



||JAI SRI GURUDEV||

# **S J C INSTITUTE OF TECHNOLOGY**

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING** 



# DESIGN AND ANALYSIS OF ALGORITHMS LAB MANUAL [18CSL47]

# (IV SEM ISE-CBCS SCHEME)

**Prepared By:** 

Mrs. BHANUMATHI S

Assistant Professor

Department of ISE

SJCIT, Chickballapur

# S.J.C.INSTITUTE OF TECHNOLOGY

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING** 

CHICKBALLAPUR -562101

Year 2020





|| Jai Sri Gurudev || Sri Adichunchanagiri Shikshana Trust ®

## SJC INSTITUTE OF TECHNOLOGY

Chickballapur – 562 101

**Department of Information Science & Engineering** 

VISION OF THE DEPARTMENT

Educating Students to Engineer Information Science and Technology for Advancing the Knowledge as to best serve the Real-world.

MISION OF THE DEPARTMENT

M1: Focusing on Fundamentals and Applied Aspects in both Information Science Theory and Programming practices.
M2: Training comprehensively and encouraging R & D culture in trending areas on Information Technology.
M3: Collaborating with premier Institutes and Industries to nurture Innovation and learning, in cutting-edge Information Technology.
M4: Preparing the students who are much Sought-after, Productive and Well-respected for their work culture having Lifelong Learning practice.
M5: Promoting ethical and moral values among the students so as to enable them as responsible professionals.



|| Jai Sri Gurudev || Sri Adichunchanagiri Shikshana Trust ®

SJC INSTITUTE OF TECHNOLOGY

Chickballapur – 562 101

**Department of Information Science & Engineering** 

#### PROGRAM EDUCATIONAL OBJECTIVES

**PEO1:** Engage in a Successful professional career in Information Science and Technology.

**PEO2:** Pursue higher studies and research to advance the knowledge for Solving Contemporary Problems in the IT industry.

**PEO3:** Adapt to a constantly changing world through Professional Development and Sustained Learning.

**PEO4:** Exhibit professionalism and teamwork with social concern.

**PEO5:** Develop Leadership and Entrepreneurship Skills by incorporating

Organizational goals.

#### PROGRAM SPECIFIC OUTCOMES

**PSO1:** Apply the knowledge of data structures, database systems, system programming, networking web development, and AI & ML techniques in engineering the software.

**PSO2:** Exhibit solid foundations and advancements in developing software / hardware systems for solving contemporary problems.

#### DESIGN AND ANALYSIS OF ALGORITHM LABORATORY

#### [As per Choice Based Credit System (CBCS) scheme]

#### (Effective from the academic year 2018 -2019)

#### SEMESTER - IV

Subject Code	18CSL47	IA Marks	40
Number of Lecture Hours/Week	2:2	Exam Marks	60
<b>Total Number of Lecture Hours</b>	36	Exam Hours	03

Course objectives: This course will enable students to

- Design and implement various algorithms in JAVA
- Employ various design strategies for problem solving.
- Measure and compare the performance of different algorithms.

#### Description

Design, develop, and implement the specified algorithms for the following problems using Java language under LINUX /Windows environment. Net beans/Eclipse IDE tool can be used for development and demonstration.

#### LIST OF EXPERIEMENTS

1a) Create a Java Class Called Student with the following details as variables within it

- (i) USN
- (ii) Name
- (iii) Branch
- (iv) Phone

Write a Java program to create n Student objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

- **1b)** Write a Java program to implement the Stack using arrays. Write Push(), Pop(), and Display() methods to demonstrate its working.
- 2a) Design a super class called *Staff* with details as Staff Id, Name, Phone, Salary. Extend this class by writing three subclasses namely *Teaching* (domain, publications), *Technical* (skills), and *Contract* (period). Write a Java program to read and display at least 3 staff objects of all three categories.
- 2b) Write a Java class called *Customer* to store their name and date\_of\_birth. The date\_of\_birth format should be dd/mm/yyyy. Write methods to read customer data as <name, dd/mm/yyyy> and display as <name, dd, mm, yyyy> using StringTokenizer class considering the delimiter character as "/".
- 3a) Write a Java program to read two integers a and b. Compute a/b and print, when b is not zero. Raise an exception when b is equal to zero.
- 3b) Write a Java program that implements a multi-thread application that has three threads. First

thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.

- 4) Sort a given set of *n* integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus *n* on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.
- 5)Sort a given set of *n* integer elements using **Merge Sort** method and compute its time complexity. Run the program for varied values of n> 5000, and record the time taken to sort. Plot a graph of the time taken versus *n*on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide- and-conquer method works along with its time complexity analysis: worst case, average case and best case.
- 6) Implement in Java, the 0/1 Knapsack problem using (a) Dynamic Programming method (b) Greedy method.
- 7) From a given vertex in a weighted connected graph, find shortest paths to other vertices using **Dijkstra's algorithm**. Write the program in Java.
- 8) Find Minimum Cost Spanning Tree of a given connected undirected graph using **Kruskal'salgorithm.** Use Union-Find algorithms in your program.
- 9) Find Minimum Cost Spanning Tree of a given connected undirected graph using **Prim's algorithm** 
  - 10)Write Java programs to
  - (a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.
  - (b) Implement Travelling Sales Person problem using Dynamic programming.
  - 11) Design and implement in Java to find a **subset** of a given set  $S = \{S1, S2,....,Sn\}$  of *n* positive integers whose SUM is equal to a given positive integer *d*. For example, if  $S = \{1, 2, 5, 6, 8\}$  and *d*= 9, there are two solutions  $\{1, 2, 6\}$  and  $\{1, 8\}$ . Display a suitable message, if the given problem instance doesn't have a solution.
  - 12) Design and implement in Java to find all **Hamiltonian Cycles** in a connected undirected Graph G of *n* vertices using backtracking principle.

Course Outcomes: The students should be able to:

- Design algorithms using appropriate design techniques (brute-force, greedy, dynamic programming, etc.)
- Implement a variety of algorithms such assorting, graph related, combinatorial, etc., in a high level language.
- Analyze and compare the performance of algorithms using language features.
- Apply and implement learned algorithm design techniques and data structures to solvereal-world problems.

#### **Graduate Attributes**

- Engineering Knowledge
- Problem Analysis
- Modern Tool Usage
- Conduct Investigations of Complex Problems
- Design/Development of Solutions

Marks distribution: Procedure + Conduction + Viva: 20 + 50 + 10 (80). Change of experiment is allowed only once and marks allotted to the procedure part to be made zero

#### **COURSE OUTCOMES**

CO1: Design algorithms using appropriate design Techniques (brute-

force, reedy, Dynamic programming, etc.)

**CO2**: Implement a variety of algorithms such as sorting, graph related, combinatorial, etc., in a high level language.

CO3: Analyze and compare the performance of algorithms using language features.

CO4: Apply and implement learned algorithm design techniques and

datastructures to solve real-world problems.

**CO5:** Analyze algorithms to deduce their time complexities.

CO6: Choose appropriate algorithms techniques to solve computational

problems.

CO-PO Mapping Table (In the scale of 3)										<b>CO-PSO Mapping Table</b>				le			
1	2	3	4	5	6	7	8	9	10	11	12	CO/PSO	1	2	3	4	
3	3	3							2			1					
2	2								2			2					
3	3	2	2						2			3					
3	3	2	2	2					2			4		2			
3	3	1	1						2		2	5	3				
3	2	3	2	1	1			3	1	1	3	6	2				
	PO 1 3 2 3 3 3 3	PO Ma       1     2       3     3       2     2       3     3       3     3       3     3       3     3       3     3       3     3	PO         Wapping           1         2         3           3         3         3           2         2         2           3         3         2           3         3         2           3         3         2           3         3         1           3         2         3           3         3         1           3         2         3	PO     Wapping       1     2     3     4       3     3     3     2       2     2     2     2       3     3     2     2       3     3     2     2       3     3     2     2       3     3     2     2       3     3     2     2       3     3     1     1       3     2     3     2	PO Mapping Tal         1       2       3       4       5         3       3       3       4       5         2       2       2       2         3       3       2       2       2         3       3       2       2       2         3       3       2       2       2         3       3       2       2       2         3       3       1       1         3       2       3       2       1	PO       Wapping Table         1       2       3       4       5       6         3       3       3            2       2            3       3       2       2           3       3       2       2           3       3       2       2       2          3       3       2       2       2          3       3       1       1          3       2       3       2       1       1	PO       Wapping       Table (In         1       2       3       4       5       6       7         3       3       3       4       5       6       7         3       3       3       4       5       6       7         2       2       2       4       5       6       7         3       3       2       2       4       5       6       7         3       3       2       2       4       5       6       7         3       3       2       2       4       5       6       7         3       3       2       2       4       5       6       7         3       3       2       2       4       5       6       7         3       3       2       2       2       4       5       6       7         3       3       1       1       4       5       6       7         3       2       3       2       1       1       1       1	PO       Harping       Table       (In the field of	PO Mapping Table (In the scalar)         1       2       3       4       5       6       7       8       9         3       3       3       4       5       6       7       8       9         3       3       3       4       5       6       7       8       9         3       3       3       4       5       6       7       8       9         3       3       2       2       4       5       6       7       8       9         3       3       2       2       4       5       6       7       8       9         3       3       2       2       4       5       6       7       8       9         3       3       2       2       4       5       6       7       8       9         3       3       2       2       2       4       5       6       7       8       9         3       3       2       2       2       4       5       6       7       8       9       1         3       3       1       1       4	PO Mapping Table (In the scale of 3         1       2       3       4       5       6       7       8       9       10         3       3       3       .       .       .       .       2       2         2       2       .       .       .       .       .       2       2         3       3       2       2       .       .       .       .       2         3       3       2       2       .       .       .       .       .       2         3       3       2       2       .       .       .       .       .       .       .         3       3       2       2       .       .       .       .       .       .         3       3       1       1       .	PO Mapping Table (In the scale of 3)         1       2       3       4       5       6       7       8       9       10       11         3       3       3       .       .       .       .       2       2         2       2       .       .       .       .       .       2       .         3       3       2       2       .       .       .       2       .         3       3       2       2       .       .       .       .       2       .         3       3       2       2       .       .       .       .       2       .         3       3       2       2       .       .       .       .       2       .         3       3       1       1       .       .       .       .       2       .         3       3       1       1       .       .       .       .       2       .         3       3       1       1       .       .       .       .       2       .       .       .       .       .       .       .	PO Mapping Table (In the scale of 3)         1       2       3       4       5       6       7       8       9       10       11       12         3       3       3       4       5       6       7       8       9       10       11       12         3       3       3       4       5       6       7       8       9       10       11       12         2       2       4       5       6       7       8       9       10       11       12         3       3       2       2       4       4       5       6       7       8       9       10       11       12         3       3       2       2       4       4       5       6       7       8       9       10       11       12         3       3       2       2       4       4       5       6       7       8       9       10       11       12         3       3       2       2       2       4       4       4       2       2       2       2       2       2       2       2	PO Mapping Table (In the scale of 3)       CO-PSO Map         1       2       3       4       5       6       7       8       9       10       11       12       CO/PSO         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO         3       3       2       2       4       2       4       3       3       3       1       1       4       2       4       4       3       3       1       1       3       6	PO Mapping Table (In the scale of 3)       CO-PSO Mappin         1       2       3       4       5       6       7       8       9       10       11       12       CO/PSO       1         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1         2       2       4       2       4       2       1       1       1         3       3       2       2       4       2       2       3       3         3       3       1       1       4       2       2       4       4         3       3       1       1       2       2       2       5       3         3       3       1       1       3       1       1       3       6       2	PO Mapping Table (In the scale of 3)       CO-PSO Mapping Table (In the scale of 3)         1       2       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2         3       3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2         3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2         3       3       3       2       2       4       2       2       1 <t< th=""><th>PO Mapping Table (In the scale of 3)       CO-PSO Mapping Table         1       2       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         2       2       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       2       2       4       2       4       2       4       2       4       2       4       2       4       3       4       3       4       4       2       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4</th></t<>	PO Mapping Table (In the scale of 3)       CO-PSO Mapping Table         1       2       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       3       3       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         2       2       4       5       6       7       8       9       10       11       12       CO/PSO       1       2       3         3       3       2       2       4       2       4       2       4       2       4       2       4       2       4       3       4       3       4       4       2       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4       4	

### CONTENTS

PROGRAM TITLE	PAGE NO
1a)Program to read student details	
	1-7
1b) Java program to implement the Stack using arrays. Write Push(),	8-18
Pop() and Display()	
2a) Java program to read and display at least 3 <i>staff</i> objects of all three categories.	19-25
2b) Java class called <i>Customer</i> to store their name and date_of_birth. The date_of_ birth format should be dd/mm/yyyy. Write methods to read customer data as <name, dd="" mm="" yyyy=""> and display as <name, dd,="" mm,="" yyyy=""> using StringTokenizer class considering the delimiter character as "/".</name,></name,>	26-29
3a) Java program to read two integers $a$ and $b$ . Compute $a/b$ and print, when $b$ is not zero. Raise an exception when $b$ is equal to zero.	30-33
3b) Java program that implements a multi-thread application that has three	
threads. First thread generates a random integer for every 1 second; second	
thread computes the square of the number and prints; third	34-42
thread will print the value of cube of the number	
4) Quick Sort	43-49
5) Merge Sort	50-55
6) 0/1 Knapsack Problem using	
(i)Dynamic Programming	56-62
(ii) Greedy Method	63-67
7) Dijkstra's algorithm	68-71
8) Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm.	72-80
9) Find Minimum Cost Spanning Tree of a given connected undirected graph using <b>Prim's algorithm</b>	81-86
10a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.	87-90
10b) Implement <b>Travelling Sales Person problem</b> using Dynamic	91-94
Programming.	
11) SubSet Problem	95-99
12)Hamiltonian Cycle Problem	100-105
Additional Programs	106-110
APPENDIX:-VIVA VOICE QUESTIONS	111-120
LAB EVALUATION RUBRICS	121

### **EXPERIMENT 1A**

#### **PROGRAM STATEMENT**

Create a Java class called *Student* with the following details as variables within it.

