COMPUTERCANARA ENGINEERING COLLEGE

(Affiliated to VTU, Approved by AICTE)

Benjanpadavu – 574 219, Bantwal Taluk, D.K. Dist. Karnataka

Department of Computer Science & Engineering

B.E 4th semester

DESIGN AND ANALYSIS OF ALGORITHM LABORATORY (15CSL47)

Lab Manual

Prepared By

Prof. Santosh Hiremath

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", BELAGAVI – 590 018

KARNATAKA

INDEX

	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.	**24 – 25**
6	**Implement in Java, the 0/1 Knapsack problem using** **(a) Dynamic Programming method** **(b) Greedy method.**	**26 – 30**
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.	**31 – 32**
8	Find Minimum Cost Spanning Tree of a given undirected graph using (a) Kruskal's algorithm (b) Prim's algorithm. Implement the program in Java language.	**33 - 37**
9	Write Java programs to (a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm. (b) Implement Travelling Sales Person problem using Dynamic programming.	**38 – 43**
10	(a) Design and implement in Java to 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}. Display a suitable message, if the given problem instance doesn't have a solution. (b) Design and implement the presence of Hamiltonian Cycle in an undirected Graph G of n vertices.	**44 – 49**

CONDUCTION OF PRACTICAL EXAMINATION:

Instructions:

1. **All laboratory experiments (TEN problems) are to be included for practical examination. Students are allowed to pick one experiment from the lot.**
2. **To generate the data set use random number generator function.**
3. **Strictly follow the instructions as printed on the cover page of answer script for breakup of marks.**
4. **Marks distribution: Procedure + Conduction + Viva: 20 + 50 + 10 (80). Change of experiment is allowed only once and marks allotted to the procedure**

****

COURSE OBJECTIVE

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.

COURSE DESCRIPTION

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

COURSE OUTCOMES

After studying this course, students will 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 solve real world problems.

1.

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

(v) USN
(vi) Name
(vii) Branch
(viii) Phone

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

```
import java.util.*;
class Student
{
        String usn;
        String name;
        String branch;
        int phone;
        Student(String usn,String name,String branch,int phone)
        {
                this.usn=usn;
                this.name=name;
                this.branch=branch;
                this.phone=phone;
        }
        public static void main(String a[])
        {
                Scanner s1=new Scanner(System.in);
                Student s[]=new Student[5];
                String usn;
                String name;
                String branch;
                int phone;
                int n;
                System.out.println("Enter the number of Students");
                n=s1.nextInt();
                System.out.println("Enter   Student   Details:   USN   NAME   BRANCH
                PHONE_NO");
                for(int i=0;i<n;i++)
                {
                        usn=s1.next();
                name=s1.next();
                branch=s1.next();
                phone=s1.nextInt();
                 s[i]=new Student(usn,name,branch,phone);
            }
            System.out.println("Student Details");
            System.out.println("USN\tNAME\tBRANCH\tPHONE");
            for(int i=0;i<n;i++)
            {
                System.out.println(s[i].usn+"\t"+s[i].name+"\t"+s[i].branch
                +"\t"+s[i].phone);
            }
        }
    }
```

Output:

```
Enter the number of Students
3

Enter Student Details: USN NAME BRANCH PHONE_NO
100 Abhi CSE 8888
200 Santosh ISE 9999
300 Amar ECE 7777

Student Details
USN   NAME        BRANCH      PHONE
100   Abhi        CSE         8888
200   Santosh     ISE         9999
300   Amar        ECE         7777
```

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

```
import java.io.IOException;
import java.util.*;

class Stack
{
        static  int top=-1;
        static int item[];


        static void pushItem(int data)
        {
                if (top == item.length - 1)
                {
                        System.out.println("Stack is Full");
                }
                else
                {
                        item[++top] = data;
                        System.out.println("Pushed Item :" + item[top]);
                }
        }

        static void popItem()
        {
                if (top < 0)
                {
                        System.out.println("Stack Underflow");
                        return 0;
                }
                else
                {
                        System.out.println("Pop Item : " + item[top]);
                        return item[top--];
                }
        }
        public void displayStack()
        {
                if (top >= 0)
                {
                        System.out.println("Elements in stack are:");
                        for (int i = top; i>=0; i--)
                        {
```

```
                        System.out.println(item[i]);
                    }
                }
                else
                {
                    System.out.println("Stack is Empty");
                }
            }
        }

        class StackExample
        {
            public static void main(String[] args) throws IOException
            {
                            int choice;
                Scanner s=new Scanner(System.in);
                System.out.println("Enter size of stack");
                int size=s.nextInt();
                item=new int[size];
                for(;;)
                {
                    System.out.println("\n");
                    System.out.println("1).Push\n2).Pop\n3).Display\n4).Exit\n\nEnter
                    Choice");
                    choice = s.nextInt();
                    switch(choice)
                    {
                        case 1: System.out.println("Enter Push Item: ");
                        pushItem(s.nextInt());
                        break;
                        case 2: popItem();break;
                        case 3: displayStack();break;
                        case 4: System.exit(0);break;
                        default: System.out.println("Invalid Choice");
                    }
                }
                 s.close();
            }
        }
```

