DSA Self-Paced

GeeksForGeeks

A training report

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

Bachelor of Technology

(Computer Science and Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 02nd June 2024 to 25th July 2024

SUBMITTED BY

Name of student: Biswajit Kumar Pradhan

Registration Number: 12208973

Student Declaration

To whom so ever it may concern

I, Biswajit Kumar Pradhan,12208973, hereby declare that the work done by me on "**DSA**" from **02nd June, 2024** to **25t\h July, 2024**, is a record of original work for the partial fulfilment of the requirements for the award of the degree, **Bachelor of Technology.**

Biswajit Kumar Pradhan

(12208973)

Dated: 29-08-2024

ACKNOWLEDGEMENT

First of all I would like to thank God that he made me capable enough to learn a new technology. Then I would like to express my sincere gratitude to the teacher and instructor of the course DSA Self-Paced from GeeksForGeeks who provide me the golden opportunity to learn a new technology from home.

I would also like to thank my own college Lovely Professional University for offering such a course which not only improve my programming skill but also taught me other new technology.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance for choosing this course.

Last but not least I would like to thank my all classmates who have helped me a lot.

Biswajit Kumar Pradhan

12208973

Date: Aug 29th, 2024

SUMMER TRAINING CERTIFICATE

https://www.geeksforgeeks.org/certificate/047692f7e4d42953e13e73829d20b003

TABLE OF CONTENT

INTRODUCTION OF THE PROJECT UNDERTAKEN

- The course name DSA stands for Data Structures and Algorithms and Self-paced means, one can join the course anytime. All of the content will be available once one gets enrolled. One can finish it at his own decided speed.

- **What is Data Structure?** Data Structure is a way of collecting and organizing datain such a way that we can perform operations on these data in an effective way. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have some data which has, player's **name** "Virat" and **age** 26. Here "Virat" is of **String** data type and 26 is of **integer** data type.

- **What is Algorithm?** An algorithm is a finite set of instructions or logic, written in order, to accomplish a certain predefined task. Algorithm is not the complete codeor program, it is just the core logic(solution) of a problem, which can be expressed either as an informal high-level description as **pseudocode** or using a **flowchart.**

- This course is a complete package that helped me learn Data Structures and Algorithms from basic to an advanced level. The course curriculum has been divided into 8 weeks where one can practice questions & attempt the assessment tests according to his own pace. The course offers me a wealth of programming challenges that will help me to prepare for interviews with top-notch companies like Microsoft, Amazon, Adobe etc.

TECHNOLOGY LEARNT

- Learn Data Structures and Algorithms from basic to an advanced level like:
- Learn Topic-wise implementation of different Data Structures & Algorithms as follows

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

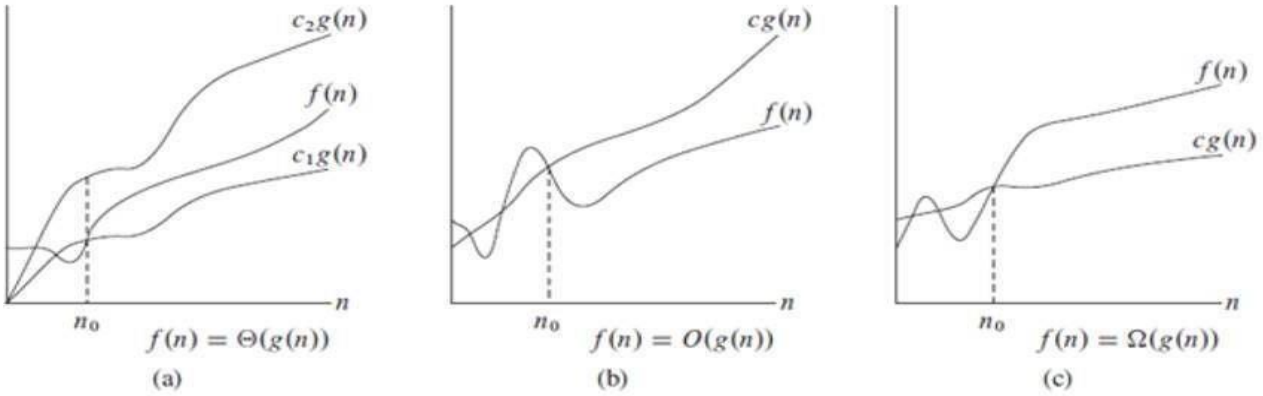
Omega Notation

- Graphical and mathematical explanation.
- Calculation.