(i) USN

(ii) Name

(iii) Branch

(iv) Phone

Write a Java program to create *n Student* objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

## **CONCEPT**

In Java everything is encapsulated under classes. Class is the core of Java language. Class can be defined as a template/ blueprint that describe the behaviors /states of a particular entity. A class defines new data type. This type can be used to create object of that type. Object is an instance of class. You may also call it as physical existence of a logical template class.

A class is declared using **class** keyword. A class contains both data and code that operate on that data. The data or variables defined within a **class** are called **instance variables** and the code that operates on this data is known as **methods**.

A simple class example

class Student String USN, name, branch; int phoneno;

}

ł

Object is an instance of a class created using a new operator. The new operator returns a reference to a new instance of a class. This reference can be assigned to a reference variable of the class. The process of creating objects from a class is called instantiation. An object encapsulates state and behavior.

An object reference provides a handle to an object that is created and stored in memory. In Java, objects can only be manipulated via references, which can be stored in variables.

Creating variables of your class type is similar to creating variables of primitive data types, such as integer or float. Each time you create an object, a new set of instance variables comes into existence which defines the characteristics of that object. If you want to create an objectof the class and have the reference variable associated with this object, you must also allocate memory for the object by using the **new operator**. This process is called instantiating an object or creating an object instance. In following statement **obj** is instance/object of Student class.

#### Student obj=new Student();

An array of objects is created just like an array of primitive type data items in the following way.

```
Student[] obj_Array = new Student[7];
```

However, in this particular case, we may use a for loop since all Student objects are created with the same default constructor.

```
for ( int i=0; i<obj_Array.length; i++)
{
    obj_Array[i]=new Student();
}Constructor</pre>
```

Constructor in java is a *special type of method* that is used to initialize the object. Java constructor is *invoked at the time of object creation*. It constructs the values i.e. provides data for the object that is why it is known as constructor.

#### Types of java constructors

There are two types of constructors:

- 1. Default constructor (no-arg constructor)
- 2. Parameterized constructor

A constructor that have no parameter is known as default constructor. Student()

```
//block of code
```

{

}

Object creation: *Student obj=new Student();* Constructor with arguments is known as parameterized constructor.

```
Student(int i,String n)
{
    id = i;
    name = n;
}
```

```
Object creation:
Student4 s1 = new Student4(111,''Karan'');
```

## PROGRAM

/\* 1a. Create a Java class called Student with the following details as variables within it.

(i) USN

(ii) Name

(iii) Branch

(iv) Phone

Write a Java program to create nStudent objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

\*/

ł

import java.util.Scanner;

class Student

String USN, Name, Branch, Phone;

Scanner input = new Scanner(System.in);

void read()

{

```
System.out.println("Enter Student Details");
System.out.println("Enter USN");
USN = input.nextLine();
```

System.out.println("Enter Name"); Name = input.nextLine();

System.out.println("Enter Branch"); Branch = input.nextLine();

```
System.out.println("Enter Phone");
Phone = input.nextLine();
```

```
}
```

{

ł

void display()

System.out.printf("%-20s %-20s %-20s %-20s", USN, Name, Branch, Phone);

```
class studentdetails
```



}

```
public static void main(String[] args)
          Scanner input = new Scanner(System.in);
          System.out.println("Enter number of student details to be created");
          int number = input.nextInt();
          Student s[] = new Student[number];
          // Read student details into array of student objects
          for (int i = 0; i < number; i++)
          {
                 s[i] = new Student();
                 s[i].read();
          }
          // Display student information
          System.out.printf("%-20s %-20s %-20s", "USN", "NAME", "BRANCH",
          "PHONE");
          for (int i = 0; i < number; i++)
          {
                        System.out.println();s[i].display();
          }
   input.close();
  }
}
```

Design & Analysis of Algorithm Lab Manual

# 2020

OUTPUT Enter number	of student detai	ls to be created	
Enter Student	Details		
Enter USN			
ISJ16CS097 Enter Name			
ramu			
Enter Branch			
cse			
Enter Phone			
992343078 Enter Student	Details		
Enter USN	Details		
1si15me45			
Enter Name			
harish			
Enter Branch			
med Enter Phone			
984512345			
Enter Student	Details		
Enter USN			
1sj17ec111			
Enter Name			
Shallin Enter Branch			
ece			
Enter Phone			
9982345678			
USN	NAME	BRANCH	PHONE
1sj10cs097 1sj15me45	harish	med	984512345
1sj17ec111	shalini	ece	9982345678
~			

#### NOTES AND OUTPUT:

Dept of ISE, SJCIT

Dept of ISE, SJCIT

2	0	2	0
_	~	_	~

Dept of ISE, SJCIT

### **EXPERIMENT 1B**

### **PROGRAM STATEMENT**

Write a Java program to implement the Stack using arrays. Write Push(), Pop(), and Display() methods to demonstrate its working

### CONCEPT

In Java everything is encapsulated under classes. Class is the core of Java language. Class can be defined as a template/ blueprint that describe the behaviors /states of a particular entity. A class defines new data type. This type can be used to create object of that type. Object is an instance of class. You may also call it as physical existence of a logical template class. A class is declared using **class** keyword. A class contains both data and code that operate on that data. The data or variables defined within a **class** are called **instance variables** and the code that operates on this data is known as **methods**.

A simple class example

class Student

#### String USN, name, branch; int phoneno;

}

ł

Object is an instance of a class created using a new operator. The new operator returns a reference to a new instance of a class. This reference can be assigned to a reference variable of the class. The process of creating objects from a class is called instantiation. An object encapsulates state and behavior. An object reference provides a handle to an object that is created and stored in memory. In Java, objects can only be manipulated via references, which can be stored in variables. Creating variables of your class type is similar to creating variables of primitive data types, such as integer or float. Each timeyou create an object, a new set of instance variables comes into existence which defines the characteristics of that object. If you want to create an object of the class and have the reference variable associated with this object, you must also allocate memory for the object by using the **new operator**. This process is called instantiating an object or creating an object instance. In following statement **obj** is instance/object of Student class.

#### Student obj=new Student();

An array of objects is created just like an array of primitive type data items in the following way.

#### Student[] obj\_Array = new Student[7];

However, in this particular case, we may use a for loop since all Student objects are created with the same default constructor.

```
for ( int i=0; i<obj_Array.length; i++)
{
    obj_Array[i]=new Student();</pre>
```

## }

#### Constructor

Constructor in java is a *special type of method* that is used to initialize the object. Java constructor is *invoked at the time of object creation*. It constructs the values i.e. provides data for the object that is why it is known as constructor.

#### Types of java constructors

There are two types of constructors:

- 1. Default constructor (no-arg constructor)
- 2. Parameterized constructor

A constructor that have no parameter is known as default constructor.

```
Student()
{
   //block of code
}
```

```
Object creation:
Student obj=new Student();
```

Constructor with arguments is known as parameterized constructor.

```
Student(int i,String n)
{
        id = i;
        name = n;
}
```

```
Object creation:

Student4 s1 = new Student4(111, ''Karan'');
```

#### PROGRAM

```
/* 1b. Write a Java program to implement the Stack using arrays. Write Push(), Pop(), and Display() methods
to demonstrate its working. */
import java.util.*;
class arrayStack
{
        int arr[];
        int top, max;
        arrayStack(int n)
        {
                max = n;
                arr = new int[max];
                top = -1;
        }
        void push(int i)
        {
                if (top == max - 1)
                        System.out.println("Stack Overflow");
                else
                        arr[++top] = i;
        }
        void pop()
        {
```

{
 System.out.println("Stack Underflow");
 }
 else
 {
 int element = arr[top--];
 System.out.println("Popped Element: " + element);
 }
}
void display()
{
 System.out.print("\nStack = ");
 if (top == -1)
 {
}

if (top == -1)

Dept of ISE, SJCIT

System.out.print("Empty\n");

Dept of ISE, SJCIT

```
return;
                }
                for (int i = top; i \ge 0; i--)
                        System.out.print(arr[i] + " ");
                System.out.println();
        }
}
class Stack
{
        public static void main(String[] args)
        {
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter Size of Integer Stack ");
                int n = scan.nextInt();
                boolean done = false;
                arrayStack stk = new arrayStack(n);
                char ch;
                do
                {
                        System.out.println("\nStack Operations");
                         System.out.println("1. push");
                         System.out.println("2. pop");
                        System.out.println("3. display");
                        System.out.println("4. Exit");
                        int choice = scan.nextInt();
                        switch (choice)
                         {
                        case 1:
                                 System.out.println("Enter integer element to push");
                                 stk.push(scan.nextInt());
                                 break;
                         case 2:
                                 stk.pop();
                                 break;
                         case 3:
                                 stk.display();
                                 break;
                         case 4:
                                 done = true;
                                 break;
                         default:
                                 System.out.println("Wrong Entry \n ");
```

break;
<pre>} while (!done):</pre>
}
}
<u>OUTPUT</u>
Enter Size of Integer Stack
5
Stack Operations
1. push
2. pop
3. display
4. Exit
1
Enter integer element to push
1
Stack Operations
1. push
2. pop
3. display
4. Exit
1
Enter integer element to push
2
Stack Operations
1. push
2. pop
3. display

Dept of ISE, SJCIT

Enter integer element to push

3

Stack Operations

1. push

2. pop

3. display

4. Exit

1

Enter integer element to push

4

Stack Operations

1. push

2. pop

3. display

4. Exit

1

Enter integer element to push

5

Stack Operations

1. push

2. pop

3. display

4. Exit

1

Dept of ISE, SJCIT

		Design & Analysis of Algorithm Lab Manual	2020
	Enter integer element to push		
	6		
	Stack Overflow		
	Stack Operations		
	1. push		
	2. pop		
	3. display		
	4. Exit		
	3		
	Stack = 5 4 3 2 1		
	Stack Operations		
	1. push		
	2. pop		
	3. display		
	4. Exit		
	2		
	Popped Element: 5		
	Stack Operations		
	1. push		
	2. pop		
	3. display		
	4. Exit		
	2		
	Popped Element: 4		
	Stack Operations		
	1. push		
_			
Dept	of ISE, SJCIT		

2. pop
--------

3. display

4. Exit

2

Popped Element: 3

Stack Operations

1. push

2. pop

3. display

4. Exit

2

Popped Element: 2

Stack Operations

1. push

2. pop

3. display

4. Exit

3

Stack = 1

Stack Operations

1. push

2. pop

3. display

4. Exit

2

Popped Element: 1

Dept of ISE, SJCIT

1. 2. 3. 4.	. push . pop		
2. 3. 4.	. pop		
3. 4.			
4.	. display		
	. Exit		
2			
St	tack Underflow		
St	tack Operations		
1.	. push		
2.	. pop		
3.	. display		
4.	. Exit		
3			
St	tack = Empty		
St	tack Operations		
1.	. push		
2.	. pop		
3.	. display		
4.	. Exit		
4			

NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT

#### **EXPERIMENT 2A**

#### **PROGRAM STATEMENT:**

Design a super class called Staff with details as StaffId, Name, Phone, Salary. Extend this class by writing three subclasses namely Teaching (domain, publications), Technical (skills), and Contract (period). Write a Java program to read and display at least 3 staff objects of all three categories.

#### **CONCEPT:**

Here in this given problem we shall use inheritance for extending Staff class into: Teaching, Technical and Contract 3 subclasses using extends keyword. Each class is having the variables as given in the bracket. We will import the util package Scanner class to read 3 objects of each class. Create a constructor of Staff class to initialize StaffId, Name, Phone, Salary. And one display function into the Staff class to display the entered values. All the data members of Staff will be inherited in Teaching, Technical and Contract using super keyword that calls the super class constructor to base class. Other additional data members of the subclasses would be initialized using there own constructor. Also along with there own constructors all 3 subclasses will have there own display method that invokes display method of super class Staff. Now in main() method using Scanner class we will read the values accordingly. To display these values we will create array of object of size 3 for each subclassTeaching, Technical and Contract. Using this array of objects we will display the values entered previously by invoking display method of each subclass. Below is the program that demonstrates the same.

#### **PROGRAM:**

/\*\* Design a super class called Staff with details as StaffId, Name, Phone, Salary. Extend this class by writing three subclasses namely Teaching (domain, publications), Technical (skills), and Contract (period).