Output:

```
1).Push
2).Pop
3).Display
4).Exit

Enter Choice
1
Enter Push Item:
10
Pushed Item: 10

1).Push
2).Pop
3).Display
4).Exit

Enter Choice
1
Enter Push Item:
20
Pushed Item: 20

1).Push
2).Pop
3).Display
4).Exit

Enter Choice
3
Elements in stack are:
10
20

1).Push
2).Pop
3).Display
4).Exit

Enter Choice
2
Pop Item: 20

1).Push
2).Pop
3).Display
4).Exit
```

```
Enter Choice
3
Elements in stack are:
10

1).Push
2).Pop
3).Display
4).Exit

Enter Choice
2
Pop Item: 10

1).Push
2).Pop
3).Display
4).Exit

Enter Choice
3
Stack is Empty


1).Push
2).Pop
3).Display
4).Exit

Enter Choice
2
Stack Underflow


1).Push
2).Pop
3).Display
4).Exit

Enter Choice
4
```

2.

a. 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.

```
package adaprograms;

import java.io.IOException;
import java.util.*;

class Staff
{
        int Staffid;
        String Name;
        long Phone;
        float Salary;

}

class Teaching extends Staff
{
        String Domain;
        String Publication;

        Teaching(int staffid, String name, long phone, float salary, String domain, String
        publication)
        {

                Staffid=staffid;
                Name=name;
                Phone=phone;
                Salary=salary;
                Domain=domain;
                Publication=publication;
        }

        void display()
        {
                System.out.println(Staffid +"\t\t"+ Name+"\t\t"+ Phone+"\t"+ Salary+"\t"+
                Domain+"\t"+ Publication );
        }
}
class Technical extends Staff
{
		String Skills;
		Technical(int staffid, String name, long phone, float salary, String skills)
		{
			Staffid=staffid;
			Name=name;
			Phone=phone;
			Salary=salary;
			Skills=skills;
		}
		void display()
		{
			System.out.println(Staffid  +"\t\t"+  Name+"\t\t"+  Phone+"\t"+  Salary+"\t"+
			Skills);
		}
	}
	class Contract extends Staff
	{
		int Period;
		Contract(int staffid, String name, long phone, float salary, int period)
		{
			Staffid=staffid;
			Name=name;
			Phone=phone;
			Salary=salary;
			Period=period;
		}
		void display()
		{
			System.out.println(Staffid  +"\t\t"+  Name+"\t\t"+  Phone+"\t"+  Salary+"\t"+
			Period);
		}
	}
	public class StaffDetails
	{
		public static void main(String argrs[]) throws IOException
		{
			Staff s[]=new Staff[10];
			int choice;
			Scanner scn=new Scanner(System.in);
			for(;;)
			{
			System.out.println("1).Teaching\n2).Technical\n3).Contract\n4).Exit\n\nEnt
			er Choice");
			choice = scn.nextInt();
			switch(choice)
```

```
{
        case 1:System.out.println("Enter number of teaching staff details");
                int n=scn.nextInt();
                System.out.println("Enter Teaching Staff Details: ID Name
                Phone Salary Domain Publication");
                 for(int i=0;i<n;i++)
                 {
                        int id=scn.nextInt();
                        String name=scn.next();
                        long phone=scn.nextLong();
                        float sal=scn.nextFloat();
                        String domain=scn.next();
                        String publ=scn.next();
                        s[i]=new Teaching(id,name,phone,sal,domain,publ);
                }

                System.out.println("Staff
                ID\tName\tPhone\tSalary\tDomain\tPublication");
                for(int i=0;i<n;i++)
                {
                        ((Teaching) s[i]).display();
                }
                break;
         case 2: System.out.println("Enter number of Technical Staff
                Details");
                int t=scn.nextInt();
                System.out.println("Enter Technical Staff Details: ID Name
                Phone Salary Skill");
                for(int i=0;i<t;i++)
                {
                        int id=scn.nextInt();
                        String name=scn.next();
                        long phone=scn.nextLong();
                        float sal=scn.nextFloat();
                        String skill=scn.next();
                        s[i]=new Technical(id,name,phone,sal,skill);
                }
                System.out.println("Staff ID\tName\tPhone\tSalary\tSkills");
                for(int i=0;i<t;i++)
                {
                        ((Technical) s[i]).display();
                }
                break;
         case 3: System.out.println("Enter number of Contract Staff Details");
                int c=scn.nextInt();
```

```
                                        System.out.println("Enter Contract Staff Details: ID Name
                                        Phone Salary Period");
                                        for(int i=0;i<c;i++)
                                        {
                                              int id=scn.nextInt();
                                              String name=scn.next();
                                              long phone=scn.nextLong();
                                              float sal=scn.nextFloat();
                                              int period=scn.nextInt();
                                              s[i]=new Contract(id,name,phone,sal,period);
                                        }
                                        System.out.println("Staff ID\tName\tPhone\tSalary\tPeriod")
                                        for(int i=0;i<c;i++)
                                        {
                                              ((Contract) s[i]).display();
                                        }
                                        break;
                                  case 4: scn.close();
                                        System.exit(0);break;
                                  default:System.out.println("Invalid Choice");
                            }
                      }

                      }
                }
```

