DATA STRUCTURE AND ALGORITHM (SELF PACED)

Geeks for Geeks

A Training report

Submitted in partial fulfillment of the requirements for the award of degree of

B. Tech

(School of Computer Science & Engineering)

Submitted to
LOVELY PROFESSIONAL UNIVERSITY
PHAGWARA, PUNJAB

From 04/06/24 to 27/08/24

SUBMITTED BY

Name of student: Mohd Juned khan
Registration Number:12220071
Signature of the student:

Mohd Juned Khan.

DECLARATION

I hereby declare that I have completed my six weeks summer training at Geeks for Geeks platform from June 06,2024 to July 27,2024 under the guidance of MR. Sandeep Jain. I have declared that I have worked full dedication during their 8 weeks of training and my learning outcomes fulfill the requirements of training for the award of degree of B.tech. CSE, Lovely Professional University, Phagwara.

Name of Student: Mohd Juned khan
Registration no: 12220071

ACKNOWLEDGEMENT

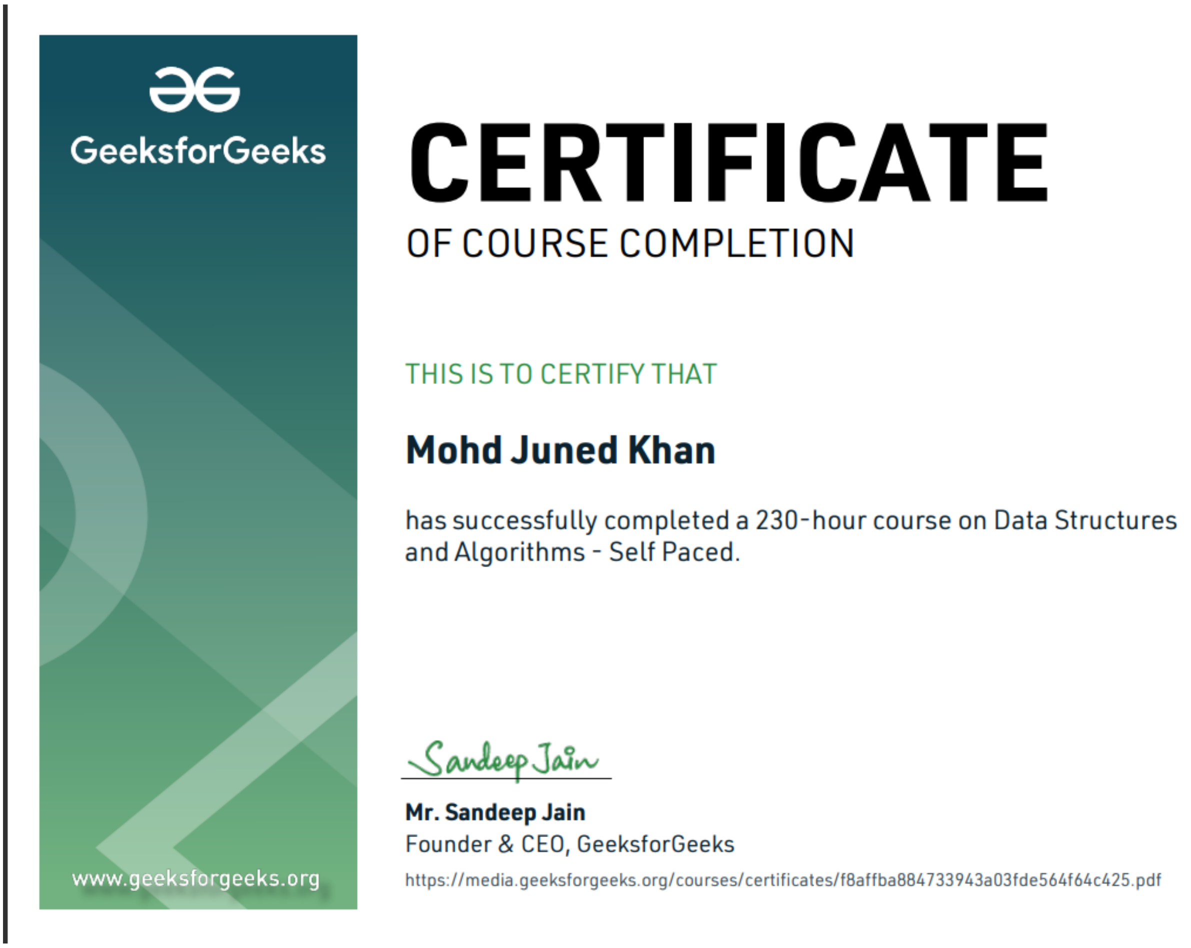
I would like to express my gratitude towards my university as well as Geeks for Geeks for providing me with the golden opportunity to do this wonderful summer training regarding DSA, which also helped me in doing a lot of homework and learning. As a result, I came to know about so many new things. So, I am really thanking full to them.

Moreover, I would like to thank my friends who helped me a lot whenever I got stuck in some problem related to my course. I am thankful to have such a good support from them as they always have my back whenever I need it.

Also, I would like to mention the support system and consideration of my parents who have always been there in my life to make me choose the right thing and oppose the wrong. Without them I could never have learned and become the person who I am now.

I have made efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

Summer Training Certificate by Geeks for Geeks



GeeksforGeeks

CERTIFICATE

OF COURSE COMPLETION

THIS IS TO CERTIFY THAT

Mohd Juned Khan

has successfully completed a 230-hour course on Data Structures and Algorithms - Self Paced.

Sandeep Jain

Mr. Sandeep Jain

Founder & CEO, GeeksforGeeks

https://media.geeksforgeeks.org/courses/certificates/f8affba884733943a03fde564f64c425.pdf

www.geeksforgeeks.org

Link: https://www.geeksforgeeks.org/certificate/f8affba884733943a03fde564f64c425

INTRODUCTION

DSA self-paced course is a complete package that helped me to learn Data Structures and Algorithms from Basic to an Advance level. The course curriculum has been divided into 10 weeks, where I practiced questions, and I have attempted the assessment tests accordingly. The course offers a wealth of programming challenges that helped me to learn all about DSA and making of an algorithm and how to solve problems and the logic behind the Algorithm.

The course was Self placed means I could join the course anytime and all the content will be available to me once I get enrolled. There were video lectures to learn form and multiple-choice questions to practice.

I learned Algorithmic techniques for solving various problems with full flexibility of time as I was not time bounded.

This course does not require any prior knowledge of Data Structure and Algorithms, but a basic knowledge of any programming language (C++ / Java) will be helpful.

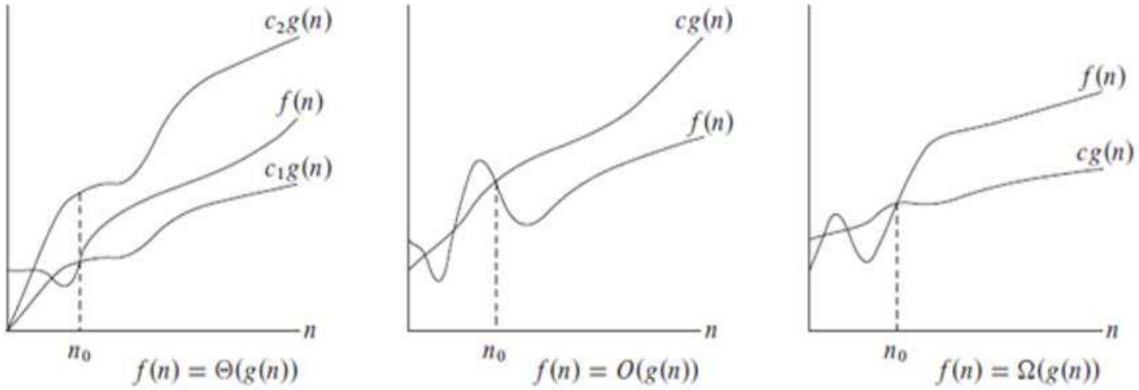
And as we all know Data Structure and Algorithm is a must skill in terms of Placement in any company because it helps us to increase our problem-solving skill.

TECHNOLOGY LEARNT

It had 24 units which was further divided into chapters and then topics so during my whole 8-week **course I learned the following**:

INTRODUCTION TO DSA

- **Analysis of Algorithm**
  - In this I learned about background analysis through a Program and its functions.
- **Order of Growth**
  - A mathematical explanation of the growth analysis through limits and functions.
  - A direct way of calculating the order of growth
- **Asymptotic Notations**
  - Best, Average and Worst-case explanation through a program.
- **Big O Notation**
  - Graphical and mathematical explanation.
  - Calculation
  - Applications at Linear Search
- **Omega Notation**
  - Graphical and mathematical explanation.
  - Calculation.
- **Theta Notation**
  - Graphical and mathematical explanation.
  - Calculation.