Write a Java program to read and display at least 3 staff objects of all three categories.\*/

import java.util.Scanner;

```
class Staff
```

```
{
```

String StaffID, Name, Phone, Salary;

Scanner input = new Scanner(System.in);

```
void read()
```

```
{
```

```
System.out.println("Enter StaffID");
StaffID = input.nextLine();
```

```
System.out.println("Enter Name");
Name = input.nextLine();
```

```
System.out.println("Enter Phone");
              Phone = input.nextLine();
              System.out.println("Enter Salary");
              Salary = input.nextLine();
       }
       void display()
       ł
              System.out.printf("\n%-15s", "STAFFID: ");
              System.out.printf("%-15s \n", StaffID);
              System.out.printf("%-15s", "NAME: ");
              System.out.printf("%-15s \n", Name);
              System.out.printf("%-15s", "PHONE:");
              System.out.printf("%-15s \n", Phone);
              System.out.printf("%-15s", "SALARY:");
              System.out.printf("%-15s \n", Salary);
       }
}
class Teaching extends Staff
{
       String Domain, Publication;
       void read_Teaching()
       {
              super.read(); // call super class read method
              System.out.println("Enter Domain");
              Domain = input.nextLine();
              System.out.println("Enter Publication");
              Publication = input.nextLine();
       }
       void display()
       {
              super.display(); // call super class display() method
              System.out.printf("%-15s", "DOMAIN:");
              System.out.printf("%-15s \n", Domain);
              System.out.printf("%-15s", "PUBLICATION:");
              System.out.printf("%-15s \n", Publication);
       }
}
class Technical extends Staff
Dept of ISE, SJCIT
```

```
{
       String Skills;
       void read_Technical()
       {
               super.read(); // call super class read method
               System.out.println("Enter Skills");
               Skills = input.nextLine();
       }
       void display()
               super.display(); // call super class display() method
               System.out.printf("%-15s", "SKILLS:");
               System.out.printf("%-15s \n", Skills);
       }
}
class Contract extends Staff
{
       String Period;
       void read_Contract()
               super.read(); // call super class read method
               System.out.println("Enter Period");
               Period = input.nextLine();
       }
       void display()
       {
               super.display(); // call super class display() method
               System.out.printf("%-15s", "PERIOD:");
               System.out.printf("%-15s \n", Period);
       }
}
class Staffdetails
{
       public static void main(String[] args)
       {
               Scanner input = new Scanner(System.in);
               System.out.println("Enter number of staff details to be created");
Dept of ISE, SJCIT
```

```
int n = input.nextInt();
Teaching steach[] = new Teaching[n];
Technical stech[] = new Technical[n];
Contract scon[] = new Contract[n];
// Read Staff information under 3 categories
for (int i = 0; i < n; i++)
{
       System.out.println("Enter Teaching staff information");
       steach[i] = new Teaching();
       steach[i].read_Teaching();
}
for (int i = 0; i < n; i++)
       System.out.println("Enter Technical staff information");
       stech[i] = new Technical();
       stech[i].read_Technical();
}
for (int i = 0; i < n; i++)
{
       System.out.println("Enter Contract staff information");
       scon[i] = new Contract();
       scon[i].read_Contract();
}
// Display Staff Information
System.out.println("\n STAFF DETAILS: \n");
System.out.println("-----TEACHING STAFF DETAILS----- ");
for (int i = 0; i < n; i++)
{
       steach[i].display();
ł
System.out.println();
System.out.println("-----TECHNICAL STAFF DETAILS ---- ");
for (int i = 0; i < n; i++)
{
       stech[i].display();
}
```

```
System.out.println();
             System.out.println("-----CONTRACT STAFF DETAILS ---- ");
             for (int i = 0; i < n; i++)
             {
                   scon[i].display();
             }
             input.close();
      }
}
Enter number of staff details to be created
1
Enter Teaching staff information
Enter StaffID
111
Enter Name
MAHESH n
Enter Phone
9916130045
Enter Salary
40000
Enter Domain
ise
Enter Publication
ijaret
Enter Technical staff information
Enter StaffID
222
Enter Name
kumar
Enter Phone
984512345
Enter Salary
10000
Enter Skills
c/c++
Enter Contract staff information
Enter StaffID
123
Enter Name
rajesh
Enter Phone
789123545
Enter Salary
8000
Enter Period
1 year
STAFF DETAILS:
----TEACHING STAFF DETAILS-----
Dept of ISE, SJCIT
```

STAFFID:	111
NAME:	MAHESH n
PHONE:	9916130045
SALARY:	40000
DOMAIN:	i <u>se</u>
PUBLICATION:	ijaret
TECHNICAL	STAFF DETAILS
STAFFID:	222
NAME :	kumar
PHONE:	984512345
SALARY:	10000
SKILLS:	c/c++
CONTRACT S	STAFF DETAILS
STAFFID:	123
NAME:	rajesh

PHONE :	789123545
SALARY:	8000
PERIOD:	1 year

#### NOTES AND OUTPUT SPACE

2	0	2	0
	υ		υ

Dept of ISE, SJCIT

#### **EXPERIMENT 2B**

## **PROGRAM STATEMENT**

Write a Java class called Customer to store their name and date\_of\_birth. The date\_of\_birth format should be dd/mm/yyyy. Write methods to read customer data as <name, dd/mm/yyyy> and display as <name,dd,mm,yyyy> using StringTokenizer class considering the delimiter character as "/".

#### CONCEPT

The **java.util.StringTokenizer** class allows you to break a string into tokens. It is simple way to break string. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments. The set of delimiters (the characters that separate tokens) may be specified either at creation time or on a per-token basis.

#### Constructors of StringTokenizer class

There are 3 constructors defined in the StringTokenizer class.

Constructor	Description	
StringTokonizor(Strin.str)	creates StringTokenizer with	
Sumgrokemzer(Sum Su)	specified string.	
StringTokenizer(String str, String delim)	creates StringTokenizer with specified string and	
	delimiter.	
StringTokenizer(String str,String delim,Boolean returnValue)	creates StringTokenizer with specified string, delimiter and returnValue. If return value is true, delimiter characters are considered to be tokens. If it is false, delimiter characters serve to separate tokens.	

The 6 useful methods of StringTokenizer class are as follows:

Public method	Description
boolean hasMoreTokens()	checks if there is more tokens available.
String nextToken()	returns the next token from the StringTokenizer
	object.
String nextToken(String delim)	returns the next token based on the delimeter.
boolean hasMoreElements()	same as hasMoreTokens() method.
Object nextElement()	same as nextToken() but its return type is Object.

int countTokens()

returns the total number of tokens.

#### **PROGRAM**:

/\* 2b. Write a Java class called Customer to store their name and date\_of\_birth. The date\_of\_birth format should be dd/mm/yyyy. Write methods to read customer data as <name, dd/mm/yyyy> and display as <name, dd, mm, yyyy> using StringTokenizer class considering the delimiter character as "/".

\*/

```
import java.util.Scanner;
import java.util.StringTokenizer;
public class Customer
{
        public static void main(String[] args)
        ł
                String name;
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter Name and Date_of_Birth in the format
                                        <Name,DD/MM/YYYY>");
                name = scan.next();
                // create stringTokenizer with delimiter "/"
                StringTokenizer st = new StringTokenizer(name, ",/");
                // Count the number of tokens
                int count = st.countTokens();
                // Print one token at a time and induce new delimiter ","
                for (int i = 1; i \le count \&\& st.hasMoreTokens(); i++)
                {
                        System.out.print(st.nextToken());
                        if (i < count)
                                System.out.print(",");
                }
        }
}
```

#### OUTPUT:-

Enter Name and Date\_of\_Birth in the format <Name,DD/MM/YYYY>

rajesh,05/04/1984

rajesh,05,04,1984

NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT
2	0	2	0
_	v	_	v

## **EXPERIMENT 3A**

## PROGRAM STATEMENT

Write a java program to read two integers and b. Compute a/b and print when b is not zero. Raise an exception when b is equal to zero.

## **CONCEPT** Exception Handling

- 1. An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.
- 2. All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.
- 3. The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.



## PROGRAM

/\* 3a. Write a Java program to read two integers a and b. Compute a/b and print, when b is not zero. Raise an exception when b is equal to zero.\*/

import java.util.Scanner;

```
class exception
{
```

public static void main(String[] args)
{
 int a, b, result;

```
Scanner input = new Scanner(System.in);
                     System.out.println("Input two integers");
                     a = input.nextInt();
                     b = input.nextInt();
                     try
                      {
                            result = a / b;
                            System.out.println("Result = " + result);
                      }
                     catch (ArithmeticException e)
                      {
                            System.out.println("Exception caught: Division by zero.");
                      }
              }
       }
       OUTPUT
       RUN1
       Input two integers
       4
       5
       Result = 0
       RUN2
       Input two integers
       5
       4
       Result = 1
       RUN3
       Input two integers
       5
       4
       Result = 1
Dept of ISE, SJCIT
```

	Design & Analysis of Algorithm Lab Manual	2020
NOTES AND OUTPUT SPACE		

#### **EXPERIMENT 3B**

## **PROGRAM STATEMENT**

Write a Java program that implements a multi-thread application that hashtree threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number.

#### **CONCEPT:**

Thread is basically a lightweight sub-process, a smallest unit of processing. Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

#### Life Cycle of a Thread



Following are the stages of the life cycle -

- □ New A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.
- $\square$  **Runnable** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
- **Waiting** Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
- Timed Waiting A runnable thread can enter the timed waiting state for a specified interval of time. A

thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs. □ **Terminated (Dead)** – A runnable thread enters the terminated state when it completes its task or otherwise terminates.

Create a Thread by Extending a Thread Class

The second way to create a thread is to create a new class that extends **Thread** class using the following two simple steps. This approach provides more flexibility in handling multiple threads created using available methods in Thread class.

**Step 1:** You will need to override **run(**) method available in Thread class. This method provides an entry point for the thread and you will put your complete business logic inside this method. Syntax of run() method – public void run()

**Step 2 :** Once Thread object is created, you can start it by calling **start()** method, which executes a call to run() method. Syntax of start() method – void start();

#### Synchronization :

Inter thread communication is important when you develop an application where two or more threads exchange some information. At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called **synchronization**. To do this we can use either Synchronized block or Synchronized method.

There are three simple methods which makes thread communication possible, listed below  $- \setminus$ 

Sr.No.	Method & Description
	public void wait()
1	Causes the current thread to wait until another thread invokes the notify().
	public void notify()
2	Wakes up a single thread that is waiting on this object's monitor.
	public void notifyAll()
3	Wakes up all the threads that called wait() on the same object.

These methods have been implemented as **final** methods in Object, so they are available in all the classes. All three methods can be called only from within a **synchronized** context.

## PROGRAM

#### MultiThread.java

/\* 3b. Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer for every 1 second; second thread computes the square of the number and prints; third thread will print the value of cube of the number \*/

```
import java.util.Random;
class SquareThread implements Runnable
{
       int x:
       SquareThread(int x)
              this.x = x;
       }
       public void run()
            System.out.println("Thread Name:Square Thread and Square of +x + " is: +x * x);
       }
}
class CubeThread implements Runnable
{
       int x;
       CubeThread(int x)
              this.x = x;
       }
       public void run()
       {
           System.out.println("Thread Name:Cube Thread and Cube of " + x + " is: " + x * x * x);
       }
}
```

class RandomThread implements Runnable

Dept of ISE, SJCIT

{

}

{

2020

```
Random r;
              Thread t2, t3;
              public void run()
              {
                    int num;
                    r = new Random();
                    try
                     {
                            while (true)
                            {
                                   num = r.nextInt(10);
                                   System.out.println("Main Thread and Generated Number is " + num);
                                   t2 = new Thread(new SquareThread(num));
                                   t2.start();
                                   t3 = new Thread(new CubeThread(num));
                                   t3.start();
                                   Thread.sleep(1000);
                                   System.out.println("_____");
                            }
                     }
                    catch (Exception ex)
                     {
                            System.out.println("Interrupted Exception");
                     }
              }
       public class MultiThread
              public static void main(String[] args)
              {
                    RandomThread thread_obj = new RandomThread();
                    Thread t1 = new Thread(thread_obj);
                    t1.start();
                                                                                               Page 37
Dept of ISE, SJCIT
```

}

#### **OUTPUT**

}

Main Thread and Generated Number is 3 Thread Name: Square Thread and Square of 3 is: 9 Thread Name: Cube Thread and Cube of 3 is: 27 ------Main Thread and Generated Number is 6 Thread Name: Square Thread and Square of 6 is: 36 Thread Name: Cube Thread and Cube of 6 is: 216 Main Thread and Generated Number is 6 Thread Name: Square Thread and Square of 6 is: 36 Thread Name: Cube Thread and Cube of 6 is: 216 \_\_\_\_\_ Main Thread and Generated Number is 0 Thread Name: Square Thread and Square of 0 is: 0 Thread Name: Cube Thread and Cube of 0 is: 0 Main Thread and Generated Number is 4 Thread Name: Square Thread and Square of 4 is: 16 Thread Name: Cube Thread and Cube of 4 is: 64 Main Thread and Generated Number is 5 Thread Name: Square Thread and Square of 5 is: 25

Dept of ISE, SJCIT

Thread Name:Cube Thread and Cube of 5 is: 125	
Main Thread and Generated Number is 6	
Thread Name:Square Thread and Square of 6 is: 36	
Thread Name:Cube Thread and Cube of 6 is: 216	
Main Thread and Generated Number is 8	
Thread Name:Square Thread and Square of 8 is: 64	
Thread Name:Cube Thread and Cube of 8 is: 512	
Main Thread and Generated Number is 2	
Thread Name:Square Thread and Square of 2 is: 4	
Thread Name:Cube Thread and Cube of 2 is: 8	
Main Thread and Generated Number is 0	
Thread Name:Square Thread and Square of 0 is: 0	
Thread Name:Cube Thread and Cube of 0 is: 0	
Main Thread and Generated Number is 6	
Thread Name:Square Thread and Square of 6 is: 36	
Thread Name:Cube Thread and Cube of 6 is: 216	
Main Thread and Generated Number is 0	
Thread Name:Square Thread and Square of 0 is: 0	
Thread Name: Cube Thread and Cube of 0 is: 0	

Main Thread and Generated Number is 9
Thread Name:Square Thread and Square of 9 is: 81
Thread Name:Cube Thread and Cube of 9 is: 729
Main Thread and Generated Number is 3
Thread Name:Square Thread and Square of 3 is: 9
Thread Name:Cube Thread and Cube of 3 is: 27
Main Thread and Generated Number is 6
Thread Name:Cube Thread and Cube of 6 is: 216
Thread Name:Square Thread and Square of 6 is: 36
Main Thread and Generated Number is 1
Thread Name:Cube Thread and Cube of 1 is: 1
Thread Name:Square Thread and Square of 1 is: 1
Main Thread and Generated Number is 2
Thread Name:Square Thread and Square of 2 is: 4
Thread Name:Cube Thread and Cube of 2 is: 8

	Design & Analysis of Algorithm Lab Manual	2020
NOTES AND OUTPUT SPACE		

2	0	2	0
_	v	_	v

## **EXPERIMENT 4**

### **PROGRAM STATEMENT**

Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n > 5000 and record the time taken to sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide- and-conquer method works along with its time complexity analysis: worst case, average case and best case.

