেকে C এবং D তে যাওয়া যাবে, যার
মধ্যে minimum হলো D। তাই D তে যাবে। তারপর D এর পরে
destination এবং এটাই হলো Minimum path S A D G।

প্রশ্ন ১. What are the difference between graph
and tree ?[BCS-36, এবারী কল্যাণ ও বৈদেশিক কর্মসূচি অন্তর্ভুক্ত
কর্মসূচি বৃত্তি-প্রশিক্ষণ একাডেমি (নাটা)-২০১৮]

প্রবলেমে কনভার্ট করা যাব এবং এই প্রবলেমের সার্টিফিকেটকে
পলিনমিয়াল টাইমে ভেরিফাই করা যাব।

Turning halting
Problem

Vertex covering
Problem

Shortest path
Problem

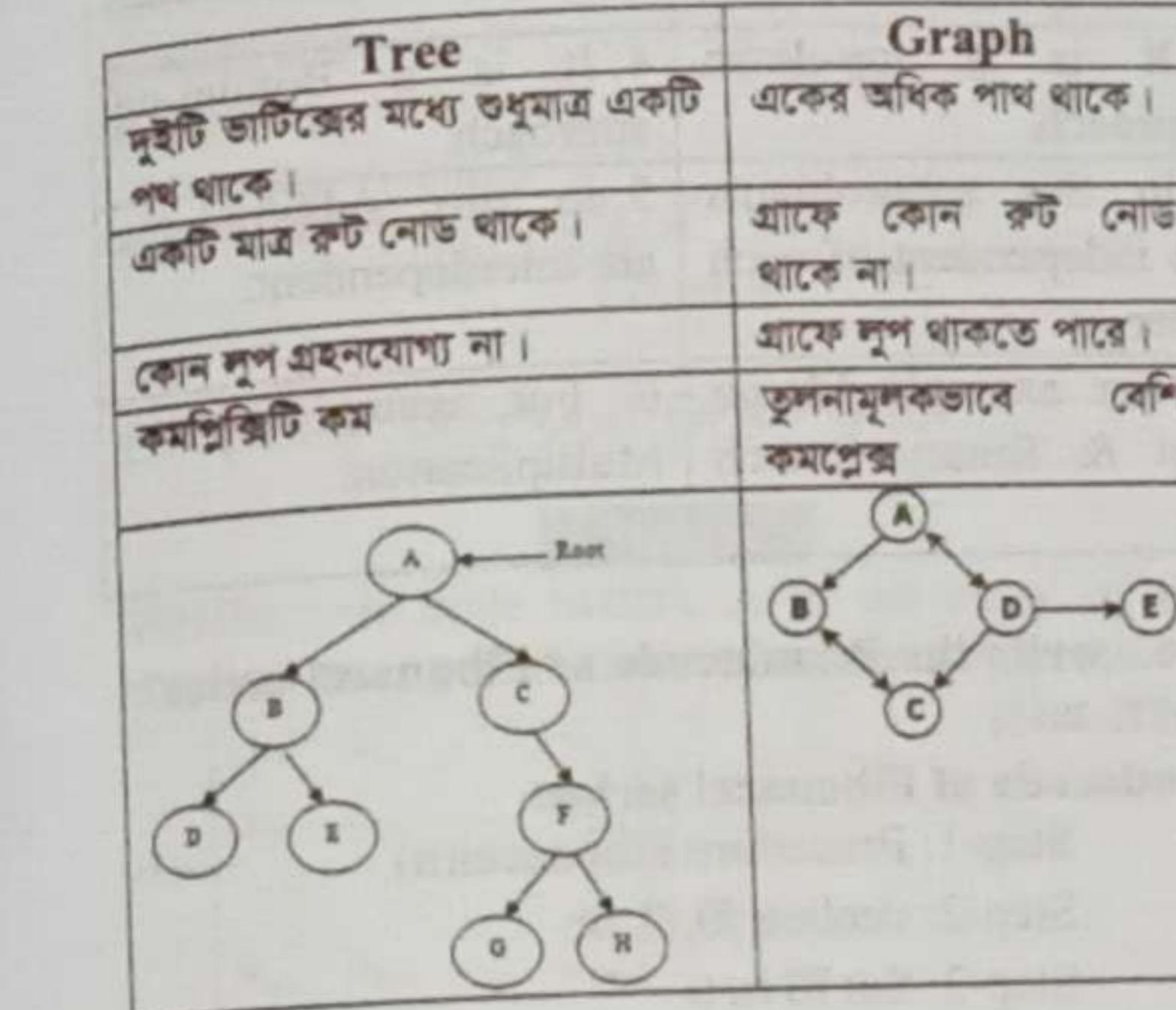
NP-Hard

NP-Complete

NP

P

Here $P \neq NP$



প্রশ্ন ২. Which numerical method are use for
solving linear system? Explain any one? [BCS-35]

পলিনমিয়াল স্টেটেমকে সমাধানের জন্য দুইটি মেথড ব্যবহার করা যাব।

Direct methods: রাউটঅফ ট্যুটির অনুপ্রযোগিতে এ মেথড
সীমাবদ্ধ সংখ্যাক স্টেপের মধ্যে সঠিক সমাধান দেয়।

Iterative methods: এই মেথড বিশেষ, খুব বড় ম্যাট্রিক্সের সাথে
জড়িত সমস্যার জন্য কার্যকর।

প্রশ্ন ৩. What is NP-hard, NP-complete and NP
with example? [BCS-35]

এমন কিছু প্রবলেম আছে যেতো সলভ করতে গেলে সবথেকে
শক্তিশালী কল্পিতারাও হাজার হাজার বছর লাগিয়ে দিবে। বড় বড়
কল্পিতার বিজ্ঞানীরা যখন দিন-রাত চিন্তা করেও এগুলো সলভ করতে
গোরবেন্না তখন তারা এই প্রবলেমগুলোকে কিছু ক্যাটাগরিতে ভাগ
করে। সেগুলোকেই আমরা NP-complete, NP-hard ইত্যাদি
নামে ডিনি।

১. P বা পলিনমিয়াল ক্যাটাগরি প্রবলেমকে ডিটারমিনিস্টিক টুরিং
মেশিন দিয়ে পলিনমিয়াল টাইমে সলভ করা যাব। যেমন শ্চেষ্ট পথ
প্রবলেম, ন্যাপস্যাক প্রবলেম। শুধু যে সেগুলো সলভ করা যাব সেটা না,
একটা "সার্টিফিকেট" দিলে সেটা পলিনমিয়াল টাইমে ভেরিফাই ও করা
যাব।

২. NP প্রবলেমকে ডিটারমিনিস্টিক টুরিং মেশিন দিয়ে পলিনমিয়াল
টাইমে ভেরিফাই করা যাব। কিন্তু পলিনমিয়াল টাইমে সলভ করা যাব কি
যাব না সেটা প্রমাণ করা সম্ভব হ্যানি। যেমন হাইমিটন পথ প্রবলেম।

৩. এই ক্যাটাগরির প্রবলেমগুলো "অস্ত এবং NP প্রবলেমের মতো"
কঠিন। NP hard প্রবলেমকে অনেক সময় NP ক্যাটাগরিতে ফেলা
যায়না কারণ সেটা পলিনমিয়াল টাইমে ভেরিফাই করা যাব না। NP hard
কে পলিনমিয়াল টাইমে সলভ করা যায়না, ভেরিফাই করা যাব না।

৪. NP complete: একটা প্রবলেম NP-complete হবে কেবল
যদি প্রবলেমটা NP-hard হব এবং সেটা NP ক্যাটাগরীতেও পড়ে।

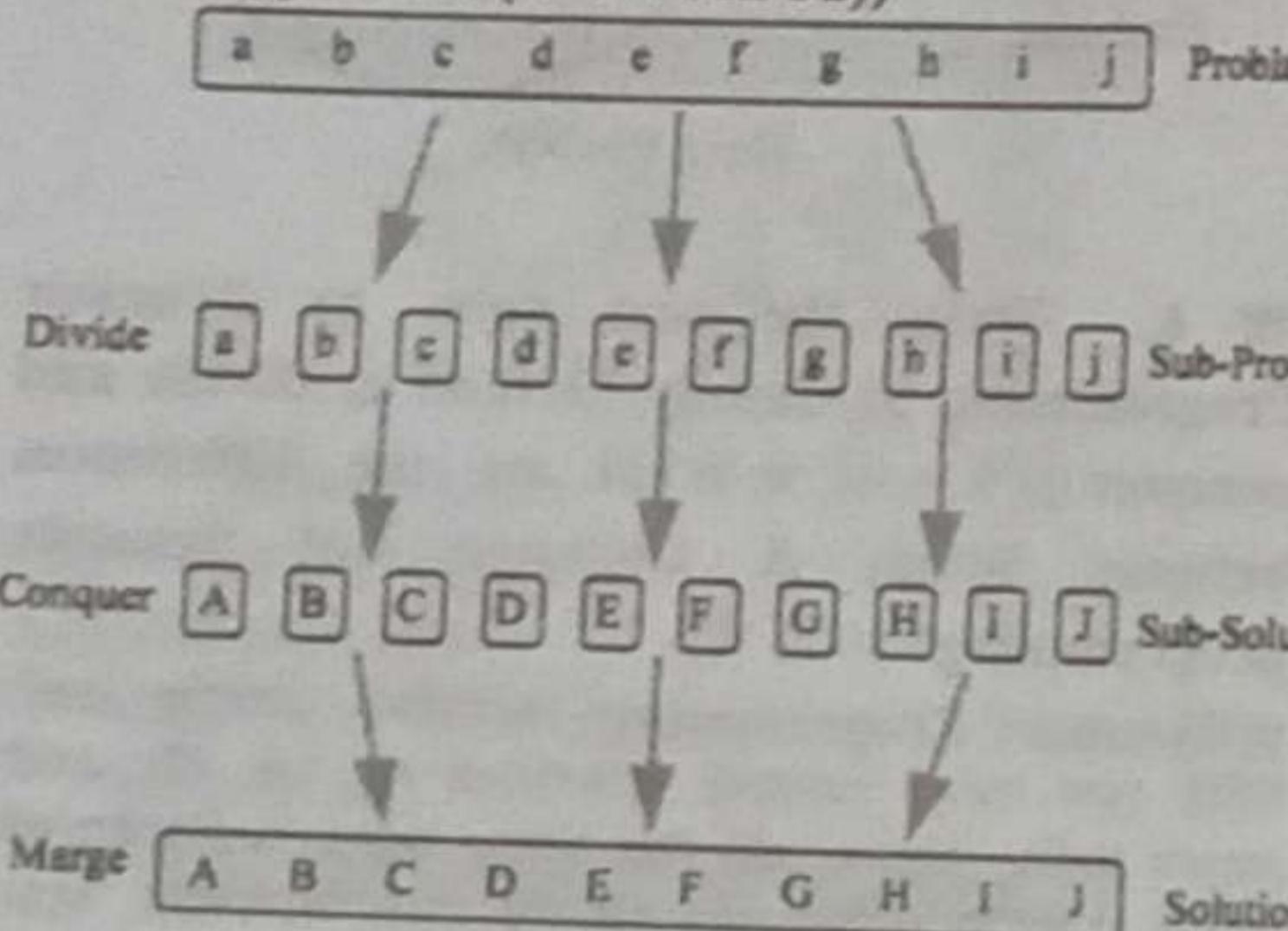
NP complete প্রবলেমকে পলিনমিয়াল টাইমে অন্য যেকোনো NP
complete প্রবলেমকে পলিনমিয়াল টাইমে ভেরিফাই করা যাব।

c) Divide and conquer: "Divide and Conquer" এর
পক্ষতি টা হলো-একটা সমস্যা (problem) কে অনেকগুলো
সমস্যাগুলো (sub problem) এ বিভক্ত (divide) করা এবং তারপর
মূল সমস্যার (problem) সমাধান করার জন্য sub problem

অলোকে recursively সমাধান করা এবং sub problem এর সমাধান অলোকে একত্রিত করা। সুতরাং, "Divide and Conquer" দিয়ে ধাপ অনুসরণ করে:

- একটি problem কে কঠভঙ্গা sub problem এ বিভক্ত করা;
- sub problem অলোকে recursively সমাধান করে conquer করা;
- "original problem" সমাধান এর জন্য sub problem এর সমাধান অলোকে combine করা।

Broadly, we can understand divide-and-conquer approach in a three-step process.(lower case to upper case (letter ascii-32))



নিম্ন Divide & Conquer Method এবং Dynamic Programming এর মাঝে পার্থক্য দেখো হলো:

Divide & Conquer Method	Dynamic Programming
Divide and Conquer" দিয়ে ধাপ অনুসরণ করে:	1. It involves the sequence of four steps: <ul style="list-style-type: none"> Characterize the structure of optimal solutions. Recursively defines the values of optimal solutions. Compute the value of optimal solutions in a Bottom-up minimum. Construct an Optimal Solution from computed information.
• একটি problem কে কঠভঙ্গা sub problem এ বিভক্ত করা;	Step-1: START
• sub problem অলোকে recursively সমাধান করে conquer করা;	Step-2: Input Number N
• "original problem" সমাধান এর জন্য sub problem এর সমাধান অলোকে combine করা।	Step-3: If N=2 then: print "co-prime" go to Step-11 Step-4: D<=2 Step-5: Q <= N/D Step-6: R <= N - Q*D Step-7: If R = 0 then: go to Step-10 Step-8: D<= D+1 Step-9: if D<= N/2 then: go to Step-5 Step-10: If R= 0 then: print "Not Prime" Else Print "prime" Step-11: STOP
2. It is Recursive.	2. It is non-Recursive.
3. It does more work on subproblems and hence has more time	3. It solves subproblems only once and then stores in the table.

প্রশ্ন ৮. Find time and space complexity like below pseudo code

For 1 to n

 For j=1 to 3

 For i=j to n

 Count ++

Complexity: $O(3)*O(n)*O(1) = O(n)$

Matrix

Matrix: An $m \times n$ matrix A is an array of $m \cdot n$ numbers arranged in m rows and n columns as

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Addition and Subtraction of Matrix:

শর্ত : অথবা Matrix এর Row এবং Column এর element সংখ্যা সমান হতে হবে।

উদাঃ ১: Let $A = \begin{bmatrix} 1 & -2 & 3 \\ 0 & 4 & 5 \end{bmatrix}$ and $B = \begin{bmatrix} 3 & 0 & -6 \\ 2 & -3 & 1 \end{bmatrix}$ হলে
(i) $A+B=?$ (ii) $A-B=?$ (iii) $3A=?$

Solⁿ:

$$\begin{aligned} (i) A+B &= \begin{bmatrix} 1+3 & -2+0 & 3+(-6) \\ 0+2 & 4+(-3) & 5+1 \end{bmatrix} = \begin{bmatrix} 4 & -2 & -3 \\ 2 & 1 & 6 \end{bmatrix} \\ (ii) A-B &= \begin{bmatrix} 1-3 & -2-0 & 3-(-6) \\ 0-2 & 4-(-3) & 5-1 \end{bmatrix} = \begin{bmatrix} -2 & -2 & 9 \\ -2 & 7 & 4 \end{bmatrix} \\ (iii) 3A &= \begin{bmatrix} 3.1 & 3.(-2) & 3.3 \\ 3.0 & 3.4 & 3.5 \end{bmatrix} = \begin{bmatrix} 3 & -6 & 9 \\ 0 & 12 & 15 \end{bmatrix} \end{aligned}$$

Self Study

Let $A = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 2 & -3 \\ 3 & 4 & 0 \end{bmatrix}$ and $B = \begin{bmatrix} 3 & 4 & -1 \\ 1 & -3 & 0 \\ -1 & 1 & 2 \end{bmatrix}$ হলে
(i) $A+B=?$ (ii) $A-B=?$ (iii) $5A=?$

Matrix Multiplication:

শর্ত : অথবা Matrix এর Column সংখ্যা ২য় Matrix এর Row সংখ্যা সমান হতে হবে।

উদাঃ ২: $A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 0 & -4 \\ 3 & 2 & 6 \end{bmatrix}$ হলে $AB=?$

Solⁿ:

$$\begin{aligned} A &= \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & -4 \\ 3 & 2 & 6 \end{bmatrix} = \begin{bmatrix} 1.2+3.3 & 1.0+3.2 & 1.(-4)+3.6 \\ 2.2+4.3 & 2.0+4.2 & 2.(-4)+4.6 \end{bmatrix} \\ &= \begin{bmatrix} 11 & 6 & 14 \\ 16 & 8 & 16 \end{bmatrix} \end{aligned}$$

উদাঃ ৩: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$ হলে $AB=?$

Solⁿ:

$$AB = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

উদাঃ ৪: $A = \begin{bmatrix} 1 & 0 & 4 \\ 2 & 1 & 1 \\ 3 & 1 & 0 \\ 0 & 2 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 4 \\ 1 & 1 \\ 3 & 0 \end{bmatrix}$ হলে $AB=?$

Solⁿ:

$$AB = \begin{bmatrix} 1.2+0.1+4.3 & 1.4+0.1+4.0 \\ 2.2+1.1+1.3 & 2.4+1.1+1.0 \\ 3.2+1.1+0.3 & 3.4+1.1+0.0 \\ 0.2+2.1+2.3 & 0.4+2.1+2.0 \end{bmatrix} = \begin{bmatrix} 14 & 4 \\ 8 & 9 \\ 7 & 13 \\ 8 & 2 \end{bmatrix}$$

উদাঃ ৫: $B = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$ হলে $AB=BA=?$

Solⁿ:

$$AB = \begin{bmatrix} 3 & 2 \\ 5 & 3 \end{bmatrix} \text{ and } BA = \begin{bmatrix} 4 & 3 \\ 3 & 2 \end{bmatrix} \quad AB \neq BA$$

উদাঃ ৬: Find the join and meet of the zero one matrix

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\text{Join } A \vee B = \begin{bmatrix} 1 \vee 0 & 0 \vee 1 & 1 \vee 0 \\ 0 \vee 1 & 1 \vee 1 & 0 \vee 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\text{Meet } A \wedge B = \begin{bmatrix} 1 \wedge 0 & 0 \wedge 1 & 1 \wedge 0 \\ 0 \wedge 1 & 1 \wedge 1 & 0 \wedge 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

NB: Zero one matrix এর "AND", "OR" operation

Vector: An n-element vector V is a list of n numbers.
As $V=(v_1, v_2, \dots, v_n)$

উপর 7 : $U=(1, -3, 4, 5)$, $V=(2, -3, -6, 0)$ and $W=(3, -5, 2, -1)$ হলে (i) $U.V=?$ (ii) $U.W=?$ (iii) $V.W=?$ (iv) $|U|=?$

$$(i) U.V = 1.2 + (-3).(-3) + 4.(-6) + 5.0 = -13$$

$$(ii) U.W = 1.3 + (-3).(-5) + 4.2 + 5.(-1) = 21$$

$$(iii) V.W = 2.3 + (-3).(-5) + (-6).2 + 0.(-1) = 9$$

$$(iv) |U| = \sqrt{1^2 + (-3)^2 + 4^2 + 5^2} = \sqrt{51}$$

Sparse Matrix: একটি স্পার্স ম্যাট্রিক্স এমন একটি ম্যাট্রিক্স যেখানে অনেকগুলি বা বেশিরভাগ উপাদানের শূন্য হাল থাকে। এটি একটি ম্যাট্রিক্সের বিপরীতে যেখানে অনেকগুলি বা বেশিরভাগ উপাদানের একটি শূন্য-হাল থাকে: স্পার্স ম্যাট্রিক্সগুলি কম্পিউটার বিজ্ঞানের নির্দিষ্ট উপায়ে ব্যবহার করা হয় এবং তাদের ব্যবহার সম্পর্কিত চোটি বিশ্বেতে এবং স্টোরেজ প্রোটোকল এবং কৌশল রয়েছে।

Example:

0	0	1	2
1	0	0	3
1	0	0	0

Why to use sparse matrix instead of simple matrix?