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

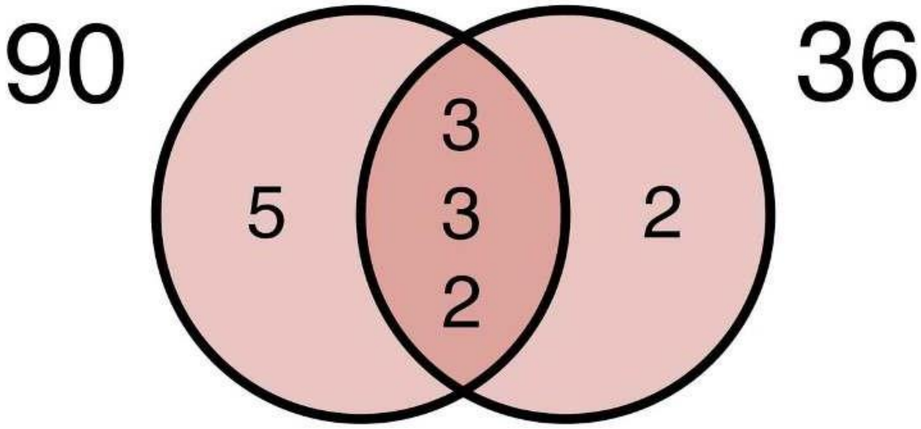
$f(n) = O(g(n))$ (b)

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

- **Analysis of common loops**
  - Single, multiple and nested loops
- **Analysis of Recursion**
  - Various calculations through Recursion Tree method
- **Space Complexity**
  - Basic Programs
  - Auxiliary Space
  - Space Analysis of Recursion
  - Space Analysis of Fibonacci number

MATHEMATICS

- **Finding the number of digits in a number.**
- **Arithmetic and Geometric Progressions.**
- **Quadratic Equations.**
- **Mean and Median.**
- **Prime Numbers.**
- **LCM and HCF**
- **Factorials**
- **Permutations and Combinations**
- **Modular Arithmetic**



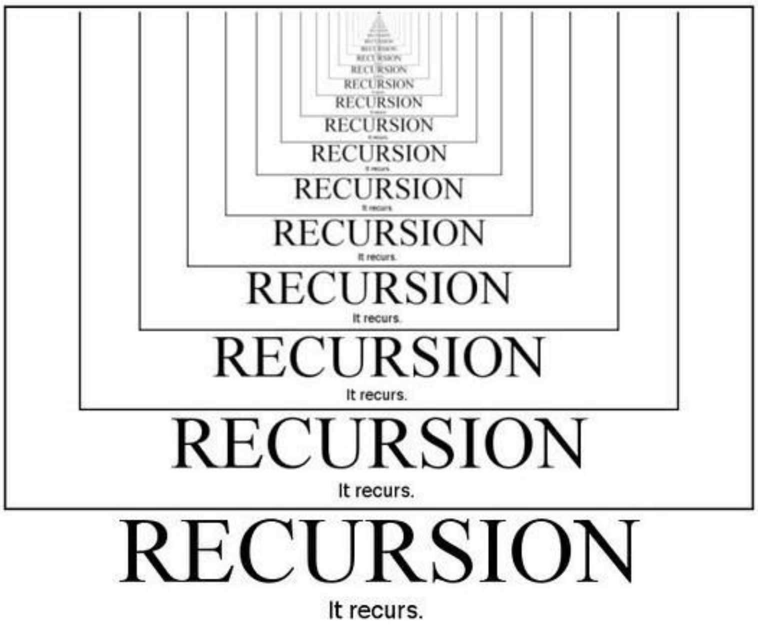
<u>GCD of two number 90 and 36</u>

BITMAGIC

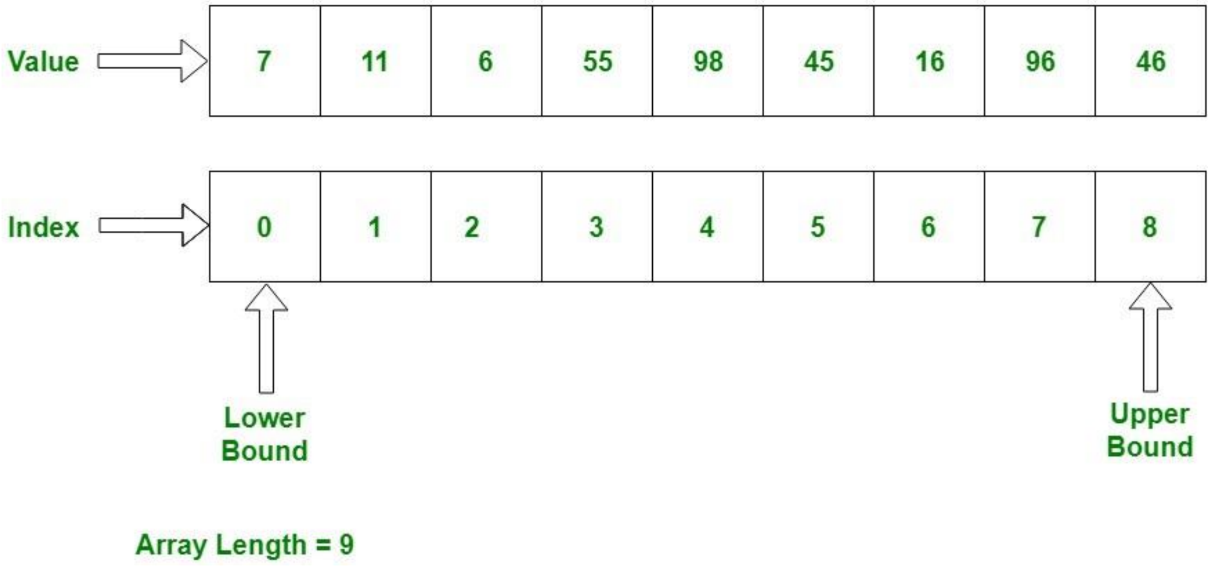
- **Bitwise Operators in C++**
    - Operation of AND, OR XOR operators
    - Operation of Left Shift, Right Shift and Bitwise Not
- **Bitwise Operators in Java**
    - Operation of AND, OR
    - Operation of Bitwise Not, Left Shift
    - Operation of Right Shift and unsigned Right Shift
- **Problem (With Video Solutions): Check Kth bit is set or not**
    - Method 1: Using the left Shift.
    - Method 2: Using the right shift

RECURSION

- **Introduction to Recursion**
- **Applications of Recursion**
- **Writing base cases in Recursion**
    - Factorial
    - N-th Fibonacci number