#### CONCEPT

**Divide and Conquer:** In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.



Broadly, we can understand **divide-and-conquer** approach in a three-step process. Divide/Break: This step involves breaking the problem into smaller sub-problems. Subproblems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, subproblems become atomic in nature but still represent some part of the actual problem. Conquer/Solve: This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

Merge/Combine: When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer & merge steps works so close that they appear as one.

The following computer algorithms are based on **divide-and-conquer** programming approach .

- Merge Sort
- Quick Sort Method:

Quick Sort divides the array according to the value of elements. It rearranges elements of a given array A[0..n-1] to achieve its partition, where the elements before position s are smaller than or equal to A[s] and all the elements after position s are greater than or equal to A[s].

A[0]A[s-1]	A[s]	A[s+1]A[n-1]
All are <=A[s]		all are $\geq = A[s]$

#### Algorithm : QUICKSORT(a[1..r]) //Sorts a subarray by quicksort

//Input: A subarray A[l..r] of A[0..n-1],defined by its left and right indices l and r //Output: Subarray A[l..r] sorted in nondecreasing order

```
if l<r
{
    s← Partition(A[l..r]) //s is a split position
    QUICKSORT(A[l..s-1])
    QUICKSORT(A[s+1..r])
}
```

```
}
```

{

#### Algorithm :

Partition(A[l..r])

//Partition a subarray by using its first element as its pivot

//Input:A subarray A[l..r] of A[0..n-1],defined by its left and right indices l and r (l<r) //Output:A partition of A[l..r],with the split position returned as this function's value

{

Dept of ISE, SJCIT

```
p \leftarrow A[l]
i \leftarrow l; j \leftarrow r+1 \text{ repeat}
\{ \text{ repeat } i \leftarrow i+1 \text{ until } A[i] \ge p \text{ repeat } j \leftarrow j-1 \text{ until } A[j] <= p \text{ swap}(A[i],A[j]) \} \text{ until } i>= j \text{ swap}(A[i],A[j]) // \text{ undo last swap when } i>= j \text{ swap}(A[l],A[j]) \text{ return } j \text{ swap}(A[l],A[j]) \text{ return } j \}
Complexity: C_{best} (n) = 2 C_{best} (n/2) + n \text{ for } n>1
```

Complexity: Cbest (II) =2 Cbest (II/2) +11 for II>1  $C_{best}$  (1) =0 Cworst (n) (n<sup>2</sup>) Cavg (n)  $\approx 1.38nlog_{2}n$ 

## PROGRAM

/\*Program 4.

{

Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n > 5000 and record the time taken to sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide-and-conquer method works along with its time complexity analysis: worst case, average case and best case.

import java.util.Scanner; import java.util.Arrays; import java.util.Random;

public class QuickSortComplexity

Dept of ISE, SJCIT

```
// for Random entry
       // a[i] = random.nextInt(1000); // generate
   // random numbers – uniform distribution
               // a = Arrays.copyOf(a, n); // keep only non zero elements
               // Arrays.sort(a); // for worst-case time complexity
               System.out.println("Input Array:");
               for (int i = 0; i < n; i++)
                      System.out.print(a[i] + " ");
               // set start time
               long startTime = System.nanoTime();
               QuickSortAlgorithm(0, n - 1);
               long stopTime = System.nanoTime();
               long elapsedTime = stopTime - startTime;
               System.out.println("\nSorted Array:");
               for (int i = 0; i < n; i++)
                      System.out.print(a[i] + " ");
               System.out.println();
       System.out.println("Time Complexity in ms for n=" + n + " is: " + (double) elapsedTime / 1000000);
       public static void QuickSortAlgorithm(int p, int r)
               int i, j, temp, pivot;
               if (p < r)
               {
                      i = p;
                      i = r + 1;
                       pivot = a[p]; // mark first element as pivot
                       while (true)
                       {
                              i++;
                              while (a[i] < pivot && i < r)
                                      i++;
                              j--;
                              while (a[j] > pivot)
                                      j--;
                              if (i < j)
                              {
                                      temp = a[i];
                                      a[i] = a[j];
                                      a[j] = temp;
                               } else
                                      break; // partition is over
Dept of ISE, SJCIT
                                                                                                        Page 46
```



#### NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

2	0	2	0
_	v	_	v

## **EXPERIMENT 5**

## **PROGRAM STATEMENT**

Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n > 5000, and record the time taken to sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide- and-conquer method works along with its time complexity analysis: worst case, average case and best case.

## CONCEPT

Merge sort is a perfect example of a successful application of the divide-and-conquer technique.

- Split array A[1..*n*] in two and make copies of each half in arrays B[1.. *n*/2] and C[1.. *n*/2]
- Sort arrays B and C
- Merge sorted arrays B and C into array A as follows:
  - a) Repeat the following until no elements remain in one of the arrays:
    - (i) compare the first elements in the remaining unprocessed portions of the arrays
    - (ii) copy the smaller of the two into A, while incrementing the index indicating the unprocessed portion of that array
    - b) Once all elements in one of the arrays are processed, copy the remaining unprocessed elements from the other array into A.

#### Algorithm: MergeSort (A [0...n-1])

```
{

Copy A [0... \ln/2^{j}-1] to B [0... \ln/2^{j}-1]

Copy A [\ln/2^{j}-... n-1] to C [0... \ln/2^{j}-1]

MergeSort (B [0... \ln/2^{j}-1])

MergeSort (C [0... \ln/2^{j}-1]) Merge (B, C,A)

}
```

```
Algorithm: Merge (B [0...p-1], C [0...q-1], A [0...p+q-1])

//Merges two sorted arrays into one sorted array.

//Input: Arrays B [0...p-1] and C [0...q-1] both sorted.

//Output: Sorted array A [0...p+q-1] of the elements of B and C.

{
```

```
Dept of ISE, SJCIT
```

}

```
i \leftarrow 0; j\leftarrow 0; k\leftarrow 0
while i\{ if B[i] <= C[j] \\ A[k] \leftarrow B[i]; i\leftarrow i+1 \\ else \\ A[k] \leftarrow C[j]; \\ j\leftarrow j+1 \ k\leftarrow k+1; \}
if i=p
copy C [j...q-1] to A[k...p+q-1]
else
copy B [i...p-1] to A[k...p+q-1]
```

#### **Complexity:**

}

All cases have same efficiency:  $\Theta(n \log n)$ Number of comparisons is close to theoretical minimum for comparison-based sorting:  $\log n$  $! \approx n \lg n - 1.44 n$ 

#### **PROGRAM:**

```
/* Program 5
Sort a given set of n integer elements using Merge Sort method and compute its time
complexity. Run the program for varied values of n > 5000, and record the time taken to
sort. Plot a graph of the time taken versus n on graph sheet. The elements can be read
from a file or can be generated using the random number generator. Demonstrate using
Java how the divide-and-conquer method works along with its time complexity analysis:
worst case, average case and best case.
*/
```

import java.util.Random; import java.util.Scanner;

public class MergeSort2

```
{
```

static final int MAX = 10005; static int[] a = new int[MAX];

```
public static void main(String[] args)
```

Scanner input = **new** Scanner(System.*in*); System.*out*.print("Enter Max array size: "); **int** n = input.nextInt(); Random random = **new** Random(); //20 System.*out*.println("Enter the array elements: ");

Dept of ISE, SJCIT

```
Design & Analysis of Algorithm Lab Manual
                                                                                          2020
              for (int i = 0; i < n; i++)
                      a[i] = input.nextInt(); // for keyboard entry
               a[i] = random.nextInt(1000); // generate random numbers -
// uniform distribution
              for (int i = 0; i < n; i++)
              System.out.println(a[i] + " "); // display unsorted numbers(input values)
              long startTime = System.nanoTime();
              MergeSortAlgorithm(0, n - 1);
              long stopTime = System.nanoTime();
              long elapsedTime = stopTime - startTime;
              System.out.println("Time Complexity (ms) for n = " +
              n + " is : " + (double) elapsedTime / 1000000);
              System.out.println("Sorted Array (Merge Sort):");
              for (int i = 0; i < n; i++)
                      System.out.print(a[i] + " ");
              input.close();
       }
       public static void MergeSortAlgorithm(int low, int high)
              int mid;
              if (low < high)
              {
                      mid = (low + high) / 2;
                      MergeSortAlgorithm(low, mid);
                      MergeSortAlgorithm(mid + 1, high);
                      Merge(low, mid, high);
               }
       }
       public static void Merge(int low, int mid, int high)
              int[] b = new int[MAX];
              int i, h, j, k;
              h = i = low;
              j = mid + 1;
              while ((h <= mid) && (j <= high))
                      if (a[h] < a[j])
                             b[i++] = a[h++];
                      else
                             b[i++] = a[i++];
              if (h > mid)
                      for (k = j; k <= high; k++)
                             b[i++] = a[k];
              else
```



	Design & Analysis of Algorithm Lab Manual	2020
NOTES AND OUTPUT SPACE		

2	0	2	0
	υ		υ

## **EXPERIMENT 6A**

### **PROGRAM STATEMENT**

Implement in Java, the 0/1 Knapsack problem using Dynamic Programming method

## CONCEPT

Dynamic-Programming Solution to the 0-1 Knapsack Problem: Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization. Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of the previously solved sub-problems. The solutions of sub-problems are combined in order to achieve the best solution. $w_i$ .In the knapsack problem we are given a set of n items, where each item i is specified by a size and a value  $v_i$ . We are also given a size bound W (the size of our knapsack). The goal is to find the subset of items of maximum total value such that sum of their sizes is

The goal is to find the subset of items of maximum total value such that sum of their sizes is at most W (they all fit into the knapsack).Now, instead of being able to take a certain weight of an item, you can only either take the item or not take the item.

Algorithm:

W<sub>i</sub>V<sub>i</sub>(n items, W weight of sack)

//Input: n and W – all integers

//Output: V(n,W)

//Initialization of first column and first row elements

Repeat for i = 0 to n

set V(i,0) = 0

• Repeat for j = 0 to W

Set V(0,j) = 0

//complete remaining entries row by row

• Repeat for i = 1 to n

```
if ( > j) V(i,j) = V(i-1,j)
```

Dept of ISE, SJCIT

Print V(n,W)

## PROGRAM

/\* Knapsack DP \*/

import java.util.Scanner;

public class KnapsackDP

{

```
static final int MAX = 20; // max. no. of objects
static int w[];
                 // weights 0 to n-1
static int p[]; // profits 0 to n-1
static int n;
                // no. of objects
static int M;
                      // capacity of Knapsack
static int V[][]; // DP solution process - table
static int Keep[][]; // to get objects in optimal solution
public static void main(String args[ ])
{
       w = \text{new int}[MAX];
       p = \text{new int}[MAX];
       V = new int [MAX][MAX];
       Keep = new int[MAX][MAX];
       int optsoln;
       ReadObjects();
       for (int i = 0; i <= M; i++)
               V[0][i] = 0;
       for (int i = 0; i <= n; i++)
               V[i][0] = 0;
       optsoln = Knapsack();
       System.out.println("Optimal solution = " + optsoln);
}
static int Knapsack()
       int r; // remaining Knapsack capacity
       for (int i = 1; i <= n; i++)
               for (int j = 0; j \le M; j ++)
                       if ((w[i] \le j) && (p[i] + V[i - 1][j - w[i]] > V[i - 1][j]))
                               V[i][j] = p[i] + V[i - 1][j - w[i]];
                               Keep[i][j] = 1;
```

Dept of ISE, SJCIT

```
else
                               {
                                       V[i][j] = V[i - 1][j];
                                      Keep[i][j] = 0;
                               }
               // Find the objects included in the Knapsack
               \mathbf{r} = \boldsymbol{M};
               System.out.println("Items = ");
               for (int i = n; i > 0; i--) // start from Keep[n,M]
                       if (Keep[i][r] == 1)
                       {
                               System.out.println(i + " ");
                               r = r - w[i];
               System.out.println();
               return V[n][M];
       }
       static void ReadObjects()
       {
               Scanner scanner = new Scanner(System.in);
               System.out.println("Knapsack Problem - Dynamic Programming Solution: ");
               System.out.println("Enter the max capacity of knapsack: ");
               M = \text{scanner.nextInt()};
               System.out.println("Enter number of objects: ");
               n = scanner.nextInt();
               System.out.println("Enter Weights: ");
               for (int i = 1; i <= n; i++)
                       w[i] = scanner.nextInt();
               System.out.println("Enter Profits: ");
               for (int i = 1; i <= n; i++)
                       p[i] = scanner.nextInt();
               scanner.close();
        }
}
OUTPUT
Knapsack Problem - Dynamic Programming Solution:
Enter the max capacity of knapsack:
15
Enter number of objects:
4
Enter Weights:
2
5
```

Dept of ISE, SJCIT

	Design & Analysis of Algorithm Lab Manual	2020
7 3 Enter Profits: 23 78 34 12 Items = 3 2 1		
Optimal solution = 135		
NOTES AND SPACE AREA		

2	0	2	0
_	v	_	v

2	0	2	0
	υ		υ

## **EXPERIMENT 6B**

### **PROGRAM STATEMENT**

Implement in Java, the Fractional Knapsack problem using Greedy method

## CONCEPT

Greedy Solution to the Fractional Knapsack Problem

The basic idea of greedy approach is to calculate the ratio value/weight for each item and sort the item on basis of this ratio. Then take the item with highest ratio and add them until we can't add the next item as whole and at the end add the next item as much as we can. This will always be optimal solution of this problem.