Output:

```
1).Teaching
2).Technical
3).Contract
4).Exit

Enter Choice
1
Enter number of teaching staff details
2
Enter Teaching Staff Details: ID Name Phone Salary Domain Publication
100 AMAR 9999 1000 JAVA PEARSON
200 AJAY 8888 2000 C# PEARSON
StaffID      Name       Phone  Salary      Domain      Publication
100          AMAR       9999   1000.0      JAVA        PEARSON
200          AJAY       8888   2000.0      C#          PEARSON
1).Teaching
2).Technical
3).Contract
4).Exit
```

```
Enter Choice
2
Enter number of Technical Staff Details
1
Enter Technical Staff Details: ID Name Phone Salary Skill
300 AMAR 8888 4000 C
StaffID         Name            Phone  Salary          Skills
300             AMAR            8888   4000.0          C
1).Teaching
2).Technical
3).Contract
4).Exit

Enter Choice
3
Enter number of Contract Staff Details
1
Enter Contract Staff Details: ID Name Phone Salary Period
400 AJITH 9999 2000 1
StaffID         Name            Phone  Salary          Period
400             AJITH           9999   2000.0          1
1).Teaching
2).Technical
3).Contract
4).Exit

Enter Choice
4
```

b. 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.*;
import java.util.StringTokenizer;
public class Customer
{
        static String data;
static void read()
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter Customer  NAME and DATE of BIRTH(DD/MM/YYYY) :");
data=sc.nextLine();
}
static void  Display()
        {

                StringTokenizer st = new StringTokenizer(data, "/");
                System.out.print("Customer DATA IS ");
                while(st.hasMoreTokens())
                {
                        String val = st.nextToken();
                        System.out.print(val);
                        if(st.countTokens()!=0)
                                System.out.print(","+" ");
                }

        }
        public static void main(String[] args)
        {
                read();
                Display();
        }
}
```

Output:

```
Enter Customer  NAME and DATE of BIRTH(DD/MM/YYYY) :
santosh 21/11/2018
Customer DATA IS santosh 21, 11, 2018
```

3.

a. 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.*;

class divide
{
        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
                        {
                                if(b!=0)
                                {
                                System.out.println("Result="+(a/b));
                                }
                                else
                                {
                                result=a/b;
                                }
                        }
                        catch(ArithmeticException  e)
                        {
                        System.out.println("exception  caught: Divide by   zero error"+e);
                        }
        }
}
```