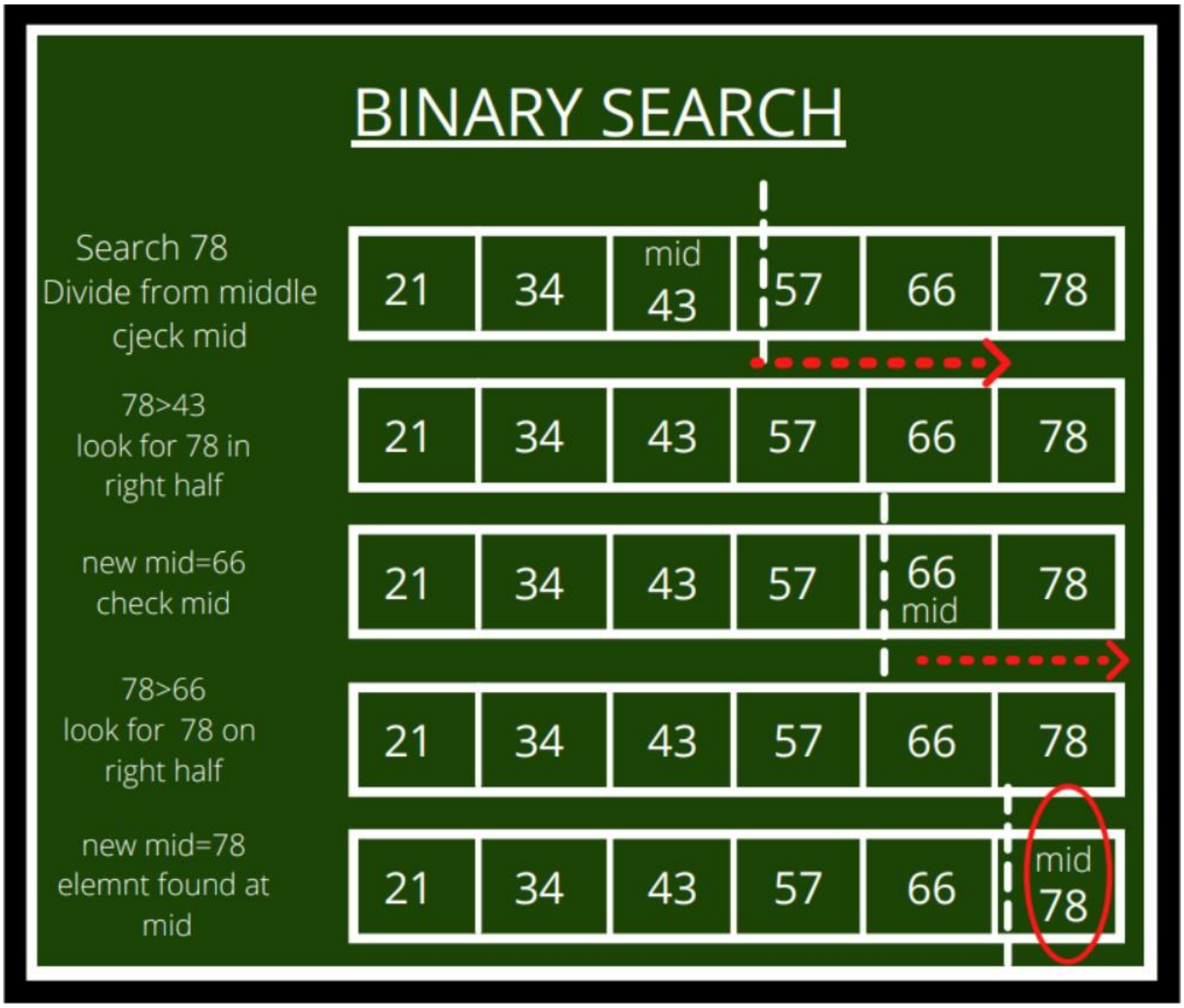
ARRAYS



- **Introduction and Advantages**
- **Types of Arrays**
    - Fixed-sized array
    - Dynamic-sized array
- **Operations on Arrays**
    - Searching
    - Insertions
    - Deletion
    - Arrays vs other DS
    - Reversing - Explanation with complexity

SEARCHING

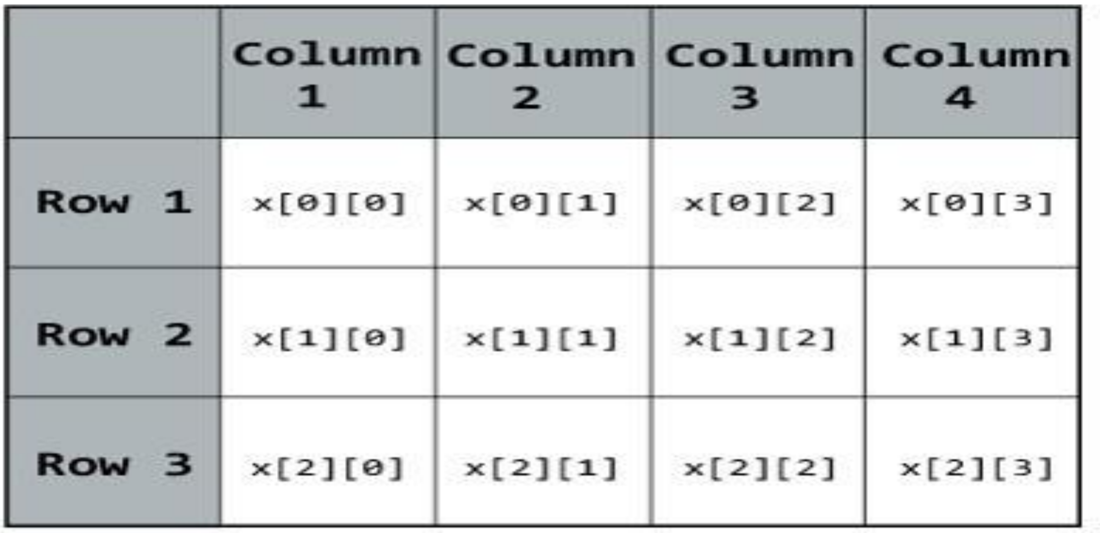
- **Binary Search Iterative and Recursive**
- **Binary Search and various associated problems**
- **Two Pointer Approach Problems**



SORTING

- **Implementation of C++ STL sort () function in Arrays and Vectors**
  - Time Complexities
- **Sorting in Java**
- **Arrays. Sort () in Java**
- **Collection. Sort () in Java**
- **Stability in Sorting Algorithms**
  - Examples of Stable and Unstable Algos
- **Insertion Sort**
- **Merge Sort**
- **Quick Sort**
  - Using Lomuto and Hoare
  - Time and Space analysis
  - Choice of Pivot and Worst case
- **Overview of Sorting Algorithms**

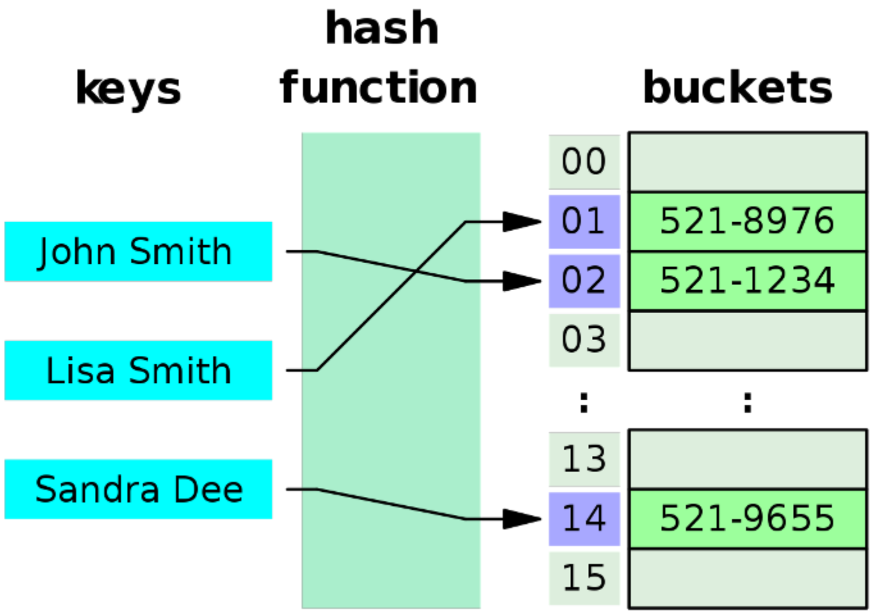
MATRIX



	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

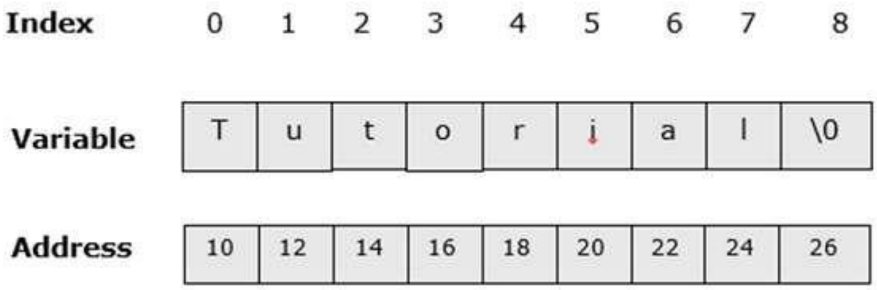
- **Introduction to Matrix in C++ and Java**
- **Multidimensional Matrix**
- **Pass Matrix as Argument**
- **Printing matrix in a snake pattern**
- **Transposing a matrix**
- **Rotating a Matrix**
- **Check if the element is present in a row and column-wise sorted matrix.**
- **Boundary Traversal**
- **Spiral Traversal**
- **Matrix Multiplication**
- **Search in row-wise and column-wise Sorted Matrix**

HASHING



- Introduction and Time complexity analysis
- Application of Hashing
- Discussion on Direct Address Table
- Working and examples on various Hash Functions
- Introduction and Various techniques on Collision Handling
- Chaining and its implementation
- Open Addressing and its Implementation
- Chaining V/S Open Addressing
- Double Hashing
- C++
  - Unordered Set
  - Unordered Map

