

**SIX WEEKS SUMMER TRAINING REPORT**

**on**

**DATA STRUCTURE AND ALGORITHM ( SELF PACED )**

Submitted by

**Aditya Kumar**

**Registration No : 11902266**

**Programme Name : Btech. CSE**

Under the Guidance of

Mr. Sandeep Jain

**School of Computer Science & Engineering**

**Lovely Professional University, Phagwara**

(June-July,2021)

# DECLARATION

I hereby declare that I have completed my six weeks of summer training at Geeks for Geeks platform from June 1,2021 to July 15,2021 under the guidance of MR. Sandeep Jain. I have declared that I have worked with full dedication during these 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.

Date – 25 Sep. 2021 Name of Student – Aditya Kumar

Registration no: 11902266

# ACKNOWLEDGEMENT

I would like to express my gratitude towards my University as well as Geeks for Geeks for providing me 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 thankful 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 really thankful to have such good support from them as they always have my back whenever I need.

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 became the person who I am now.

I have taken 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



# Table of content

[**DECLARATION**](#_heading=h.kp54agwvnfjn)2

[**ACKNOWLEDGEMENT**](#_heading=h.iuh0ct793czj)3

[**Summer Training Certificate By Geeks for Geeks**](#_heading=h.9hbxdp146r2)4

[**Table of content**](#_heading=h.8s4amx6yac1n)5

[**INTRODUCTION**](#_heading=h.r5vm2d601cpm)10

[**TECHNOLOGY LEARNT**](#_heading=h.961kbogwijrj)11

[INTRODUCTION TO DSA](#_heading=h.e8z8p5zgrlz8) 11

[Analysis of Algorithm](#_heading=h.6bsuwwd8d45c) 11

[Order of Growth](#_heading=h.v3ksyyzcv6gw) 11

[Asymptotic Notations](#_heading=h.8zwn9nxsp4sv) 11

[Omega Notation](#_heading=h.9vdpn7vj3rqq) 11

[Theta Notation](#_heading=h.ojobnuyz3l0o) 11

[Analysis of common loops](#_heading=h.9ltkpbfbn3y2) 12

[Analysis of Recursion](#_heading=h.bx13g84khtl4) 12

[Space Complexity](#_heading=h.fhv2rfwo8dc2) 12

[**MATHEMATICS**](#_heading=h.uboktxggne62)13

[Finding the number of digits in a number.](#_heading=h.p8lchjbtyj1b) 13

[Arithmetic and Geometric Progressions.](#_heading=h.czwa868o5gxr) 13

[Quadratic Equations.](#_heading=h.5zkdn6edase) 13

[Mean and Median.](#_heading=h.e93i1hia41t9) 13

[Prime Numbers.](#_heading=h.cv9qznhyp4es) 13

[LCM and HCF](#_heading=h.fugxdhk9bqgb) 13

[Factorials](#_heading=h.cdmz8j8l037h) 13

[Permutations and Combinations](#_heading=h.rkbf94a5jo8p) 13

[Modular Arithmetic](#_heading=h.pv850v4ko07z) 13

[**BITMAGIC**](#_heading=h.i2o6xd3ko5lx)14

[Bitwise Operators in C++](#_heading=h.dz2ln41uv77) 14

[Bitwise Operators in Java](#_heading=h.kbgsvkzhgme1) 14

[Problem(With Video Solutions): Check Kth bit is set or not](#_heading=h.kqedmnxohlbr) 14

[**RECURSION**](#_heading=h.6wpk65aee20r)15

[Introduction to Recursion](#_heading=h.95fexhvoxl1u) 15

[Applications of Recursion](#_heading=h.mzoh96f1kn0f) 15

[Writing base cases in Recursion](#_heading=h.hyrrbmfn0fkf) 15

[Example for Recursion](#_heading=h.70fn9xsljke1) 15

[Output](#_heading=h.hq28kdzhn4ey) 16

[**ARRAYS**](#_heading=h.sexqp15hwqtn)17

[Introduction and Advantages](#_heading=h.cqcuz2uomk2v) 17

[Types of Arrays](#_heading=h.po1hf63lbu5h) 17

[Operations on Arrays](#_heading=h.3gl5vs8r3o7n) 17

[SEARCHING](#_heading=h.sljxhw1845ju) 18

[SORTING](#_heading=h.k7q7pzyrm1td) 18

[**MATRIX**](#_heading=h.qp43g8a2afhi)19

[Introduction to Matrix in C++ and Java](#_heading=h.5qzqoiryiek8) 19

[Multidimensional Matrix](#_heading=h.9a3z7c6139no) 19

[Pass Matrix as Argument](#_heading=h.xen6wzyd338o) 19

[Printing matrix in a snake pattern](#_heading=h.52tm61j0ra73) 19

[Transposing a matrix](#_heading=h.33so9ts4btjk) 19

[Rotating a Matrix](#_heading=h.n0ppurtp4wcc) 19

[Check if the element is present in a row and column-wise sorted matrix.](#_heading=h.knw8u5t9jxbm) 19