Theta Notation

- Graphical and mathematical explanation.

- Calculation.



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

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**

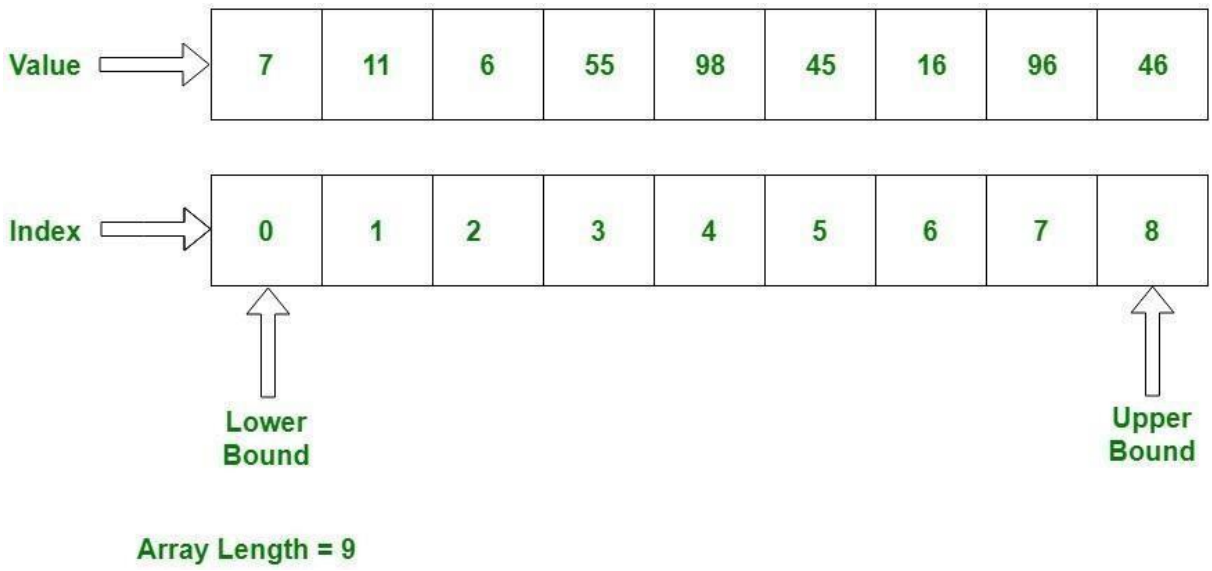
  - Method 1: Using the left Shift.

- 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 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

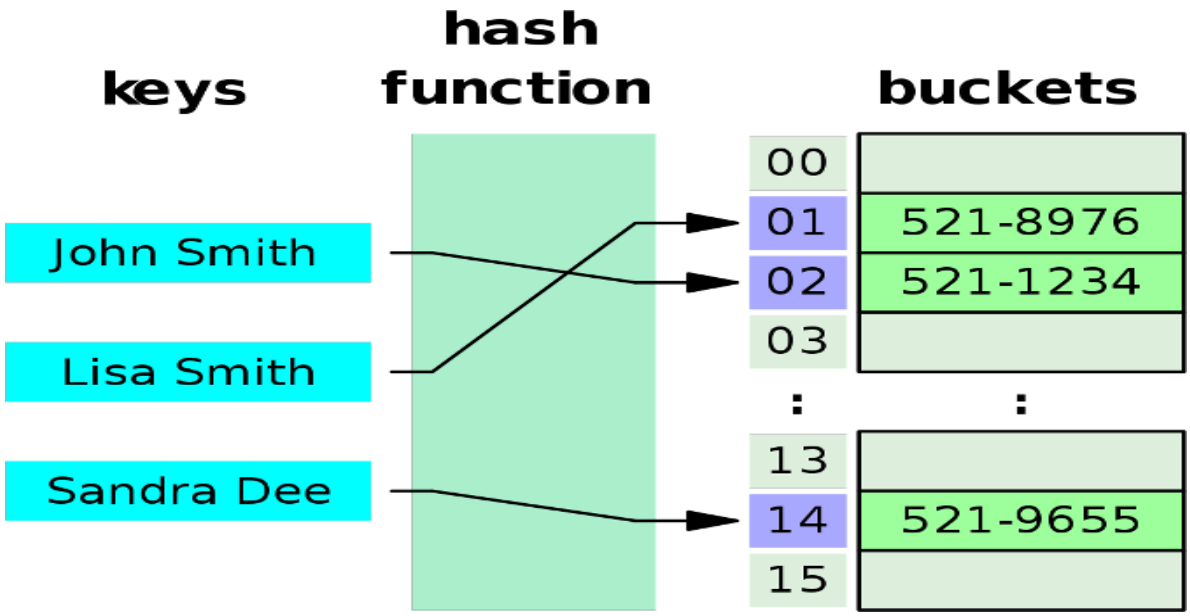
- Rotating a Matrix
- Check if the element is present in a row and column-wise sorted matrix.
- Boundary Traversal
- Spiral Traversal
- Matrix Multiplication
- Search in row-wise and column-wise Sorted Matrix