Storage: সেহেতু এইধৰনে নন জিরো ইলিমেন্ট কম থাকে তাই নন জিরো ইলিমেন্ট সংরক্ষণ করতে কম মেমরির ধরণেজন হয়।

Computing Time:

সেহেতু অপারেশনের ক্ষেত্রে তথ্যাবলী ইলিমেন্ট প্রিসেটিং করতে হয়, সেই জন্য কম্পিউটিং টাইম কম হয়ে। স্পার্স ম্যাট্রিক্স বিপ্রজেক্টেশনের ছুটি আরে ব্যবহার করলে অনেক মেমরি লস হয়, কারণ নন জিরো ইলিমেন্টের কোন ব্যবহার নেই ব্যক্তিক পকে। জিরো ও নন জিরো ইলিমেন্ট সংরক্ষণের পরিবর্তে তথ্যাবলী নন জিরো ইলিমেন্ট সংরক্ষণ করবে। এর অর্থ হলো টিপ্প (ROW,COLUMN,VALUE) অ-শূন্য উপাদান সংরক্ষণ করা। স্পার্স ম্যাট্রিক্সে আমরা দুইভাবে বিপ্রজেক্টেশন করতে পারি।

1. Array Representation

2. Linked list Representation

Array Representation: ছুটি আরে এর মাধ্যমে স্পার্স ম্যাট্রিক্স বিপ্রজেক্টেশনে আমরা সিস্টি সাবি করবার করবে। সেজলো হলো,

Row: index of row, where non-zero elements is located.

Column: index of column, where non-zero elements is located.

Value: Value of the non-zero elements located at index(Row,Column)

Ex-I:

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

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

Ex-2:

0	1	2	3
0	0	0	1
1	1	0	0
2	0	0	6

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

Linked List Representation: Here each node has four fields-

Row: index of row, which non-zero elements is located.

Column: index of column, which non-zero elements is located.

Value: Value of the non-zero elements located at the index of (Row,Column).

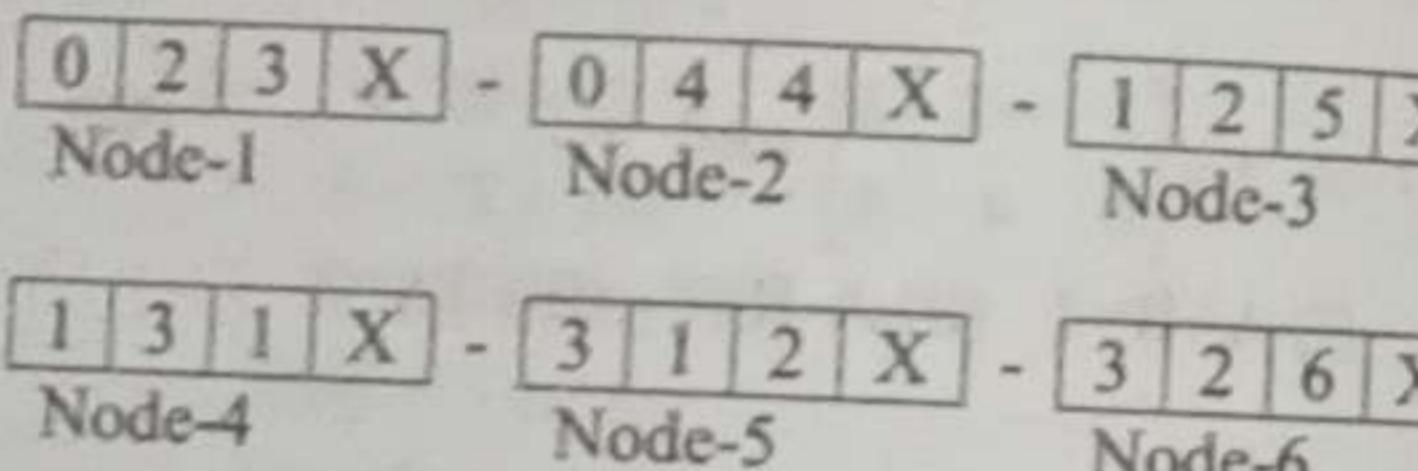
Next Node: Address of next node.

Ex:

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

Node:

Start	column	Value	Next address
0	1	2	3
0	0	3	0
1	0	5	1
2	0	0	0
3	0	2	6



Sorting

এখন 1. Sorting কাকে বলে? কত প্রকার ও কি কি?

উত্তর: Sort শব্দটি আমরা আমাদের মোবাইল বা কম্পিউটারে হয়ে থাকে। Sort শব্দটির অর্থ হলো তাটাকে একটি নির্দিষ্ট ক্রমে সাজানো। সাজানোটা হতে পারে ছেট থেকে বড়, বড় থেকে ছেট ইত্যাদি। ক্রমানুসারে সাজানোর এই প্রক্রিয়াকেই Sorting (Sorting) বলে।

দুইভাবে sorting করা হয়। যথা:

- Assending order (ছেট থেকে বড়)
- Descending order (বড় থেকে ছেট সাজানো)

এখন 2. বিভিন্ন প্রকার sorting টেকনিক লিখ।

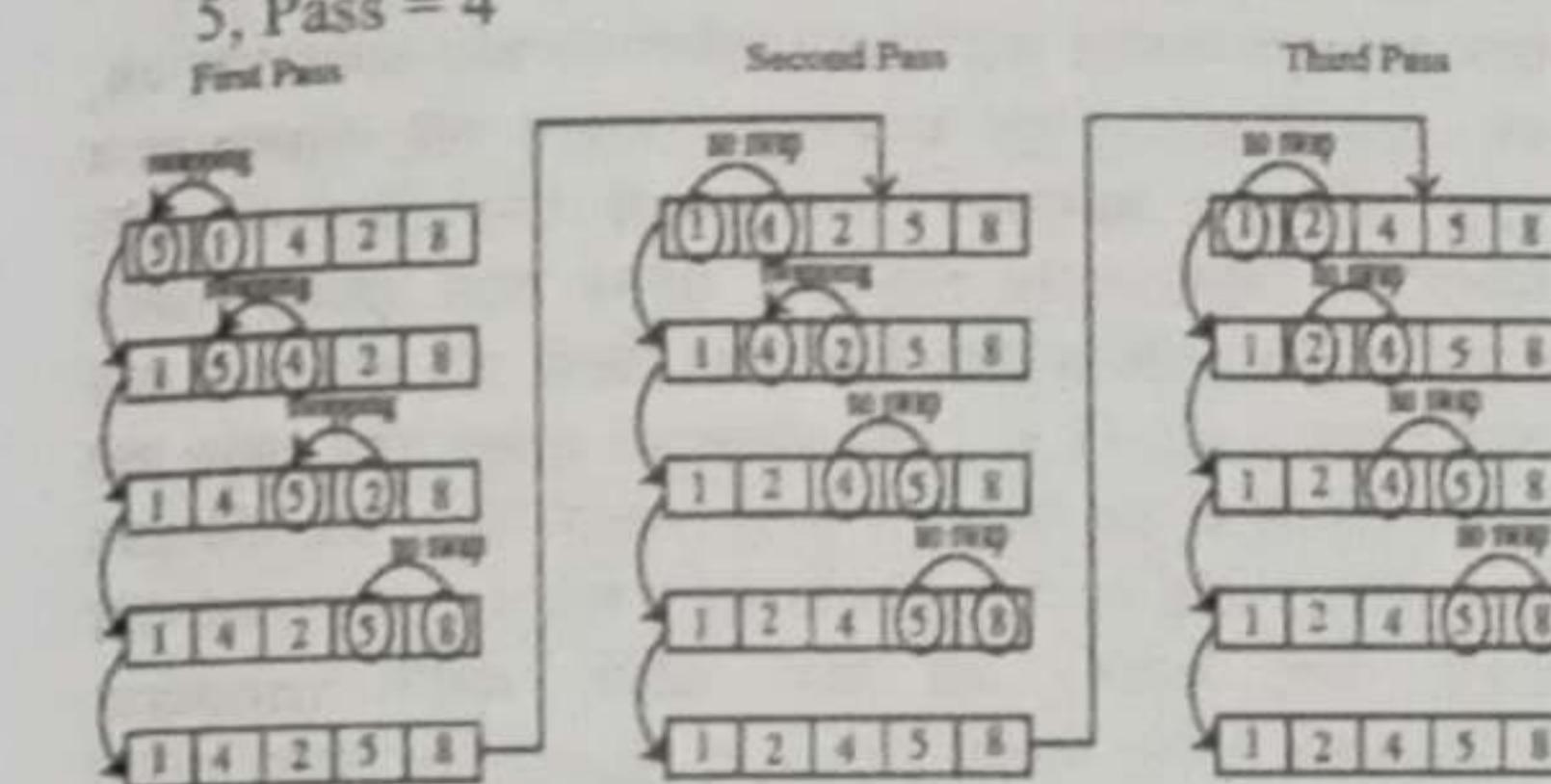
উত্তর: বিভিন্ন প্রকার sorting টেকনিক-

- Bubble sort
- Insertion sort
- Selection sort
- Quick sort
- Heap sort (Describe on Heap topics)
- Merge sort
- Radix sort
- Topological sort

এখন 3. বাবল সর্টের ব্যাখ্যা কর। [BRTA -2012] How bubble sort works ? Consider the array [68, 27, 56, 13, 87, 42,66] using bubble sort? [BRTA (senior Computer Operator)-2012] বিভিন্ন মানদণ্ড -২০১০-সহকারী মেইন্টেনেনেস ইন্জিনিয়ার (মন-ক্লাসের)

উত্তর: বাবল সর্টেতে বুদবুদ দেখায়। বুদবুদ যেমন তরলের উপরে উচ্চে আসে তেমনি এই সর্টের মাধ্যমে ভাটা element উপরে উচ্চে আসে (ছেট element বা বড় element)

মনে করি 5, 1, 4, 2, 8 এটাকে Assending order এ সাজাবে। এখন element থাকে তবে Pass হবে N -1 সংখ্যক , এখনে N = 5, Pass = 4



বাবল সর্টের কোড:

```
for(i=1; i<=n ; i++){
    for(j=0; j<n-1; j++){
        if(arr[j+1]>arr[j]){
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
```

কোডের আলোচনা:

এখানে প্রথম for লুপটি ব্যবহার করা হয়েছে আরের প্রতিটা ইনডেক্সের ক্ষেত্রে সর্টেড করার জন্য। সেজন্য এটা আরের উপাদানের সম্পরিমাণ iteration হবে (iteration means the repetition of a process)। আর দ্বিতীয় for লুপটি ব্যবহার করা হয়েছে একটা ইনডেক্সের ভ্যালুর সাথে অপর ইনডেক্সের ভ্যালু তুলনা করার জন্য। যদি তুলনা করা সময় প্রথম ইনডেক্সটি বিভিন্ন চেয়ে বড় হয়, তবে তার জায়গা বদল করবে। জায়গা বদলের জন্য একটা temporary variable রাখা হয়েছে। এখানেও আমরা swap() নামক বিভিন্ন ফাংশন ব্যবহার করতে পারি।

N.B: same procedure perform this array [68,27,56,13,87,42,66]

এখন 4. Bubble sort algorithm টি লিখুন। এই algorithm টির Time complexity নির্ণয় করুন। [বিনিষ্ঠ মানদণ্ড - ২০১৩-সহকারী মেইন্টেনেনেস ইন্জিনিয়ার (মন-ক্লাসের)]

উত্তর:

Bubble sort algorithm টি হলো:

Step 1: Repeat Step 2

For i = 0 to N-1

Step 2: Repeat For J = i + 1 to N - i

Step 3: IF A[J] > A[i]

SWAP A[J] and A[i]

[END OF INNER LOOP]

[END OF OUTER LOOP]

Step 4: EXIT

Complexity:

Scenario	Complexity
Space	O(1)

Worst case	Complexity
running time	$O(n^2)$

<tbl

Step-1: Make all the number same as maximum number digit.

608,005,768,298,576,975,090,080

[Maximum 3 digit number, so all numbers are now 3 digit number... ex.: 5 converted with 005 and 90 converted with 090]

Step-2: Sorting the least leftmost bit. Like 0, 1 and so on.

090 080 975 005 576 298 768 608

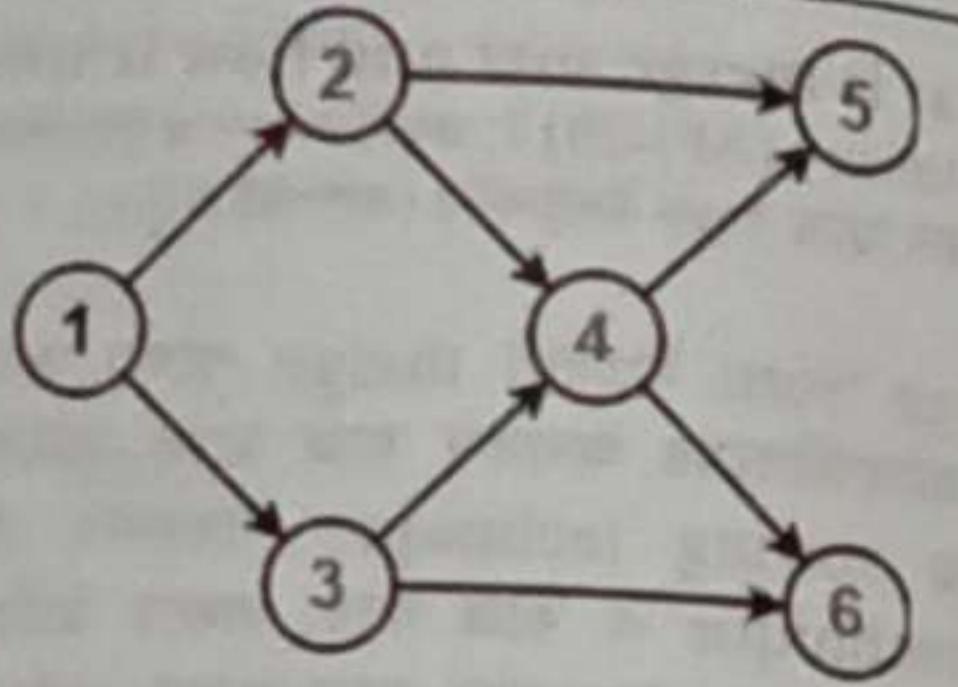
Step-3: Sorting the 2nd digit from left using step 2.

005 608 975 768 576 080 090 298

Step-4: Sorting the 3rd digit from left using step 3.

005 080 090 298 576 608 768 975

Now the list is sorted. When digit is more step will be increased.



Topological Sort Example

এই ধারে ৮ ধরনের টপোলজিক্যাল অর্ডারিং সম্ভব তা হলো:

⇒ 1 2 3 4 5 6

⇒ 1 2 3 4 6 5

⇒ 1 3 2 4 5 6

⇒ 1 3 2 4 6 5

টপোলজিক্যাল সর্ট মূলত অর্ডারিং, অর্থাৎ কোনটার পর কেন্তী আসবে, এই সংটি এ Directed Acyclic Graph (DAG) ব্যবহার হয়। উপরের চিত্রে নোড 8 অবশ্যই নোড 2 অথবা 3 এর উপর ডিপেন্ড, তাই নোড 8 এর আগে অবশ্যই নোড 2,3 আসবে। অনুরূপ নোড 6 আগে নোড 8 বা 3 আসবে।

Topological sort: Topological sort এমন একটি sorting technic যেখানে একটি directed acyclic graph এর vertex তলো sort করা হয় এবং sorted vertex তলো ধারের অন্যান্য node তলোর থেকে আগে দাকবে। একটি উদাহরনের মাধ্যমে বিষয়টি উপস্থাপন করলে অনেক সহজে বোঝা যাবে।

Quick Sort: হলো একটি fastest shorting algorithm। প্রার্থিক্যাল কাজের ক্ষেত্রে Quick Sort এর ব্যবহার ব্যাপক। Complexity এ কথা সিদ্ধ করলে quick sort এর complexity $O(n\log n)$ । অনেক ভাটার ক্ষেত্রে suitable sort হলো quick sort। quick sort এ ক্ষেত্রে একটি নতুন term আসবে তা হলো Pivot value। একটি উদাহরনের মাধ্যমে বিষয়টা ভালোভাবে বোঝা দেখানো হলোঃ
দেয়া আছে একটি unsorted array [5,2,8,6,4,9,3,17]। যেখানে বলা আছে প্রথম Value টা হবে Pivot। তাহলে পরবর্তি ধাপগুলো দেখা যাবে:

5	2	8	6	4	9	3	17
Pivot	5						
i					j		

2	8	6	4	9	3	17	P<j
i					j		

Pivot এর value কি array এর j তম লোকেশনের থেকে ছেট।

যদি ছেট হয় তাহলে j তম লোকেশনের value i তম লোকেশনে

পাঠিয়ে দাও। আর না হলে j = j-1 কর। এখানে 5 < 17 সত্য তাই j = j-1 করা হয়েছে।

2	8	6	4	9	3	17	P<j
I					J		

5 < 3 false arr[i] = 3; i = i+1;

3	2	8	6	4	9		17	P<i
I					J			

5 < 2 false; i = i+1;

3	2	8	6	4	9		17	P<i
i					J			

5 < 8 True; arr[j] = arr[i]; j = j-1

Pivot	5							
3	2		6	4	9	8	17	P < j
i			j					

5 < 9 True; j = j-1

Pivot	5							
3	2		6	4	9	8	17	
i			j					

5 < 4 False; arr[i] = arr[j]; i = i+1;

Pivot	5							
3	2	4	6		9	8	17	
i		j						

5 < 6 True; arr[j] = arr[i]; j = j-1;

Pivot	5							
3	2	4	6		9	8	17	
i		j						

যখন i == j হবে তখন pivot এর value i == j এর স্থানে assign হবে।

3	2	4	5	6	9	8	17
---	---	---	---	---	---	---	----

Pivot	3							
		2	4					
i		j		P < j				

3 < 4 True; j = j-1;

Pivot	3							
		2	4					
i		j	P < j					

3 < 2 False; arr[i] = arr[j]; i = i+1;

Pivot	2							
		3	4					
i		j	P < j					

6 < 9 True; j = j-1;

Pivot	2							
		3	4					
i,j		9	8	17		i=j		

6 < 8 True; j = j-1;

Pivot	2							
		3	4					
i,j		9	8	17		i=j		

9 < 17 True; j = j-1;

Pivot	2							

<tbl_r

i) Difference Merge sort and Quick sort. [Barishal Engineering Collage-2021]