[Boundary Traversal](#_heading=h.m9b8g8g9lmv4) 19

[Spiral Traversal](#_heading=h.b3wbtri6vxym) 19

[Matrix Multiplication](#_heading=h.np87c37oe23a) 19

[Search in row-wise and column-wise Sorted Matrix](#_heading=h.adt6z1rwa7y5) 19

[**HASHING**](#_heading=h.thlxr7m23czv)20

[Introduction and Time complexity analysis](#_heading=h.xdgxtwqj4dmc) 20

[Application of Hashing](#_heading=h.qhu2ej1hj7f2) 20

[Discussion on Direct Address Table](#_heading=h.wi2bycl1zigv) 20

[Working and examples on various Hash Functions](#_heading=h.2mk4s5pknxx0) 20

[Introduction and Various techniques on Collision Handling](#_heading=h.d70zancum31u) 20

[Chaining and its implementation](#_heading=h.hseuahv4d7xy) 20

[Open Addressing and its Implementation](#_heading=h.bwnlv3rshwzb) 20

[Chaining V/S Open Addressing](#_heading=h.qfosqhy12x1r) 20

[Double Hashing](#_heading=h.xv4gkllx60nn) 20

[C++](#_heading=h.wlka1m4jxf6h) 20

[Java](#_heading=h.jdqx0zsz0rep) 21

[**STRINGS**](#_heading=h.6uy8ibtp2f1k)22

[Discussion of String DS](#_heading=h.vqh3f4f01rlj) 22

[Strings in CPP](#_heading=h.vizokxcd6v84) 22

[Strings in Java](#_heading=h.3bqxvbnv1nz5) 22

[Rabin Karp Algorithm](#_heading=h.4empz251h2vo) 22

[KMP Algorithm](#_heading=h.7fuiqcxmic4) 22

[**LINKED LIST**](#_heading=h.vh06vwi0str5)23

[Introduction](#_heading=h.ctv65i7r0dkx) 23

[Doubly Linked List](#_heading=h.93xbczwq8ek3) 23

[Circular Linked List](#_heading=h.y5gw06wfrzrp) 23

[Loop Problems](#_heading=h.9qhqmbw98tph) 23

[**STACK**](#_heading=h.odi2isnnz3tw)24

[Understanding the Stack data structure](#_heading=h.pdgi15rrawdd) 24

[Applications of Stack](#_heading=h.idpf34o3sb20) 24

[Implementation of Stack in Array and Linked List](#_heading=h.xbs7ers9iwss) 24

[**QUEUE**](#_heading=h.u0jc9ebivm10)25

[Introduction and Application](#_heading=h.4t8897575uun) 25

[Implementation of the queue using array and LinkedList](#_heading=h.8eot41odaf7o) 25