HASHING

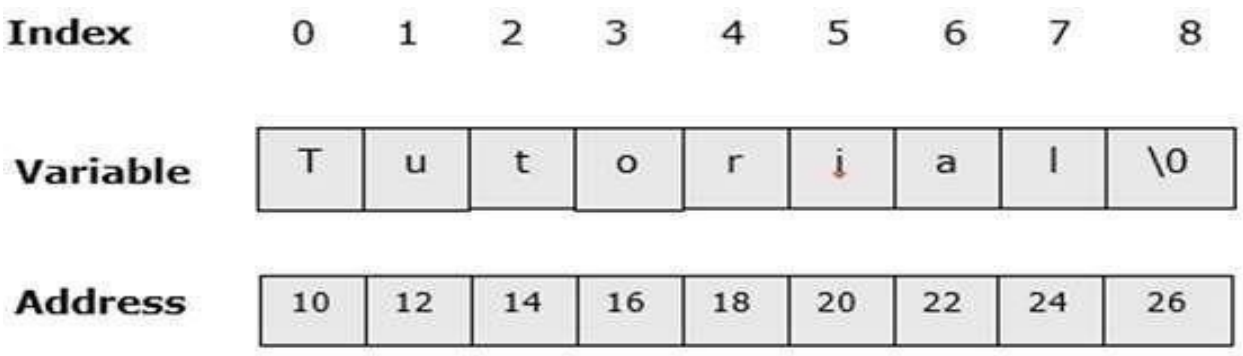
- Introduction and Time complexity analysis
- Application of Hashing
- Discussion on Direct Address Table
- Working and examples on various Hash Functions
- Introduction and Various techniques on Collision Handling
- Chaining and its implementation
- Open Addressing and its Implementation
- Chaining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