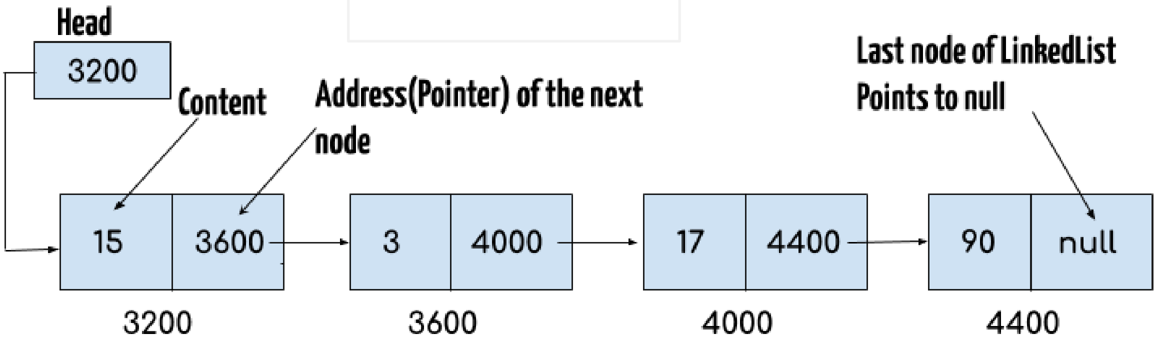
STRINGS



Index	0	1	2	3	4	5	6	7	8
Variable	T	u	t	o	r	i	a	l	\0
Address	10	12	14	16	18	20	22	24	26

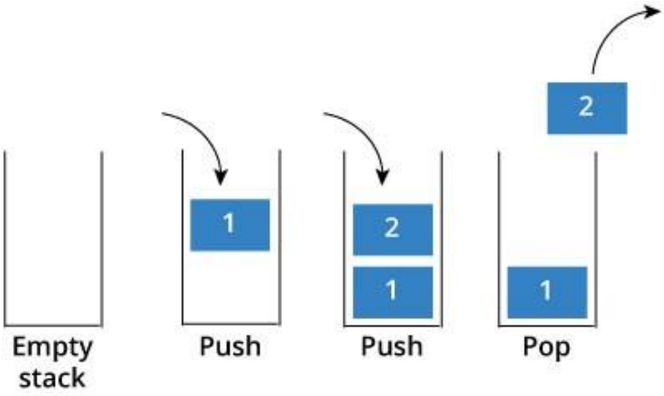
- **Discussion of String DS**
- **Strings in CPP**
- **Strings in Java**
- **Rabin Karp Algorithm**
- **KMP Algorithm**

LINKED LIST


Head
3200
Content
Address(Pointer) of the next node
Last node of LinkedList Points to null
15
3600
3
4000
17
4400
90
null
3200
3600
4000
4400

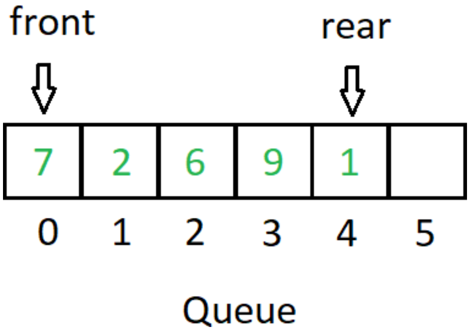
- **Introduction**
  - Implementation in CPP
  - Implementation in Java
  - Comparison with Array DS
- **Doubly Linked List**
- **Circular Linked List**
- **Loop Problems**
  - Detecting Loops
  - Detecting loops using Floyd cycle detection
  - Detecting and Removing Loops in Linked List

STACK



- **Understanding the Stack data structure**
- **Applications of Stack**
- **Implementation of Stack in Array and Linked List**
    - In C++
    - In Java

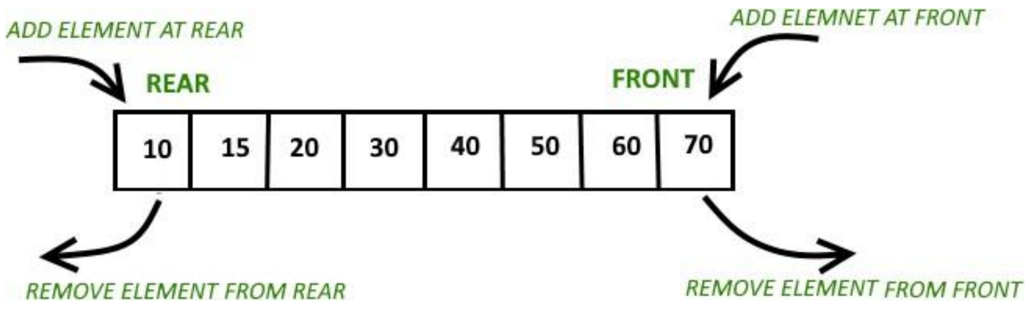
QUEUE



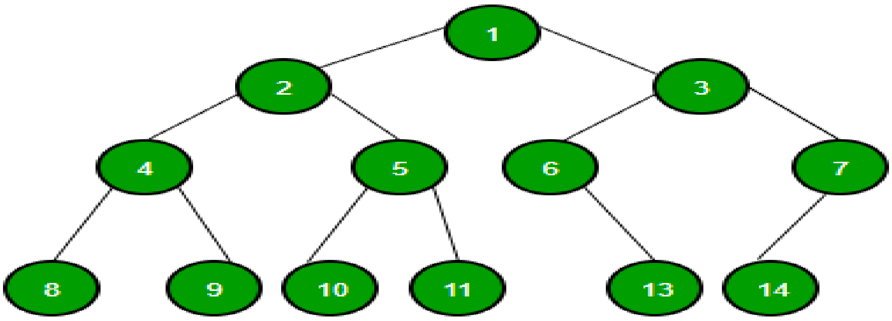
- **Introduction and Application**
- **Implementation of the queue using array and LinkedList**
    - In C++ STL
    - In Java
    - Stack using queue

DEQUE

- **Introduction and Application**
- **Implementation**
  - In C++ STL
  - In Java
- **Problems (With Video Solutions)**
  - Maximums of all subarrays of size k
  - Array Deque in Java
  - Design a DS with min max operations

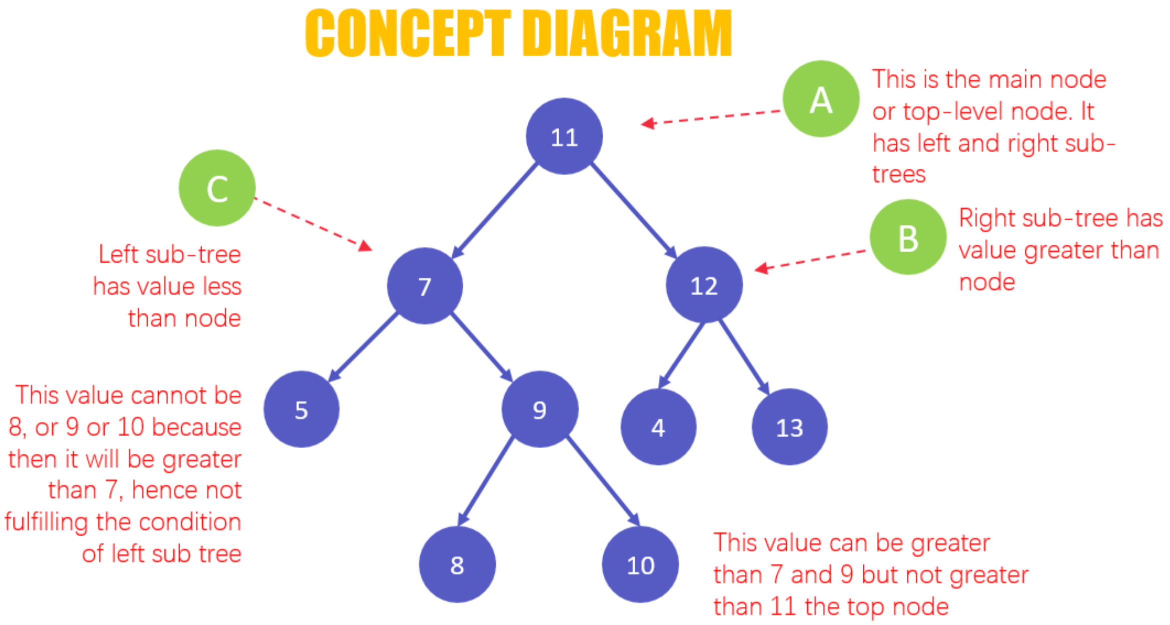


TREE



- **Introduction**
  - Tree
  - Application
  - Binary Tree
  - Tree Traversal
- **Implementation of:**
  - Inorder Traversal
  - Preorder Traversal
  - Postorder Traversal
  - Level Order Traversal (Line by Line)
  - Tree Traversal in Spiral Form