[**DEQUE**](#_heading=h.45mo8x7u0fui)26

[Introduction and Application](#_heading=h.d2t534ei0n3m) 26

[Implementation](#_heading=h.prwt05cto6s7) 26

[Problems(With Video Solutions)](#_heading=h.92mvq6p75198) 26

[**TREE**](#_heading=h.61b49rp0j86q)27

[Introduction](#_heading=h.w13gxjgq9lz1) 27

[Implementation of:](#_heading=h.1x2f20t8ivaz) 27

[BINARY SEARCH TREE](#_heading=h.dl3rwzvr83vk) 28

[Background, Introduction and Application](#_heading=h.delr0tody464) 28

[Implementation of Search in BST](#_heading=h.xgu92z9q7pcq) 28

[Insertion in BST](#_heading=h.dy7d93e6pg3) 28

[Deletion in BST](#_heading=h.i5dsg11flt35) 28

[Floor in BST](#_heading=h.y7xwitsxe8l) 28

[Self Balancing BST](#_heading=h.f9sxvlolwwl) 29

[AVL Tree](#_heading=h.t1p1xwnohiqe) 29

[Red Black Tree](#_heading=h.ap0r58w08wdv) 29

[Set in C++ STL](#_heading=h.fzylc2vbnxxi) 29

[Map in C++ STL](#_heading=h.kn03fgvsj1wi) 29

[**HEAP**](#_heading=h.46mt2illtgvp)30

[Introduction & Implementation](#_heading=h.qlcvm5ouo0jr) 30

[Binary Heap](#_heading=h.ket1olpdaaba) 30

[Heap Sort](#_heading=h.j0zmqy4lurjv) 30

[Priority Queue in C++](#_heading=h.rbbhh2o5jgaq) 30

[PriorityQueue in Java](#_heading=h.se7ev0ybd318) 30

[**GRAPH**](#_heading=h.chcnid9y8iu0)30

[Introduction to Graph](#_heading=h.ijhkfezba06) 31

[Graph Representation](#_heading=h.rvl6uy7kyeak) 31

[Breadth-First Search](#_heading=h.uji8n3uxlcsn) 31

[Depth First Search](#_heading=h.1ajjd2a7tpbn) 31

[Shortest Path in Directed Acyclic Graph](#_heading=h.rkmv0m2w2ici) 31

[Prim's Algorithm/Minimum Spanning Tree](#_heading=h.u3t3yp3a9miz) 31

[Dijkstra's Shortest Path Algorithm](#_heading=h.1lpe4gy1qrh2) 31

[Bellman-Ford Shortest Path Algorithm](#_heading=h.8yqepz5my4h1) 31

[Kosaraju's Algorithm](#_heading=h.eefoetm6sn1h) 31

[Articulation Point](#_heading=h.6dgbkfvc6srq) 31

[Bridges in Graph](#_heading=h.5abflce438pj) 31

[Tarjan’s Algorithm](#_heading=h.kkalcdipr9j5) 31

[**GREEDY**](#_heading=h.ax0vvhpsqnv)32

[Introduction](#_heading=h.dkh2tetkdtp5) 32

[Activity Selection Problem](#_heading=h.lfkqd0eop40t) 32

[Fractional Knapsack](#_heading=h.itwx7yrmet1x) 32

[Job Sequencing Problem](#_heading=h.fucr54mzs3ho) 32

[**BACKTRACKING**](#_heading=h.367zbusa56s7)32

[Concepts of Backtracking](#_heading=h.2vcg9fw004fy) 32

[Rat In a Maze](#_heading=h.42hn05ktadxo) 32

[N Queen Problem](#_heading=h.mujwqdqls9nw) 32

[**DYNAMIC PROGRAMMING**](#_heading=h.v41n7j5buc4l)32

[Introduction](#_heading=h.emwf02jx1g2c) 32

[Dynamic Programming](#_heading=h.9zy038t54vkk) 32

[Memoization](#_heading=h.md9951fw6q5l) 32

[Tabulation](#_heading=h.lwz578v5ov8v) 32

[**TREE**](#_heading=h.vlaqrnx7befq)33

[Introduction](#_heading=h.5p6stejv8j22) 33

[Representation](#_heading=h.qpsexww7f26w) 33

[Search](#_heading=h.idxreezcayky) 33

[Insert](#_heading=h.wiqa4qfl36g5) 33

[Delete](#_heading=h.a7bvyhlure79) 33

[Count Distinct Rows in a Binary Matrix](#_heading=h.1rurka6r2iv1) 33

[SEGMENT TREE](#_heading=h.35puanbspwf3) 33

[Introduction](#_heading=h.qlrzxwb3lni9) 33

[Construction](#_heading=h.3v3dm6yoyxj1) 33

[Range Query](#_heading=h.z53braf9x0px) 33

[Update Query](#_heading=h.33k5wf47iyui) 33

[DISJOINT SET](#_heading=h.r01d5jefpd82) 34

[Introduction](#_heading=h.gclqs0jpv64t) 34

[Find and Union Operations](#_heading=h.70fxcjz0y41h) 34

[Union by Rank](#_heading=h.3lw232t425xe) 34

[Path Compression](#_heading=h.ab95nfmu4osp) 34

[Kruskal's Algorithm](#_heading=h.bx980x4wk604) 34

[**Mini project outputs**](#_heading=h.yzf0qsrpojhz)35

[**REASON FOR CHOOSING DSA**](#_heading=h.oszxaempyyki)38

[**LEARNING OUTCOMES**](#_heading=h.88vsv9mr108u)39

[What I Learned from the course precisely :](#_heading=h.abxn7xs6deu3) 40

[**BIBLIOGRAPHY**](#_heading=h.gjdgxs)42

# INTRODUCTION

The 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, which 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 bound.

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

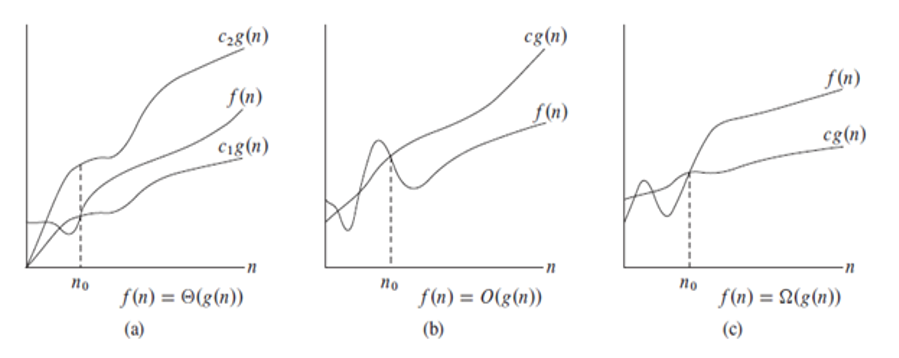
* + Best, Average and Worst case explanation through a program.
* **Big O Notation**
  + Graphical and mathematical explanation.
  + Calculation
  + Applications at Linear Search

### Omega Notation

* + Graphical and mathematical explanation.
  + Calculation.

### Theta Notation

* + Graphical and mathematical explanation.
  + Calculation.