- Unordered Map

- Java
  - HashSet
  - HashMap

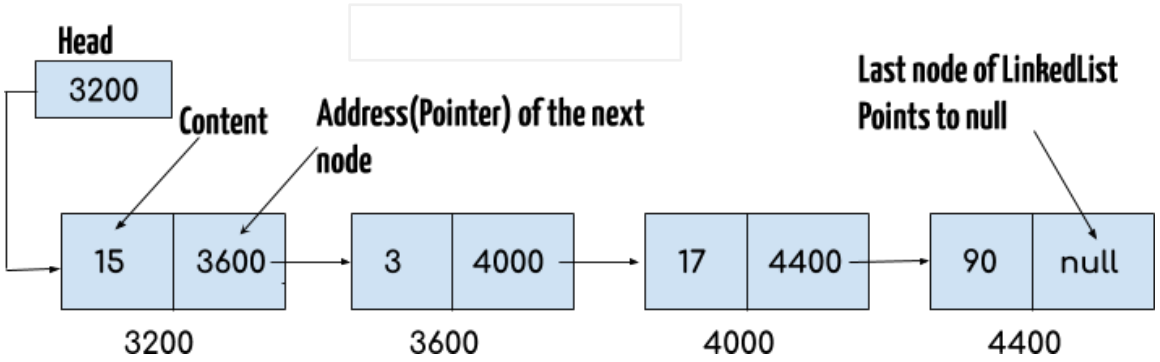


STRINGS



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

LINKED LIST



- Introduction
- Implementation in JAVA And CPP

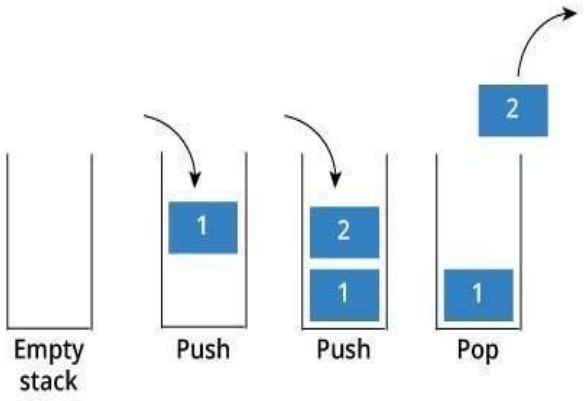
- **Doubly Linked List**

- **Circular Linked List**

- **Loop Problems**

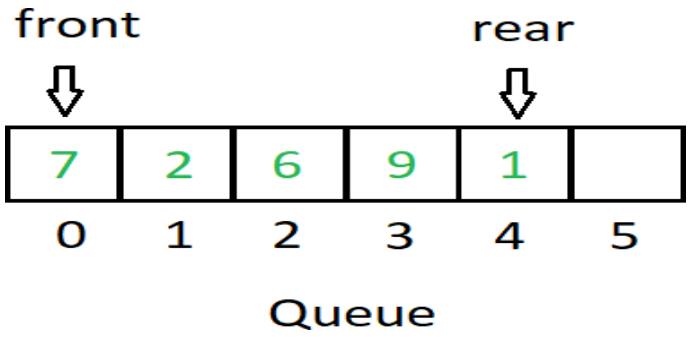
  - Detecting Loops
  - Detecting loops using Floyd cycle detection
  - Detecting and Removing Loops in Linked List

STACK



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

QUEUE



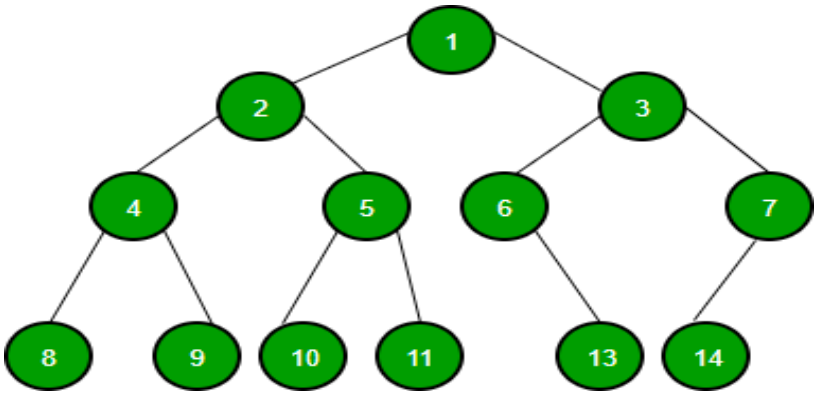
- Introduction and Application
- Implementation of the queue using array and LinkedList

- In C++ STL
- In Java
- Stack using queue

DEQUE

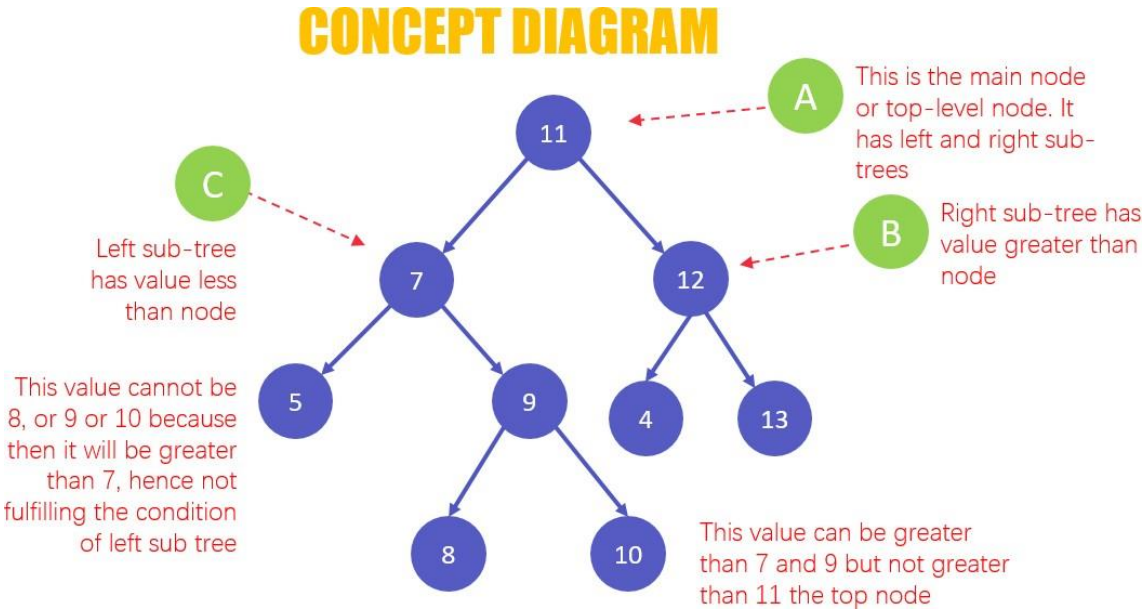
- Introduction and Application
- Implementation
  - In C++ STL
  - In Java
- **Problems (With Video Solutions)**
  - Maximums of all subarrays of size k
  - Array Deque in Java
  - Design a DS with min max operations