BINARY SEARCH TREE



This is the main node or top-level node. It has left and right sub-trees

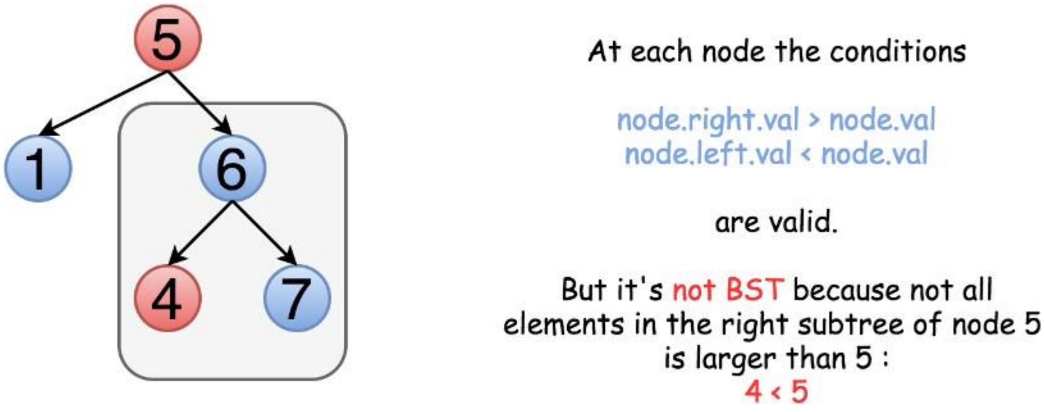
Right sub-tree has value greater than node

Left sub-tree has value less than node

This value cannot be 8, or 9 or 10 because then it will be greater than 7, hence not fulfilling the condition of left sub tree

This value can be greater than 7 and 9 but not greater than 11 the top node

- **Background, Introduction and Application**
- **Implementation of Search in BST**
    - In CPP
    - In Java
- **Insertion in BST**
    - In CPP
    - In Java
- **Deletion in BST**
    - In CPP
    - In Java
- **Floor in BST**
    - In CPP
    - In Java
- **Self-Balancing BST**
- **AVL Tree**
- **Red**, Black **Tree**
- **Set in C++ STL**
- **Map in C++ STL**



At each node the conditions

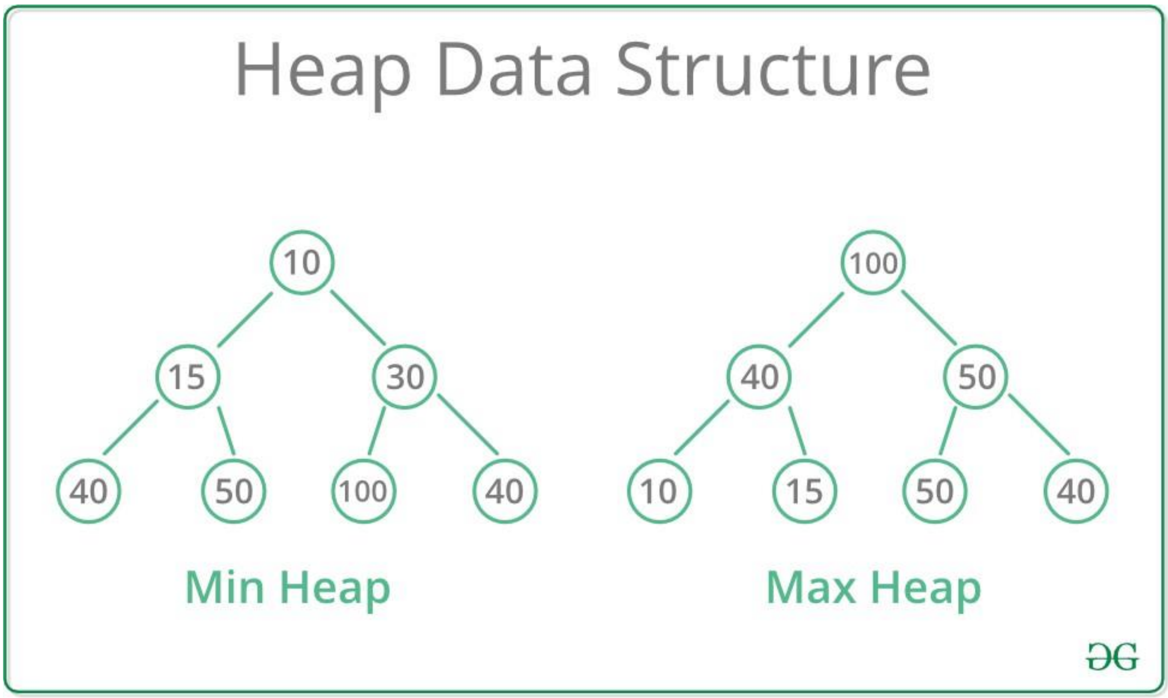
node.right.val > node.val
node.left.val < node.val

are valid.

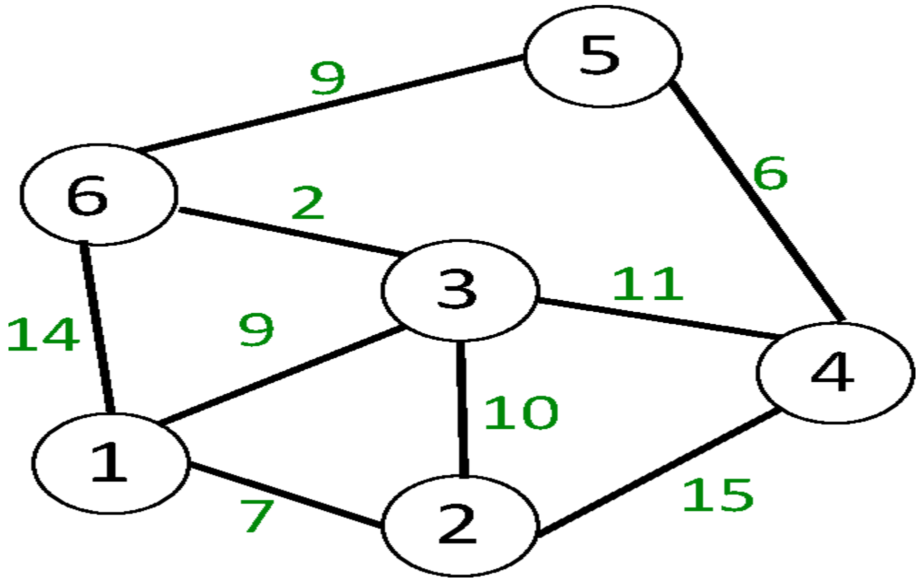
But it's not BST because not all elements in the right subtree of node 5 is larger than 5 :
4 < 5

HEAP

- **Introduction & Implementation**
- **Binary Heap**
  - Insertion
  - Heapify and Extract
  - Decrease Key, Delete and Build Heap
- **Heap Sort**
- **Priority Queue in C++**
- **Priority Queue in Java**