### 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

## Example for Recursion

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int factorial(int);**

**int fact,value;**

**cout<<"Enter any number: ";**

**cin>>value;**

**fact=factorial(value);**

**cout<<"Factorial of a number is: "<<fact<<endl;**

**return 0;**

**}**

**int factorial(int n)**

**{**

**if(n<0)**

**return(-1); /\*Wrong value\*/**

**if(n==0)**

**return(1); /\*Terminating condition\*/**

**else**

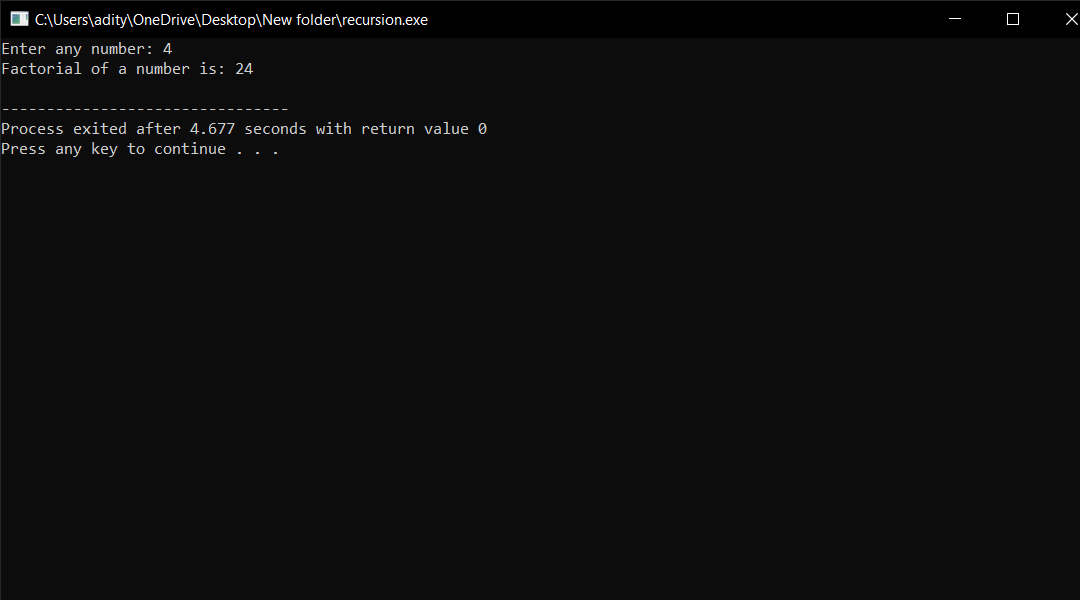
**{**

**return(n\*factorial(n-1));**

**}**

**}**

### Output

****

# ARRAYS

Table

Description automatically generated

## 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

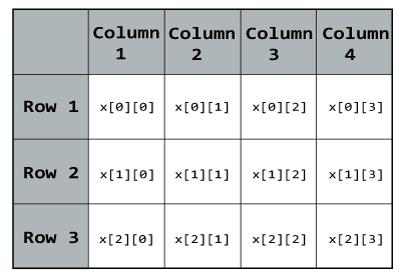
## SEARCHING

* **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



## 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

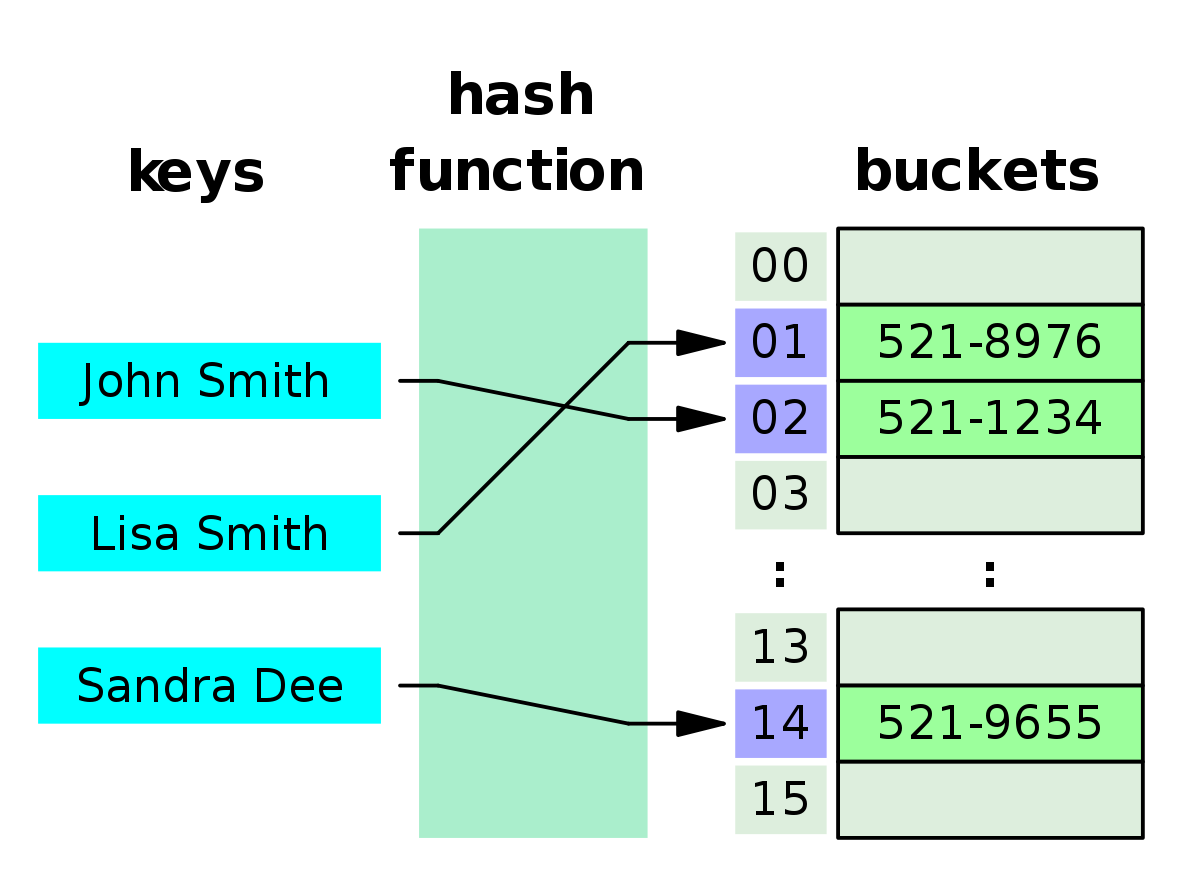