#### Algorithm:

- Assume knapsack holds weight W and items have value vi and weight wi
- Rank items by value/weight ratio:  $v_i$  /  $w_i$  Thus:  $v_i$  /  $w_i \ge v_j$  /  $w_j$ , for all  $i \le j$
- Consider items in order of decreasing ratio
- Take as much of each item as possible

## PROGRAM

```
/* Knapsack Greedy */
import java.util.Scanner;
class KObject
                                     // Knapsack object details
{
       float w:
       float p;
       float r;
}
public class KnapsackGreedy2
{
       static final int MAX = 20; // max. no. of objects
                                            // no. of objects
       static int n;
                                            // capacity of Knapsack
       static float M;
       public static void main(String args[])
              Scanner scanner = new Scanner(System.in);
              System.out.println("Enter number of objects: ");
              n = \text{scanner.nextInt()};
              KObject[] obj = new KObject[n];
              for(int i = 0; i<n;i++)
                      obj[i] = new KObject();// allocate memory for members
              ReadObjects(obj);
```

Dept of ISE, SJCIT

```
Knapsack(obj);
               scanner.close();
        }
       static void ReadObjects(KObject obj[])
       {
               KObject temp = new KObject();
               Scanner scanner = new Scanner(System.in);
               System.out.println("Enter the max capacity of knapsack: ");
               M = scanner.nextFloat();
               System.out.println("Enter Weights: ");
               for (int i = 0; i < n; i++)
                       obj[i].w = scanner.nextFloat();
               System.out.println("Enter Profits: ");
               for (int i = 0; i < n; i++)
                       obj[i].p = scanner.nextFloat();
               for (int i = 0; i < n; i++)
                       obj[i].r = obj[i].p / obj[i].w;
               // sort objects in descending order, based on p/w ratio
               for(int i = 0; i<n-1; i++)
                       for(int j=0; j<n-1-i; j++)
                              if(obj[j].r < obj[j+1].r)
                                      temp = obj[j];
                                      obj[j] = obj[j+1];
                                      obj[j+1] = temp;
               scanner.close();
       }
       static void Knapsack(KObject kobj[])
       {
               float x[] = new float[MAX];
               float totalprofit;
               int i:
               float U; // U place holder for M
               U = M;
               totalprofit = 0;
               for (i = 0; i < n; i++)
                       x[i] = 0;
               for (i = 0; i < n; i++)
Dept of ISE, SJCIT
```
```
if (kobj[i].w > U)
                               break;
                       else
                       {
                               x[i] = 1;
                               totalprofit = totalprofit + kobj[i].p;
                               U = U - kobj[i].w;
                       }
                }
               System.out.println("i = " + i);
               if (i < n)
                       x[i] = U / kobj[i].w;
               totalprofit = totalprofit + (x[i] * kobj[i].p);
               System.out.println("The Solution vector, x[]: ");
               for (i = 0; i < n; i++)
                       System.out.print(x[i] + " ");
               System.out.println("\nTotal profit is = " + totalprofit);
        }
}
/*Enter number of objects:
4
Enter the max capacity of knapsack:
20
Enter Weights:
5
9
12
3
Enter Profits:
45
78
45
65
i = 3
The Solution vector, x[]:
1.0 1.0 1.0 0.25
Total profit is = 199.25
*/<u>RUN 2</u>
Enter number of objects:
5
Enter the max capacity of knapsack:
25
Enter Weights:
12
Dept of ISE, SJCIT
```

10 9 6 14 Enter Profits: 45 67 12 78 54 i = 2 The Solution vector, x[]: 1.0 1.0 0.64285713 0.0 0.0 Total profit is = 179.7143

## NOTES AND SPACE AREA

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

## **EXPERIMENT 7**

## **PROGRAM STATEMENT**

From a given vertex in weighted connected graph, find shortest path to other vertices using Dijkstras algorithm. Write the program in java.

## CONCEPT

It is a greedy algorithm which finds the shortest path from the source vertex to all other vertices of the graph.

Steps

- 1. Input a cost matrix for graph. Read the source vertex and n from user
- 2. Create the array d [1...n] which stores the distance from source vertex to all other vertices of graph. Initialize distance to source vertex as 0(i.e. d [source] =0) and remaining vertices as 999.
- 3. Create the array visited [1...n] which keeps track of all the visited nodes. Visit the source vertex and initialize visited [source] =1.
- 4. For all adjacent vertices[v<sub>i</sub>,v<sub>i+1</sub>,...] for source vertex calculate distance using formula d[v<sub>i</sub>]=min(d[v<sub>i</sub>], d[source]+ cost[source][v1]). Update the array d [1...n].
- 5. For all adjacent vertices find vertex v<sub>i</sub> which has minimum distance from source vertex.
- 6. Initialize source =  $v_i$ . Repeat the steps 4, 5 until there all some vertices which are unvisited.
- 7. Stop

Example



Design & Analysis of Algorithm Lab Manual

## 2020

|--|

d[A]	d[B]	d[C]	d[D]	d[E]
0	3	7	1	2
visit	visit	visit	visit	visit
ed[A	ed[B	ed[C	ed[D	ed[E
]	]	]	]	]
1	1	0	1	1

## Step 4

Source	Adjacent	d[v] = min(d[v], d[u] + cost[u])	Find d[v]	New	Visited
Vertex	vertices	[v])	such that	sourc	vertex
	which		it is	e	
	are		minimum		
	unvisite				
	d				
В	С	d[C]=min(d[C],d[B] + cost[C])			
		[B])	v=C	C	C
		$=\min(7,3+5)=7$			

d[A]	d[B]	d[C]	d[D]	d[E]
0	3	7	1	2
visit	visit	visit	visit	visit
ed[A	ed[B	ed[C	ed[D	ed[E
]	]	]	]	]
1	1	1	1	1

Hence distance from A to A = 0Hence distance from A to B = 3Hence distance from A to C = 7Hence distance from A to D = 1Hence distance from A to E = 2

# Design & Analysis of Algorithm Lab Manual **2020**

d[D]=n	nin(999,1)=1		

d[A]	d[B]	d[C]	d[D]	d[E]
0	6	999	1	999
visit	visit	visit	visit	visit
ed[A	ed[B	ed[C	ed[D	ed[E
]	]	]	]	]
1	0	0	1	0

#### Step 2:

Step 2.		14 A A A A A A A A A A A A A A A A A A A			
Source	Adjacent	d[v] = min(d[v], d[u] + cost[u])	Find d[v]	New	Visited
Vertex	vertices	[v])	such that	sourc	vertex
	which		it is	e	
	are		minimum		
	unvisite				
	d				
D	B,E	d[B]=min(d[B],d[D] + cost[B]			
		[D])	v=E	E	E
		$=\min(6,1+2)=3$			
		d[E]=min(d[E],d[D]+cost[E]			
		[D])			
		$=\min(999,1+1)=2$			

d[A]	d[B]	d[C]	d[D]	d[E]
0	3	999	1	2
visit	visit	visit	visit	visit
ed[A	ed[B	ed[C	ed[D	ed[E
]	]	]	]	]
1	0	0	1	1

## Step 3

Source	Adjacent	d[v] = min(d[v], d[u] + cost[u])	Find d[v]	New	Visited
Vertex	vertices	[v])	such that	sourc	vertex
	which	TOOR - PROF.	it is	e	
	are		minimum		
	unvisite				
	d				
E	B,C	d[B]=min(d[B],d[E] + cost[B])			
		[E])	v=B	В	В
		$=\min(3,2+2)=3$			
		d[C]=min(d[C],d[E]+cost[C]			
		[E])			

```
Hence distance from A to A = 0
Hence distance from A to B = 3
Hence distance from A to C = 7
Hence distance from A to D = 1
Hence distance from A to E = 2
    PROGRAM
/* Dijikstra's */
import java.util.*;
public class DijkstrasClass
{
       final static int MAX = 20;
       final static int infinity = 9999;
       static int n;// No. of vertices of Gstatic int a[][];// Cost matrix
       static Scanner scan = new Scanner(System.in);
       public static void main(String[] args)
       ł
               ReadMatrix();
               int s = 0;
                                      // starting vertex
               System.out.println("Enter starting vertex: ");
               s = scan.nextInt();
               Dijkstras(s); // find shortest path
       }
       static void ReadMatrix()
       {
               a = new int[MAX][MAX];
               System.out.println("Enter the number of vertices:");
               n = scan.nextInt();
               System.out.println("Enter the cost adjacency matrix:");
               for (int i = 1; i <= n; i++)
                       for (int j = 1; j <= n; j++)
                              a[i][j] = scan.nextInt();
       }
       static void Dijkstras(int s)
       {
               int S[] = new int[MAX];
```

```
int d[] = new int[MAX];
       int u, v;
       int i;
       for (i = 1; i <= n; i++)
        {
               S[i] = 0;
               d[i] = a[s][i];
        }
        S[s] = 1;
       d[s] = 1;
       i = 2;
        while (i <= n)
        {
               u = Extract_Min(S, d);
               S[u] = 1;
               i++;
               for (v = 1; v <= n; v++)
                {
                       if (((d[u] + a[u][v] < d[v]) && (S[v] == 0)))
                               d[v] = d[u] + a[u][v];
                }
        for (i = 1; i <= n; i++)
               if (i != s)
                       System.out.println(i + ":" + d[i]);
}
static int Extract_Min(int S[], int d[])
{
        int i, j = 1, min;
        min = infinity;
        for (i = 1; i <= n; i++)
        {
               if ((d[i] < min) && (S[i] == 0))
                {
                       min = d[i];
                       j = i;
                }
        }
        return (j);
}
```

}

## **OUTPUT**

Enter the number of vertices: 4 Enter the cost adjacency matrix: 999 1 6 4 999 999 999 2 999 3 999 999 999 999 1 999 Enter starting vertex: 1 2:1 3:4

4:3

## NOTES AND OUPUT AREA

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

### **EXPERIMENT 8**

## **PROGRAM STATEMENT**

Find Minimum Cost Spanning Tree of a given connected undirected graph using **Kruskal's** algorithm. Implement the program in Java. Use Union-Find algorithm in your program.

## CONCEPT

A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected.

By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.





We found three spanning trees off one complete graph. A complete undirected graph can have maximum  $\mathbf{n}^{\mathbf{n}\cdot\mathbf{2}}$  number of spanning trees, where **n** is the number of nodes. In the above addressed example,  $\mathbf{3}^{\mathbf{3}-\mathbf{2}} = \mathbf{3}$  spanning trees are possible.

Dept of ISE, SJCIT

#### **General Properties of Spanning Tree**

We now understand that one graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G –

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of graph G, have the same number of edges and vertices.
- The spanning tree does not have any cycle (loops).
- Removing one edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- Adding one edge to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

#### Minimum Spanning Tree

Given a connected and undirected graph, a *spanning tree* of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A *minimum spanning tree (MST)* or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and doesnot violate MST properties.

To understand Kruskal's algorithm let us consider the following example -



#### Dept of ISE, SJCIT

#### **Step 1 - Remove all loops and Parallel Edges**

Remove all loops and parallel edges from the given graph.



In case of parallel edges, keep the one which has the least cost associated and remove all others.



#### Step 2 - Arrange all edges in their increasing order of weight

The next step is to create a set of edges and weight, and arrange them in an ascending order of weightage (cost).

#### Step 3 - Add the edge which has the least weightage

Now we start adding edges to the graph beginning from the one which has the least weight. Throughout, we shall keep checking that the spanning properties remain intact. In case, by adding one edge, the spanning tree property does not hold then we shall consider not to include the edge in the graph.

2020



We observe that edges with cost 5 and 6 also create circuits. We ignore them and move on.



Now we are left with only one node to be added. Between the two least cost edges available 7 and 8, we shall add the edge with cost 7.



By adding edge S,A we have included all the nodes of the graph and we now have minimum cost spanning tree.