উত্তর: মার্জ সর্ট এবং কুইক সর্ট এর পার্থক্য নিচে দেওয়া হলো: (নেটওর্ক সর্ট অ্যালগরিদম এবং Divide and Conquer এ কাজ করে)

Merge Sort	Quick Sort
আরে কে একদম মিড-পর্যন্তে ভাগ করে।	আরে কে যে কোন পর্যন্তে ভাগ করে। অর্ধাং সমান ভাগে ভাগ করতে পারে আবার নাও করতে পারে।
Auxiliary array দরকার হয়।	Auxiliary array দরকার হয় না।
Merge করা দরকার হয়।	merge করা দরকার হয় না।
Complexity:	Complexity:
> Best case: $O(n\log n)$	> Best case: $O(n\log n)$
> Average case: $O(n\log n)$	> Average case: $O(n\log n)$
> Worst case: $O(n^2)$	> Worst case: $O(n^2)$

ii) Analyze and compare the quick-sort and Merge-sort algorithms in terms of their time and space complexity. [বাট অনলাইন- 2021- AP]

উত্তর: Complexity Comparison:

Complexity	Merge	Quick
Best Case	$O(n\log n)$	$O(n\log n)$
Average Case	$O(n\log n)$	$O(n\log n)$
Worst Case	$O(n \log n)$	$O(n^2)$

Algorithm	Data Structure	Worst Case Auxiliary Space Complexity
Quick sort	Array	$O(n)$
Merge sort	Array	$O(n)$

Merge Sort: ইন্দোজি merge শব্দের অর্থ একজীকৃত ; অর্ধাং এই অ্যালগরিদমের অন্যতম কাজ হচ্ছে ডাটাশোলোকে একত্র করা। Merge sorting technique সাধারণত divide and conquer technique এ কাজ করে। Merge sort এর worst-case time complexity $O(n \log n)$ ।

মার্জ সর্ট কাখনে আমরা দেখতে পাই, ইন্সেমেট করতে প্রথমে ভাগ করা হয় পরে আবার মার্জ (একত্র) করা হয়। ফিগুর-১ এ লিস্টেকে দুই ভাগে ভাগ করা হয়। লিস্টের প্রথম পার্টের ভাগ করতে টাইম লাগে $1/2 \log(n)$ বেসানে n হলো নাম্বার সংখ্যা। মার্জ করার সময় কম্পেয়ার করে মার্জ আউটকাম পেতে n procedures দরকার হয়। figure এ মার্জ সর্ট এর প্রসেস দেখানো হলো:

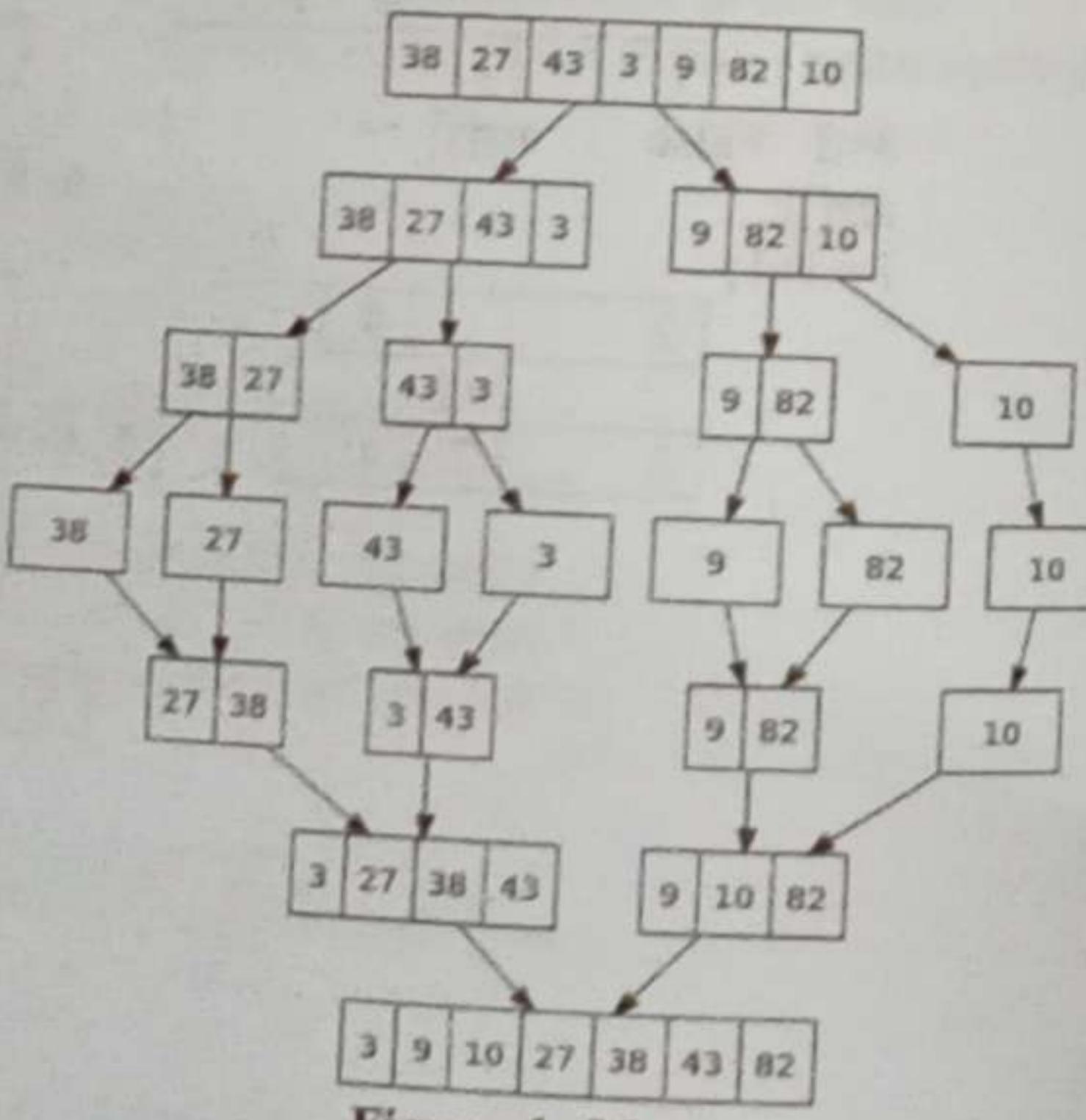


Figure-1: Merge Sort.

Quick Sort হলো একটি fastest shorting algorithm। প্রাক্তিকাল কাজের ক্ষেত্রে Quick Sort এর ব্যবহার ব্যাপক। Complexity এর ক্ষেত্রে Quick sort এর complexity $O(n\log n)$ । অনেক ডাটার ক্ষেত্রে suitable sort হলো quick sort। quick sort এর দেখা আছে একটি নতুন term আছে তা হলো Pivot value। একটি উদাহরণের মাধ্যমে বিষয়টা ভালো ভাবে বের করা দেখানো হলোঃ

দেখা আছে মাঝের Value টা হবে Pivot।

পাশের ফিগুরে অ্যালগরিদম করলে পাই n উপাদান সংখ্যাকে পাইভট ভালুর মাধ্যমে অথবা মাঝে বরাবর দুই ভাগে ভাগ করা হয় হ্যাল কম্পেন্সেশন হবে $n\log(n)$ । কিন্তু worse case কম্পিলেশনে মাঝে বরাবর ভাগ করা যানো বা মাঝে প্রথম ভালুকে পাইভট ধরতে হয় তখন লিস্টকে ০ থেকে $n-1$ পর্যন্ত কম্পেয়ার করার প্রয়োজন পরে তখন কম্পেন্সেশন হবে (n^2) ।

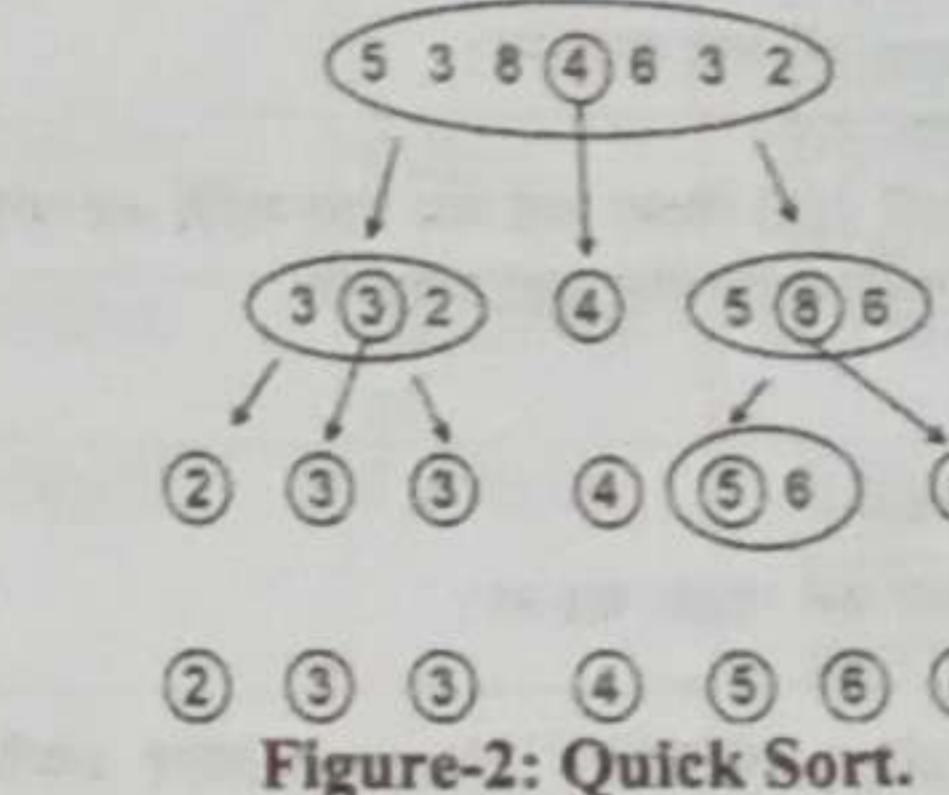


Figure-2: Quick Sort.

অধ্য 10. কত হলো সার্চ টেকনিকের নাম লিখ। ব্যাখ্যা কর।

উত্তর: কত হলো সার্চ টেকনিকের নাম-

- Linear Search
- Binary Search
- Jump Search
- Interpolation Search
- Exponential Search
- Sublist Search (Search a linked list in another list)
- Fibonacci Search
- The Ubiquitous Binary Search
- Recursive program to linearly search an element in a given array
- Recursive function to do substring search

লিনিয়ার সার্চ:

বাট অনলাইন- 2021 , উপ-সহকারী ইন্টিনিয়ার (মন-কাচুর)

কতগুলো Element এর একটা আবারের মধ্যে থেকে একটা নিদিষ্ট Element থেকার সহজ পদ্ধতি হল লিনিয়ার সার্চ। যতক্ষণ পর্যন্ত নিদিষ্ট Element টা খুজে পাবেনা ঠিক ততক্ষণ পর্যন্ত আবারের প্রতিটা element ঢেক করতে থাকবে।

Number 4 7 9 10 15 90 30 20 2 6
0 1 2 3 4 5 6 7 8 9

মনে করি Number নামের একটা আবারেতে 10 টা value আছে। আবরা এখানে 20 এবং 50 কে খুজে বের করবো।
মনে করি,

ITEM = 20 যেহেতু লিনিয়ার সার্চ পদ্ধতিতে আবারের প্রত্যেক টা element এর সাথে তুলনা করা হয়। সুতরাং ITEM কে আবারের প্রত্যেক location এর element এর সাথে তুলনা কর।

ITEM == Number [0] হয়, তাহলে তা print করে শেষ হবে না হলো Next location এর সাথে তুলনা করা হবে। ITEM == Number [1] যতক্ষণ না পাওয়া যাবে ততক্ষণ আবারের last position পর্যন্ত খুজতে থাকবে। [Number [n], Where n=1,2,3,] এখানে Array 7 পজিশন এর সাথে ITEM টা সমান হবে। তখন তা print করবে।

যদি ITEM = 50 হয় তখন ITEM এর সাথে সকল element compare করবে যদি সমান না হয় তাহলে ITEM Not found দেখাবে। এখানে 50 ITEM Not found দেখাবে।

পিনিয়ার সার্চের সূবিধা

- আরের Element সার্চ করা শোন না।
- সহজে code লেখা যাব।
- আরের এলিমেন্ট 100 কিংবা তার কম হলে সার্চ এর অন্ত কোন আলগোরিদম ব্যবহার করার জিজ্ঞ করতে হব।

পিনিয়ার সার্চের অভিযোগ

- সবচেয়ে অনেক বেশি লাগে।
- আরের দেখে ভাল ফল পাওয়া যাব না।

Best case Complexity: N সংখ্যক element থেকে একটি নিশ্চিত element খুঁজে পেতে যে সর্বনিম্ন ধাপ বা সহজের প্রয়োজন হয়, সে সংখ্যক ধাপ বা সহজের Best case Complexity বলে। এই Algorithm এর Best case Complexity হল- O(1)।

Worst case Complexity: N সংখ্যক element থেকে একটি নিশ্চিত element খুঁজে পেতে যে সর্বোচ্চ ধাপ বা সহজের প্রয়োজন হয়, সে সংখ্যক ধাপ বা সহজের Worst case Complexity বলে। এই Algorithm এর Best case Complexity হল- O(n)।

Average case Complexity: N সংখ্যক element থেকে একটি নিশ্চিত element খুঁজে পেতে সর্বোচ্চ ও সর্বনিম্ন ধাপ বা সহজের মাঝামাঝি ঘটকলো ধাপ বা ঘটাটুকু সহজের প্রয়োজন হয়, সে সংখ্যক ধাপ বা সহজের Average case Complexity বলে। এই Algorithm এর Average case Complexity হল- O(n/2)।

(ii) নাইনারি সার্চ: নিশ্চিত element খোজার সহজ এবং স্বতন্ত্র পদ্ধতি হল বাইনারী সার্চ। বাইনারী সার্চ আরের Element ক্ষেত্রে মধ্য Element টা বের কর। আরের Element সংখ্যা N হলে মধ্যে হবে $\frac{(N-1)}{2}$ অবশ্যই আরেটো sorted হতে হবে। যদে কোরি K এর সাথে তুলনা করে K এর মান সার্চ করবো, যদি K এর value Mid value এর দেখে বড় হয় তাহলে তা right size এর সাথে তুলনা করে value search করবো অন্যথায় Mid value এর বাইরের value ক্ষেত্রে সাথে তুলনা করতে হবে।

যদে কোরি Number একটা আরের মধ্যে

1	5	7	8	1	2	3	3	3	3	7	7
2	6	0	1	3	5	0	2				
0	1	2	3	4	5	6	7	8	9	1	1

BEG=0

END=11

$$MID = \frac{BEG + END}{2}$$

$$= \frac{0+11}{2} = 5$$

যদি ITEM == 8 হয় তাহলে ITEM টা MID এর সাথে তুলনা করতে হবে। সেহেতু ITEM!=MID সেহেতু ITEM যদি MID এর দেখে বড় হয় তাহলে তান দিকে search করবে আর ছোট হলে বাই দিকে search করবে।

এখন ITEM দেহেতু MID এর দেখে ছোট তাই left side এ করবে এবং END = MID -1 হবে তিনি হবে এরকম।

1	5	7	8	12
0	1	2	3	4

$$\text{আবার } MID = \frac{0+4}{2} = 2 \quad \text{ITEM} \neq \text{MID} \text{ Then right}$$

8	12
3	4

$$\text{BEG END} \\ \text{BEG} = \text{MID} + 1$$

$$\text{MID} = \frac{3+4}{2} = 3$$

ITEM found and print ITEM

প্র 11. Write a algorithm to find a node in binary search tree? [Sonali and Janata bank (IT/ICT), 2018]

Algorithm:

BinarySearch (A, Item, LB, UB)

Step-1: If LB>UB then: return -1

Step-2: End If

Step-3: MID: = [(LB+UB)/2]

Step-4: If A[MID] == Item then: return MID

Step-5: Elseif A[MID] > Item then: BinarySearch (A, Item, LB, MID -1)

Step-6: Else BinarySearch (A, Item, MID + 1, UB)

Step-7: End If

প্র 12. Time space Trade off কি?

উত্তর: Time space Trade off: কোন Algorithm এর Running time কমাতে চাইলে data storage space বেশি লাগে এবং storage space এর পরিমাণ কমাতে চাইলে Running time বেশি লাগবে এই বিশেষ অবস্থাকে বলা হয় Time space Trade off.

প্র 13. Suppose there is an array with some numbers in a random order. Can you apply binary search on it? Why or Why not? [পৃষ্ঠা অন্তর্ভুক্ত পৃষ্ঠার প্রশ্ন প্রক্রিয়া মাট্টা]-2019]

উত্তর: যখন আরে ব্যাক্ত অর্ডারে সাজানো থাকবে তখন আমরা বাইনারী সার্চ করতে পারবোনা কারণ বাইনারী সার্চ করার শর্ত হলো আরে অবশ্যই সঠিক হতে হবে।

Complexity

উত্তর: Complexity: কোন Algorithm এর Data সম্বরে যদি তুলনা করার প্রয়োজন হলে algorithm এর Execution time এর পার একে algorithm এর complexity বলে।

Big O Notation	Name	Example(s)
O(1)	Constant	# Odd or Even number,

O(log n)	Logarithmic	# Look-up table (on average)
O(n)	Linear	# Finding element on sorted array with binary search
O(n log n)	Linearithmic	# Find max element in unsorted array, # Duplicate elements in array with Hash Map
O(n^2)	Quadratic	# Sorting elements in array with merge sort
		# Duplicate elements in array,

O(n^3)	Cubic	# Sorting array with bubble sort
O(2n)	Exponential	# 3 variables equation solver
O(n!)	Factorial	# Find all subsets
		# Find all permutations of a given set/string

প্র 2. Sorting আলগোরিদমের complexity table দেখাও। [DDBL(PO)2016]

উত্তর: Sorting আলগোরিদমের complexity table-

Sl.No.	Algorithm	Worst case	Average case	Best Case
01	Bubble	$O(n^2)$	$O(n^2)$	$O(n)$
02	Insertion	$O(n^2)$	$O(n^2)$	$O(n)$
03	Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$
04	Heap	$O(nlogn)$	$O(nlogn)$	$O(nlogn)$
05	Quick	$O(n^2)$	$O(nlogn)$	$O(nlogn)$
06	Merge	$O(nlogn)$	$O(nlogn)$	$O(nlogn)$
07	Bucket	$\Omega(n+k)$	$\Theta(n-k)$	$O(n^2)$
08	Radix	$\Omega(nk)$	$\Theta(nk)$	$O(nk)$
09	Binary Search	$(logn)$	$(logn)$	$O(1)$

Find Out Complexity:

Example-1:

```
Model_solution(n){  
    result = n * (n + 1);  
    return result;}
```

Ans: Time complexity O(1).

Example-2:

```
Slow_solution(n){  
    result = 0;  
    for i in to (n){  
        for j in to (i + 1){  
            result += 1;}  
        return result;}
```

Ans: Time complexity O(n^2).