## Spiral Traversal

## 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

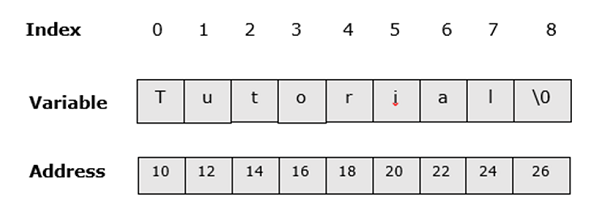
## C++

* + Unordered Set
  + Unordered Map

## Java

* + HashSet
  + HashMap

# STRINGS



## Discussion of String DS

## Strings in CPP

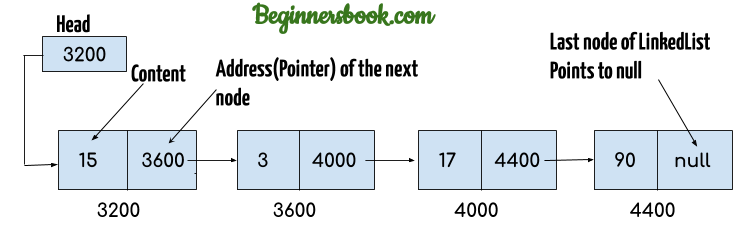
## Strings in Java

## Rabin Karp Algorithm

## KMP Algorithm

# 

# LINKED LIST



## Introduction

* + Implementation in CPP
  + Implementation in Java
  + Comparison with Array DS

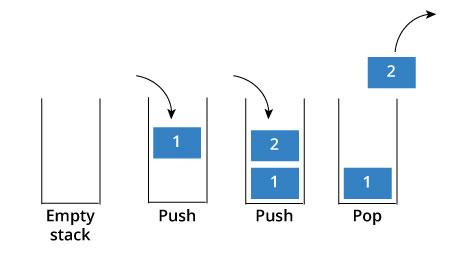
## Doubly Linked List

## Circular Linked List

## Loop Problems

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

# STACK



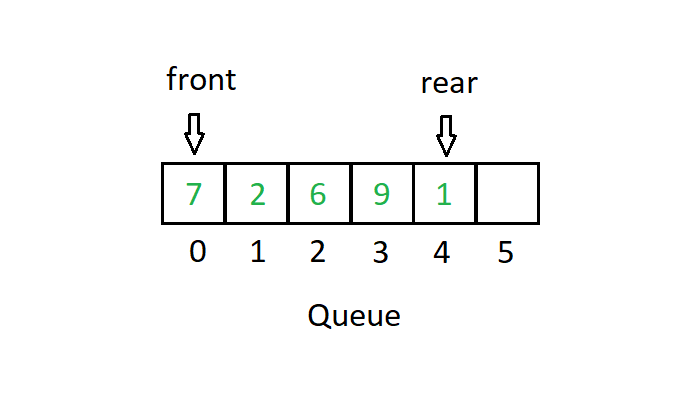
## Understanding the Stack data structure

## 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

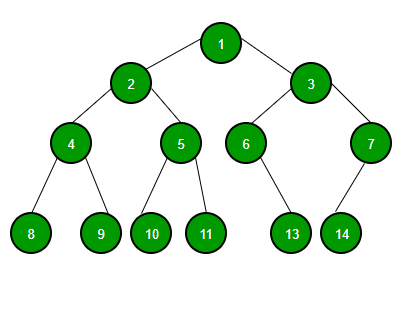
## Implementation

* + In C++ STL
  + In Java

## Problems(With Video Solutions)

* + Maximums of all subarrays of size k
  + ArrayDeque in Java
  + Design a DS with min max operations