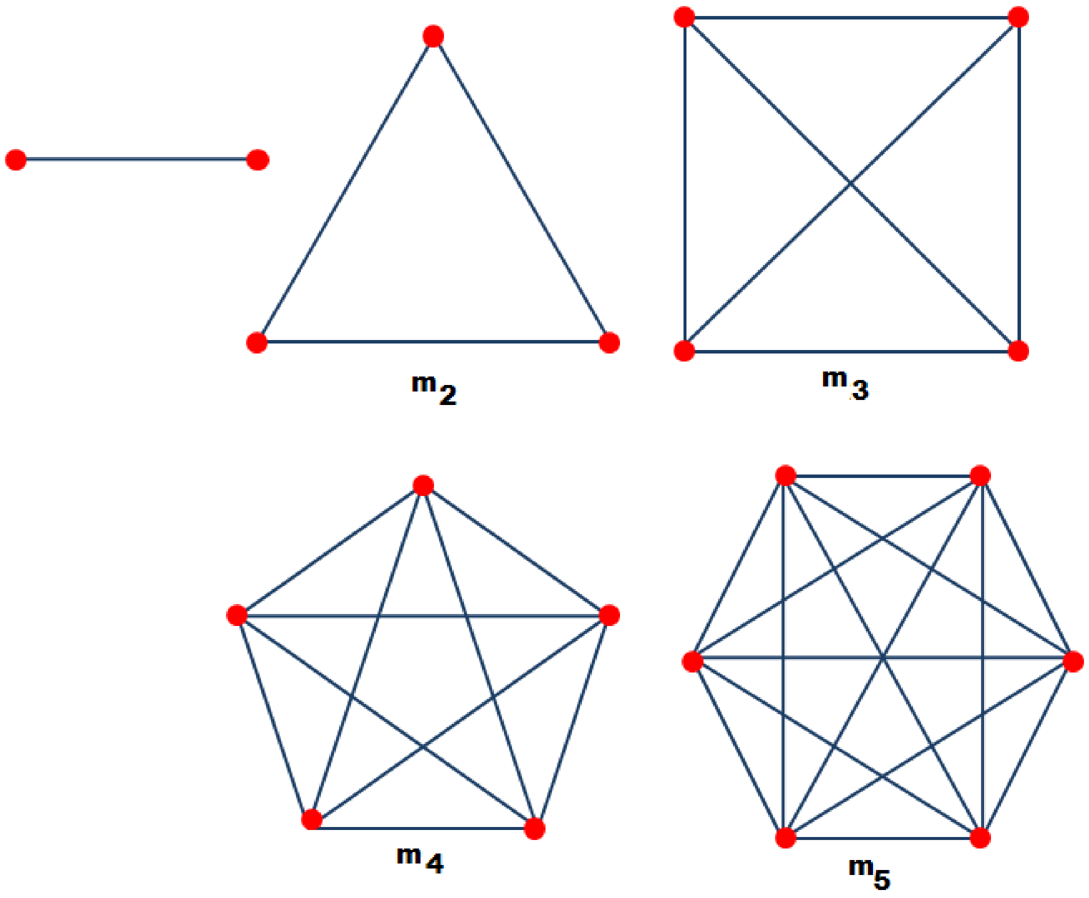


GRAPH



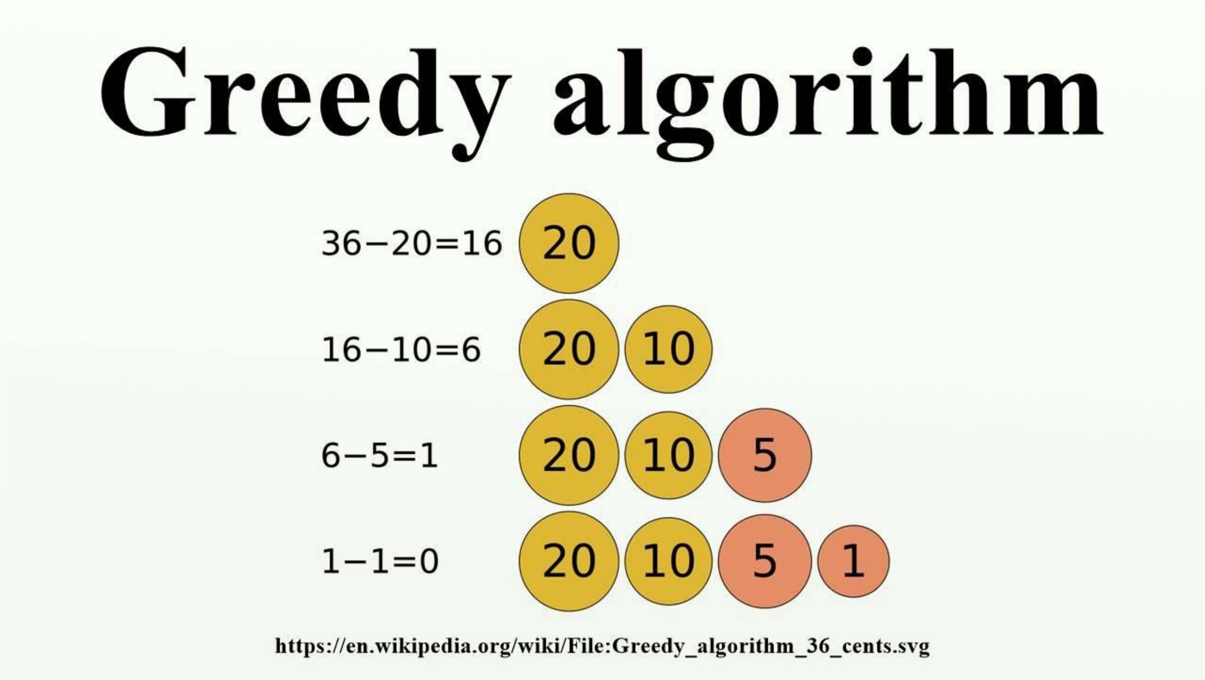
- **Introduction to Graph**
- **Graph Representation**
  - Adjacency Matrix
  - Adjacency List in CPP and Java
  - Adjacency Matrix VS List
- **Breadth-First Search**
  - Applications
- **Depth First Search**
  - Applications
- **Shortest Path in Directed Acyclic Graph**
- **Prim's Algorithm/Minimum Spanning Tree**
  - Implementation in CPP
  - Implementation in Java
- **Dijkstra's Shortest Path Algorithm**
  - Implementation in CPP
  - Implementation in Java
- **Bellman-Ford Shortest Path Algorithm**

- **Kosaraju's Algorithm**
- **Articulation Point**
- **Bridges in Graph**
- **Tarjan’s Algorithm**



GREEDY

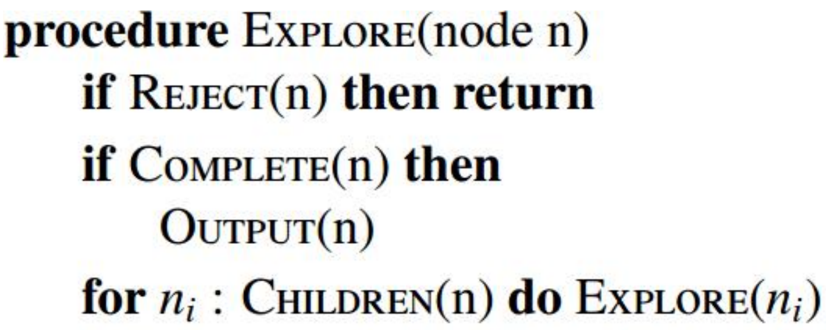
- **Introduction**
- **Activity Selection Problem**
- **Fractional Knapsack**
- **Job Sequencing Problem**



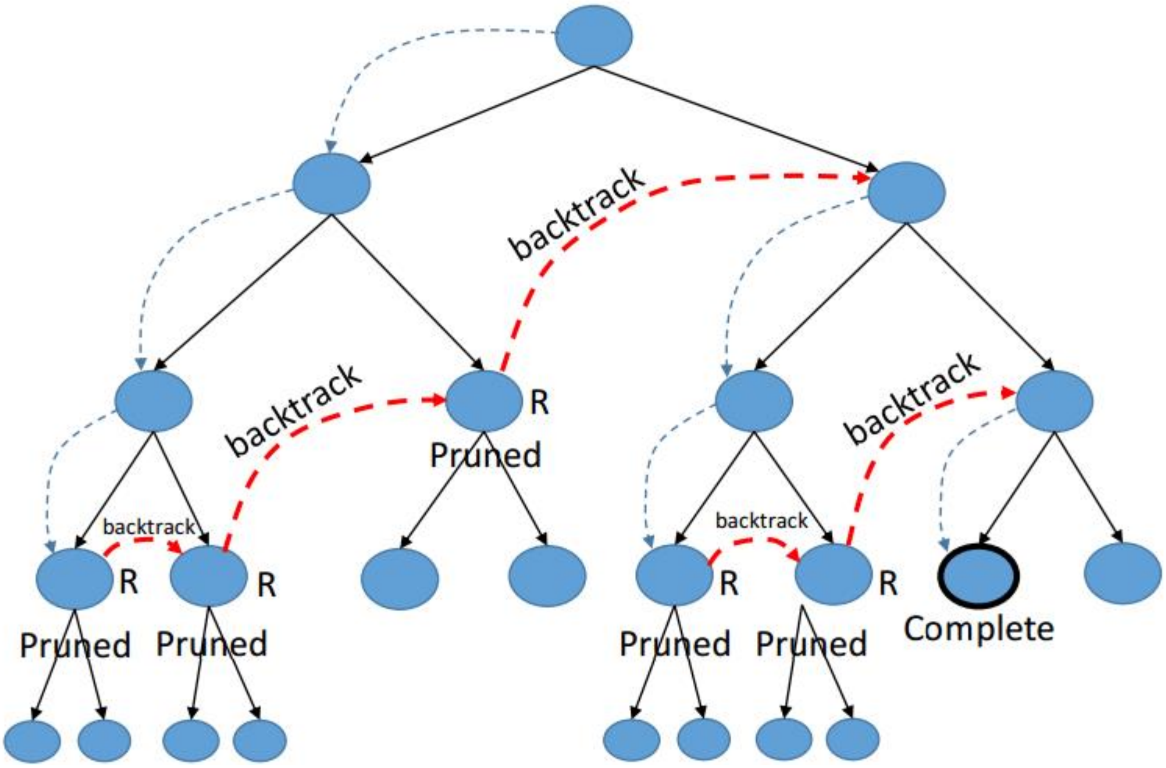
https://en.wikipedia.org/wiki/File:Greedy_algorithm_36_cents.svg