Example-3:

```
int a = 0, i = n;  
while (i > 0) {  
    a += i;  
    i /= 2;}
```

Ans: Complexity will be $\rightarrow O(\log(n))$

Example-4:

```
int m=0;  
for (int i = 0; i < n; i++) {  
    m=m+1;}  
for (int i = 0; i < n; i++) {  
    for(int j = 0; j < n; j++) {  
        m=m+1;}}
```

Ans: Complexity will be $n+n*n \rightarrow O(n^2)$

Example-5:

```
int m=0;  
for (int i = 0; i < n; i++) {  
    for(int j = n/2; j < n; j++) {  
        for(int k=0;k*k < n; k++) {  
            m=m+1;}}}
```

Ans: Complexity will be $n*n/2*\log(n) \rightarrow n^2\log(n)$

Example-6:

```
int m=0;  
for (int i = n/2; i < n; i++) {  
    for(int j = n/2; j < n; j++) {  
        for(int k=0;k < n; k++) {  
            m=m+1;}}}
```

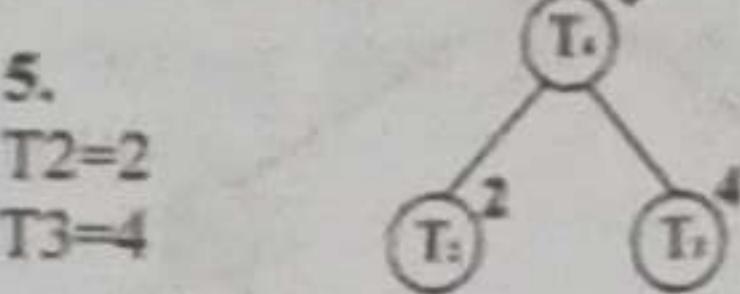
Ans: Complexity will be $n/2*n/2*n \rightarrow n^3$

T2=2

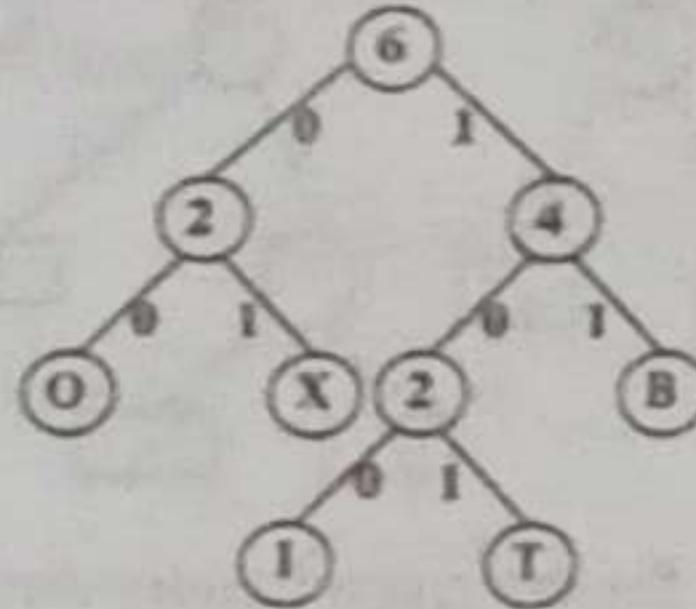
5.

T2=2

T3=4



Final Tree:



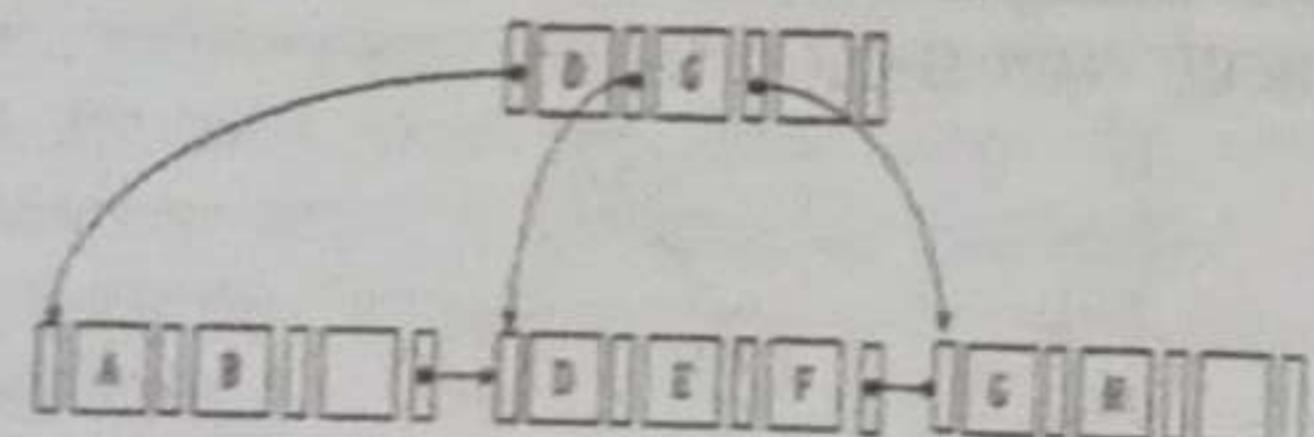
B=11 = 2X2=4 Bits
I=100 = 3X1=3 Bits
T=101 = 3X1=3 Bits
O=00 = 2X1=2 Bits
X=01 = 2X1=2 Bits
So, BITBOX=11100101110001

Time complexity: $O(n \log n)$ is the overall time complexity. Where n is the number of characters.

Q8 B+ tree indexing

প্রশ্ন ১. B+ tree কি?

উত্তর: B+ tree মূলত বিভিন্ন কার তারণাদিক ইনডেক্সিং এর জন্য ব্যবহৃত হয়। বিপুস টি পদেটারের মাধ্যমে শীক নোড এ ভাটা স্টোর করে, যা স্থান ভাটা অনুসরণ করে।



Rules of B+ tree:

- শীক নোড সমূহ তে রেকর্ড সংরক্ষণ করতে ব্যবহৃত হয়।
- ভাটা সমূহ ইন্টানাল নোডগুলিতে স্টোর হয়।
- যদি টার্ণেটি নোড এর ভ্যালু ইন্টানাল নোডের চেয়ে ছোট হয়, তবে তার বাব নিকে reference বিদ্যুতি অনুসরণ করা হবে।
- যদি টার্ণেটি নোড এর ভ্যালু ইন্টানাল নোডের চেয়ে বড় বা সমান হয় তবে তার ভানানিকে reference বিদ্যুতি অনুসরণ করা হবে।
- রুট নোড এর সর্বনিম্ন দুটি চাইক নোড থাকবে।

Applications

Importance of B+ Tree:

- Using B+, we can retrieve range retrieval or partial retrieval. Traversing through the tree structure makes this easier and quicker.
- As the number of record increases/decreases, B+ tree structure grows/shrinks. There is no restriction on B+ tree size, like we have in ISAM.
- Since we have all the data stored in the leaf nodes and more branching of internal nodes makes height of the tree shorter. This reduces disk I/O. Hence it works well in secondary storage devices.

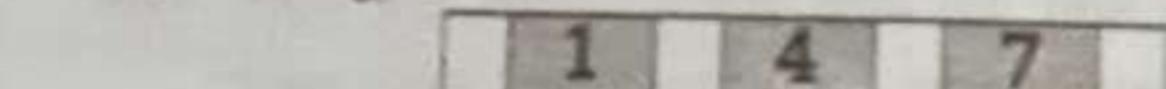
Example1: Construct a B+ tree for (1, 4, 7, 10, 17, 21, 31, 25, 19, 20) with n=4.

Solution:

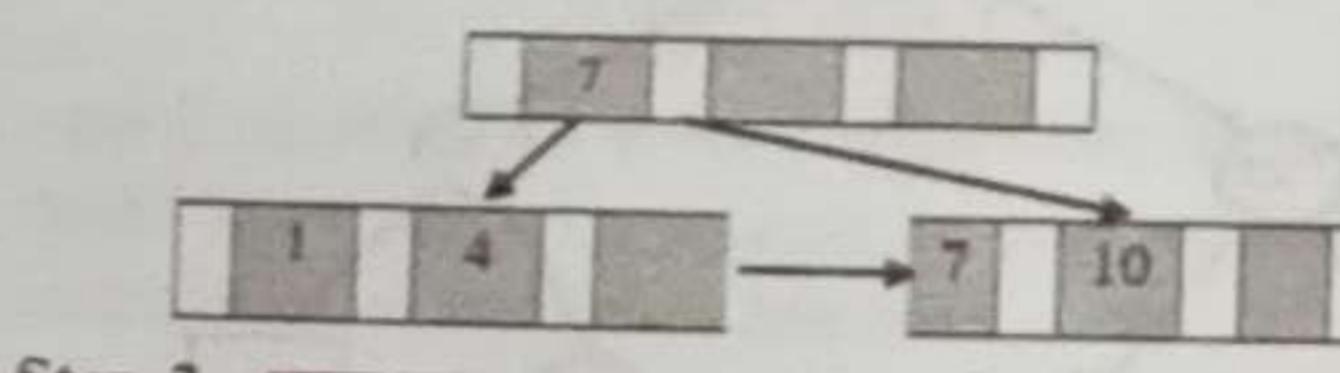
We Know, B+ Tree is constructed by parameter n
• Each Node (except root) has $(n/2)$ to n pointers
• Each Node (except root) has $(n/2)-1$ to $n-1$ search-key values

[সহজ কথায়, নোডের সার্স কী ভ্যালু সংখ্যা নোড পয়েন্টার থেকে ১ কম হবে।]

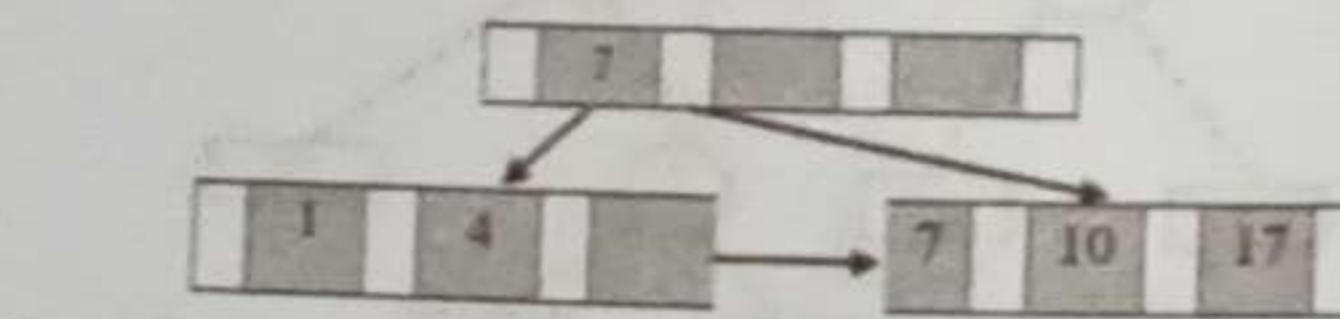
Step-1: যেহেতু এখানে $n=4$ তাই আমরা 1, 8, 7 বসাতে পার। উপরের শর্ত অনুসারে।



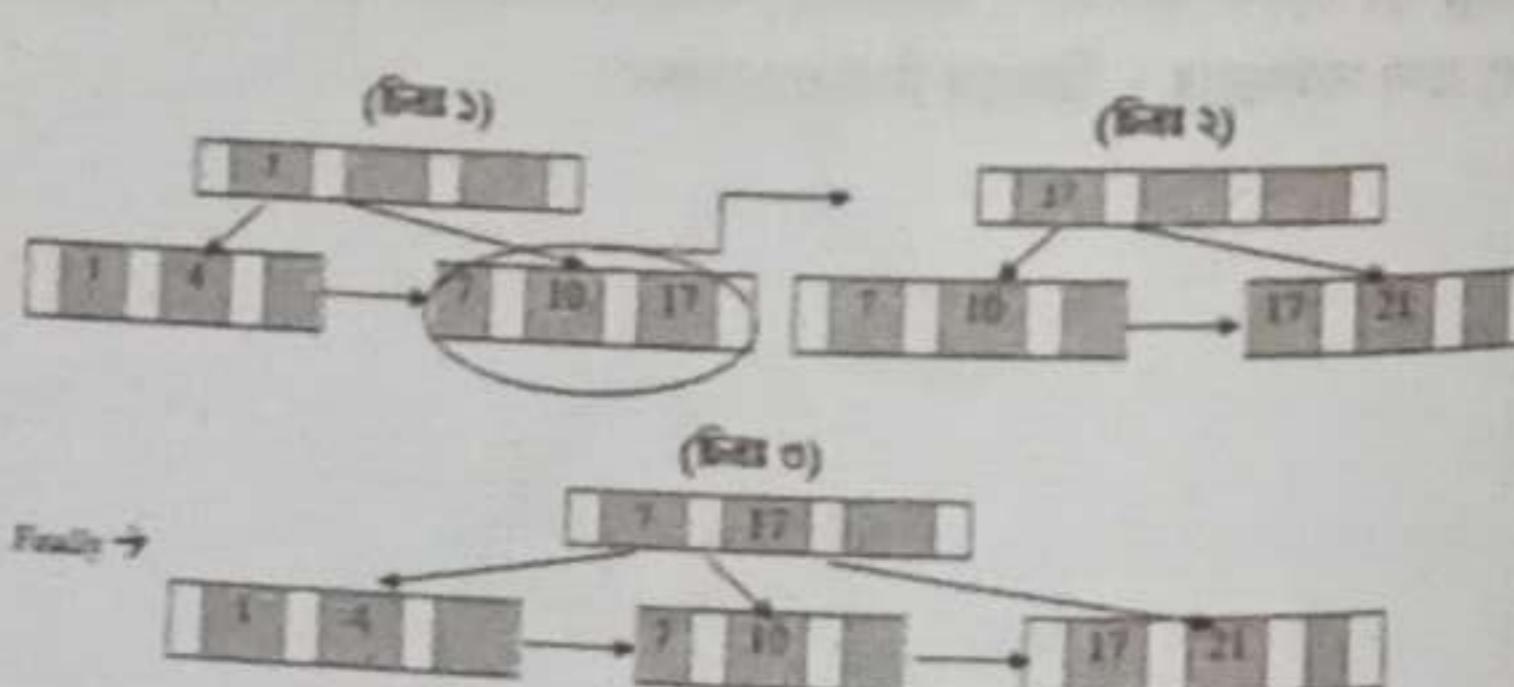
Step-2: যেহেতু তিনি ($n=4$) এবং key = $(n-1)=3$ যা অন্তরে পূরণ হয়ে গেছে তাই 10 ইন্সার্ট করতে হলে নোডকে স্পিট করতে হবে।



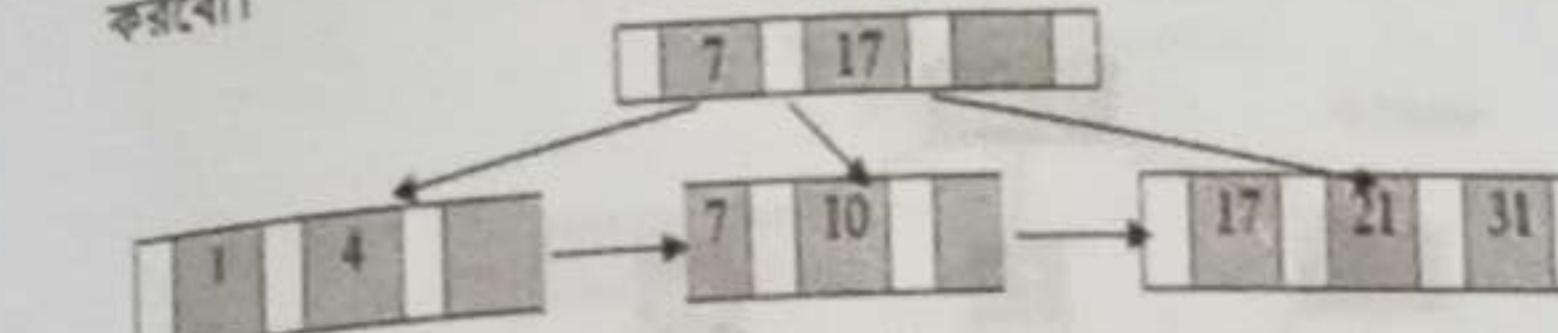
Step-3: পরের ভ্যালু 17 কে 10 এর পরে Insert করবো।



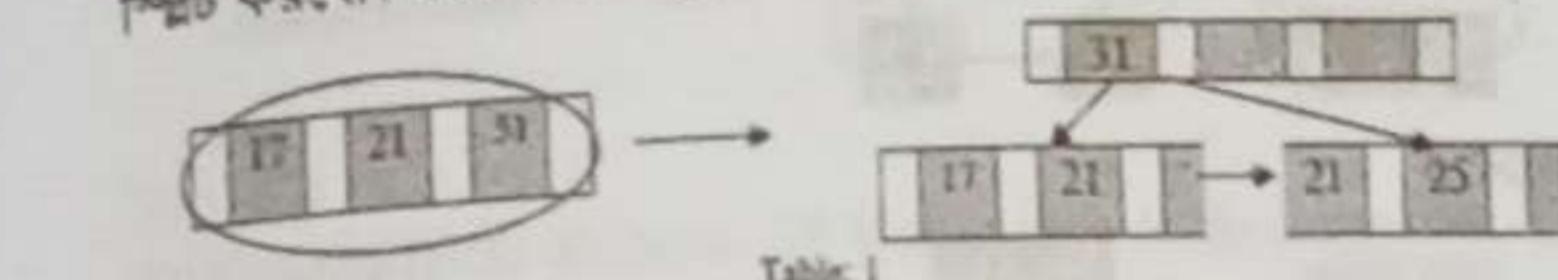
Step-4: পরের ভ্যালু 21 কে ইন্সার্ট করতে গেলে শেষ নোডকে আবার স্পিট করবো।