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

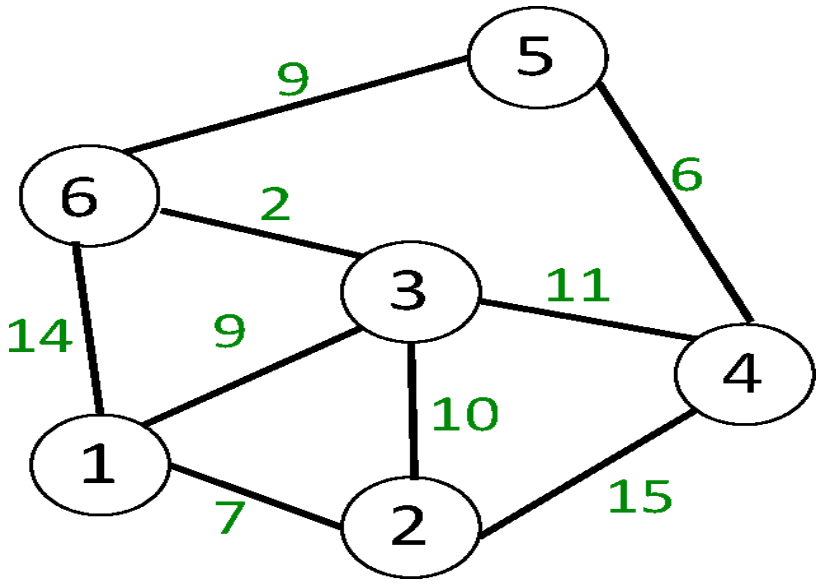
BINARY SEARCH TREE



- Background, Introduction and Application
- Implementation of Search in BST
- Insertion in BST
- Deletion in BST
- Floor in BST
- Self Balancing BST
- AVL Tree

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

BACKTRACKING

- Concepts of Backtracking
- Rat In a Maze

- N Queen Problem **DYNAMIC**

PROGRAMMING

- Introduction
- Dynamic Programming
  - Memorization
  - Tabulation33

TREE

- Introduction
  - Representation
  - Search
  - Insert
  - Delete
- **Count Distinct Rows in a Binary Matrix**

SEGMENT TREE

- Introduction
- Construction
- Range Query
- Update Query

DISJOINT SET

- Introduction

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

- Improved my problem-solving skills by practicing problems to become a stronger developer
  - **Practice problems**
    - This track contains many practice problems for the users which are considered important and must-do as far as Data Structure and Algorithm is concerned
- Developed my analytical skills on Data Structures to use them efficiently
- Solved problems asked in product-based companies' interviews
- Solved problems in contests similar to coding round for SDE role

REASON FOR CHOOSING THIS TECHNOLOGY

- With advancement and innovation in technology, programming is becoming a highly in-demand skill for Software Developers. Everything you see aroundyourself from Smart TVs, ACs, Lights, Traffic Signals uses some kind of programming for executing user commands.

> *<In order to be irreplaceable, one must always be efficient.=*

Data Structures and **Algorithms** are the identity of a good Software Developer. The interviews for technical roles in some of the tech giants like *Google, Facebook, Amazon, Flipkart* is more focused on measuring the knowledge of Data Structures and Algorithms of the candidates. The main reason behind this is Data Structures and Algorithms improves the problem-solving ability of a candidate to a great extent.