### PROGRAM

/\* 8. <u>Kruskals</u> \*/

import java.util.Scanner;

public class KruskalsClass
{

final static int MAX = 20; static int n; // No. of vertices of G static int cost[ ][ ]; // Cost matrix static Scanner scan = new Scanner(System.in);

```
public static void main(String[ ] args)
```

Dept of ISE, SJCIT

```
{
        ReadMatrix();
        Kruskals();
}
static void ReadMatrix()
ł
        int i, j;
        cost = new int[MAX][MAX];
        System.out.println("Implementation of Kruskal's algorithm");
        System.out.println("Enter the no. of vertices");
        n = scan.nextInt();
        System.out.println("Enter the cost adjacency matrix");
        for (i = 1; i <= n; i++)
        {
                for (j = 1; j <= n; j++)
                ł
                        cost[i][j] = scan.nextInt();
                        if (cost[i][j] == 0)
                                cost[i][j] = 999;
                }
        }
}
static void Kruskals()
{
        int a = 0, b = 0, u = 0, v = 0, i, j, ne = 1, min, mincost = 0;
        System.out.println("The edges of Minimum Cost Spanning Tree are");
        while (ne < n)
        ł
                for (i = 1, min = 999; i <= n; i++)
                {
                        for (j = 1; j <= n; j++)
                                if (cost[i][j] < min)
                                {
                                        min = cost[i][j];
                                        a = u = i;
                                        \mathbf{b} = \mathbf{v} = \mathbf{j};
                                }
                        }
```

Dept of ISE, SJCIT

```
}
                       \mathbf{u} = find(\mathbf{u});
                       \mathbf{v} = find(\mathbf{v});
                       if (u != v)
                       {
                               uni(u, v);
                               System.out.println(ne++ + "edge (" + a + "," + b + ") =" + min);
                               mincost += min;
                        }
                       cost[a][b] = cost[b][a] = 999;
               System.out.println("Minimum cost :" + mincost);
        }
       static int find(int i)
       {
               int parent[] = new int[9];
               while (parent[i] == 1)
                       i = parent[i];
               return i;
        }
       static void uni(int i, int j)
       {
               int parent[] = new int[9];
               parent[j] = i;
        }
OUTPUT
Implementation of Kruskal's algorithm
Enter the no. of vertices
4
Enter the cost adjacency matrix
999164
999 999 999 2
999 3 999 999
999 999 1 999
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) = 1
2edge(4,3) = 1
3edge (2,4) =2
Minimum cost :4
```

Dept of ISE, SJCIT

}

NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

### **EXPERIMENT 9**

## **PROGRAM STATEMENT**

Find Minimum Cost Spanning Tree of a given connected undirected graph using **Prim's** algorithm.

### CONCEPT

Prim's algorithm to find minimum cost spanning tree uses the greedy approach. Prim's algorithm shares a similarity with the **shortest path first** algorithms.

Prim's algorithm, in contrast with Kruskal's algorithm, treats the nodes as a single tree and keeps on adding new nodes to the spanning tree from the given graph.

To contrast with Kruskal's algorithm and to understand Prim's algorithm better, we shall use the same example -



#### Step 1 - Remove all loops and parallel edges



Remove all loops and parallel edges from the given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.



#### Step 2 - Choose any arbitrary node as root node

In this case, we choose S node as the root node of Prim's spanning tree. This node is arbitrarily chosen, so any node can be the root node. One may wonder why any video can be root node. So the answer is, in the spanning tree all the nodes of a graph are included and because it is connected then there must be at least one edge, which will join it to the rest of the tree.

### Step 3 - Check outgoing edges and select the one with less cost

After choosing the root node **S**, we see that S,A and S,C are two edges with weight 7 and 8, respectively. We choose the edge S,A as it is lesser than the other.



Now, the tree S-7-A is treated as one node and we check for all edges going out from it. We select the one which has the lowest cost and include it in the tree.



After this step, S-7-A-3-C tree is formed. Now we'll again treat it as a node and will check all the edges again. However, we will choose only the least cost edge. In this case, C-3-D is the new edge, which is less than other edges' cost 8, 6, 4, etc.





After adding node **D** to the spanning tree, we now have two edges going out of it having the same cost, i.e. D-2-T and D-2-B. Thus, we can add either one. But the next step will again yield edge 2 as the least cost. Hence, we are showing a spanning tree with both edges included.



# PROGRAM

/\* 9. Prim's \*/

import java.util.Scanner;

#### public class PrimsClass

{

```
final static int MAX = 20;
static int n; // No. of vertices of G
static int cost[][]; // Cost matrix
static Scanner scan = new Scanner(System.in);
```

```
public static void main(String[] args)
{
    ReadMatrix();
    Prims();
    }
    static void ReadMatrix()
    {
```

Dept of ISE, SJCIT

```
int i, j;
                        cost = new int[MAX][MAX];
                        System.out.println("\n Enter the number of nodes:");
                        n = scan.nextInt();
                        System.out.println("\n Enter the adjacency matrix:\n");
                        for (i = 1; i \le n; i++)
                        for (j = 1; j \le n; j++)
                         {
                                cost[i][j] = scan.nextInt();
                                if (cost[i][j] == 0)
                                        cost[i][j] = 999;
                        }
                }
                static void Prims()
                 {
                        int visited[] = new int[10];
                        int ne = 1, i, j, min, a = 0, b = 0, u = 0, v = 0;
                        int mincost = 0;
                        visited[1] = 1;
                        while (ne < n)
                         {
                                for (i = 1, min = 999; i \le n; i++)
                                         for (j = 1; j <= n; j++)
                                                 if (cost[i][j] < min)
                                                         if (visited[i] != 0)
                                                          {
                                                                 min = cost[i][j];
                                                                 a = u = i;
                                                                 \mathbf{b} = \mathbf{v} = \mathbf{j};
                                if (visited[u] == 0 \parallel visited[v] == 0)
                                 {
                                   System.out.println("Edge" + ne++ + ":(" + a + "," + b + ")" + "cost:" +
    min);
                                        mincost += min;
                                         visited[b] = 1;
                                 }
                                cost[a][b] = cost[b][a] = 999;
                         ł
                        System.out.println("\n Minimun cost" + mincost);
Dept of ISE, SJCIT
                                                                                                             Page 84
```

}

## OUTPUT

}

Enter the number of nodes: 4

Enter the adjacency matrix:

Minimun cost10

### NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

## **EXPERIMENT 10A**

## **PROGRAM STATEMENT**

Implement All-Pairs Shortest Paths problem using Floyd's algorithm.

## CONCEPT

The Floyd Warshall Algorithm is for solving the All Pairs Shortest Path problem. The problem is to find shortest distances between every pair of vertices in a given edge weighted directed Graph. Following Algorithm is used to find shortest path: given: w[i][j] is the wieghted matrix that contains weights k is the iteration variable for intermidiate vertices Following is the algorithm function floyd for i=1 to n for j=1 to n w[i][j]=infinity//if entered w[i][j] == 0 for each edge (i,j) in E dmatrix[i][i] = amatrix[i][j]://copy the entered weighted matrix to distance matrix //calculate distance matrix for k=1 to n for j=1 to n if (dmatrix[i][k] + dmatrix[k][j]

## PROGRAM

import java.util.Scanner;
public class FloydsClass
{

```
static final int MAX = 20; // max. size of cost matrix
static int a[][]:
                                      // cost matrix
                                      // actual matrix size
static int n;
public static void main(String args[])
       a = \text{new int}[MAX][MAX];
       ReadMatrix();
       Floyds();
                                              // find all pairs shortest path
       PrintMatrix();
}
static void ReadMatrix()
       System.out.println("Enter the number of vertices\n");
       Scanner scanner = new Scanner(System.in);
       n = \text{scanner.nextInt()};
       System.out.println("Enter the Cost Matrix (999 for infinity) \n");
       for (int i = 1; i <= n; i++)
```

Dept of ISE, SJCIT

```
{
                       for (int j = 1; j <= n; j++)
                              a[i][j] = scanner.nextInt();
               }
               scanner.close();
       }
       static void Floyds()
       {
               for (int k = 1; k <= n; k++)
               {
                       for (int i = 1; i <= n; i++)
                              for (int j = 1; j <= n; j++)
                                      if ((a[i][k] + a[k][j]) < a[i][j])
                                              a[i][j] = a[i][k] + a[k][j];
               }
       }
       static void PrintMatrix()
       {
               System.out.println("The All Pair Shortest Path Matrix is:\n");
               for(int i=1; i<=n; i++)
               {
                       for(int j=1; j<=n; j++)
                              System.out.print(a[i][j] + "\t");
                       System.out.println("\n");
               }
       }
}
OUTPUT
Enter the number of vertices
4
Enter the Cost Matrix (999 for infinity)
999146
999 999 7 4
2 999 999 3
999 999 999 999
The All Pair Shortest Path Matrix is:
6
       1
               4
                       5
Dept of ISE, SJCIT
```

				Design & Analysis of Algorithm Lab Manual	2020
	10	7	4		
9	10	1	4		
2	3	0	3		
999	999	999	999		
NOT	ES ANI	D OUT	PUT SPACE		

2	0	2	0
_	v	_	v

## **EXPERIMENT 10B**

## PROGRAM STATEMENT

Write a Java program to implement 4 Salesman Problem using Dynamic programming

## CONCEPT

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Note the difference between <u>Hamiltonian Cycle</u> and TSP. The Hamiltonian cycle problem is to find if there exist a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle.



For example, consider the above graph. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is 80.

### PROGRAM

/\* 10b. TSP - DP \*/

import java.util.Scanner;

public class TravSalesPerson

static int MAX = 100; static final int infinity = 999;

public static void main(String args[])

Dept of ISE, SJCIT

{

```
Design & Analysis of Algorithm Lab Manual
                                                                                             2020
       {
               int cost = infinity;
               int c[][] = new int[MAX][MAX];
                                                     // cost matrix
               int tour[] = new int[MAX];
                                                      // optimal tour
                                                                     // max. cities
               int n:
               System.out.println("Travelling Salesman Problem using Dynamic Programming\n");
               System.out.println("Enter number of cities: ");
               Scanner scanner = new Scanner(System.in);
               n = scanner.nextInt();
               System.out.println("Enter Cost matrix:\n");
               for (int i = 0; i < n; i++)
                       for (int j = 0; j < n; j++)
                       {
                               c[i][j] = scanner.nextInt();
                               if (c[i][j] == 0)
                                      c[i][i] = 999;
               for (int i = 0; i < n; i++)
                       tour[i] = i;
               cost = tspdp(c, tour, 0, n);
               // print tour cost and tour
               System.out.println("Minimum Tour Cost: " + cost);
               System.out.println("\nTour:");
               for (int i = 0; i < n; i++)
                       System.out.print(tour[i] + " -> ");
               System.out.println(tour[0] + "\n");
               scanner.close();
       }
       static int tspdp(int c[][], int tour[], int start, int n)
       {
               int i, j, k;
               int temp[] = new int[MAX];
               int mintour[] = new int[MAX];
               int mincost;
               int cost:
               if (start == n - 2)
                       return c[tour[n - 2]][tour[n - 1]] + c[tour[n - 1]][0];
               mincost = infinity;
               for (i = start + 1; i < n; i++)
               {
                       for (j = 0; j < n; j++)
                               temp[j] = tour[j];
                       temp[start + 1] = tour[i];
Dept of ISE, SJCIT
                                                                                                       Page 92
```

```
temp[i] = tour[start + 1];
if (c[tour[start]][tour[i]] + (cost = tspdp(c, temp, start + 1, n)) < mincost) {
            mincost = c[tour[start]][tour[i]] + cost;
            for (k = 0; k < n; k++)
                 mintour[k] = temp[k];
            }
            for (i = 0; i < n; i++)
            tour[i] = mintour[i];
            return mincost;
        }
}
```

}

### **OUTPUT**

Travelling Salesman Problem using Dynamic Programming

Enter number of cities: 4 Enter Cost matrix:

0 1 2 4 1 0 999 1 2 999 0 1 4 1 1 0 Minimum Tour Cost: 5

Tour: 0 -> 1 -> 3 -> 2 -> 0

## NOTES AND OUTPUT SPACE

Dept of ISE, SJCIT

2	0	2	0
_	v	_	v

## **EXPERIMENT 11**

## **PROGRAM STATEMENT**

Find a subset of a given set  $S = \{sl, s2, ..., sn\}$  of n positive integers whose sum is equal to a given positive integer d. For example, if  $S = \{1, 2, 5, 6, 8\}$  and d = 9 there are two solutions  $\{1,2,6\}$  and  $\{1,8\}$ . A suitable message is to be displayed if the given problem instance doesn't have a solution.

## CONCEPT

Subset-Sum Problem is to find a subset of a given set  $S = \{s1, s2... sn\}$  of n positive integers whose sum is equal to a given positive integer d. It is assumed that the set's elements are sorted in increasing order. The state-space tree can then be constructed as a binary tree and applying backtracking algorithm, the solutions could be obtained. Some instances of the problem may have no solutions.

#### PROGRAM

{

import java.util.Scanner;

```
public class SumOfsubset
```

```
final static int MAX = 10;
static int n;
static int S[]:
static int soln[];
static int d;
public static void main(String args[])
{
       S = new int[MAX];
       soln = new int[MAX];
       int sum = 0;
       Scanner scanner = new Scanner(System.in);
       System.out.println("Enter number of elements: ");
       n = \text{scanner.nextInt()};
       System.out.println("Enter the set in increasing order: ");
       for (int i = 1; i <= n; i++)
               S[i] = \text{scanner.nextInt}();
       System.out.println("Enter the max. subset value(d): ");
       d = \text{scanner.nextInt()};
       for (int i = 1; i <= n; i++)
               sum = sum + S[i];
       if (sum < d || S[1] > d)
               System.out.println("No Subset possible");
```

Dept of ISE, SJCIT

```
else
               SumofSub(0, 0, sum);
        scanner.close();
}
static void SumofSub(int i, int weight, int total)
{
        if (promising(i, weight, total) == true)
               if (weight == d) {
                       for (int j = 1; j <= i; j++)
                        ł
                               if (soln[j] == 1)
                                        System.out.print(S[j] + " ");
                       System.out.println();
                }
               else
                {
                       soln[i + 1] = 1;
                       SumofSub(i + 1, weight + S[i + 1], total - S[i + 1]);
                       soln[i + 1] = 0;
                       SumofSub(i + 1, weight, total - S[i + 1]);
                }
}
static boolean promising(int i, int weight, int total)
{
        return ((weight + total >= d) && (weight == d \parallel weight + S[i + 1] \leq d);
}
```

### **OUTPUT**

}

Enter number of elements: 4 Enter the set in increasing order: 4 6 9 12 Enter the max. subset value(d): 13 4 9

Dept of ISE, SJCIT
	Design & Analysis of Algorithm Lab Manual	2020
NOTES AND OUTPUT SPACE		
Dept of ISE, SJCIT		Page 98

2	0	2	0
_	v	_	v

Dept of ISE, SJCIT

### **EXPERIMENT 12**

# PROGRAM STATEMENT

Design and implement in java to find all Hamiltonian Cycles in a connected undirected graph G with n vertices using backtracking principle.