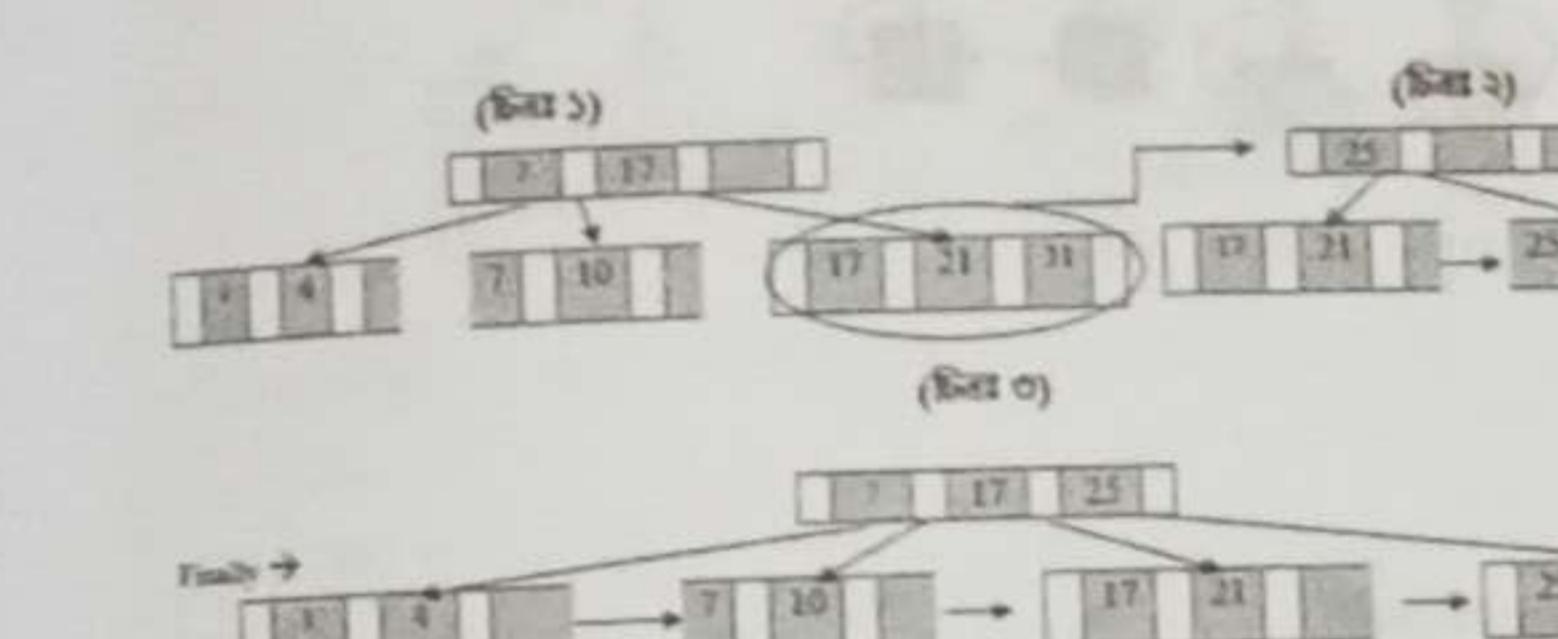
Step-5: পরের ভ্যালু 31 কে ইন্সার্ট কে Last Node এ ইন্সার্ট করবো।



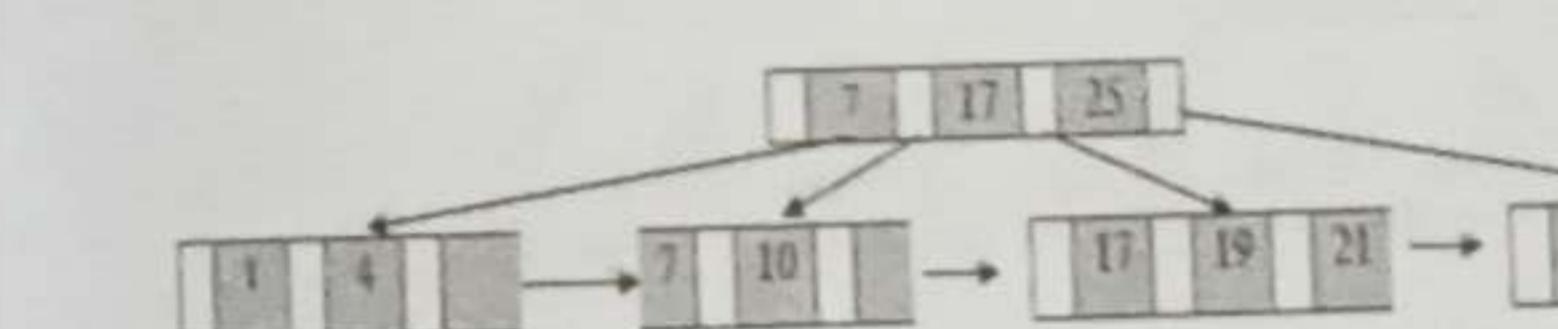
Step-6: পরের ভ্যালু 25 কে ইন্সার্ট করতে গেলে শেষ নোডকে আবার স্পিট করবো। কারণ নোড ফুল।



অপরের টেবিলে (Table:1) মতো করে split করতে পারবো না, কারণ Binary Tree এর শর্ত হল Left <=Node <=Right, তাই ১১ এর right node ২৫ হতে পারে না। তাই নিচের মত করে split করবো।

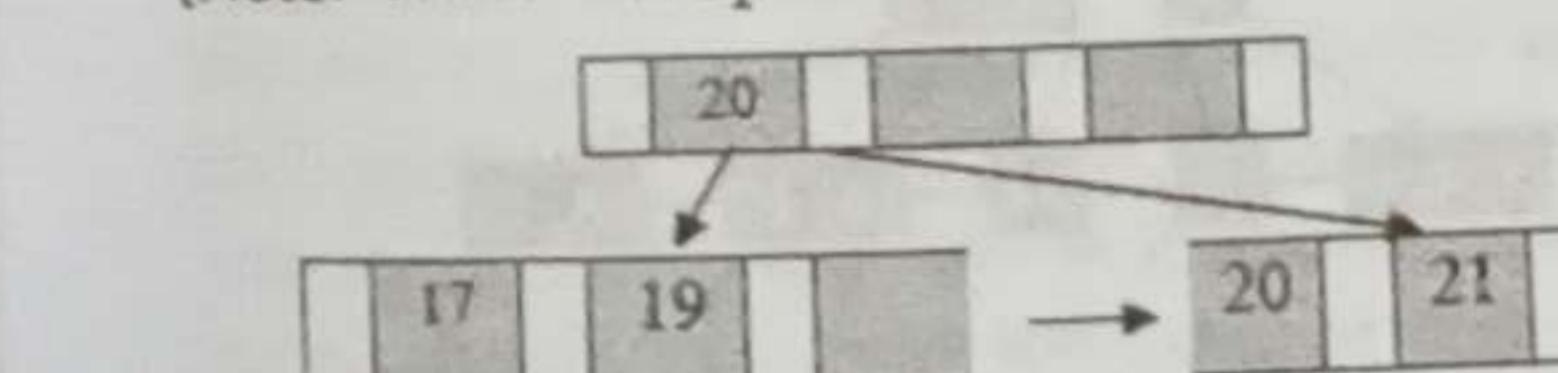


Step-7: পরের ভ্যালু 19, Binary Tree এর শর্ত মতে ১৭ এবং ২১ এর মাঝে বসে, আর যেহেতু এই Node এ একটি key ফীকা আছে তাই নিচের মত করে বসাতে পারি।

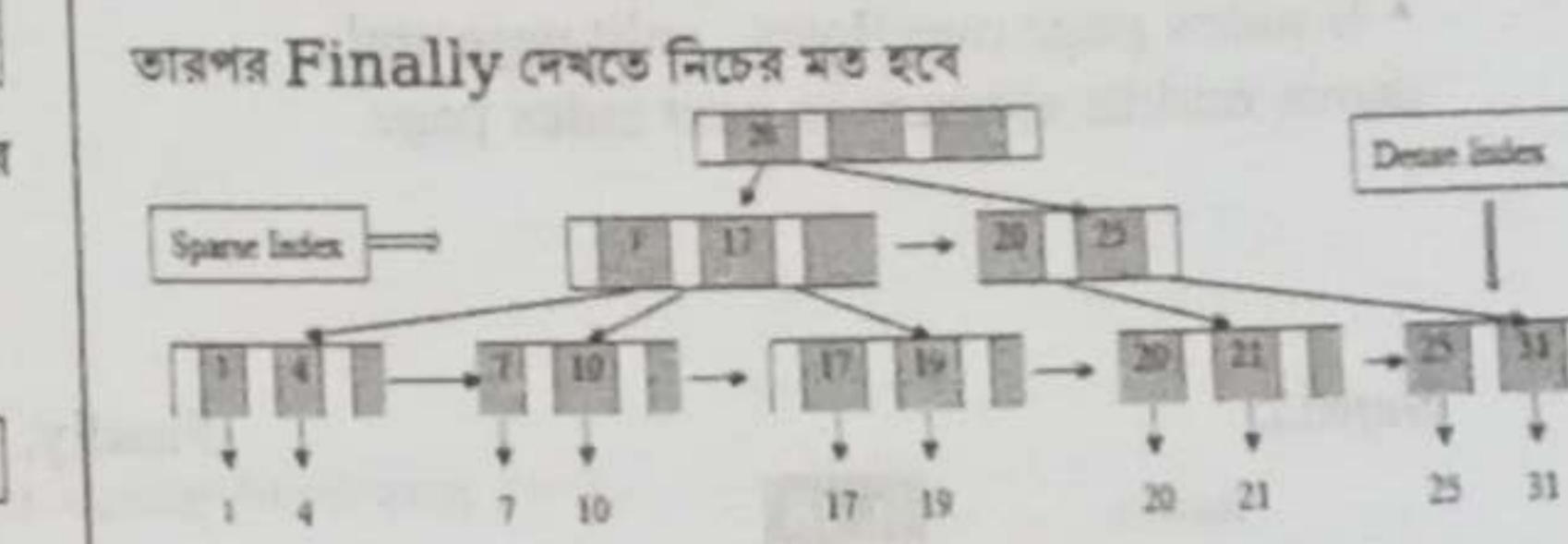
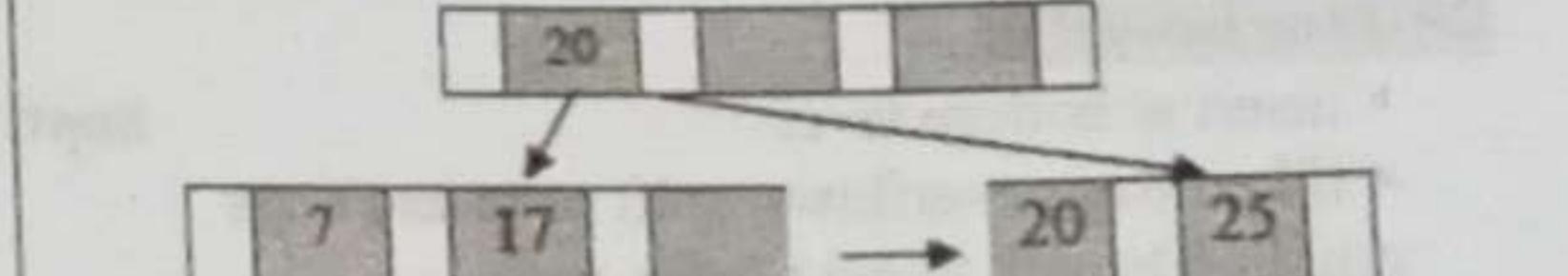


Step-8: পরের ভ্যালু 20, Binary Tree এর শর্ত মতে ২১ এবং ২৫ এর মাঝে বসবে কিন্তু ২১ এর নোড (১৭, ১৯, ২১) ফুল একই ভাবে ২৫ এর নোড (১৭, ২৫) ও ফুল তাই নিচের মত করে আবার Split করতে হবে। ১৭, ১৯, ২০, ২১ কে নিচের মত করে স্পিট করা যায়।

[Note: একাধিক ভাবে split করা যাবে।]



এখন মতুন key (20) উপরের Keys (7, 17, 25) এর সাথে একত্রে 7, 17, 20, 25 অব্যাখ্যা 4 টি হয়ে যায়, যা শর্ত ভঙ্গ করে। তাই 7, 17, 20, 25 কে split করবো নিচের মত করে।



চির্তা B+ Tree

[Note: dense index এ প্রতিটি search key value এর জন্য database এ একটি index record থাকে। sparse index এ প্রতিটি search key value এর জন্য database এ index record থাকে না।]

Q Construct a B+ tree index structure of emp-id for given relation employee as shown below with n=4.

emp-id	Name	Designation
e1	Harun	Chief Engineer
e2	X	Superintending Engineer
e3	Y	Superintending Engineer
e4	Z	Executive Engineer
e5	W	Executive Engineer
e6	Q	Executive Engineer

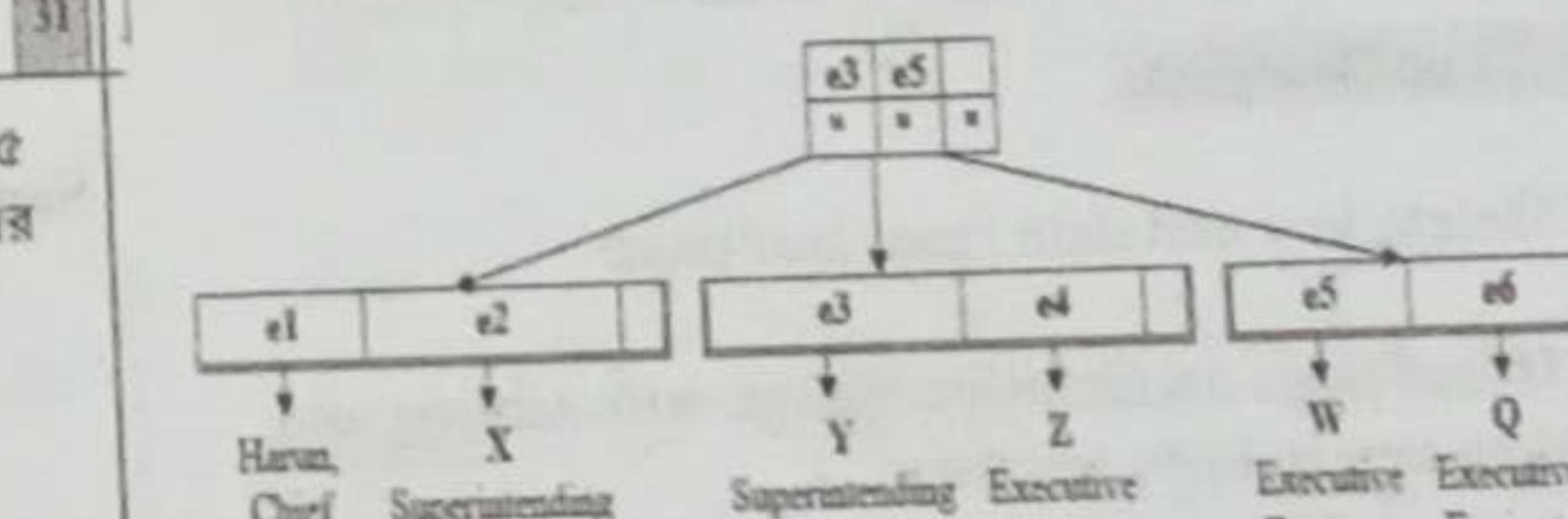
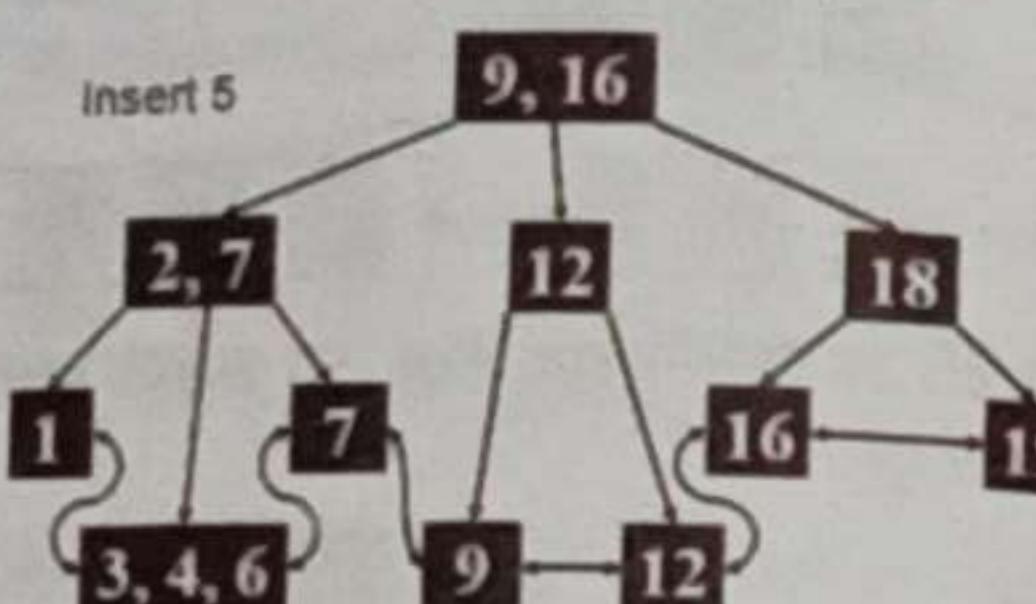


Fig: B+ Tree Relational Table

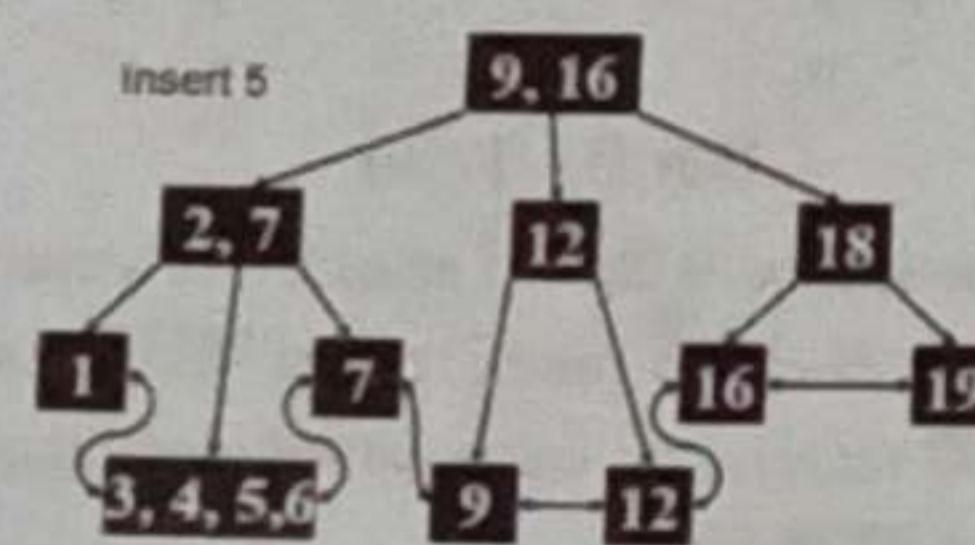
B+ Tree Insertion:

- Insert at bottom level
- If leaf page overflows, split page and copy middle element to next index page
- If index page overflows, split page and move middle element to next index page

