Sub: Basic

Day

Time: Date: / /

DAA LAB theory for viva

1. Algorithm (type of Algorithm)
2. Complexity বলতে কী বুঝায়, type of complexity.
3. Asymptotic notation?
4. কোন Complexity বোঝার জন্য notation কি কি ধরনের এবং কোনো টিকিয়ে দাও?
5. Topological sort কিভাবে করতে হয়?
6. Eulerian path and point?
7. Breadge কী?
8. Shortest path খোঁজার Algorithm গুলো কোন কোন?
9. Dijkstra and Bellmanford কে কতটি Main অপারেশন?
10. Dijkstra বা Bellman Any graph? এর complexity কী বের করো?

11. ~~David~~ Devide and conquire কি? উদাহরণ
সহ Algorithm Devide and conquire
follow করে।

12. Greedy Algorithm কয় প্রকার? কী কী
Greedy Algorithm ব্যবহৃত?

13. Dynamic এবং Greedy এর পার্থক্য?

14. Network flow Diagram কে বলে এবং?
example?

Solve by H.I Niloy

CSE-37

Q1: What Is Algorithm? Say some type of algorithm?

Ans: An algorithm is a procedure used for solving a problem or performing a computation. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines.

1. Brute Force Algorithm:

This is the most basic and simplest type of algorithm. A Brute Force Algorithm is the straightforward approach to a problem i.e., the first approach that comes to our mind on seeing the problem. More technically it is just like iterating every possibility available to solve that problem.

Example:

2. Recursive Algorithm:

This type of algorithm is based on recursion. In recursion, a problem is solved by breaking it into subproblems of the same type and calling own self again and again until the problem is solved with the help of a base condition.

Some common problem that is solved using recursive algorithms are Factorial of a Number, Fibonacci Series, Tower of Hanoi, DFS for Graph, etc.

a) Divide and Conquer Algorithm:

In Divide and Conquer algorithms, the idea is to solve the problem in two sections, the first section divides the problem into subproblems of the same type. The second section is to solve the smaller problem independently and then add the combined result to produce the final answer to the problem.

Some common problem that is solved using Divide and Conquers Algorithms are Binary Search, Merge Sort, Quick Sort, Strassen's Matrix Multiplication, etc.

b) Dynamic Programming Algorithms:

This type of algorithm is also known as the memoization technique because in this the idea is to store the previously calculated result to avoid calculating it again and again. In Dynamic Programming, divide the complex problem into

smaller overlapping subproblems and store the result for future use.
The following problems can be solved using the Dynamic Programming algorithm Knapsack Problem, Weighted Job Scheduling, Floyd Warshall Algorithm, etc.

c) Greedy Algorithm:

In the Greedy Algorithm, the solution is built part by part. The decision to choose the next part is done on the basis that it gives an immediate benefit. It never considers the choices that had been taken previously.
Some common problems that can be solved through the Greedy Algorithm are Dijkstra Shortest Path Algorithm, Prim's Algorithm, Kruskal's Algorithm, Huffman Coding, etc.

d) Backtracking Algorithm:

In Backtracking Algorithm, the problem is solved in an incremental way i.e. it is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.
Some common problems that can be solved through the Backtracking Algorithm are the Hamiltonian Cycle, M-Coloring Problem, N Queen Problem, Rat in Maze Problem, etc.

3. Randomized Algorithm:

In the randomized algorithm, we use a random number.it helps to decide the expected outcome. The decision to choose the random number so it gives the immediate benefit
Some common problems that can be solved through the Randomized Algorithm are Quicksort: In Quicksort we use the random number for selecting the pivot.

4. Sorting Algorithm:

The sorting algorithm is used to sort data in maybe ascending or descending order. Its also used for arranging data in an efficient and useful manner.
Some common problems that can be solved through the sorting Algorithm are Bubble sort, insertion sort, merge sort, selection sort, and quick sort are examples of the Sorting algorithm.

5. Searching Algorithm:

The searching algorithm is the algorithm that is used for searching the specific key in particular sorted or unsorted data. Some common problems that can be solved through the Searching Algorithm are Binary search or linear search is one example of a Searching algorithm.

6. Hashing Algorithm:

Hashing algorithms work the same as the Searching algorithm but they contain an index with a key ID i.e a key-value pair. In hashing, we assign a key to specific data.

Q2: What you understand by complexity? Say some type of complexity?