BACKTRACKING

- **Concepts of Backtracking**
- **Rat In a Maze**
- **N Queen Problem**

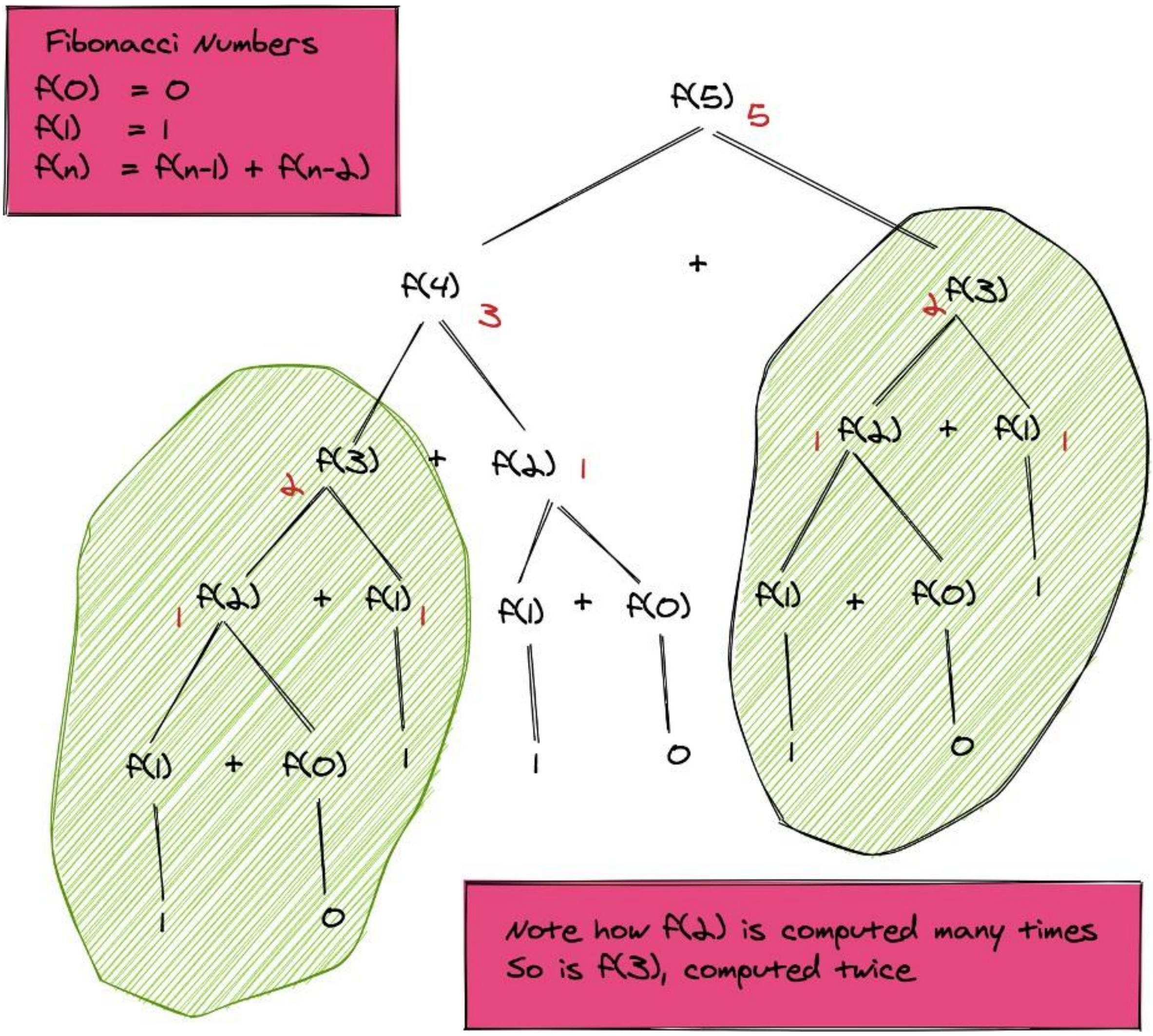


```
procedure EXPLORE(node n)
    if REJECT(n) then return
    if COMPLETE(n) then
        OUTPUT(n)
    for $n_i$ : CHILDREN(n) do EXPLORE($n_i$)
```



DYNAMIC PROGRAMMING

- **Introduction**
- **Dynamic Programming**
    - Memorization
    - Tabulation

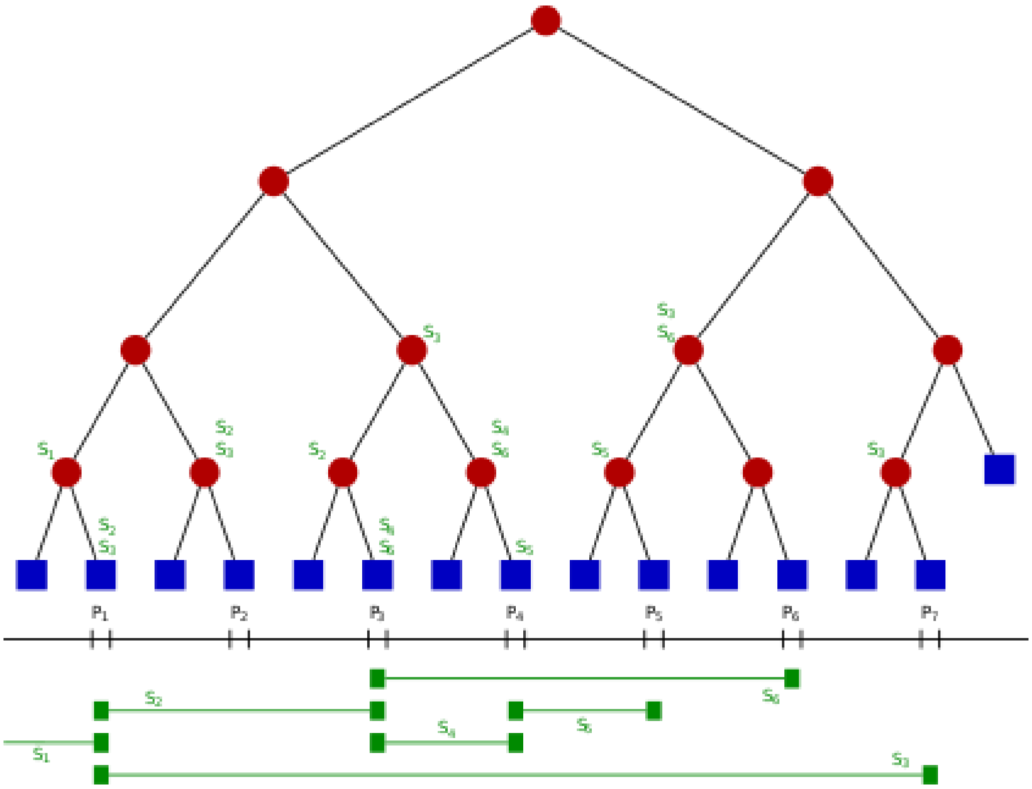


Fibonacci Numbers
f(0) = 0
f(1) = 1
f(n) = f(n-1) + f(n-2)