# CONCEPT

A *Hamiltonian cycle*, *Hamiltonian circuit* is a <u>cycle</u> that visits each vertex exactly once (except for the vertex that is both the start and end, which is visited twice). A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**.

Following is one way of checking whether a graph contains a Hamiltonian Cycle or not. A Hamiltonian Path in a graph having N vertices is nothing but a permutation of the vertices of the graph  $[v_1, v_2,$ 

 $v_3, \dots, v_{N-1}, v_N$ ], such that there is an edge between  $v_i$  and  $v_{i+1}$  where  $1 \le i \le N-1$ . So it can be checked for all permutations of the vertices whether it represents a Hamiltonian Cycle or not.



#### INPUT

# PROGRAM

/\* 12. <u>Hamiltonian</u> cycle \*/

import java.util.Scanner;

```
public class Hamiltonian
```

```
{
```

```
boolean found = false;
int G[][];// = new int[n + 1][n + 1];
int x[];// = new int[n + 1];
int n;
```

public static void main(String args[])

```
{
```

Hamiltonian hamiltonian = new Hamiltonian(); hamiltonian.getData();

Dept of ISE, SJCIT

```
System.out.println("\nSolution:");
                hamiltonian.HamiltonianMethod(2);
                hamiltonian.printNoSlnPossible();
        }
        public void printNoSlnPossible()
                if (found == false)
                       System.out.println("No Solution possible!");
        }
        public void getData()
                Scanner scanner = new Scanner(System.in);
                System.out.println("\t\t\t\tHamiltonian Cycle");
                System.out.print("\nEnter the number of the vertices: ");
                // <u>int</u> n;
                n = scanner.nextInt();
                G = new int[n + 1][n + 1];
                \mathbf{x} = \mathbf{new int}[\mathbf{n} + 1];
                System.out.print("\nIf edge between the following vertices enter 1 else 0:\n");
                for (int i = 1; i <= n; i++)
                       for (int j = 1; j <= n; j++)
                               if ((i != j) \&\& (i < j))
                                       System.out.print(i + " and " + j + ": ");
                                        G[i][i] = G[i][j] = scanner.nextInt();
                               if (i == j)
                                       G[i][j] = 0;
                        }
                for (int i = 1; i <= n; i++)
                       x[i] = 0;
                x[1] = 1;
                scanner.close();
        }
        void HamiltonianMethod(int k)
        {
                while (true)
                {
                       NextValue(k, G, x, n);
                       if (x[k] == 0)
Dept of ISE, SJCIT
```

```
return;
                       if (k == n)
                       {
                               for (int i = 1; i <= k; i++)
                                       System.out.print(x[i] + " ");
                               System.out.println(x[1]);
                               System.out.println();
                               found = true;
                               return;
                       }
                       else
                               HamiltonianMethod(k + 1);
               }
        }
       void NextValue(int k, int G[][], int x[], int n)
        {
               while (true)
               {
                       x[k] = (x[k] + 1) \% (n + 1);
                       if (x[k] == 0)
                               return;
                       if (G[x[k - 1]][x[k]] != 0)
                       {
                               int j;
                               for (j = 1; j < k; j++)
                                       if (x[k] == x[j])
                                               break;
                               if (j == k)
                                       if ((k < n) \parallel ((k == n) \&\& G[x[n]][x[1]] != 0))
                                               return;
                       }
               }
        }
OUTPUT Hamiltonian Cycle
Enter the number of the vertices: 4
```

If edge between the following vertices enter 1 else 0: 1 and 2: 1 1 and 3: 1 1 and 4: 1 Dept of ISE, SJCIT

}

Dept of ISE, SJCIT

	Design & Analysis of Algorithm Lab Manual	2020
2 and 3: 1 2 and 4: 1 3 and 4: 1		
Solution: 1 2 3 4 1		
1 3 2 4 1		
1 4 2 3 1		
NOTES AND OUTPUT SPACE		
Dept of ISE, SJCIT		Page 104

2	0	2	0
_	v	_	v

Dept of ISE, SJCIT

#### **ADDITIONAL PROGRAMS**

1) All pair shortest paths problem using Floyd's algorithm. Parallelize this algorithm , implement it using OPenMP and determine the speed-up achieved.

```
# include< stdio.h>
# include <omp.h>
void floyds(int a[10][10],int n);
int min(int a,int b);
int main()
{
   int thread_id,i,j,n,a[10][10];
  double starttime, endtime;
  system("clear");
   printf(" Enter the no of vertices");
    scanf("%d",&n);
    printf("Enter the weight matrix n");
    printf(" Enter 0 for self loops and 999 for no edge\n");
    for ( i=1; i<=n; i++ )
       for ( j=1; j<=n; j++ )
         {
           scanf("%d",&a[i][j]);
       starttime=omp_get_wtime();
      floyds(a,n);
      endtime= omp_get_wtime();
   printf("Speed up achieved is %f",endtime-starttime);
 }
int min( int a, int b)
    return(a<b?a:b);
void floyds(int a[10][10],int n)
{
   int b[10][10],i,j,k,thread_id;
     for ( i=1; i<=n; i++ )
       for ( j=1; j<=n; j++ )
       b[i][j]=a[i][j];
       omp_set_num_threads(2);
       # pragma omp parallel for shared(b) private(i,j,k)
      for ( i=1; i<=n; i++ )
       ł
          for ( j=1; j<=n; j++ )
          for(k=1; k<=n; k++)
          {
               thread_id=omp_get_thread_num();
```

Dept of ISE, SJCIT

•

```
b[i][j]=min(b[i][j],b[i][k]+b[k][j]);
printf("Thread %d:b[%d][%d]=%d\n",thread_id,i,j,b[i][j]);
}
printf("All pairs shortest path \n");
for ( i=1; i<=n; i++ )
{
for ( j=1; j<=n; j++ )
printf("%5d", b[i][j]);
printf("%5d", b[i][j]);
}
return;
}
```

#### **Output:**

Enter the Number of vertices 3 Enter 0 for self loops and 999 for noedges 0 5 999 999 0 3 2 999 0 All pairs shortest path is 0 5 8 5 0 3 2 7 0

Dept of ISE, SJCIT

```
2) Implement N-Queens problem using Backtracking method
       # include< stdio.h>
       # include < conio.h>
       # include <process.h>
       #include <math.h >
        int x[10];
        void main( )
        {
           int k, i, j, n, count=1;
           int place (int);
           clrscr();
           printf( "Enter the number of Queens\n");
           scanf("%d",&n);
            if( n = 0 \parallel n = 2 \parallel n = 3 )
              printf(" No Solution");
            else
              k=1;
              x[1]=0;
              while(k)
              {
                x[k] = x[k] + 1;
                while( ( x[k] <=n) && (!place(k) ) )
                    x[k] = x[k] + 1;
                  if(x[k] \leq n)
                    {
                       if(k==n)
                        {
                            getch();
                            printf("Solution %d \n",count++);
                            for ( i=1; i<=n; i++ )
                              {
                                  for ( j=1; j<x[i]; j++ )
                                  printf(" * ");
                                  printf("Q");
                                  for ( j=x[i]+1; j<=n; j++ )
                                      printf(" * ");
                                       printf("\n");
                                }
                             }
                           else
                             {
                                 k + = 1;
                                 x[k]=0;
                               }
Dept of ISE, SJCIT
```

```
}
                        else
                            k - = 1;
                     }
                   getch();
              }
        int place( int p)
           {
                int i;
                for ( i=1; i<=(p-1); i++ )
                if( (x[i]=x[p]) \parallel ((abs(x[i]-x[p])) = (abs(i-p))))
                    return 0;
                  return 1;
            }
Output:
Enter the no of queens
4
Solution 1
   Q * *
   ***Q
   Q * * *
   * * Q *
Solution 2
    * * Q *
    Q * * *
    ***Q
    * Q * *
Dept of ISE, SJCIT
```

Dept of ISE, SJCIT

#### VIVA VOICE QUESTIONS

#### 1. What is an Algorithm? (L2)

Algorithm is a Step by step procedure to Solve a given problem for a finite number of input producing finite number of output with desired output.

#### 2. What is a Flow Chart?(L2)

Flow chart is a Graphical Representation of a solution to the Problem.

#### 3. What is the difference between Algorithm, Flow Chart, Program?(L2)

- Algorithm specifies the different things to be followed for solving a Problem.
- Flow Chart is a Graphical Representation of a Solution to the Problem. Both Algorithm and Flow Chart are Machine Independent.
- Program is a Set of Instructions which is used as a tool to communicate to the machine to get our work done, Program is Machine Dependent for particular Machine.

#### 4. What is the Aim of DAA lab or why we need to study DAA Lab?(L3)

DAA is a discipline, where we are dealing with designing or writing the algorithm keeping in Consideration of Space and Time Complexity, Such that Our Algorithm should execute in a very minimum amount of time by Minimum Space or RAM.

#### 5. Define Space and Time Complexity? Among this which one is more prioritized?(L3)

Space Complexeity is a measure of Amount of Space taken by a Program to finish its Execution. Time Complexeity is a measure of amount of time taken by a program to complte its Execution.Depending Upon Application it is considered,EX:For Mobile or Handheld Devices,We give Prefernce for both Space and time.

For a Huge and Inter active Systems like Web Applications we give more Preferences to time Complexeity.

#### 6. What is Design and what is Analysis of a Program?(L2)

Design is a Process of Writing an algorithm to a given Problem so that it should accept finite number of input and finite number of output with a definite output and Should Exit appropriately.

Analysis: Analysis is a next Phase of Writing an Algorithm ,in this phase we calculate the Efficiency of an Algorithm i.e time and space needed by an algorithm.

# 7. Write the general plan for analyzing the recursive algorithms.(L2)

Identify the inputs.

Identify the output is depended only on number of inputs.

Identify the Basic Operation in ALgorithm.

Form or write the Recursive Relation to the Algorithm.

#### 8. What are the various notations used to write an algorithm? (L2)

(i) Pseudocode (ii) Natural Language and etc..

#### 9. What is a Pseudocode?(L2)

It's a notation which is having the combination of Programming Constructs and English like Statements.

#### **10.** What is Asymptotic Notations?Explain Various notations used for the same.(L2)

Asymptotic Notation is a way of Representing an algorithm time Complexeity in terms of Best, Average, Worst.

There are 3 Notations used to represent (i) $\Omega(n)$ -Best Case :-Here the algorithm runing time g(n) is less than f(n). for all n>=n<sub>o</sub>.Such that there exist a constant C.

 $f(n) \ge c.g(n)$ , for all  $n \ge n_o$ 

Refer Text book for Graph.

(ii) O-Worst Case:-Here the Algorithm runing time g(n) is greater than F(n) for all  $n \ge n_0$ .

 $f(n) \le c.g(n)$ , for all  $n \ge n_0$ . Such that there exist a constant C.

Refer Text book for Graph.

(iii)  $\Theta$ -Average Case: When there is more than one parameter controlling the Execution time and We should go for Amortized analysis or Series of Execution for the Same Input and the Resultant Would be Average Case is which is c1.g(n)<f(n)<c2.g(n), for all n>=n<sub>o</sub>, Such that there exist a constant C.

Refer Text book for Graph.

#### 11. List out the few topics you have studied or Explain few topics of your Favourite list.(L2)

1. Mention a topics Which you are really good in, the topics like

- Searching-Explain any one Searching Algorithm
- Sorting-Explain Any one Sorting Algorithm.
- String Matching-Explain any one method ,Select either Brute Force or Horspool Method.
- Divide and Conquer-Explain either Merge sort or Quick sort which You are More Comfortable.
- Greedy Technique-Define Greedy and Explain Any problem with Greedy Method Ex:Knapsack Problem, Job Sequencing with Deadlines, Prims, Kruskals and etc..
- dynamic Programming-Explain any Problem with Dynamic Programming Approach

- Decrease and Conquer:Explain any problem which can be Solved Using decrease and Conquer,EX:DFS,BFS,Topological Sorting.
- Limitations of Algorithms:Explain few things about Decision trees and its use in solving a Problem.
- Coping with Limitations of Algorithms:Explain Backtracking and one problem which illustrates about Backtacking method and Wind Up.

# 12. What is the Time Complexeity of Bubble Sort, Selection Sort , Merge Sort, Quick Sort?(L3)

Bubble Sort-n<sup>2,</sup>

Selection Sort- n<sup>2</sup>

Merge Sort-nlog.n

Quick Sort -nLogn, Worst case for Quick Sort- n2

#### 13. Which sorting agorithm is more Efficient and why?(L3)

Quick Sorting is More Efficient ,because this algorithm is instable algorithm and inplace.