Output:

```
Enter the value of a
3
Enter the value of b
2
value of a is: 3
value of b is: 2
a/b is: 1
Enter the value of a
3
Enter the value of b
0
exception  caught: Divide by   zero error java.lang.ArithmeticException:  Divide by Zero Error
        at adaprograms.Arithmetic.<init>(Arithmetic.java:18)
        at adaprograms.Arithmetic.main(Arithmetic.java:31)
```

b. 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 Square extends Thread
{
        public int X;
        public Square(int x)
        {
                X = x;
        }
        public void run()
        {
                System.out.println("Square of " + x + " is: " + X * X);
        }
}
class Cube extends Thread
{
        public int X;
        public Cube(int x)
        {
                X = x;
        }
        public void run()
        {
                System.out.println("Cube of " + x + " is: " + X * X * X);
        }
}
class JavaThread extends Thread
{
        public void run()
        {
                int num = 0;
                Random r = new Random();
                try
                {
                        for (int i = 0; i < 5; i++)
                        {

                                System.out.println("-------------------------------------");
                                num = r.nextInt(10);
                                System.out.println("Random Number Generated is " + num);
                                Square s=new Square(num);
                                 s.start();
                                 Cube c = new  Cube(num);
                                 c.start();
                                 Thread.sleep(1000);
                        }
                }
                catch (Exception ex)
                {
                        System.out.println(ex.getMessage());
                }
        }
        public static void main(String args[])
        {
                JavaThread j=new JavaThread();
                j.start();
        }

}
```

Output:

```
--------------------------------------
Random Number Generated is 9
Square of 9 is: 81
Cube of 9 is: 729
--------------------------------------
Random Number Generated is 6
Square of 6 is: 36
Cube of 6 is: 216
--------------------------------------
Random Number Generated is 2
Square of 2 is: 4
Cube of 2 is: 8
--------------------------------------
Random Number Generated is 5
Square of 5 is: 25
Cube of 5 is: 125
--------------------------------------
Random Number Generated is 4
Square of 4 is: 16
Cube of 4 is: 64
```

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.*;
public class Qsort
{
        static int a[]=new int[100];
        static void qsort(int a[],int low,int high)
        {
                int j;
                if(low<high)
                {
                j=partition(a,low,high);
                qsort(a,low,j-1);
                qsort(a,j+1,high);
                }
        }

        static int partition(int a[], int low,int high)
        {
                int pivot,i,j,temp;
                 pivot=a[low];
                i=low+1;
                j=high;
                while(true)
                {
                while(pivot>a[i] && i<=high) i++;
                 while(pivot<a[j]) j--;
                        if(i<j)
                         {
                        temp=a[i];
                        a[i]=a[j];
                        a[j]=temp;
                         }
                        else
                        {
                        temp=a[j];
                        a[j]=a[low];
                        a[low]=temp;
                        return j;
```

```
                    }
                }
            }

public static void main(String[] args)
                {
                    Scanner scan = new Scanner( System.in );
                   System.out.println("enter nos of elements");
                   int n=scan.nextInt();

                    int i;
                   System.out.println("Random Numbers generated are: ");
                     Random r=new Random();
                    for(i=0;i<n;i++)
                    {
                            a[i]=r.nextInt(100);
                            System.out.println(a[i]);
                    }

                    long startTime = System.nanoTime();
                    qsort(a,0,n-1);
                    double endTime = System.nanoTime();
                    double duration = (endTime - startTime);

                    System.out.println("\nElements after sorting ");
                    for (int k = 0; k <n; k++)
                         System.out.print(a[k]+" ");
                         System.out.println();
                        double milliseconds=duration/1000000;
System.out.println("Total Time taken to sort the numbers is: "+milliseconds+" MilliSeconds");

            }
}
```

Output:

```
Quick Sort Test
Random Numbers generated are:
9811    8613    5565    4860    3580    7293    1014    9565    424     2456

Elements after sorting
424 1014 2456 3580 4860 5565 7293 8613 9565 9811
Total Time taken to sort the numbers is: 0.02415 MilliSeconds
```

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.*;
public class MergeSort
{
static int a[]=new int[500];
static int b[]=new int[500];
        static void merge(int a[],int low,int mid,int high)
        {
        int i,j,k;
        i=low;
        j=mid+1;
        k=low;
                while(i<=mid && j<=high)
                {
                if(a[i]<=a[j])
                b[k++]=a[i++];
                        else
                b[k++]=a[j++];