- This course has video lectures of 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 lecture or face to face lectures make it easy to understand faster as we are involved Practically.
- It has 200+ algorithmic coding problems with video explained solutions.
- It has 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.
- This was a lifetime accessible course which I can use to learn even after my training whenever I want to revise.

LEARNING OUTCOMES

Programming is all about data structures and algorithms. Data structures are used to hold data while algorithms are used to solve the problem using that data.

Data structures and algorithms (DSA) goes through solutions to standard problems in detail and gives you an insight into how efficient it is to use each one of them. It also teaches you the science of evaluating the efficiency of an algorithm. This enables you tochoose the best of various choices.

For example, you want to search your roll number in 30000 pages of documents, for thatyou have choices like Linear search, Binary search, etc. So, the more efficient way will be Binary search for searching something in a huge number of data.

So, if you know the DSA, you can solve any problem efficiently. The main use of DSA is to make your code scalable because

- Time is precious
- Memory is expensive

One of the use DSA is to crack the interviews to get into the product-based companies

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 things happen with these companies. The problem faced by these companies is much harder and at a much larger scale. Software developers also have to make the right decisions when it comes to solving the problems of these companies.

For example, in an interview, you're given a problem to find the sum of first N natural numbers.

One candidate solves it by using loop like

Initialize sum = 0

for every natural number n in range 1 to N (inclusive):

add n to sum

sum is the answer

And you solve it using the sum of first N natural numbers is given by the formula:

Sum=N*(N+1)/2

Obviously, they will choose you over other one because your solution is more efficient.

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. Just like a car mechanic needs the right tool to fix a car and make it run properly, a programmer needs the right tool (algorithm and data structure) to make the software run properly. So, the interviewer wants to find a candidate who can apply the right set of tools to solve the given problem. If you know the characteristics of one data structure in contrast to another you will be able to make the right decision in choosing the right data structure to solve a problem.

Another use of DSA, if you love to solve the real-world complex problems.

Let's take the example of Library. If you need to find a book on Set Theory from a library, you will go to the math section first, then the Set Theory section. If these books are not organized in this manner and just distributed randomly then it will be frustrating to find a specific book. So, data structures refer to the way we organize information on our computer. Computer scientists process and look for the best way we can organize the data we have, so it can be better processed based on input provided.

A lot of newbie programmers have this question that where we use all the stuff of data structure and algorithm in our daily life and how it's useful in solving the real-world complex problem. We need to mention that whether you are interested in getting into thetop tech giant companies or not DSA still helps a lot in your day to day life.

Let's consider some examples

- In Facebook you can represent your friends on Facebook, friends of friends, mutual friends easily by Graph.

- If you need to keep a deck of cards and arrange it properly how would you do that? You will throw it randomly or you will arrange the cards one over another and from a proper deck. You can use Stack here to make a proper arrangement of cards one over another.

- If you need to search a word in the dictionary, what would be your approach? Do you go page by page or you open some page and if the word is not found you open a page prior/later to one opened depending upon the order of word to the current page (Binary Search).

The first two were a good example of choosing the right data structure for a real-world problem and the third one is a good example of choosing the right algorithm to solve a specific problem in less amount of time.