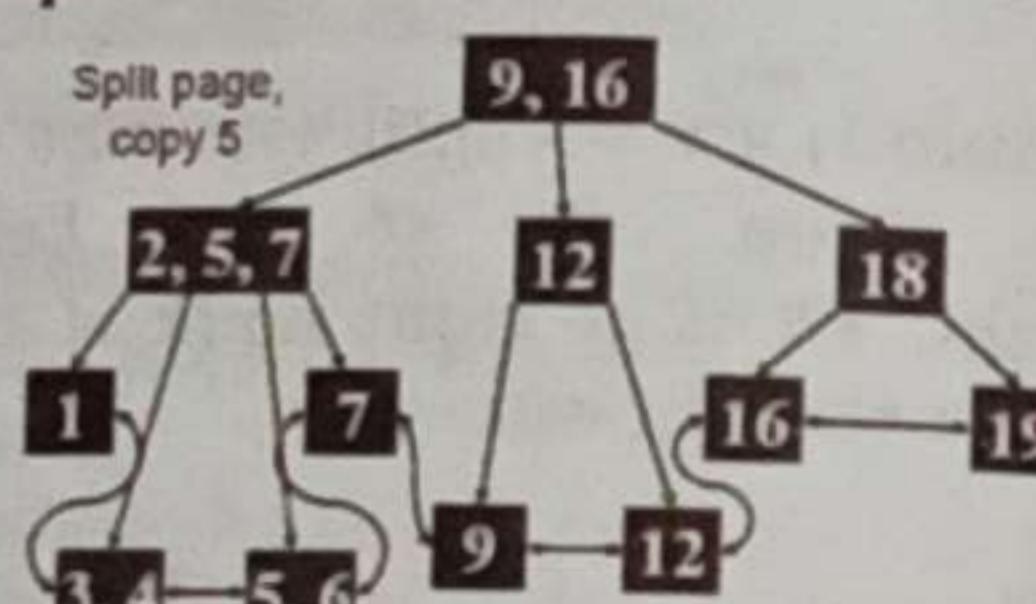
Sept:1



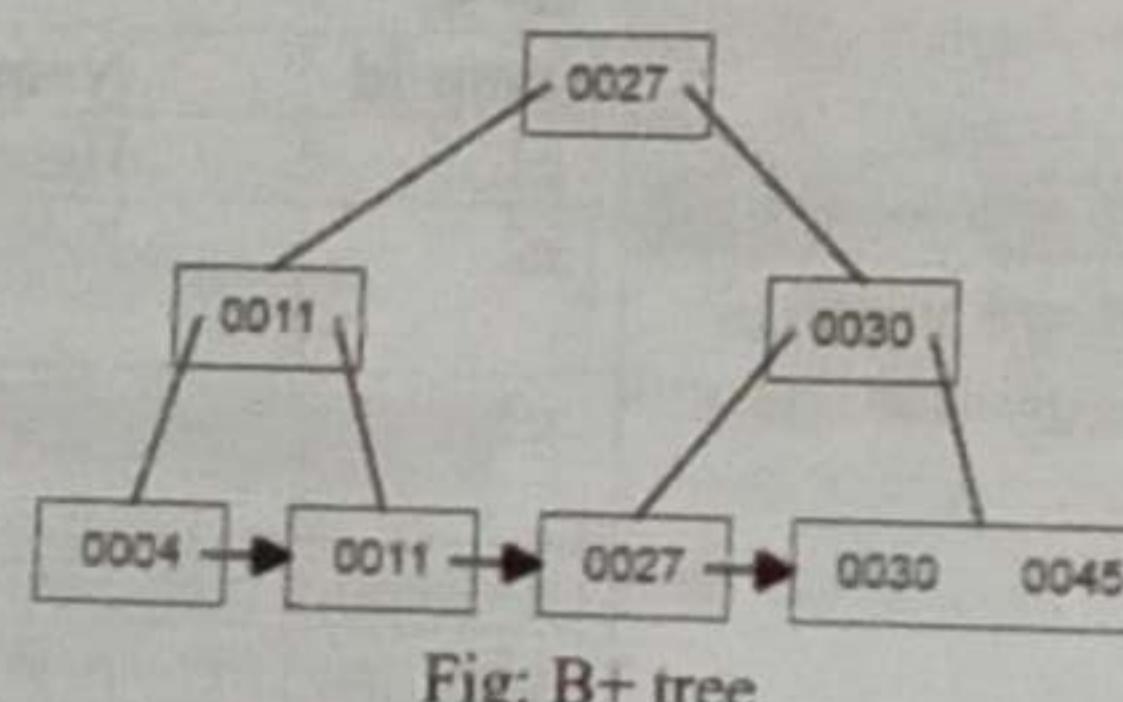
Sept:2



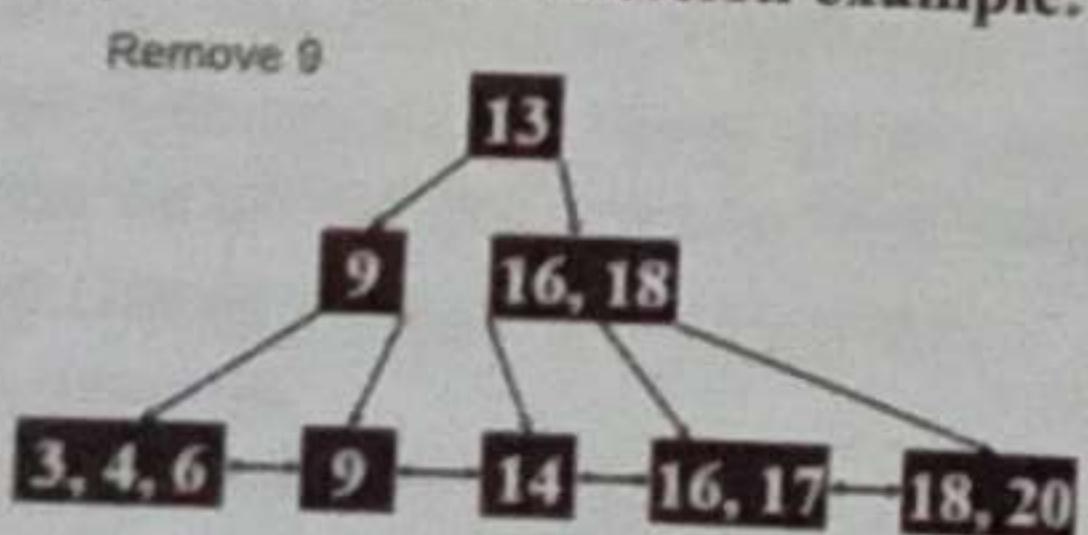
Finally, Sept:3



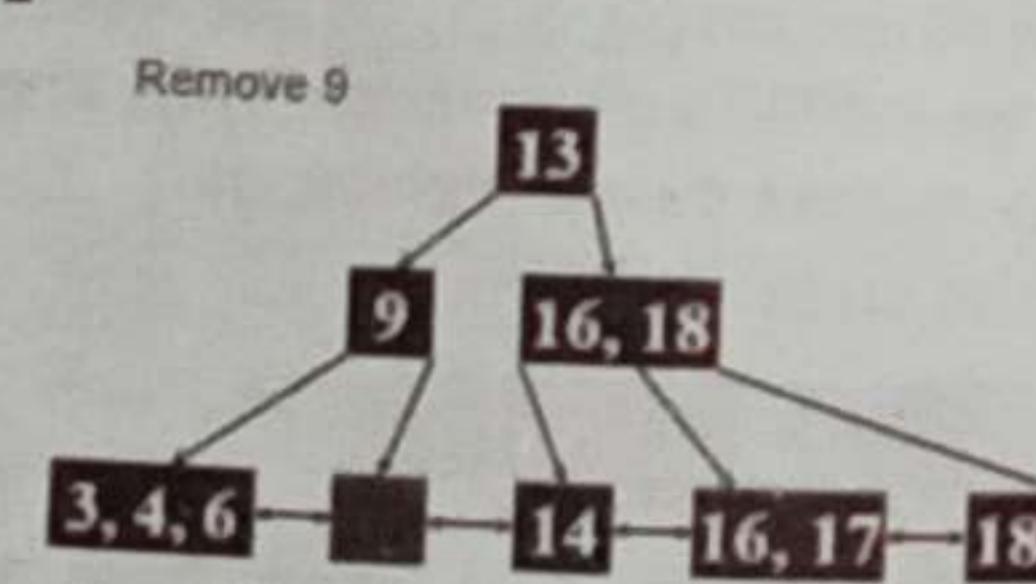
☞ Example-2: Draw B+ Tree 11, 27, 4, 45, 30 with n=3.
Ans:

**B+ Tree Deletion:**

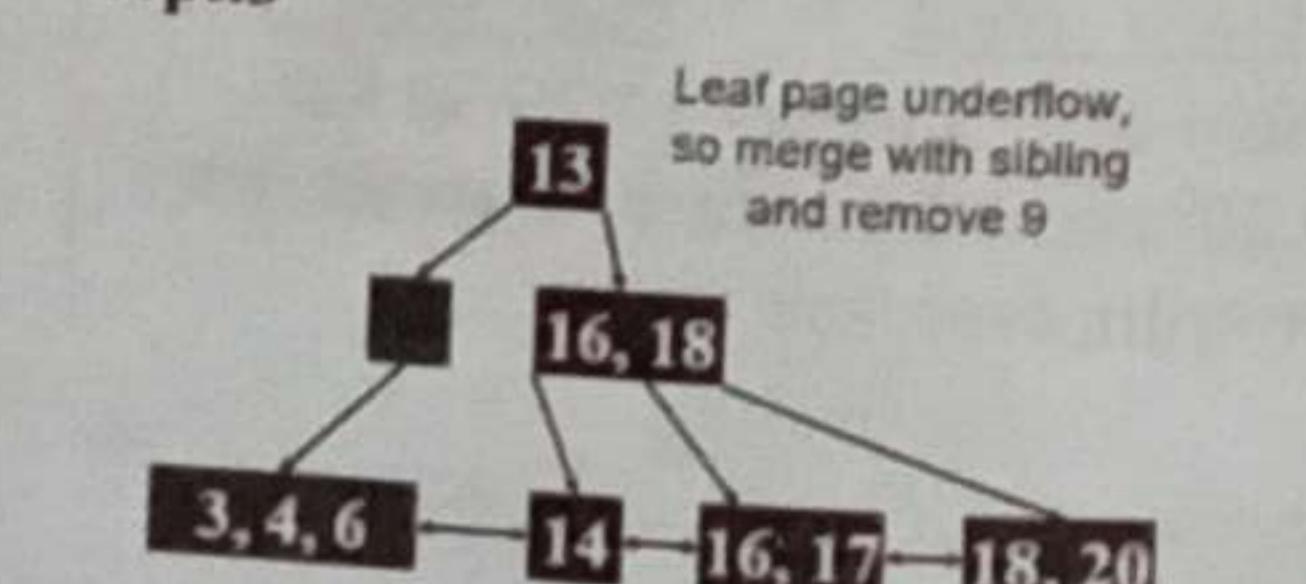
- Delete key and data from leaf page
- If leaf page underflows, merge with sibling and delete key in between them
- If index page underflows, merge with sibling and move down key in between them

Sept:1 B+ Tree deletion example:

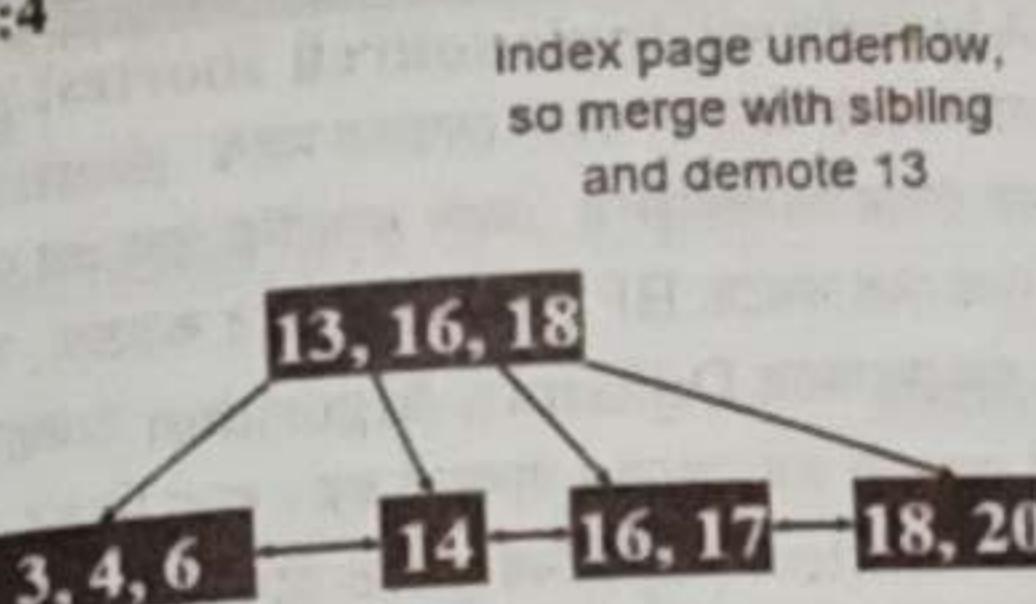
Sept:2



Sept:3

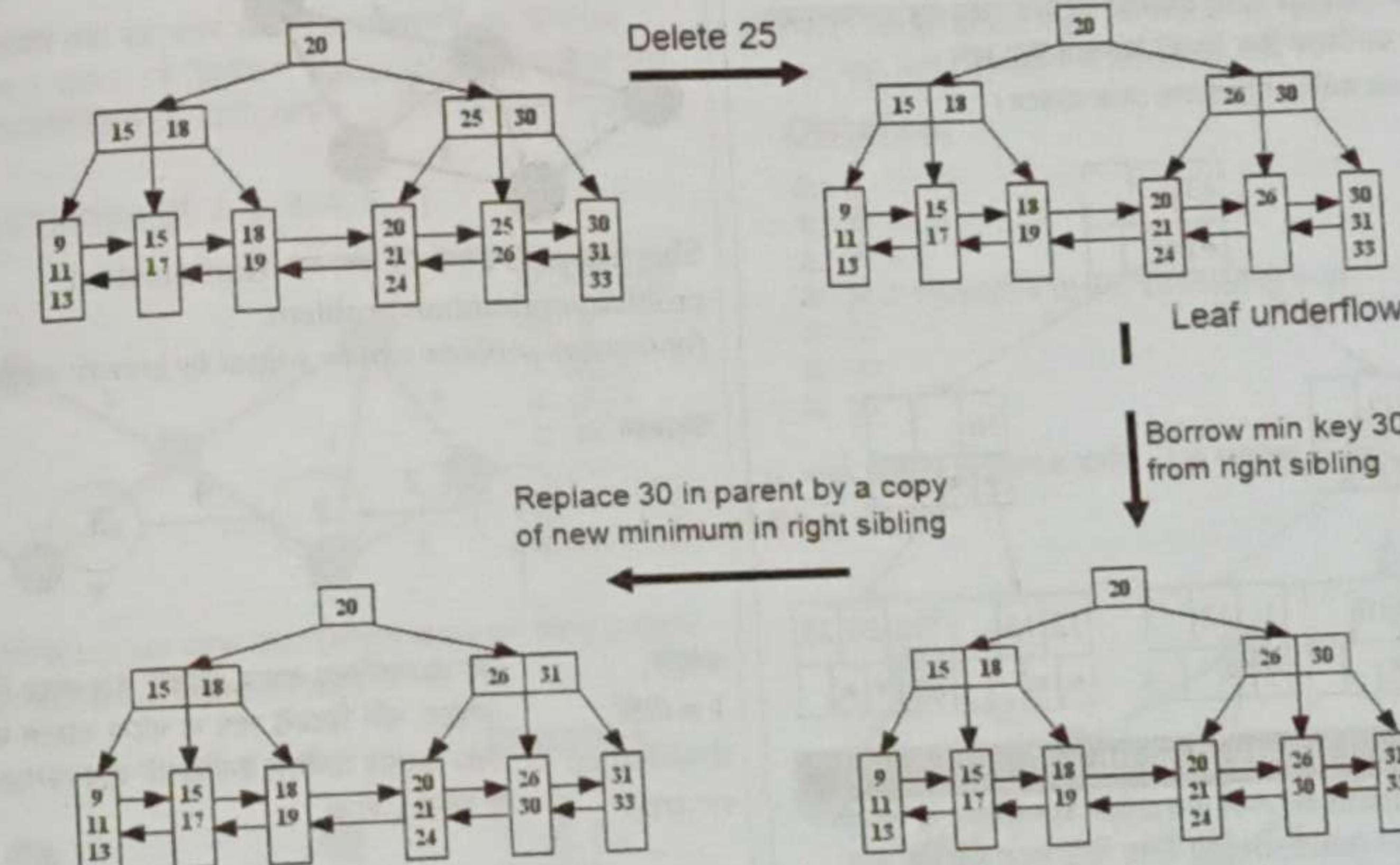


Sept:4



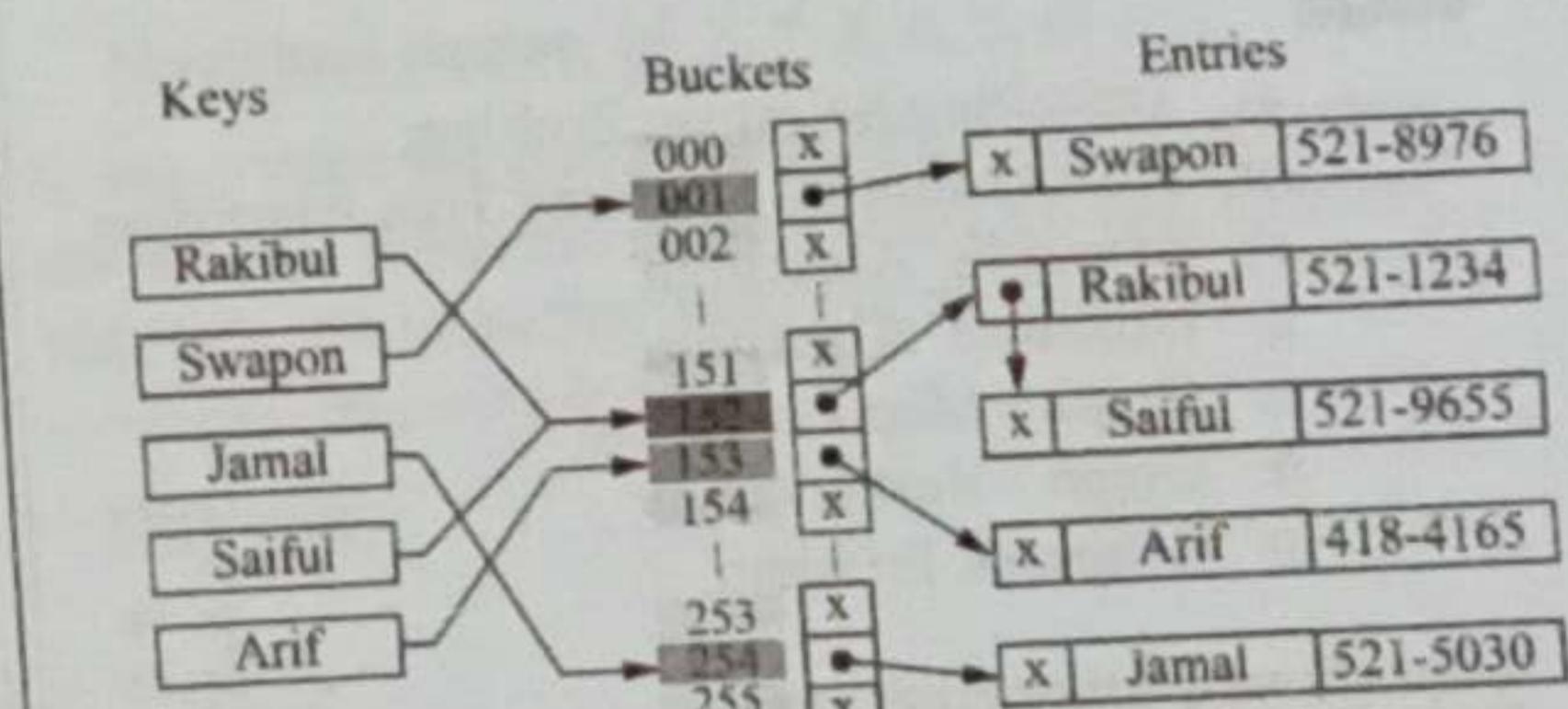
index page underflow,
so merge with sibling
and demote 13

☞ Example 3: Delete 25 from the following B+ tree order M=3 and L=3.