                }
         while(i<=mid) b[k++]=a[i++];
         while(j<=high) b[k++]=a[j++];
        for(k=low;k<=high;k++) a[k]=b[k];
        }
        static void mergesort(int a[],int low,int high)
        {
        int mid;
        if(low>=high) return;
         mid=(low+high)/2;
        mergesort(a,low,mid);
        mergesort(a,mid+1,high);
        merge(a,low,mid,high);
        }

public static void main(String[] args)
        {
    Scanner scan = new Scanner( System.in );
  System.out.println("enter nos of elements");
   int n=scan.nextInt();
```

```
        System.out.println("Random Numbers generated are: ");
         Random r=new Random();
                          for(int i=1;i<=n;i++)
                          {
                                a[i]=r.nextInt(100);
                                System.out.println(a[i]);
                          }

                          long startTime = System.nanoTime();
                          mergesort(a,1,n);
                          double endTime = System.nanoTime();
                          double duration = (endTime - startTime);

                          System.out.println("\nElements after sorting ");
                          for (int k = 1; k <=n; k++)
         System.out.print(a[k]+" ");
         System.out.println();
         double milliseconds=duration/1000000;
System.out.println("Total Time taken to sort the numbers is: "+milliseconds+" MilliSeconds");


           }
}
```

Output:

```
enter nos of elements
5
Random Numbers generated are:
64 982 784 914 359
Elements after sorting
64 359 784 914 982
Total Time taken to sort the numbers is: 0.012831 MilliSeconds
```

6. Implement in Java, the 0/1 Knapsack problem using:
a. Dynamic Programming method

```
import java.util.*;
public class knapsak {
static    int n,p;
static    int w[]=new int[20];
static    int c[]=new int[20];
static    int v[][]=new int[20][20];

 static int knapsack(int n,int p)
{
 int value;
 if(v[n][p]<0)
 {
     if(p-w[n]<0)
         value=knapsack(n-1,p);
     else
         value=max(knapsack(n-1,p),(c[n]+knapsack(n-1,p-w[n])));

     v[n][p]=value;
 }
 return v[n][p];
 }

static int max(int a,int b)
{
 if(a>b)
  return a;
 else
  return b;
}
static void knaps()
{
    int i,j;
    for(i=0;i<=n;i++)
    {
        for(j=0;j<=p;j++)
        {
           if(i==0 || j==0)
               v[i][j]=0;
           else
               v[i][j]=-1;
        }
    }
}

public static void main(String arg[])
{ int d,i,k,j;
            Scanner s = new Scanner( System.in );

System.out.println("Enter no. of items: ");
 n=s.nextInt();

System.out.println("\nEnter the capasity:\n");

p=s.nextInt();

System.out.println("\nEnter the weights:\n");
 for(i=1;i<=n;i++)
  w[i]=s.nextInt();

 System.out.println("\nEnter the cost:\n");
 for(i=1;i<=n;i++)
  c[i]=s.nextInt();


knaps();
 d=knapsack(n,p);


 System.out.println("\nThe optimal soluntion is:"+d);

}

}
```

Output:

```
Enter no. of items:
4
Enter the capasity:
5
Enter the weights:
2 1 3 2
Enter the cost:
12 10 20 15
The optimal soluntion is:37
```

b. 0/1 Knapsack problem using Greedy method

```
import java.util.*;

class  knaps {

public static  void main(String[] args) {

int i,j=0,max_qty,m,n;
float sum=0,max;
Scanner sc =  new  Scanner(System.in);
int array[][]=new  int[2][20];
System.out.println("Enter no   of items");
n=sc.nextInt();

 System.out.println("Enter  the weights of each items");
 for(i=0;i<n;i++)
array[0][i]=sc.nextInt();
System.out.println("Enter  the values of each knapsack :");
for(i=0;i<n;i++)
array[1][i]=sc.nextInt();
System.out.println("Enter maximum  volume  of max_knapsack");
max_qty=sc.nextInt();
m=max_qty;
while(m>=0)
{

max=0;
for(i=0;i<n;i++)
{

if(((float)array[1][i])/((float)array[0][i])>max)
{

max=((float)array[1][i])/((float)array[0][i]);
j=i;
}
}

if(array[0][j]>m)
{
System.out.println("Quantity  of item number: " +    (j+1) +  " added is " +m);

sum+=m*max;
m=-1;

}
else
{
System.out.println("Quantity  of item number: " +  (j+1) +  " added is " +  array[0][j]);
m-=array[0][j];
sum+=(float)array[1][j];
array[1][j]=0;
}
}
System.out.println("The total profit is " +  sum);
sc.close();

}

}
```

Output:

```
Enter no   of items
4
Enter  the weights of each items
2 1 3 2
Enter  the values of each knapsack :
12 10 20 15
Enter maximum  volume  of max_knapsack
5
Quantity  of item number: 2 added is 1
Quantity  of item number: 4 added is 2
Quantity  of item number: 3 added is 2
The total profit is 38.333332
BUILD SUCCESSFUL (total time: 18 seconds)
```

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.