Ans:

In algorithmic complexity, we're concerned with understanding how the performance of an algorithm scales with the size of its input. Here are some types of complexity commonly discussed:

1. **Time Complexity:** This measures the amount of time an algorithm takes to run as a function of the size of its input. It answers questions like "How many basic operations does the algorithm perform?" Examples include:
    - **Constant Time (O(1)):** The running time of the algorithm does not change with the size of the input.
    - **Linear Time (O(n)):** The running time increases linearly with the size of the input.
    - **Logarithmic Time (O(log n)):** The running time grows logarithmically with the size of the input.
    - **Quadratic Time (O(n^2)):** The running time grows quadratically with the size of the input.

- **Exponential Time (O(2^n)):** The running time grows exponentially with the size of the input.

2. **Space Complexity:** This measures the amount of memory space an algorithm uses as a function of the size of its input. It answers questions like "How much memory is required by the algorithm?" Space complexity is also expressed using big O notation, similar to time complexity.
3. **Best, Worst, and Average Case Complexity:** Algorithms may perform differently depending on the input data. The best-case complexity represents the minimum amount of resources an algorithm requires, the worst-case complexity represents the maximum amount, and the average-case complexity represents an expected performance under some probabilistic assumptions.
4. **Amortized Complexity:** This considers the average time taken per operation in a sequence of operations, even though a single operation might be more expensive. It's often used in data structures where some operations are more costly than others, but the overall cost is spread out over multiple operations.
5. **Asymptotic Complexity:** This describes the behavior of the algorithm as the input size approaches infinity. It focuses on the dominant term of the complexity function and ignores constant factors and lower-order terms, providing a high-level understanding of how the algorithm scales with input size.

Q2: what is Asymptotic Notation?

The main idea of asymptotic analysis is to have a measure of the efficiency of algorithms that don't depend on machine-specific constants and don't require algorithms to be implemented and time taken by programs to be compared. Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.

Asymptotic Notations:

.Asymptotic Notations are mathematical tools that allow you to analyze an algorithm's running time by identifying its behavior as its input size grows.

.This is also referred to as an algorithm's growth rate.

.You can't compare two algorithm's head to head.

.You compare space and time complexity using asymptotic analysis.

.It compares two algorithms based on changes in their performance as the input size is increased or decreased.

There are mainly three asymptotic notations:

1. ***Big-O Notation (O-notation)***
2. ***Omega Notation (Ω-notation)***
3. ***Theta Notation (Θ-notation)***

1. Theta Notation (Θ-Notation):

Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, *it is used for analyzing the* ***average-case*** *complexity of an algorithm.*

2. Big-O Notation (O-notation):

Big-O notation represents the upper bound of the running time of an algorithm*. Therefore, it gives the worst-case complexity of an algorithm.*

3. Omega Notation (Ω-Notation):

Omega notation represents the lower bound of the running time of an algorithm*. Thus, it provides the best case complexity of an algorithm.* **The execution time serves as a lower bound on the algorithm's time complexity.**

Q4: What do you understand by complexity or notation? Which is good use for us?

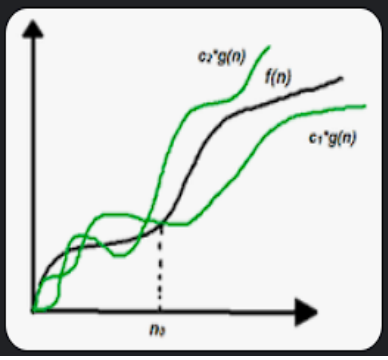
Ans: (What do you understand by complexity or notation?) Follow previous question.

What is the best complexity notation?

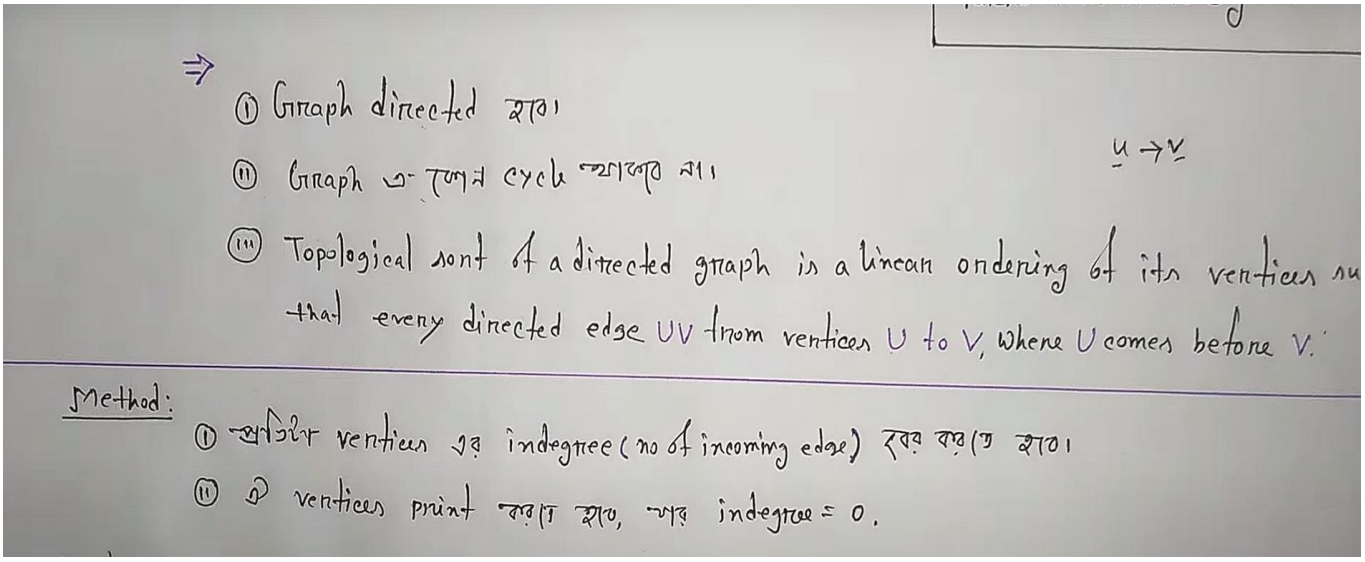
Omega Notation (Ω-Notation):