#### 14. What do you mean by the term Instable Algorithms?(L2)

The Instable Algorithms are one, which divides the array as certainly depending upon pivot or key element and hence i index precedes index j

#### **15.** Which algorithms are faster?(L3)

Instable Algorithms are much Faster compared to Stable Algorithms.

#### 16. For what type of instance Merge sort do better than Quick Sort?(L3)

For a Larger input and a sorted input values.

# 17. For what type of instance Quick sort do better than Merge Sort? (L3)

For Smaller Set of input numbers.

# 18. What are Inplace Algorithms?

Inplace Algorithms are the one which doesn't occupies Extra Space.

# **19.** Write the order of growth terms as per the time Execution in Ascending Order.(L3)

 $logn,n,nlogn,n^2,n^3,\ldots,n^n,2^n,n!$ 

# 20. What is Brute Force Technique? When We Should Use? (L3)

Brute Force is a straight Forward Technique to solve a problem, We used to solve a Problem through this approach when we don't have sufficient data to solve a problem in Efficient Way.

# 21. What is the difference between Divide and Conquer, Decrease and Conquer? (L2)

Dept of ISE, SJCIT

Divide and Conquer can be solved to solve a problem with a larger data set and when there is no dependency between any of the data sets.

Divide and Solve as Small as Small sets.

Conquer or Merge it get one final resultant data set.

Decrease and Conquer is almost similar to Divide and Conquer but we are finding a solutions to the problem in a different variations,EX:Decrease by Constant (Usually by One),Decrease by Constant factor which is almost similar to Divide and Conquer Technique(Usually by two),Decrease by Variable(The Dividing Criteria changes for each iteration depends upon the data set.

#### 22. Define Greedy Technique. (L2)

Greedy Technique is always applied for the problem of the type optimization type, which reduces loss and increases profit.

#### 23. Define Optimal and Feasible Solution. (L2)

Optimal Solution is a solution which is best among N Feasible Solution.

Feasible Solution is a solution which Satisfies a Problem Constraints/conditions.

#### 24. Can A Problem solved by all the algorithmic Techniques.(L3)

Yes, but some problems will give better results with some Algorithmic Technique and it may give worst result when it is applied with other technique.

#### 25. State and Explain Knapsack Problem. (L2)

Filling the Maximum number of items to the Knapsack (Container) Which Increases the profit and decreases the Loss.

#### 26. State Few Algorithmic Techniques which you have studied. (L2)

Brute Force Technique

Divide and Conquer

Greedy Technique

**Dynamic Programming** 

Decrease and Conquer

## 27. Which one is Most Admired algorithmic Technique?(L3)

Dynamic Programming.

#### 28. What is Spanning tree and Minimum Spanning tree?(L2)

A tree Without Cycles are called as Spanning tree .

A Minimum Spanning Tree is a spanning tree which yeilds the very less Cost when all the edges cost summed up.

#### **29.** How Many Spanning Tree can a Tree can have?(L3)

A tree can have 1 to many number of Possible ways of Spanning Tree.

#### **30.** Differentiate between Prims and Kruskals Algorithm for finding MST.(L2)

In Prims We consider any one vertex in the graph as Source and We compute the distance from that source to other vertices ,after computing the vertices which has minimum value among (n-1) vertices is added to tree vertices and that respective edges added to tree Edges Set. The above mentioned Process continues till we reach (n-1) vertices.

In Kruskals we first arrange the edges in Ascending Order and then we start to form the tree which wont form cycles, if adding that edges forms cycles then that edges is dropped from adding to tree edges. The above said process is continues till we reach the count of

(n-1) Vertices.

#### 31. What is the Application of Prims and Kruskals Algorithm?(L3)

In Networks to remove the Cyclicity of the Network.

#### 32. Explain Job Sequencing With Deadlines?(L2)

Placing or scheduling the maximum number of Jobs to a machine without violating the deadlines constraint of any of the Jobs in Sequence.

#### **33.** Why the Name Bubble Sort named?(L3)

Because in first Pass the first highest data will bubbles up, so since the largest element bubbles up in the first and second largest element bubbles up in the Second pass and so on, so hence the name bubble sort.

#### 34. Why the Name Selection Sort?(L3)

The Selection sort is named because we initially first select an arrays first element as minimum and will compare with other elements ,so in pass one first least element goes to the first position and so on so forth for 2nd,3rd and so on. Selecting

#### 35. What is the difference between Brute force strings matching to Horspool String Matching

**Method?** (L2)In brute Force we compare each and every element of the text to the pattern by shifting the text positionby one and in Horspool method we shift it by number of shift positions recorded in the shift table.

#### 36. Explain Merge Sort?(L2)

In Merge Sort will divide the entire input set by 2 until we reach low<high and later will find a solution to each item by comparing half of the array data set to the other half array data set and finally we mergeit to form a sinle array(conquer)

#### **37.** What is the Basic Operations in Merge sort and Quick sort?(L2)

In Merge Sort the Basic Operations is Comparisions and in Quick sort basic Operations is Partitioning and hence also known as partitioning sort.

#### 38. Why the Insertion Sort?(L3)

We are Inserting an element to its suitable place by comparing n elements for each pass.

#### **39.** What is the Use of DFS and BFS?(L2)

DFS and BFS both used to check the Connectivity of a graph,Cyclicity in a graph,Spanning tree of a graph.

#### 40. Differentiate between DFS and BFS.(L2)

DFS and BFS are both the Graph Traversing Technique, in which DFS Traverse the Graph in a depth wise(Vertical) and BFS Traverse the Graph from left to right(Horizontal)

#### 41. Which Data structures used in BFS and DFS.(L2)

BFS USes Queue as its data structure and DFS uses as stack its Data structure.

#### 42. What are back edges in DFS and Cross Edges in BFS.(L2)

Back Edges and Cross edges are the Edges which already visited by a ancestor node.

#### 43. What is Topological Sorting?(L2)

Topological Sorting is a Sorting Technique used for sorting Vertices in the Graph.

#### 44. What is the Conditions necessary for a Topological Sorting? (L3)

For a Topological Sorting the Graph Should be DAG(Directed Acyclic Graph)

#### 45. What are the Different methods used to solve a topological Sorting ?(L3)

- (i) Source Removal Method
- (ii) Using DFS based Scheme.

#### 46. What is the Use of Topological Sorting?(L3)

Use of Topological Ordering is in the field of Operating System for Scheduling and in Networks, Automation and Robotics.

#### 47. What is Dijikstra's Algorithm?(L2)

Dijikstra's Algorithm is Used to find the Single shortest Path from source to the other vertex.

#### 48. What is a graph?(L2)

Graph is a component which is having a set of Edges and vertices  $G = \{V, E\}$ 

#### 49. What are the different ways that can be represents a graph?(L2)

Adjaceny Matrix and Adjacency List.

#### 50. What is Adjacency Matrix?(L2)

Is a Matrix which illustrates the Graph in the form of Matrix, if it is a weights Graph then we initialize the value of the cost in that position(i,j) or else simply we write 1 to mention ther exist an edge between (i,j) OR else we use 0 or 9999 to mention non connectivity of a graph.

#### **51.** What is the limitations of Algorithms?(L2)

Algorithm can't find the better the soltions when we come across the tight lower bound, So we can find the better solutins which is not possible with Algorithmic way. To find the Tight lower bound we use Decision Trees.

#### 52. What is Tight lower Bound?(L2)

It is a Lower bound which is a best lower bound for an problem, beyond that no algorithm will produce better results.

#### **53.** What are Decision Trees?(L2)

Decision trees are also known as Comparision Trees used to find the tight lower bound for a particular Problem EX:Tight Lower Bound For Sorting is n.logn

and tight lower bound for Searching is logn which is not possible to get the better result.

#### 54. What is a polynomial problem (P-type) (L2)

P-type problem are decision problems in which we can find the solutions in a polynomial time and is of type deterministic.

#### 55. What is NP-problem?(L2)

NP-Problem belongs to decision problem and these problems are Non Deterministic Polynomial i.e for which the problem doesn't have deterministic solutions and Can be solved in Polynomial time

There are 2 phases in solving a problem.

(i) Guessing(Non-Deterministic stage) Producing N number of Candidate Outputs.

(ii) Verification(Deterministic Stage) Verifying The correctness of N Number of Candidate Outputs.

#### 56. What is NP-Complete Problems? (L2)

Np\_Complete Problems belongs to Decision problems and NP type Problems .

These problems can be find the solutions by converting or reducing to the problem which we know the Solutions.

Dept of ISE, SJCIT

#### 57. What is a trivial lower bound?(L2)

Trivial bound can be derived by formulating the number of inputs that has to be given and number of outputs that has to be generated.

#### 58. Explain Bactracking W.r.t

#### (I)Subset Problem (ii)N-Queens Problem

#### **59. Explain Subset Problem.(L2)**

In a given Set S ,find the Subset, in which the sum of all subset elements is equal to the sum d which is predefined in a problem.

#### 60. Explain N-Queens Problem.(L2)

N-Queens Problem is of Placing a N-Queens in a N\*N Chess board such that No 2-Queens Should be placed in the same Row,Column and same diagnol(N=Should consider both principal diagonal elements)

#### 61. What is Hamiltonian Circuit?(L2)

Hamiltonian circuit is a problem in which that circuit starts from a source vertex and has other vertex in any order without repeating and Should end with the Source vertex only i.e source and Destination vertex should be same.

#### 62. Explain the Problem of TSP.(L2)

Travelling Sales Person Problem is a problem in which he should Visit N number of cities with a minimum number of Cost by visiting every city Exactly one and the city what he is started should end with same city.

#### **63.** What is the Concept of Dynamic Programming?(L3)

Deriving a Solution to the basic Condition and Extracting the solutions for the rest of the other data sets by Previously drawnd Solution.Computer to other algorithmic Technique Dynamic Programming because it avoids lot of reworking on the same Solution what we have solved in the earlier phases of deriving the solution to the problem.

#### 64. What is the goal of Warshalls Algorithm?(L3)

Warshall's algorithm is to find the shortest distance between a node to all the other nodes in the graph. It Uses the Property of Transitive Closure i.e if there exist a path between (i,k) and (k,j) then there surely exist a path between (i,j)

 $(i,k) \And (k,j) \dashrightarrow \rightarrow (I,J)$ 

**65.** What is the use of Floyds algorithm?(L3)

It is use to find the All pairs shortest Path of an Graph.

#### **66.** What is Parallel Programming?(L2)

Parallel programming is a technique of writing a Program to Execute a series of instructions parallel to gain Speed up in Execution.

#### **67.** What is openMp?(L2)

OpenMp is an abbreviation of Open Multi programming and it is a programming tool which creates an environment to execute a program in the Parallel fashion i.e, N number of Instructions at the same time so as to achieve the spped up.

#### **68.** When one can go for Parallel Programming?(L3)

Parallel Programming Fashion is can be opted when there is no dependency between the series of Instructions that has to be Executed Parallely.

69. What is the Command used to compile a parallel Program?

#### (2) Command is **cc-fopen filename.c**

#### 70. What is cc stands for and –fopenmp stands for?(L2)

Cc stands for invoking a turbo C Compiler and –fopenmp is an option to the command which directs a compiler to invoke or use the parallel programming API so as to Execute the program in a parallel Way.

# 71. What are the Maximum Number of Possible Ways for a travelling Sales Person with N number of Cities?(L4)

(N-1)! i.e for 5 cities (5-1)!=4 !=24 ways.

## 72. Give Some Examples for a problem of type NP and NP-Complete?(L2)

Knapsack, Job Sequenncing with Deadlines, N-Queens Problem, Subset Problem, TSP and Hamiltonian circuit.

**73.** Give few Examples for a problem of type P?(L2)

Sorting Searching, String Matching and Hashing Problem.

# 74. What is the difference between Functions,Procedures,SubRoutines,Macros and Pseudo Code. Functions:are the Constructs which consists of set of Instructions which is meant to Perform some Specific Task.It is the Programming Constructs used in Middle Level/High Level Languages like C/C++.(L3)

**Procedures:**It is same as the Functions but we call it Procedures in Assembly languages like Masm,Tasm.The Size of the Procedures is small Compared to Functions and it uses the Mnemonics like JLR,ADD,SUB,DIV,CMP and etc.

SubRoutines: Same as above(Procedures, used in languages like FORTRAN)

**Macros:**Macros are the functions Which is Small in Size ,usually used as Library files ,it is invoked only if it is needed.

#### 75. Name Standard Input, Standard output, Standard Error.(L2)

Standard Input-Keyboard

Standard Output-Monitor

Standard Error-Monitor

76. What are standard I/O functions? (L2)

Printf() and Scanf()

77. What is the use of using getch() in every Program?Does it Impact to the ouput of the Program? (L3)

Getch() is a informal input functions which makes the ouput screen to be displayed until we enter a

character(Read-input operations)

Getch() doesn't Impact in anyways

#### 78. What are Command line Arguments (L2)

The Arguments which are passed at command line to the function main()

Are called as Comman line Arguments.

# **LAB EVALUATION RUBRICS**

#### FOR 40 MARKS (2018 NEW SCHEME)

SI. No.	DESCRIPTION	MARKS
1.	CONTINUOUS EVALUATION	<u>25</u>
	a. Observation write up & punctuality	5.0
		8.0
	b. Conduction of experiment and output	4.0
		8.0
	c. Viva voce	
	d. Record write up	
2.	INTERNAL TEST	15.0

Dept of ISE, SJCIT