```
import java.util.*;
public class Dijkstra {
   static int cost[][]=new int[20][20];
   static int d[]=new int[20];
   static int visited[]=new int[20];
   static int i,j,min,u,w;
static void dij(int source, int[][] cost, int[] visited, int[] d, int n) {

 for(i=1;i<=n;i++)
 {
  visited[i]=0;
  d[i]=cost[source][i];
 }

 visited[source]=1;
 d[s
ource]=0;
 for(j=2;j<=n;j++)
{
 min=999;
 for(i=1;i<=n;i++)
 {
  if(visited[i]!=1)
  {
        if (d[i]<min) {
            min=d[i];
            u=i;
        }
  }
  }    //for i
  visited[u]=1;

  for(w=1;w<=n;w++)
  {
   if(cost[u][w]!=999 && visited[w]==0)
   {
    if(d[w]>cost[u][w]+d[u])
     d[w]=cost[u][w]+d[u];
}
              }   //for w
}   // for j
}

public static void main(String arg[])
{
    int i,j,n,source;
            Scanner s = new Scanner( System.in );
        System.out.println("enter nos of vertices:");
          n=s.nextInt();


          System.out.println("Enter the cost adjacency matrix: ");


                        for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   cost[i][j]=s.nextInt();
  }
 }
System.out.println("\nEnter the source node: ");
 source=s.nextInt();
 dij(source,cost,visited,d,n);
 for(i=1;i<=n;i++)
  if(i!=source)
   System.out.println("\nShortest path from "+ source+"to"+i + " is "+d[i]);
}

}
```

Output:

```
enter nos of vertices:
6
Enter the cost adjacency matrix:
999 3 999 999 6 5
3 999 1 999 999 4
999 1 999 6 999 4
999 999 6 999 8 5
6 999 999 8 999 2
5 4 4 5 2 999

Enter the source node:
1
Shortest path from 1to2 is 3
Shortest path from 1to3 is 4
Shortest path from 1to4 is 10
Shortest path from 1to5 is 6
Shortest path from 1to6 is 5
```

8. Find Minimum Cost Spanning Tree of a given undirected graph using
a. Kruskal's algorithm

```
import java.util.*;

public class Kruskal {

  static  int i,j,k,a,b,u,v,n,ne=1;
static int min,mincost=0;
static int cost[][]=new int[10][10];
static int parent[]=new int[10];

static int find(int i)
{
 while(parent[i]!=0)
  i=parent[i];
 return i;
}
static void uni(int i,int j)
{
 if(i<j)

  parent[i]=j;
  else
  parent[j]=i;
}
public static void main(String arg[])
{
 Scanner s=new Scanner(System.in);

 System.out.println("Implementation of Kruskal's algorithmnn");
 System.out.println("Enter the no. of verticesn");
 n=s.nextInt();
 System.out.println("Enter the cost adjacency matrixn");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   cost[i][j]=s.nextInt();
   if(cost[i][j]==0)
    cost[i][j]=999;
  }
 }
 System.out.println("The edges of Minimum Cost Spanning Tree arenn");
 while(ne<n)
 {
  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;
     b=v=j;
    }
   }
  }
  u=find(u);
  v=find(v);
 // int uni=uni(u,v);
  if(u!=v)
  {   uni(u,v);
  System.out.println("edge"+(ne++) +"("+a+"->"+b+")"+min);
      mincost +=min;
  } else {
  }
  cost[a][b]=cost[b][a]=999;
 }
 System.out.println("ntMinimum cost = "+mincost);

}
}
```

Output:

```
Implementation of Kruskal's algorithmnn
Enter the no. of verticesn
4
Enter the cost adjacency matrixn
0 1 2 999
1 0 3 999
2 3 0 4
999 999 4 0
The edges of Minimum Cost Spanning Tree arenn
edge1(1->2)1
edge2(1->3)2
edge3(3->4)4
ntMinimum cost = 7
```

(b) Prim's algorithm. Implement the program in Java language