Thus, it provides the best case complexity of an algorithm.

Nov 22, 2023



Q5: How does topological sort work?



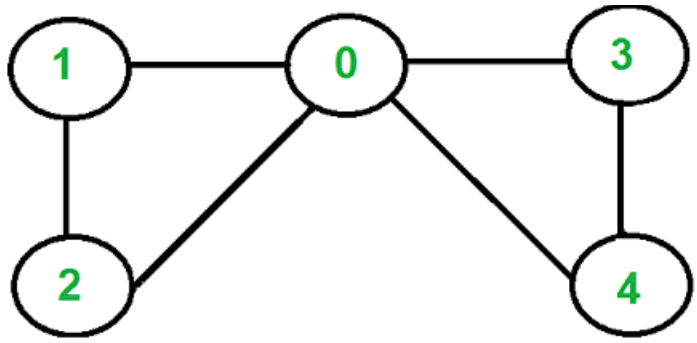
Q6: Eulerian path and point?

Ans:

An Euler path is a path that uses every edge of a graph exactly once.

An Euler circuit is a circuit that uses every edge of a graph exactly once. ▶ An Euler path starts and ends at different vertices. ▶ An Euler circuit starts and ends at the same vertex.

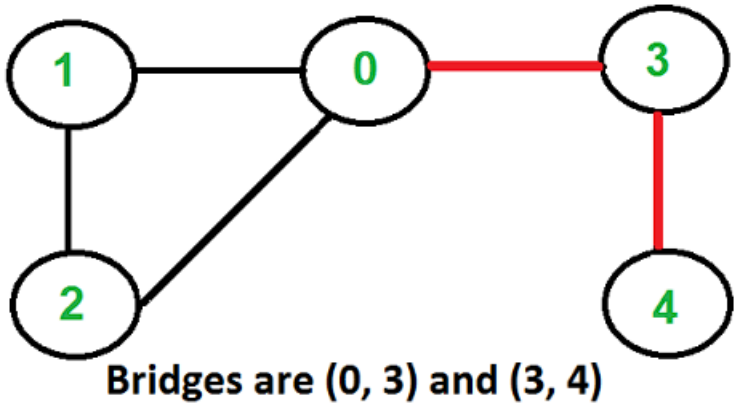
An example of an Euler path is **0,2,1,0,3,4**Each number represents a point, or vertex, on the path. The path starts at vertex 0 and ends at vertex 4.



The graph has Eulerian Cycles, for example "2 1 0 3 4 0 2"
Note that all vertices have even degree

Q7: What is Bridge?

Ans: ***An edge in an undirected connected graph is a bridge if removing it disconnects the graph.*** *For a disconnected undirected graph, the definition is similar, a bridge is an edge removal that increases the number of disconnected components.*



Q8: Find shortest path algorithm name?

Ans:

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.

Algorithms [edit]

The most important algorithms for solving this problem are:

- Dijkstra's algorithm solves the single-source shortest path problem with non-negative edge weight.
- Bellman–Ford algorithm solves the single-source problem if edge weights may be negative.
- A* search algorithm solves for single-pair shortest path using heuristics to try to speed up the search.
- Floyd–Warshall algorithm solves all pairs shortest paths.
- Johnson's algorithm solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs.
- Viterbi algorithm solves the shortest stochastic path problem with an additional probabilistic weight on each node.

Additional algorithms and associated evaluations may be found in Cherkassky, Goldberg & Radzik (1996).