# TREE



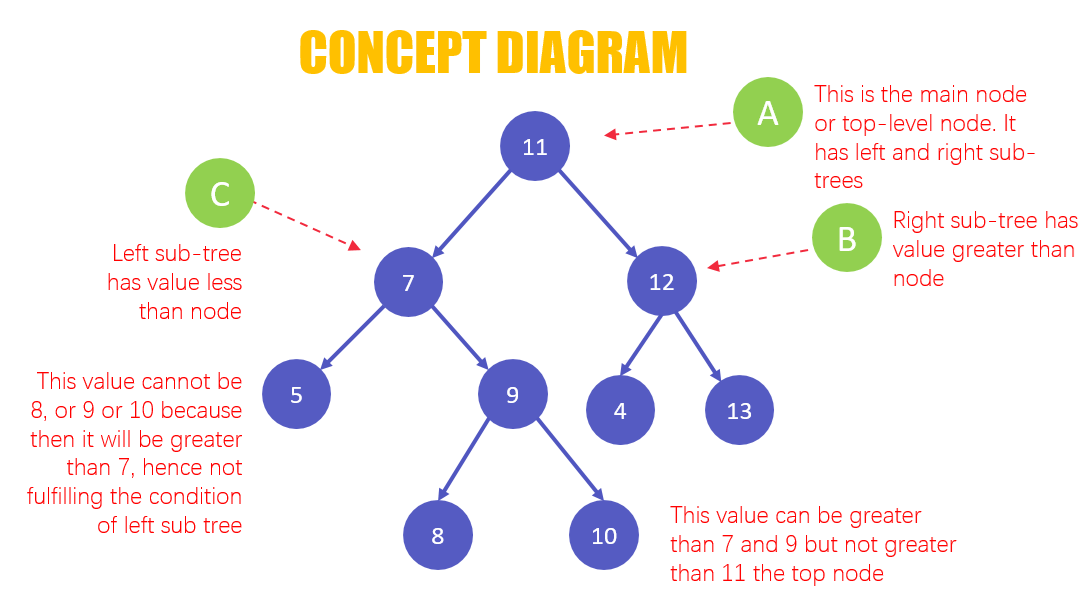
## Introduction

* + 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

* + 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

# HEAP

## Introduction & Implementation

## Binary Heap

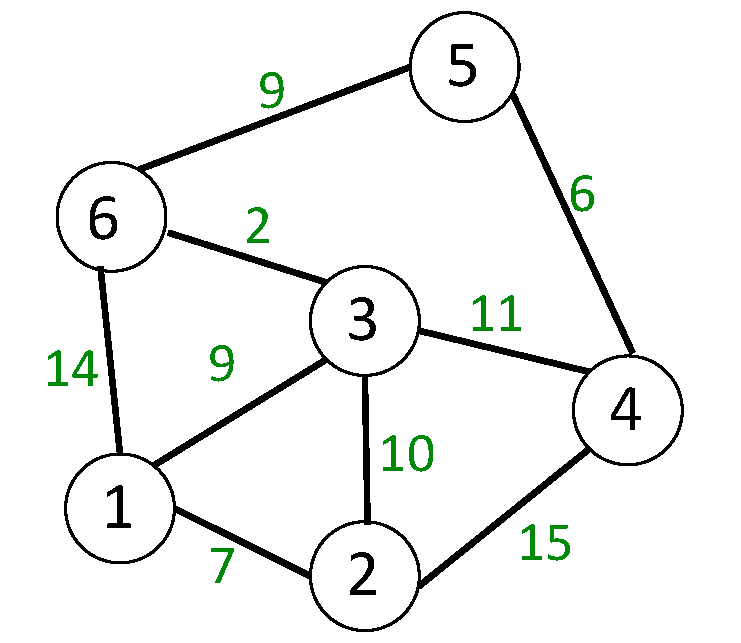
* + Insertion
  + Heapify and Extract
  + Decrease Key, Delete and Build Heap

## Heap Sort

## Priority Queue in C++

## PriorityQueue 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

### Memoization

### Tabulation

# TREE

## Introduction

### Representation

### Search

### Insert

### Delete

## Count Distinct Rows in a Binary Matrix

## SEGMENT TREE

### Introduction

 A segment tree, also known as a statistic tree, is a tree data structure used for storing information about intervals, or segments. It allows querying which of the stored segments contain a given point.

### Construction

### Range Query

### Update Query

Like tree construction and query operations, the update can also be done recursively. We are given an index which needs to be updated. Let diff be the value to be added. We start from the root of the segment tree and add diff to all nodes which have given index in their range. If a node doesn’t have a given index in its range, we don’t make any changes to that node.

## DISJOINT SET

### Introduction

Disjoint set is basically as group of sets where no item can be in more than one set. It supports union and find operation on subsets. Find(): It is used to find in which subset a particular element is in and returns the representative of that particular set.