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

PROJECT CODE

Sudoku Solver:

```
#include <iostream>
#define N 9
using namespace std;
int grid[N][N] = {
  {3, 0, 6, 5, 0, 8, 4, 0, 0},
  {5, 2, 0, 0, 0, 0, 0, 0, 0},
  {0, 8, 7, 0, 0, 0, 0, 3, 1},
  {0, 0, 3, 0, 1, 0, 0, 8, 0},
  {9, 0, 0, 8, 6, 3, 0, 0, 5},
  {0, 5, 0, 0, 9, 0, 6, 0, 0},
  {1, 3, 0, 0, 0, 0, 2, 5, 0},
  {0, 0, 0, 0, 0, 0, 0, 7, 4},
  {0, 0, 5, 2, 0, 6, 3, 0, 0}
};
bool isPresentInCol(int col, int num){ //check whether num is present in col or not
  for (int row = 0; row < N; row++)
    if (grid[row][col] == num)
      return true;
  return false;
}
bool isPresentInRow(int row, int num){ //check whether num is present in row or not
  for (int col = 0; col < N; col++)
    if (grid[row][col] == num)
      return true;
  return false;
}
bool isPresentInBox(int boxStartRow, int boxStartCol, int num){
//check whether num is present in 3x3 box or not
  for (int row = 0; row < 3; row++)
    for (int col = 0; col < 3; col++)
      if (grid[row+boxStartRow][col+boxStartCol] == num)
        return true;
  return false;
```

```
}
void sudokuGrid(){ //print the sudoku grid after solve
   for (int row = 0; row < N; row++){
      for (int col = 0; col < N; col++){
         if(col == 3 || col == 6)
            cout << " | ";
         cout << grid[row][col] <<" ";
      }
      if(row == 2 || row == 5){
         cout << endl;
         for(int i = 0; i<N; i++)
            cout << "---";
      }

      cout << endl;
   }
}
bool findEmptyPlace(int &row, int &col){ //get empty location and update row and column
   for (row = 0; row < N; row++)
      for (col = 0; col < N; col++)
         if (grid[row][col] == 0) //marked with 0 is empty
            return true;
   return false;
}
bool isValidPlace(int row, int col, int num){
   //when item not found in col, row and current 3x3 box
   return !isPresentInRow(row, num) && !isPresentInCol(col, num) && !isPresentInBox(row - row%3 ,
col - col%3, num);
}
bool solveSudoku(){
   int row, col;
   if (!findEmptyPlace(row, col))
      return true; //when all places are filled
   for (int num = 1; num <= 9; num++){ //valid numbers are 1 - 9
      if (isValidPlace(row, col, num)){ //check validation, if yes, put the number in the grid
         grid[row][col] = num;
```

```
      if (solveSudoku()) //recursively go for other rooms in the grid
        return true;
      grid[row][col] = 0; //turn to unassigned space when conditions are not satisfied
    }
  }
  return false;
}
int main(){
  if (solveSudoku() == true)
    sudokuGrid();
  else
    cout << "No solution exists";
}
```

OUTPUT OF SUDOKU SOLVER

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   SQL CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS D:\Satyam B.tech\C++ coding> cd "d:\Satyam B.tech\C++ coding\" ; if ($?) { g++ Sudoku.cpp -o Sudoku } ; if ($?) { .\Sudoku }
3 1 6  | 5 7 8  | 4 9 2
5 2 9  | 1 3 4  | 7 6 8
4 8 7  | 6 2 9  | 5 3 1
-------------------------
2 6 3  | 4 1 5  | 9 8 7
9 7 4  | 8 6 3  | 1 2 5
8 5 1  | 7 9 2  | 6 4 3
-------------------------
1 3 8  | 9 4 7  | 2 5 6
6 9 2  | 3 5 1  | 8 7 4
7 4 5  | 2 8 6  | 3 1 9
PS D:\Satyam B.tech\C++ coding>
```

CONCLUSION

The Data Structures and Algorithms (DSA) self-paced course offered by GeeksforGeeks is a comprehensive and valuable resource for anyone seeking to enhance their programming and problem-solving skills. The course equips learners with a deep understanding of fundamental data structures and algorithms, enabling them to tackle a wide range of coding challenges. Throughout the course, participants are guided through a well-structured curriculum that covers essential topics such as arrays, linked lists, trees, graphs, sorting, searching, and more. The interactive nature of the course allows individuals to learn at their own pace, makingit suitable for beginners as well as intermediate programmers aiming to solidify their knowledge. By combining theory with practical implementation through coding exercises and quizzes, the course fosters a holistic learning experience. Learners not only grasp theoretical concepts but also gain the confidence to applythem in real-world scenarios. GeeksforGeeks' reputation for providing clear explanations and code examples further enhances the learning process, making complex concepts accessible to a wide audience. Moreover, theself-paced nature of the course accommodates different learning styles and schedules, allowing participants to balance their studies with other commitments. The availability of resources beyond the course, such as articles, practice problems, and an active community forum, ensures that learners can continue their growth even after completing the curriculum. In summary, the DSA self-paced course offered by GeeksforGeeks delivers a comprehensive, flexible, and interactive learning experience. Through its well-structured modules,hands-on exercises, and supportive community, the course empowers individuals to build a strong foundationin data structures and algorithms, thereby enhancing their programming skills and career prospects.

REFERENCES

- GeeksForGeeks website
  https://practice.geeksforgeeks.org/myCourses
- GeeksForGeeks DSA Self-Paced course
  https://practice.geeksforgeeks.org/batch/dsa-4?tab=Chapters
- Google.com
- w3schools.com
  https://www.w3schools.com
- Youtube.com