Q9: Main difference Dijkstra's Algorithm and Bellman Ford's Algorithm?

Ans :

Bellman Ford's Algorithm works when there is negative weight edge, it also detects the negative weight cycle. Dijkstra's Algorithm doesn't work when there is negative weight edge.

Q10: Dijkstra and bellman ford algorithm any graph complexity?

One of the main benefits of Dijkstra's Algorithm is its efficiency. It has a time complexity of O(|E| + |V| log |V|)

However, one downside to the

Bellman-Ford algorithm is its relatively slow time complexity of O(|E| * |V|), where |E| is the number of edges in the graph and |V| is the number of vertices.

The space complexity of the Bellman-Ford algorithm is O(V).

Q11: Divide and Conquer Algorithm?

A divide and conquer algorithm **is a strategy of solving a large problem by**

1. **breaking the problem into smaller sub-problems**
2. **solving the sub-problems, and**
3. **combining them to get the desired output.**

Algorithm name:

Binary Search.

Merge Sort.

Quick Sort.

Calculate pow(x, n)

Karatsuba algorithm for fast multiplication.

Strassen's Matrix Multiplication.

Convex Hull (Simple Divide and Conquer Algorithm)

Quickhull Algorithm for Convex Hull.

Q12: What is greedy algorithm? Name of greedy algorithm? Type of greedy Algorithm?

Ans:

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment.

Different Types of Greedy Algorithm

- Selection Sort
- Knapsack Problem
- Minimum Spanning Tree
- Single-Source Shortest Path Problem
- Job Scheduling Problem
- Prim's Minimal Spanning Tree Algorithm
- Kruskal's Minimal Spanning Tree Algorithm
- Dijkstra's Minimal Spanning Tree Algorithm
- Huffman Coding
- Ford-Fulkerson Algorithm

Q13: Difference between greedy and dynamic programming?

Ans:

Generally speaking, there are two approaches to solving programming problems: greedy and dynamic.

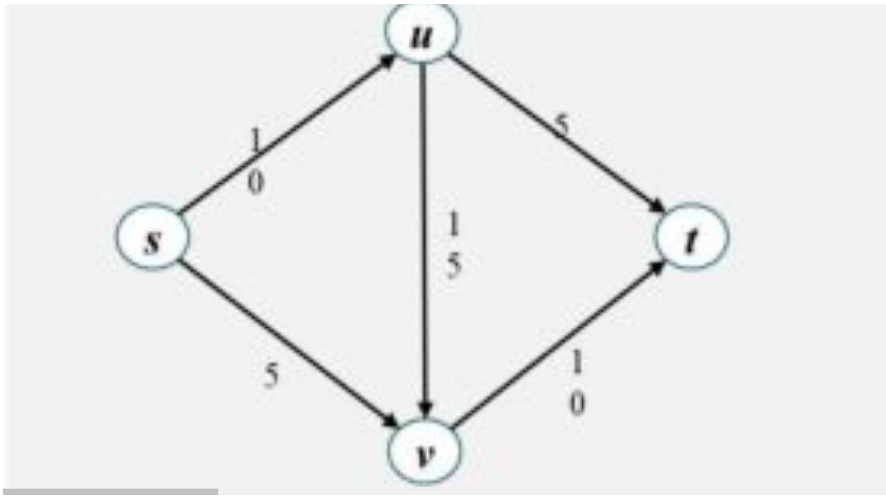
Greedy programming focuses on solving problems quickly, whereas dynamic programming focuses on solving problems efficiently.

Parameter	Greedy Programming	Dynamic Programming
Basic	Produces one decision sequence.	Generates many possible decision sequences
Reliability	There is a lower degree of reliability with this method	There is high reliability in this method
Working Approach	It prefers a top-down approach	It prefers a Bottom-up approach
Solutions	Provide a specific set of feasible solutions.	No particular set of feasible solutions exists.
Efficiency	It is more efficient than dynamic methods	A less efficient method than greedy methods
Example	Fractional knapsack, shortest path	0/1 Knapsack

Q14: what does work by Network flow diagram with example?

A network flow diagram maps the flow of data through networks .

Digital systems often involve network-connected systems with functionality distributed across multiple nodes. For example, in an ecommerce store, data might move from an order system to invoicing, payment, and logistics systems.



Examples include coordination of trucks in a transportation system, routing of packets in a communication network, and sequencing of legs for air travel. Such problems often involve few indivisible objects, and this leads to a finite set of feasible solutions.

Type of Network flow problems maximum cut or minimum cut;

1.The Ford-Fulkerson algorithm.

2.Bipartite Matching.

3.Edmonds karp algorithms.

Chill man Chill

Main Hoon Nah!

Thank you