```
public class prims {
   static int ne=1,min_cost=0;
static int cost[][]=new int[20][20];
   static int d[]=new int[20];
   static int visited[]=new int[20];
   static int v,u,a,b;
public static void main(String arg[])
{
 int n,i,j,min,source;


 Scanner s=new Scanner(System.in);
        System.out.println("enter nos of vertices:");
          n=s.nextInt();
              System.out.println("Enter the cost adjacency matrix: ");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   cost[i][j]=s.nextInt();
  }
 }
 for(i=1;i<=n;i++)
  visited[i]=0;
System.out.println("Enter the root node:");
source=s.nextInt();
 visited[source]=1;
 System.out.println("\nMinimum cost spanning tree is\n");
 while(ne<n)
 {
  min=999;
  for(i=1;i<=n;i++)
  {
   for(j=1;j<=n;j++)
   {
    if(cost[i][j]<min)
     if(visited[i]==0)
      continue;
    else
    {
     min=cost[i][j];
     a=u=i;
     b=v=j;
    }
   }
  }

if(visited[u]==0||visited[v]==0)
  {
 System.out.println("\nEdge"+ne++ +"\t"+a+"->"+b+"="+min);
   min_cost=min_cost+min;
   visited[b]=1;
  }
  cost[a][b]=cost[b][a]=999;
 }
System.out.println("\nMinimum cost="+min_cost);


}
}
```

Output:

```
enter nos of vertices:
4
Enter the cost adjacency matrix:
999 1 5 2
1 999 999 999
5 999 999 3
2 999 3 999
Enter the root node:
1
Minimum cost spanning tree is

Edge1  1->2=1
Edge2  1->4=2
Edge3  4->3=3

Minimum cost=6
```

9.Write Java programs to
(a) Implement All-Pairs Shortest Paths problem using Floyd's algorithm.

```
import java.util.*;
public class floyed {
  static int cost[][]=new int[20][20];
     static int A[][]=new int[20][20];

     static int min(int a,int b)
     {
         if(a<b)
             return a;
         else
             return b;
     }
public static void main(String arg[])
{

    int n,i,j,min,k;
 Scanner s=new Scanner(System.in);
        System.out.println("enter nos of vertices:");
          n=s.nextInt();
              System.out.println("Enter the cost adjacency matrix: ");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   A[i][j]=s.nextInt();
  }
 }

 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   cost[i][j]=A[i][j];
  }
 }

 for(k=1;k<=n;k++)
 {
     for(j=1;j<=n;j++)
     {
         for(i=1;i<n;i++)
        {
           cost[i][j]=min(cost[i][j],cost[i][k]+cost[k][j]);

        }
     }
 }
 System.out.println("Transitive closer graph");
 for(i=1;i<=n;i++)
 {
  for(j=1;j<=n;j++)
  {
   System.out.print(cost[i][j]+"\t");
  }
  System.out.println();
 }

}
}
```

Output:

```
enter nos of vertices:
4
Enter the cost adjacency matrix:
0 999 3 999
2 0 999 999
999 7 0 1

6 999 999 0
Transitive closer graph
0       10      3       4
2       0       5       6
7       7       0       1
6       999     999     0
```

(or)

```
mport java.util.Scanner;
class  floyd1 {

static void flyd(int[][] w,int n)
{
int i,j,k; for(k=1;k<=n;k++) for(i=1;i<=n;i++) for(j=1;j<=n;j++)
w[i][j]=Math.min(w[i][j],  w[i][k]+w[k][j]);
}

public static  void main(String[] args) {
int a[][]=new  int[10][10];
int n,i,j;
System.out.println("enter the number of  vertices"); Scanner  sc=new Scanner(System.in);
n=sc.nextInt();
System.out.println("Enter  the weighted matrix");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
a[i][j]=sc.nextInt();
floyd1 f=new  floyd1();
flyd(a, n);
System.out.println("The shortest  path matrix is");
for(i=1;i<=n;i++)
{
for(j=1;j<=n;j++)
{
System.out.print(a[i][j]+" ");
}
System.out.println();
}
sc.close();