### Find and Union Operations

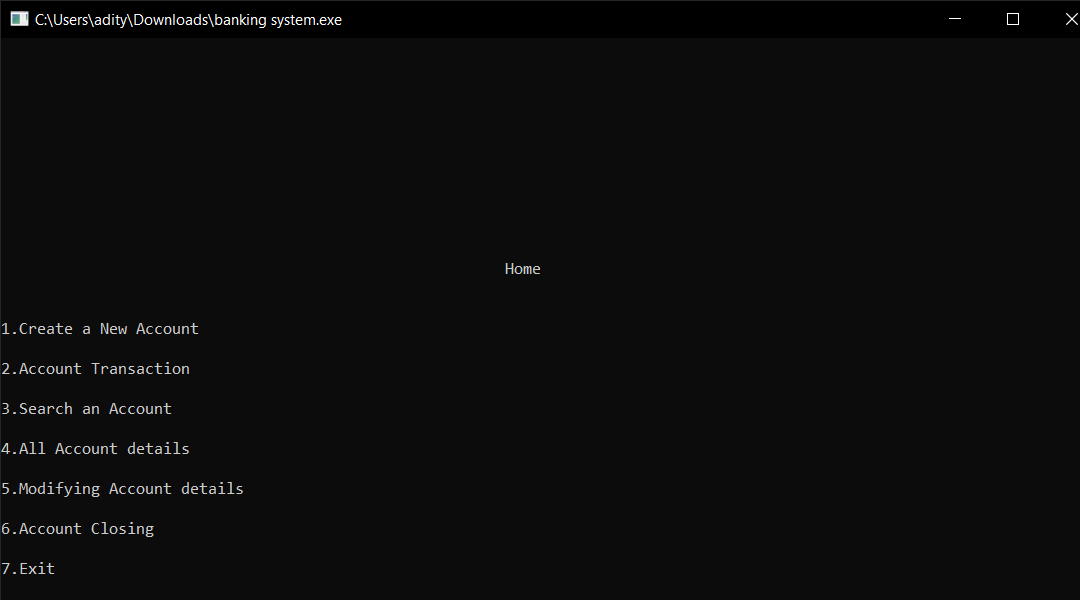
### Union by Rank

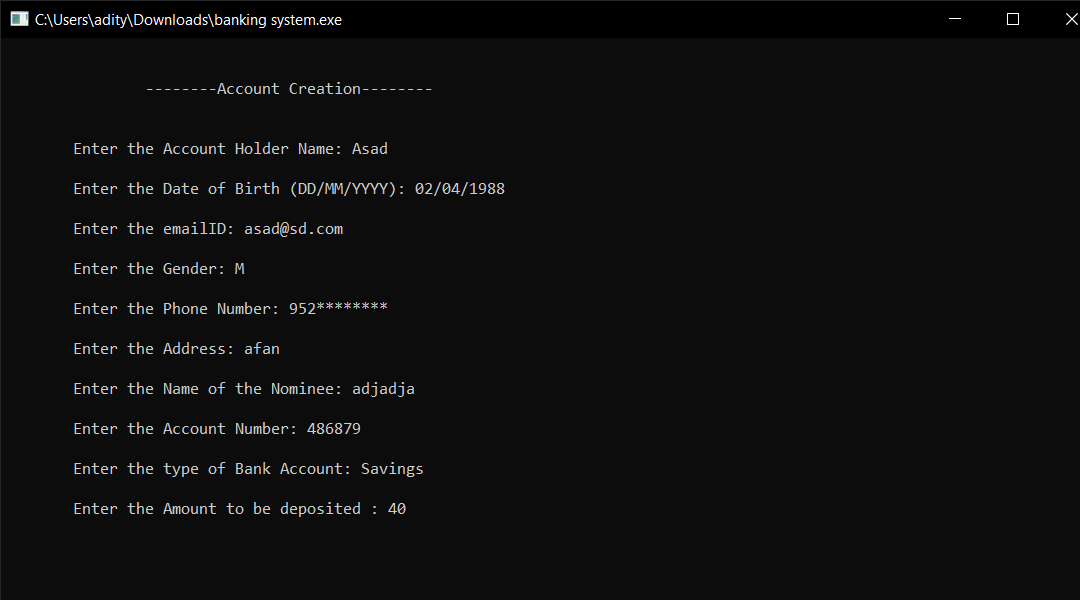
### Path Compression

### Kruskal's Algorithm

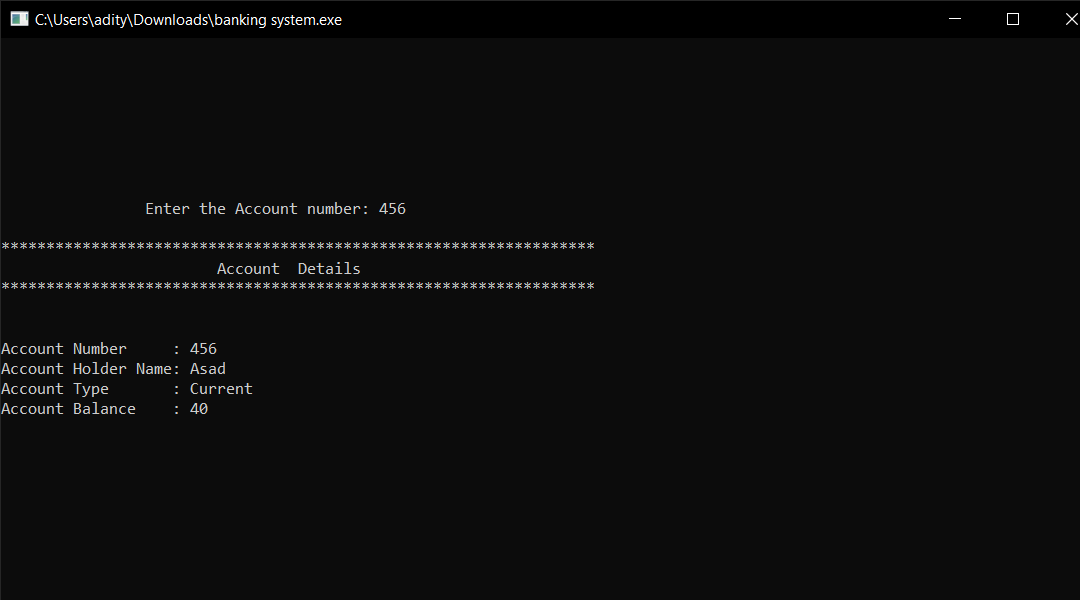
Kruskal's algorithm is a [minimum spanning tree](https://www.programiz.com/dsa/spanning-tree-and-minimum-spanning-tree#minimum-spanning) algorithm that takes a graph as input and finds the subset of the edges of that graph