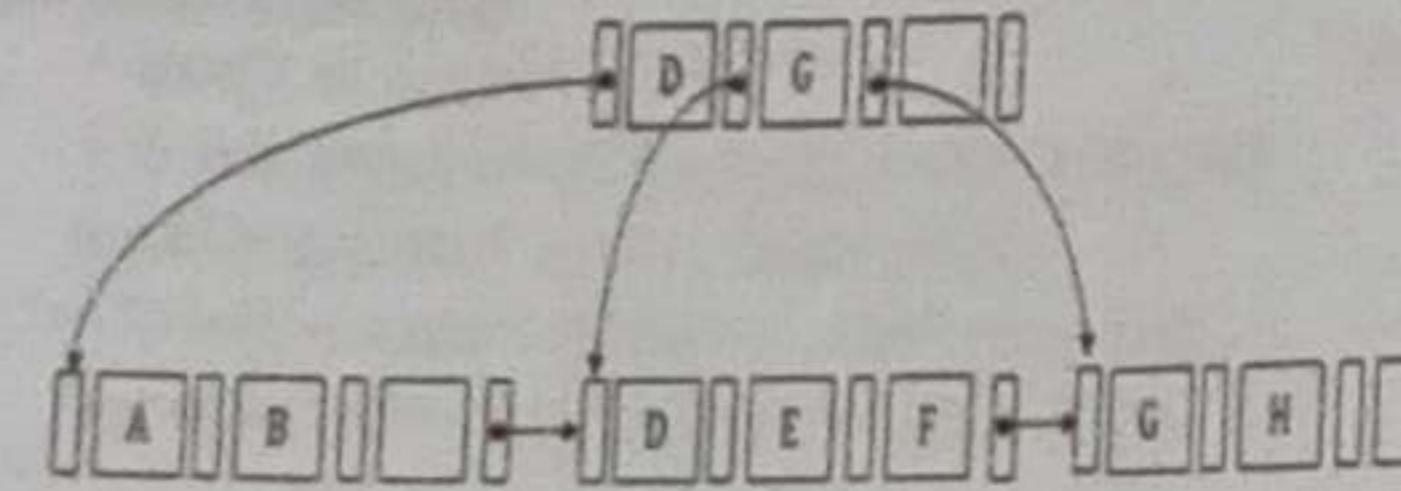


2. How Hash indexing and B+ tree indexing do work? [BCS-36]

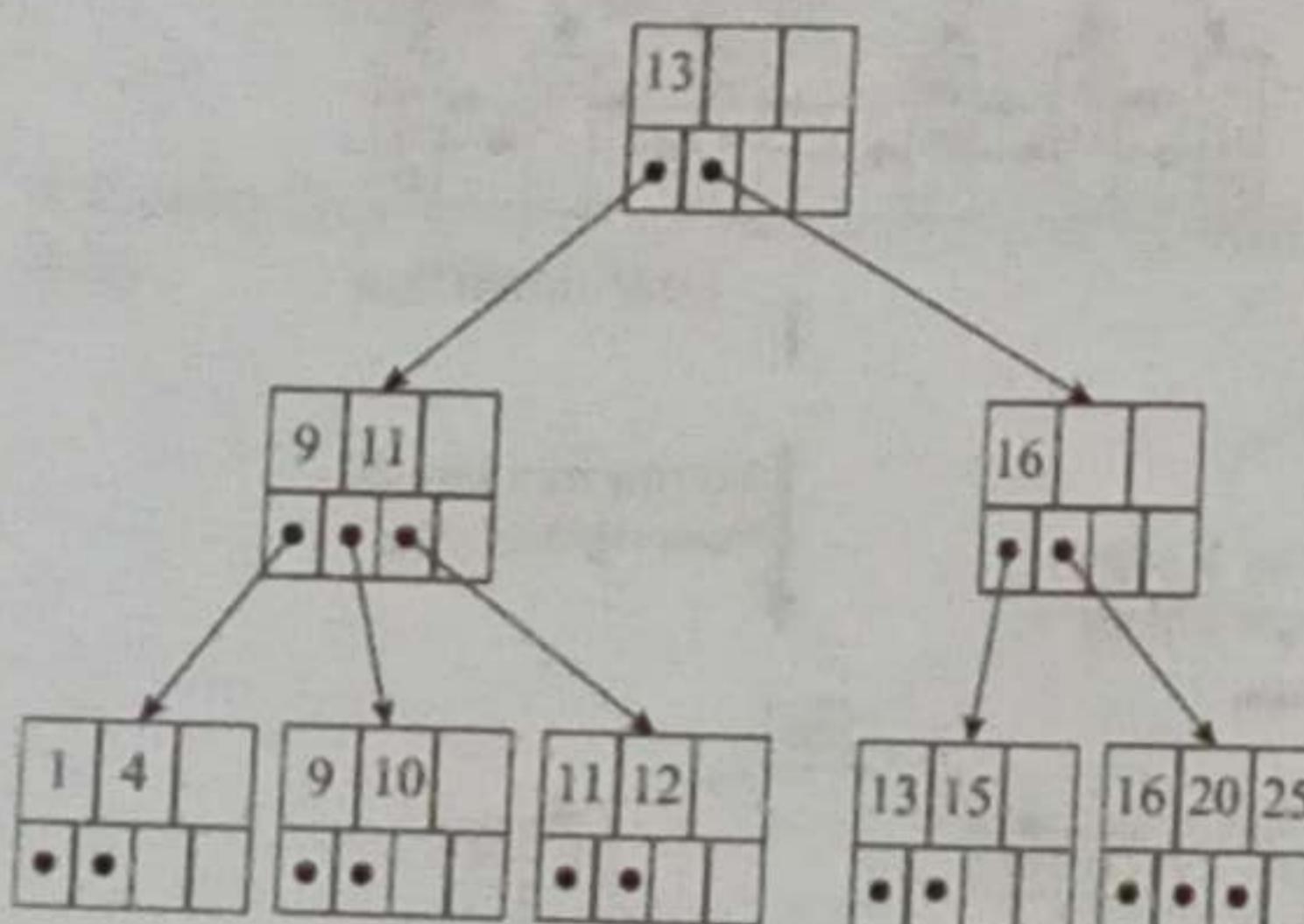
উত্তর: হ্যাশ index টি N buckets বা slot গুলির একটি আরে এবং প্রতিটি সারিতে একটি পয়েন্টার মুক্ত। হ্যাশ index গুলি হ্যাশ ফাংশন F(K, N) ব্যবহার করে যাতে একটি কী K এবং bucket N প্রদান করা হয়, ফাংশনটি key এর সাথে হ্যাশ index এর সংশ্লিষ্ট bucket map করে থাকে।



বিপ্লব মূলত বিভিন্ন ভরে ডায়নামিক ইনডেক্সিং এর জন্য ব্যবহৃত হয়।
বিপ্লব টি পয়েন্টারের মাধ্যমে লীফ নোড এ ডাটা স্টোর করে, যা দ্রুত
ডাটা অনুসরণ করে।

**Rules of B+ tree:**

- शीर्ष नोड समूह डेटा रेकर्ड संरक्षण करते वाबहत हय।
- डाटा समूह इन्टीनल नोडोंप्रियते स्टोर हय।
- यदि टार्णेट नोड एवं डालू इन्टीनल नोडोंप्रियते चेये छोट हय, तबे तार वाम दिकेके ठिक विस्तुति अनुसरण करा हवे।
- यदि टार्णेट नोड एवं डालू इन्टीनल नोडोंप्रियते चेये बड़ वा समान हय, तबे तार तानदिके ठिक विस्तुति अनुसरण करा हवे।
- बूट नोड एवं सर्वनिम्न दृष्टि चाइन्ट नोड थाकवे।

**Greedy algorithm**

Greedy algorithm हल एवन एक प्रकार अलगारिदम या अनेक उल्लेखनोंप्रियते माके heuristic एवं उपर तिति करे सबचेये ताल (optimal) स्युल्शन निर्नय करे।

उदाहरणः

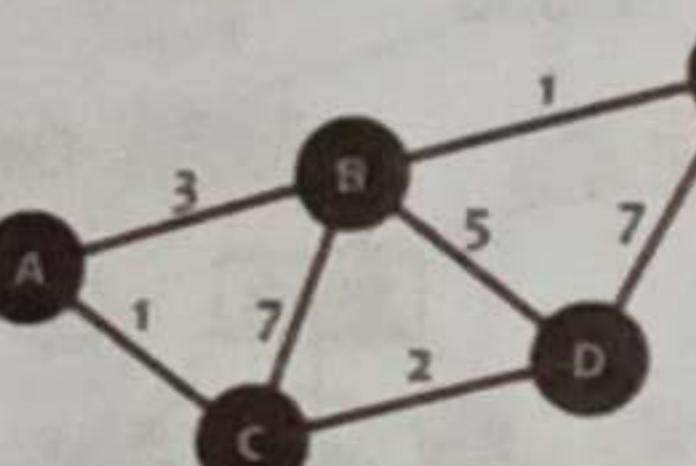
- Travelling Salesman Problem
- Prim's Minimal Spanning Tree Algorithm
- Kruskal's Algorithm
- Dijkstra's Algorithm
- Graph - Map Colouring
- Graph - Vertex Cover
- Knapsack Problem

एथाने दृष्टि समस्या आलोचना करा हलोः-

Knapsack problem: धरि एको चोर यार्केटे चूरि करते गेलो १ टा बाल नियो, बाले सर्वोक्त १० केजि जिनिस निते पारवे। मने करि १० टा आइटेम आहे १० रुकम ओजनेर एवं १० रुकम घुल्लोर। एथन चोर कि कि आइटेम कत्तुकू करे निले १० केजि बाल भर्ति कराले माझिमाम लात कराते पारवे। एवं अष्टिमाल स्युल्शन दिवे छिडि अलगारिदम।

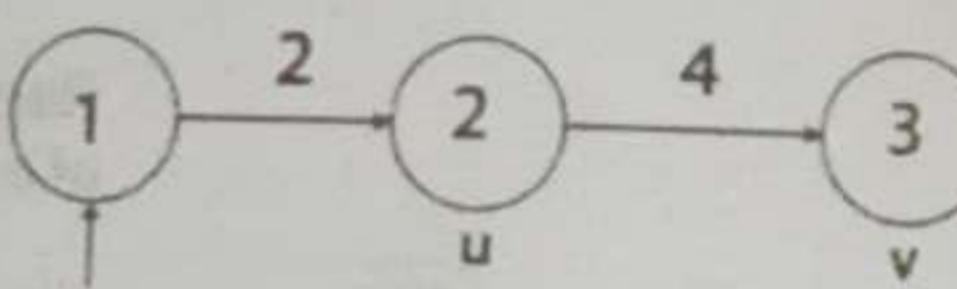
Dijkstra algorithm

Dijksta's Algorithm(single sourch shortest path): ग्राफेर एको नोड थेके अन्य ये कोन नोडोंप्रियते माके shortest path खुजे बेर करार अलगारिदम, आन ओरेटेटे ग्राफ एवं केत्र अमरा स्टेट पाथ बेर करते BFS,DFS व्यवहार कराताम, यदि ग्राफो ओरेटेट हय सेक्वेन्टे Dijksta's Algorithm व्यवहार हय, यदि non weighted graph एवं केत्रेते व्यवहार हय। धरो निचेर छोट ग्राफे A थेके E shortest path बेर करते हवे, आमरा निचे तार A थेके E याइते पारि, किन्तु सबचेये कम कस्ट ए किभावे कोन पाथ दिये गोचाते पारवो ता बेर करते साहाय्य करे एवं यिति अलगारिदम।



Shortest path खोजा के बला हय Minimization problem/optimization problem.

[optimization problem can be solved by greedy method.]

उदाहरणः

एथाने,
1 = सोर्स
d=distance
c = cost

एই अलगारिदम प्रथमे डिरेक्ट पाथ आहे किना देववे, यदि डिरेक्ट पाथ ना थाके ताहले वाकि नोड उलोर डिस्ट्राइब इनफिनिट हवे। उपरोर आवे ए सोर्स १ हते:

$d[u]=2$
 $d[v]=\infty$

(सोर्स थेके डिरेक्ट पाथ)

नाई)

$c(1,u)=2$

$c(u,v)=4$

Relaxation
If($d[u]+c(u,v) < d[v]$)

$d[v] = d(u)+c(u,v)$

सोर्स १ हते ३ ए याइते डिरेक्ट पाथ ना थाक्लेवे २ ताया हये पाथ आहे; स्टेट पाथ बेर करते Relaxation करते हवे। ताहले फर्म्ला अनुवादी-

If($2+4 < \infty$)
 $d[v] = 2+4$

एथन हवे,

$d[u]=2$
 $d[v]=6$

$c(1,u)=2$
 $c(u,v)=4$

ताहले एथाने सोर्स १ हते ३ ए येते सर्टेट डिस्ट्राइब हवे=६

उदाहरणः
एই ग्राफे सोर्स भाट्टे ० धरे सकल भाट्टे ए याओयार पाथ बेर करते हवे।

Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

Distance:

0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: ~~∞~~
5: ~~∞~~
6: ~~∞~~

१. एथन ० डिजिट करा हवे, तरन तादेव अडजासेट एवं डिस्ट्राइब चेक करा हवे। १ एवं २ के डिस्ट्राइब Relaxation फर्म्ला दिये चेक करा हवे। इनफिनिटिव थेके छोट, ताइ २,६ दिये डिस्ट्राइब रिप्रेस हवे।

Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

Distance:

0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7 from (5 + 2) vs. 14 from (6 + 8)
4: ~~∞~~
5: ~~∞~~
6: ~~∞~~

२. एथन २ डिजिट करा हवे, तार अडजासेट एवं डिस्ट्राइब चेक करा हवे। Relaxation फर्म्ला दिये चेक करा हवे। १ थेके १४ (६+८) वड, ताइ डिस्ट्राइब ७ रिप्रेस हवे ना।

Distance:

0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7 from (5 + 2) vs. 14 from (6 + 8)
4: ~~∞~~
5: ~~∞~~
6: ~~∞~~

६. एथन भाट्टे ३ तधुमात्र वाकी, ताइ भाट्टे ३ एथन डिजिटेड हवे।

Distance:

0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: ~~∞~~
5: ~~∞~~
6: ~~∞~~

Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

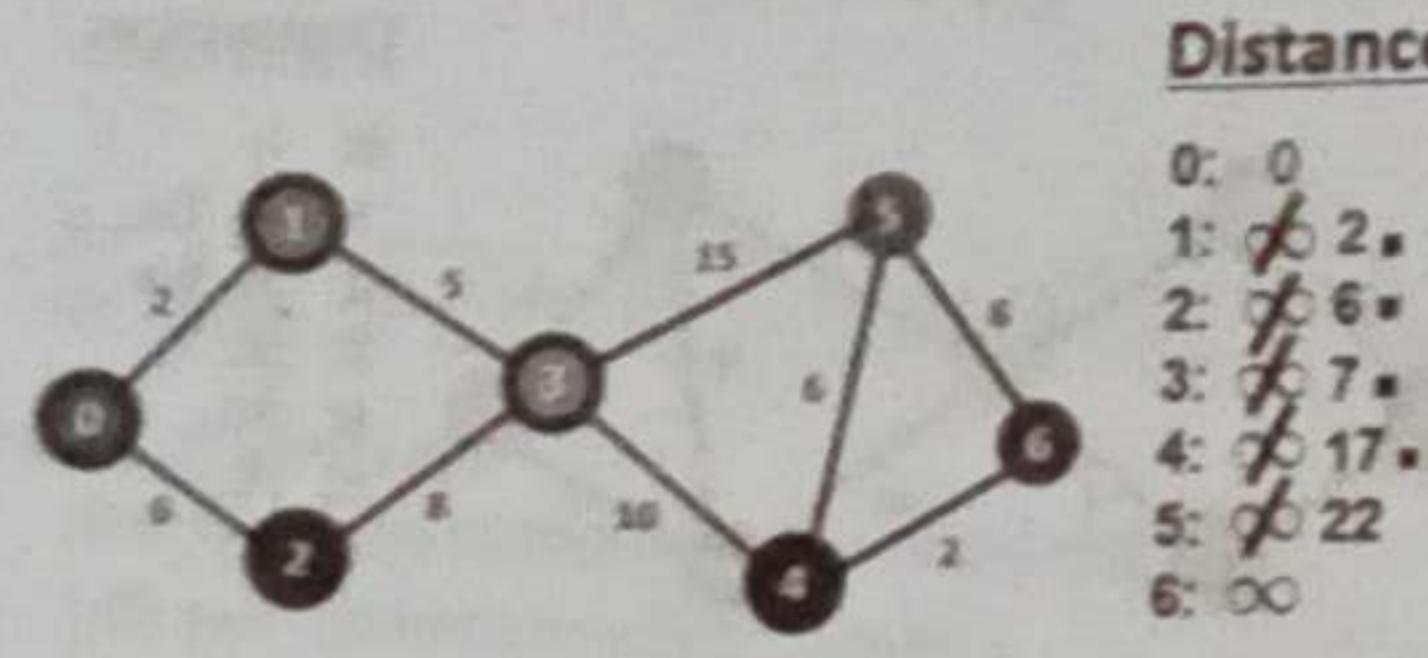
७. एथन ३ डिजिट करा हवे, तार अडजासेट एवं ८,५ एवं डिस्ट्राइब चेक करा हवे। Relaxation फर्म्ला दिये चेक करा हवे। इनफिनिटिव थेके छोट, ताइ १७,२२ दिये डिस्ट्राइब रिप्रेस हवे। ($0 \rightarrow 1 \rightarrow 3 \rightarrow 8, 0 \rightarrow 1 \rightarrow 3 \rightarrow 5$)।

Distance:

0: 0
1: ~~∞~~ 2
2: ~~∞~~ 6
3: ~~∞~~ 7
4: 17 from (2 + 5 + 10)
5: 22 from (2 + 5 + 15)
6: ~~∞~~

Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

8. এখন ভার্টের ৫,৬ এর মাঝে যার ডিস্ট্যান্স কম তাকে সিলেক্ট করতে হবে। এখনে ভার্টের ৮ এর ডিস্ট্যান্স কম।