Note how f(2) is computed many times
So is f(3), computed twice

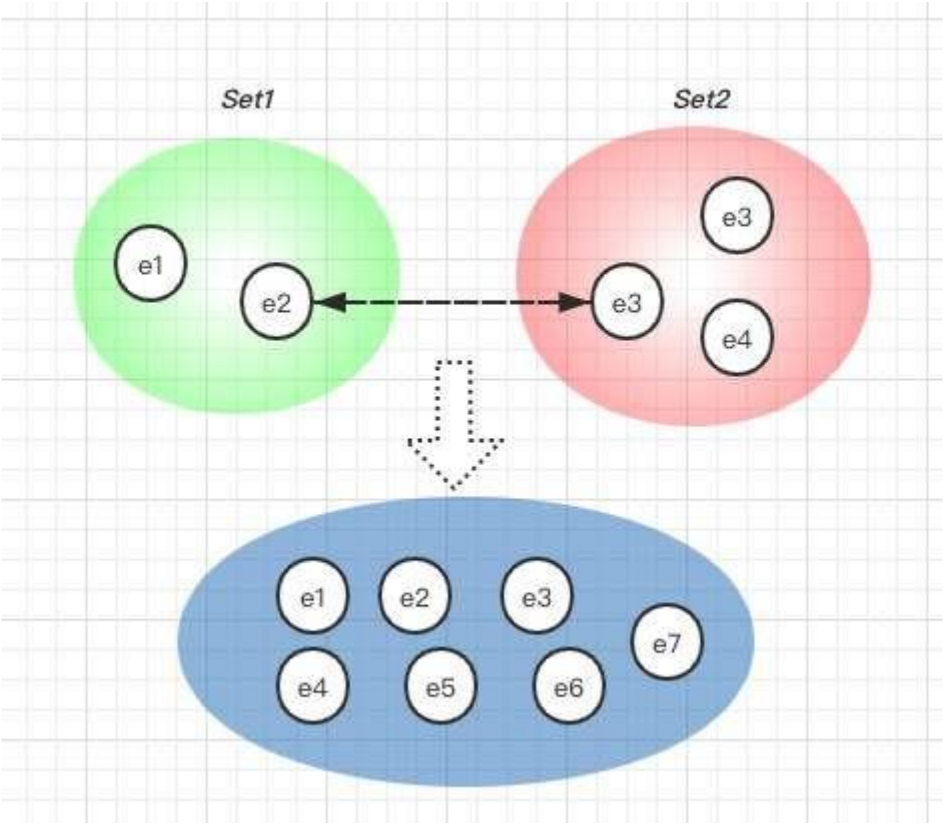
SEGMENT TREE

- **Introduction**
- **Construction**
- **Range Query**
- **Update Query**



DISJOINT SET

- **Introduction**
- **Find and Union Operations**
- **Union by Rank**
- **Path Compression**
- **Kruskal's Algorithm**



REASON FOR CHOOSING DSA

All of the above was part of my training during my summer break I specially choose the DSA by Geeks for Geeks for reasons stated below:

- I was interested in Problem Solving and Algorithms since my first semester.
- Data structure is a thing you need to know no matter in which language do you code.
- One needs to learn how to make algorithms of a real-life problem he/she is facing.
- It had video lectures on all the topics from which one can easily learn. I prefer learning from video rather than books and notes. I know books and notes and thesis have their own significance but still video lectures or face to face lectures make it easy to understand faster as we are involved Practically.
- It had 200+ algorithmic coding problems with video explained solutions.
- It had track based learning and weekly assessment to test my skills.
- It was a great opportunity for me to invest my time in learning instead of wasting it here and there during my summer break in this Covid-19 pandemic.
- It contained a lot of knowledge for such a reasonable price.
- The course was in two programing languages C++ and JAVA.
- This was a lifetime accessible course which I can use to learn even after my training whenever I want to revise.
- Along with all these reasons one of the reasons was the Geeks for Geeks platform which is offering the course because Geeks for Geeks is one of the best platform for Computer Science Students.

LEARNING OUTCOMES

A lot of beginners and experienced programmers avoid learning Data
Structures and Algorithms because it's complicated and they think that there is no use of all the above stuff in real life but there is a lot of implementation of DSA in daily life.

For example, if we must search our roll number in 2000 pages of Document how would we do that?

- If we try to search it randomly or in sequence it will take too much time.
- We can try another method in which we can directly go to page no. 1000and we can see if our roll no. is there or not, if not we can move ahead and by repeating this and eliminating, we can search our roll no. in no time.

And this is called Binary Search Algorithm.

Two reasons to Learn Data Structure and Algorithms -

- If you want to crack the interviews and get into the product-based companies
- If you love to solve real-world complex problems.

I have learnt a vast number of topics like Trees, Graphs, Linked Lists, Arrays, etc. I understood their basics, their working, their implementation, and their practical use in the problems we face while we solve a problem using coding.

When we work in the IT sector (Software or Programing part to be specific) we need to solve the problems and make programs write tons of code which will help us with the given problem and to write a program one needs to make different algorithms. Many algorithms combine to make a program. Now, algorithm is written in some languages, but they are not depended on ton them, one need to plan and algo first then write it into any language wither i tis C++ or JAVA or C or any other programing language. Algorithms are based on data structure and its implementation and working. So, basically one needs to have a good grip on DSA to work in the programming sector.

When you ask someone to decide for something the good one will be able to tell you "*I chose to do X because it's better than A, B in these ways. I could have gone with C, but*

I felt this was a better choice because of this ". In our daily life, we always go with that person who can complete the task in a short amount of time with efficiency and using fewer resources. The same thing happens with these companies. The problem faced by these companies is much harder and on a much larger scale. Software developers also must make the right decisions when it comes to solving the problems of these companies.
Knowledge of data structures like Hash Tables, Trees, Tries, Graphs, and
various algorithms goes a long way in solving these problems efficiently and the interviewers are more interested in seeing how candidates use these tools to solve a problem.

I learned about how to break a problem into pieces and then find the solution then how to make the desired algorithm which will help me to solve my respective problem.

What I Learned from the course precisely:

- I Learned Data Structures and Algorithms from basic to advanced level.
- Learned Topic-wise implementation of different Data Structures & Algorithms.
- Improved my problem-solving skills to become a stronger developer.
- Developed my analytical skills on Data Structures and use them efficiently.
- Solved problems asked in product-based companies' interviews.
- Solved problems in contests like coding round for SDE role.

This will help me during my career as a programmer and afterwards also whenever I need to code. We are surrounded by a lot of real-world complex problems for which no one has the solution. Observe the problems in depth and you can help this world by giving the solution which no one has given before.

"**Data **structure and algorithms help in understanding the nature of the problem at a deeper level and thereby a better understanding of the world. "

BIBLIOGRAPHY

- DSA Books
- Geeks for Geeks website
- Geeks for Geeks Course

Conclusion.

Algorithms are a vast topic, and it is all about making a program more efficient. These Algorithms arethe ones that make our experience smother with the software. Take google search engine for example how fast it provides the best search result in a few seconds of times. That is the power of algorithms. That is why many companies ask questions related to algorithms during interviews.

Through this Course I have Learnt a vast number of interesting topics like Trees, Graphs, LinkedList and many other more. Their implementation in practical problems and understanding their base concepts.

While working in the IT sector we need to solve the problems and make programs write tons of codewhich will help us with the given problem and to write a program one needs to make different algorithms. Many algorithms combine to make a program. Now, algorithms are written in some language, but they are not dependent on them, one need to make a plan and algo first then write it into any language whether its C++ or JAVA or C or any other programing language. The algorithmis based on data structure and its implementation and working. So, basically one needs to have a good grip on DSA to work in the programming sector.

PROJECT.

N-Queen Visualizer.

Overview.

The N-Queens problem is a classic computer science problem involving the placement of N queens on an N×N chessboard. The goal is to place the queens in such a way that no two queens can attack each other. This means that no two queens can share the same row, column, or diagonal.
This project provides a visualizer for the N-Queens problem, showcasing how the solution is found using a recursive backtracking algorithm.

Features

- **Visualization of the Algorithm: Watch how the recursive algorithm places the queens on the board.**
- **Support for Any N (≥ 4): The visualizer works for any value of N, though performance may vary for large N.**
- **Step-by-Step Execution: Pause and resume the execution to understand how the algorithm progresses.**

Requirements

- **A C++ compiler (e.g., g++, clang++)**
- **Basic knowledge of HTML, CSS, and JavaScript.**
- **A solid understanding of backtracking algorithms.**
- **Experience with event-driven programming for interactive features.**
- **Basic knowledge of data structures like arrays and matrices.**

PATH OF PROJECT

“This is the link to my project where I’ve used HTML, CSS, and JavaScript. It’s hosted on my GitHub ID, and you can look. The live working file is also available there. I hope it’s easy for you to understand, and there shouldn’t be any difficulties. The entire code is included.”

- ○ MY GITHUB ID:
- ○ https://github.com/juned-k786
- ○ PROJECT LIVE WORKING FILE:
- ○ https://nqueen.netlify.app/
- ○ SOURCE CODE OF PROJECT:
- ○ https://github.com/juned-k786/N-queen-project-DSA-Selfed-placed?tab=readme-ov-file