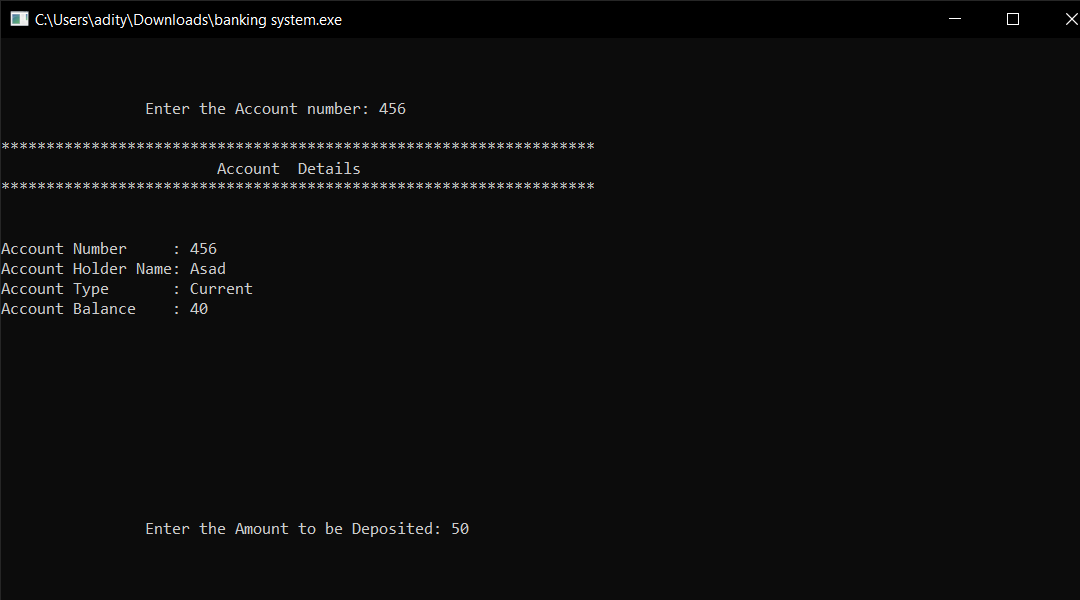
# Mini project outputs

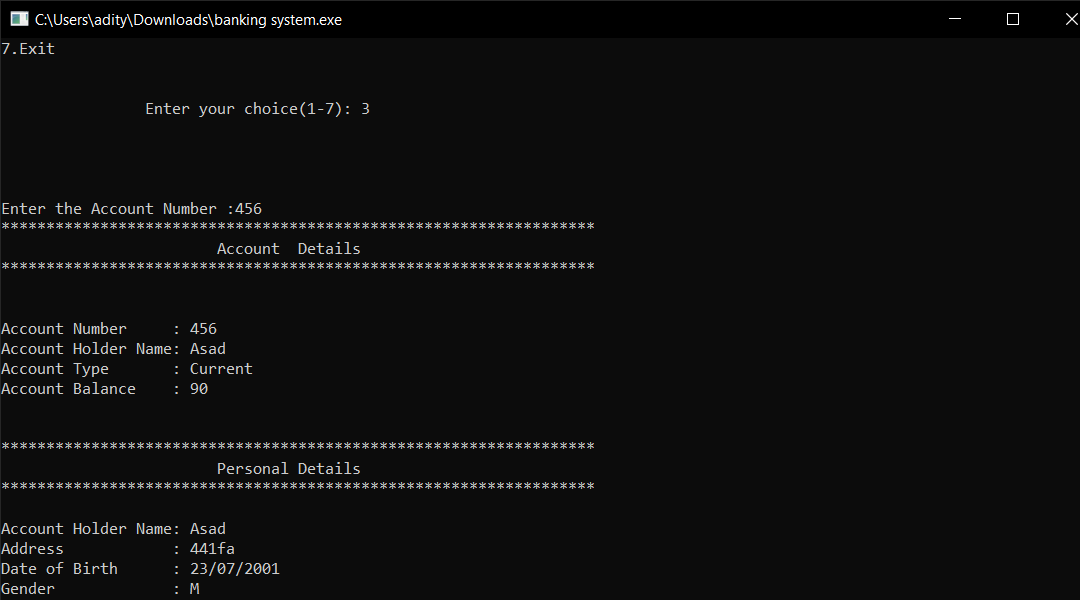
****

****

****

****

****

****

# 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 have been interested in Problem Solving and Algorithms since my first semester.
* Data structure is a thing you need to know no matter in which language you code.
* One needs to learn how to make algorithms for a real life problem he/she is facing.
* It had 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 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 programming 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 platforms for Computer Science Students.

# LEARNING OUTCOMES

A lot of beginners and experienced programmers avoid learning [Data Structures](https://www.geeksforgeeks.org/data-structures/) and [Algorithms](https://www.geeksforgeeks.org/fundamentals-of-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 have to search for 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. 1000 and 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](https://practice.geeksforgeeks.org/courses/dsa-self-paced?vb=154) -

* 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 Programming 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, algorithms are written in some languages but they are not dependent on them, one needs to make a plan and algo first then write it into any language whether it is C++ or JAVA or C or any other programming language. Algorithm is based on data structure and its implementation and working. So, basically one needs to have a good grip on DSA to work in the programing sector.

When you ask someone to make a decision 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 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.

Knowledge of data structures like Hash Tables, Trees, Tries, [Graphs](http://www.geeksforgeeks.org/graph-data-structure-and-algorithms/), and various [algorithms](https://www.geeksforgeeks.org/fundamentals-of-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 used them efficiently.
* Solved problems asked in product-based companies’ interviews.
* Solved problems in contests similar to coding rounds for SDE roles.

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

* [Geeks of geeks dsa course](https://practice.geeksforgeeks.org/batch/dsa-4)
* [Geeks of geeks website](https://www.geeksforgeeks.org/)
* [Mini Project Github file](https://github.com/aditya-since-2002/Banking-system-modified)
* [Data-structures-and-algorithm codes](https://github.com/aditya-since-2002/Data-structures-and-algorithm)