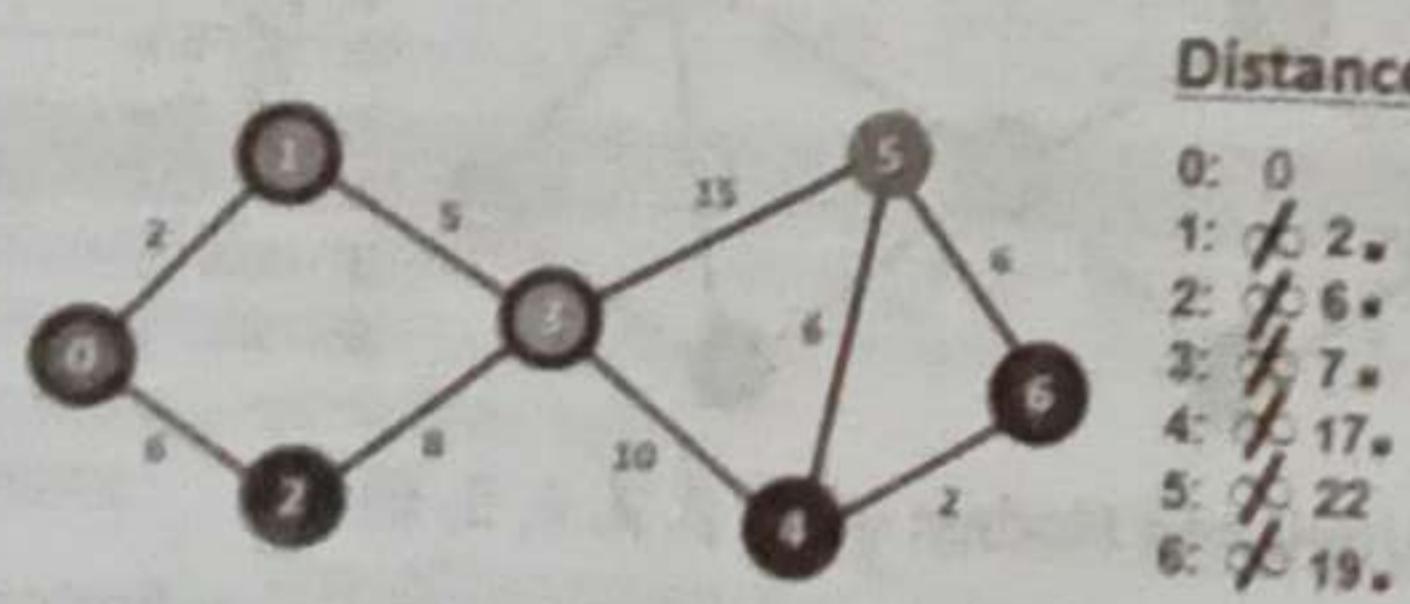
Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

9. এখন ৪ ডিজিট করা হয়েছে, তার আড়জাসেট এর ৫,৬ এর ডিস্ট্যান্স তেক করা হবে। Relaxation ফর্মুলা দিয়ে চেক করা হবে। ভার্টের ৬ এর ডিস্ট্যান্স ১৯, ইনভিনিটির থেকে ছেট, তাই ১৯ দিয়ে ডিস্ট্যান্স রিপ্রেস হবে। ভার্টের ৫ এর ডিস্ট্যান্স বর্তমান ২২ যা ২০ থেকে অপরেতি ছেট, ২০ দিয়ে রিপ্রেস হবে না। (০→১→০→৪→৫→৬, ০→১→০→৪→২)

Distance:

0: 0
1: 2.
2: 6.
3: 7.
4: 17.
5: 22 vs. 19 from (2 + 5 + 10 + 2)
6: 19 from (2 + 5 + 10 + 2)

10. এখন ভার্টের ৫,৬ এর মাঝে যার ডিস্ট্যান্স কম তাকে সিলেক্ট করতে হবে। এখনে ভার্টের ৬ এর ডিস্ট্যান্স কম।



Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

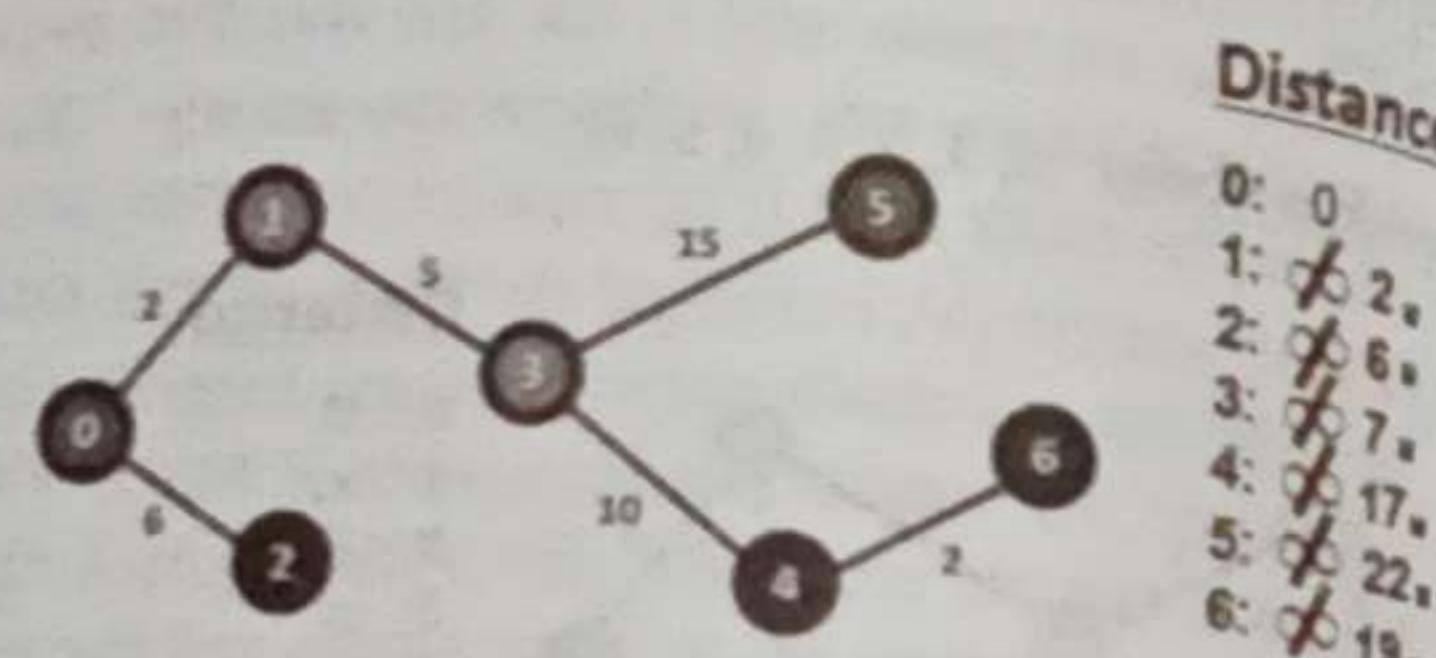
11. এখন ৬ ডিজিট করা হয়েছে, তার আড়জাসেট হলো ৪,৫। তার মাঝে ৪ অপরেতি ডিজিটেট। Relaxation ফর্মুলা দিয়ে ৫ এর ডিস্ট্যান্স ঢেক করা হবে। ভার্টের ৫ এর ডিস্ট্যান্স বর্তমান ২২ যা ২৫ থেকে অপরেতি ছেট, ২৫ দিয়ে রিপ্রেস হবে না। (০→১→০→৪→৫→৬)

Distance:

0: 0
1: 2.
2: 6.
3: 7.
4: 17.
5: 22 from (2 + 5 + 10) vs. 25 from (2 + 5 + 10 + 6) vs. 25 from (2 + 5 + 10 + 2 + 6)
6: 19.

function Dijkstra(Graph, source);

12. এখন ভার্টের ৫ অনুমত বাকী, তাই ভার্টের ৫ এখন ডিজিটেট হবে।



Unvisited Nodes: {0, 1, 2, 3, 4, 5, 6}

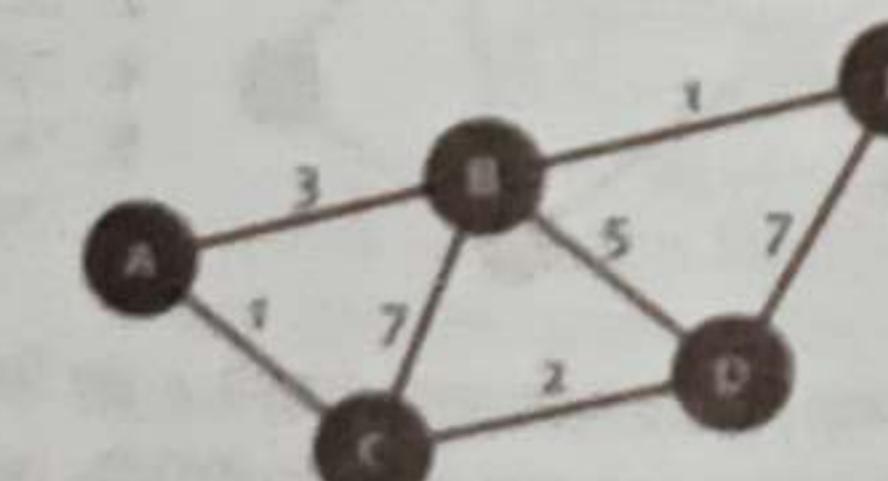
Worst case complexity: $O(n^2)$
[when graph is complete graph.]

3. ▶ Several substations of SGFL Company exist in different places of the city. You have to travel from one substation to another. Write an algorithm to travel using the shortest path between two substations for SGFL Company. [SGFL-2021-AE]

▶ shortest path algorithm [একটি দেশের বিভিন্ন শহর যাতার জন্য স্টেট পথ বের করার আলগরিদম লিখে বলেছিল।] [BPDB-AE -2021]

Ans: We can use Dijkstra algorithm.

Dijkstra's Algorithm: একেবারে একটা নোড থেকে অন্য মেরে নেতৃত্বের মাঝে shortest path খুঁজে বের করার আলগরিদম, যেখে নিজে থাকে A থেকে E shortest path বের করতে হবে, আবার বিভিন্ন থাকে A থেকে E থাইতে পারি, কিন্তু সবচেয়ে কম কষ্ট এ কিভাবে কোন পথ দিয়ে পৌছাতে পারবো তা বের করতে সাহায্য করে ছিল আলগরিদম।



```

for each vertex v in Graph:
    dist[v] := ∞
    previous[v] := undefined
    dist[source] := 0
    Q := the set of all nodes in Graph
    while Q is not empty:
        u := node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u:
            alt := dist[u] + dist_between(u, v)
            if alt < dist[v]:
                dist[v] := alt
                previous[v] := u
    return previous[]
  
```

// Initialization
// initial distance from source to vertex v is set to infinite
// Previous node in optimal path from source
// Distance from source to source
// all nodes in the graph are unoptimized - thus are in Q

// main loop
// where v has not yet been removed from Q.
// Relax (u,v)
// Relaxation (u,v)

Dijkstra algorithm এর অসুবিধা:

1. নেগেটিভ এভজ হ্যান্ডল করতে পারেনা।
2. গ্রাইড সার্চ।

নেগেটিভ edge থাকলে স্টেজ পাথ বের করতে বেলম্যান ফোর্ড আলগরিদম ব্যবহার হয়।

Bellman ford algorithm

4. Explain how to bellman ford algorithm can identify a negative cycle of a graph?

উত্তর: বেলম্যান-ফোর্ড আলগরিদম হল negative edge weights যুক্ত থাকের single source shortest path খুঁজে পাওয়ার উপর (তবে কোনও negative cycles নেই)। এই আলগরিদমের বিষ্টীয় লুপ টি negative cycles সন্দেহ করে। প্রথম লুপে এভাবে একটি edge কে n-1 বার relaxe করে। আবর্ত দাবি করি যে n-1 iteration পরে, distance সঠিক হবে (গ্যারান্টিযুক্ত)।

Algorithm:

d[s]=0

For each vertex V

d[v]=∞ // initialization

for i to (|V|-1)

for each edge (u,v)

If(d[u]+c(u,v))<d[v])

d[v]=d(u)+c(u,v) // relaxation

for each edge (u,v)

If(d[u]+c(u,v))<d[v])

Print "negative -weight cycle exists"

এই আলগরিদম Dijkstra's Algorithm এর মতই তবে নেগেটিভ edge আছে কিনা তা ডিটেক্ট করতে পারে, কারেক্ট স্টেজ পাথ ও বের করতে পারে। Dijkstra's Algorithm এ একবার মাঝে Relaxation করা হয়েছে, এই আলগরিদমে যতশূলো ভার্টের বা নোড আছে তা থেকে 1 কম বার Relaxation করা লাগবে ((|V|-1))। অর্থাৎ মোট ৭ টা থাকলে 6 বার Relaxation করা লাগবে।

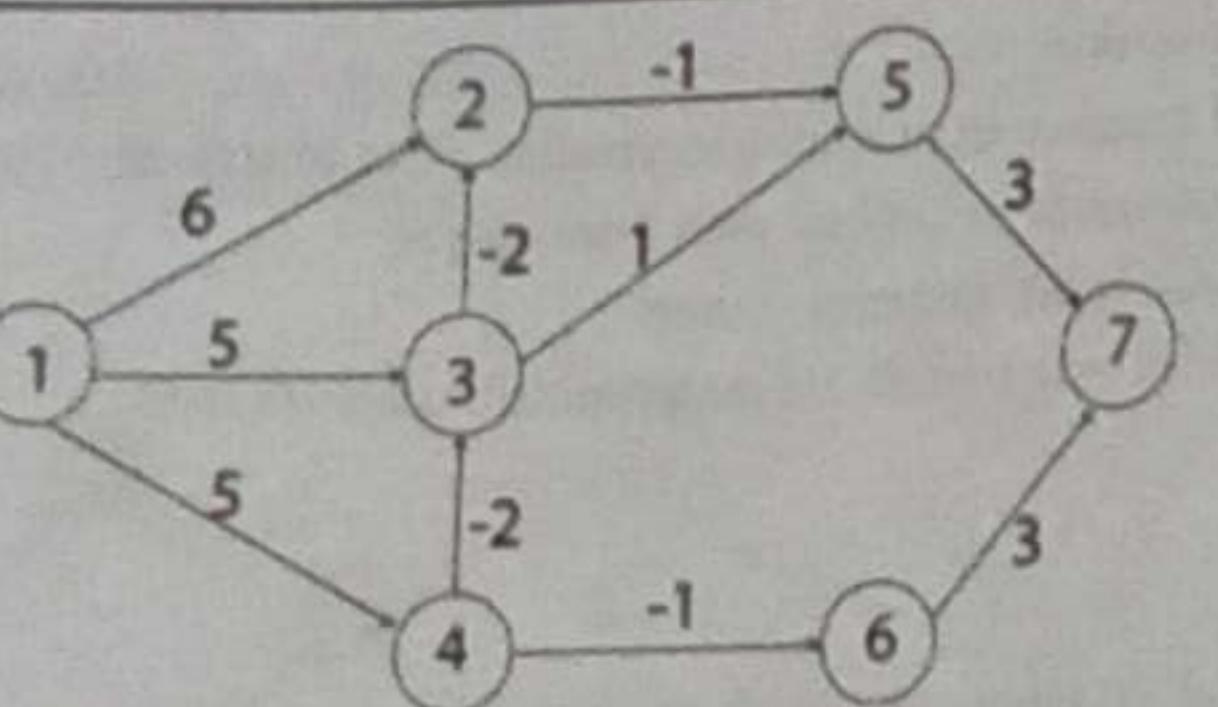
Relaxation

If(d[u]+c(u,v))<d[v])
d[v]= d(u)+c(u,v)

উদাহরণ:

পাশের থাকে 1 কে সোর্স হিসাবে ধরে একটি নোডের স্টেজ পাথ বের করবো, আমাদের 6 বার বিলাঙ্গেশন করা লাগবে, আবর্ত নিচের 6 টো যুক্ত টেবিল এ আছে করবো। প্রথমে আবর্ত ডিটেক্ট পাথ দেখুনোতে আছে তা নির্ণয় করবো, অর্থাৎ edge list তৈরী করবো: edge list = (1,2) (1,3) (1,4) (2,5) (3,2) (3,5) (4,3) (4,6) (5,7) (6,7)। প্রথমে (1,2) নোডের মাঝে দেখবো ডিস্ট্যান্স 6 দেওয়া আছে। অর্থাৎ 1 হচ্ছে u, 2 হচ্ছে

v।

Relaxation

If($d[u] + c(u,v) < d[v]$)
 $d[v] = d(u) + c(u,v)$

Relaxation করে দেখবো এখন,
If($0+6 < \infty$) //সত্তা
 $d[v] = 0+6$

তাহলে নিচের ২ এর ঘরে ৬ বসাবো, একই ভাবে (1,3) (1,4) (2,5) (3,2) (3,5) (4,3) (4,6) (5,7) (6,7) সবগুলো করবো। এবং মোট ৬ বার করলে নিচের মত টেবিল পাওয়া যাবে।

৩ বার করার পর ই স্টেজ ডিস্ট্যাল চলে এসেছে, যদি না আসতো মানিলে ৬ বারে স্টেজ ডিস্ট্যাল চলে আসবে। তাহলে সোর্স ১ থেকে বাকি মোট ৭ টেবিল হলো:

	node						
	1	2	3	4	5	6	7
initialization	0	∞	∞	∞	∞	∞	∞
i)	0	6, 3	5, 3	5	5	4	8, 7
ii)	0	3,	3	5	2	4	5
iii)	0	1	3	5	0	4	3
iv)	0	1	3	5	0	4	3
v)	0	1	3	5	0	4	3
vi)	0	1	3	5	0	4	3

নোড	ডিস্ট্যাল
2	1
3	3
4	5
5	0
6	4
7	3

5. What is a Brute-Force algorithm? Mention some of its usages.

উত্তর: **Brute-Force algorithm:** এটি সমস্যা সমাধানের একটি পদ্ধতি, যেখানে সমাধানের জন্য সম্ভাব্য সকল পদ্ধতিকে বিবেচনা করে, মূলশ্য অ্যালগরিদম। ধরো 8 ডিজিটের পিন নামার মুক্ত একটা Padlock তালা আছে, আপনি পিন নামার জানেন না। চেষ্টা করতে ধরকে $0000, 0001, 0002, \dots, 9999$ । সর্বোচ্চ 10000 বার চেষ্টা করে খুলতে পারবেন। এখানে Brute-Force algorithm এ সম্ভাব্য 10000 সম্ভ্যনাই দিবে।

বাস্তব জীবনে brute force algorithm এর কিছু ব্যবহার:

- Chess playing
- Padlock
- Rubik's cube
- Magic square
- 8-queens puzzle

Question: What is Dijkstra and Bellman ford algorithm? Also, compare them.

Answer: Bellman ford algorithm is used to find the shortest path within a graph containing negative edges. Dijkstra's algorithm, used for the same purpose works for graphs without negative edges.

Dijkstra's Algorithm

Dijkstra's Algorithm doesn't work when there is negative weight edge.

Time consuming

Need large chunk of memory for a big graph to store.

Its lot of time wasting algorithm

Greedy approach is taken to implement the algorithm.

Bellman-Ford's Algorithm

Bellman Ford's Algorithm works when there is negative weight edge, it also detects the negative weight cycle.

Does not scale well

So more complex than other.

It is relatively less time consuming.

Dynamic Programming approach is taken to implement the algorithm.

Data Communication

[**Syllabus:** BPSC CS: Introduction to OSI and TCP/IP protocol. Data transmission basics: analog and digital data, spectrum and bandwidth. Transmission impairments. Data rate channel capacity. Transmission media: Twisted pair, coaxial cable and optical fiber, wireless transmission. Data encoding: NRZ, NRZI, Manchester and differential Manchester modulation techniques-AM, FM, PM, Delta modulation, compounding Equations, ASK, PSK, FSK, QPSK. QAM sampling theorem, PCM, PPM, PAM. Data transmission: Synchronous and asynchronous and asynchronous. NULL modem configuration. Data link control error and flow control CRC and HDLC. Multiplexing: FDM, TDM, statistical TDM. Basic circuit switching and packet switching techniques.]