}

}
```

(b) Implement Travelling Sales Person problem using Dynamic programming

```
import java.util.*;
public class TSP {

public static void main(String[] args) {
int c[][]= new int [10][10];
int path[]=new int [10];
Scanner in =new Scanner(System.in);
int i, j,cost;
System.out.println("Enter the number of nodes: ");
int n = in.nextInt();

System.out.println("Enter the cost matrix");
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
c[i][j] = in.nextInt();

for(i=1;i<=n;i++)
path[i]=i;
cost = tspdp(c, path, 1, n);
System.out.println("path is");
for(i=1;i<=n;i++)
System.out.print(path[i]+"->");
System.out.println("1");
System.out.println(" mincost is "+cost);
}
static int tspdp(int c[][], int path[], int start, int n)
{
int mintour[]=new int [10], temp[]=new int[10], mincost=999, ccost, i, j, k;
if(start == n-1)  {
return(c[path[n-1]][path[n]] + c[path[n]][1]);
}
for(i=start+1; i<=n; i++)  {
for(j=1; j<=n; j++)
temp[j] = path[j];
temp[start+1] = path[i];
temp[i] = path[start+1];

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

Output:

```
Enter the number of nodes:
4
Enter the cost matrix
0 1 2 3
1 0 2 3
2 2 0 2
3 3 2 0
path is
1->2->3->4->1
 mincost is 8
```

10.

a. **Design and implement in Java to 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}. Display a suitable message, if the given problem instance doesn't have a solution.**

```
import java.util.*;

public class Subset
{
static int s[]=new int[10] ;
int x[]=new int[10];
static int d;
void sumofsub ( int m , int k , int r )
{
        int i=1 ;
        x[k]=1;
        if ( ( m + s[k] ) == d )
        {
                System.out.print("Subset:");
                for ( i = 1 ; i <= k ; i++ )
                if ( x[i] == 1 )
                        System.out.print(s[i]+"\t");
                        System.out.println("\n") ;
        }
        else
                if ( m + s[k] + s[k+1] <= d )
                        sumofsub ( m + s[k] , k + 1 , r - s[k] ) ;
                if ( ( m + r - s[k] >= d ) && ( m + s[k+1] <=d ) )
                {
                        x[k] = 0;
                        sumofsub ( m , k + 1 , r - s[k] ) ;
                }
}
public static void main (String args[])
{
        Subset sb=new Subset();
        int n , sum = 0 ;
        int i ;
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the set : " ) ;
        n=sc.nextInt();
        System.out.println("Enter the set in increasing order:" ) ;
```

```
                for ( i = 1 ; i <= n ; i++ )
                        s[i]=sc.nextInt();
                        System.out.print("Enter the value of d :") ;
                        d=sc.nextInt();
                        for ( i = 1 ; i <= n ; i++ )
                                sum = sum + s[i] ;
                                if ( sum < d || s[1] > d )
                                        System.out.println("No subset possible : ") ;
                                else
                                        sb.sumofsub ( 0 , 1 , sum ) ;
                sc.close();
        }
        }
```

Output:

```
Enter the size of the set: 5
Enter the set in increasing order:
1 2 3 4 5
Enter the value of d: 5
Subset: 1       4

Subset: 2       3

Subset: 5
```

b. Design and implement the presence of Hamiltonian Cycle in an undirected Graph G of n vertices.

```
        public class HamiltonianCycle
        {
        final int V = 5;
        int path[];


boolean isSafe(int v, int graph[][], int path[], int pos)
        {

        if (graph[path[pos - 1]][v] == 0)
                        return false;

        for (int i = 0; i < pos; i++)
                        if (path[i] == v)
                                return false;
                        return true;
        }

boolean hamCycleUtil(int graph[][], int path[], int pos)
        {

   if (pos == V)
           {

      if (graph[path[pos - 1]][path[0]] == 1)
                 return true;
              else
                 return false;
           }


   for (int v = 0; v < V; v++)
           {

      if (isSafe(v, graph, path, pos))
              {
                 path[pos] = v;

         if (hamCycleUtil(graph, path, pos + 1) == true)
                   return true;
```

```
            path[pos] = -1;
                }
            }

            return false;
        }


 int hamCycle(int graph[][])
        {
            path = new int[V];
            for (int i = 0; i < V; i++)
                path[i] = -1;

    path[0] = 0;
            if (hamCycleUtil(graph, path, 1) == false)
            {
                System.out.println("\nSolution does not exist");
                return 0;
            }

            printSolution(path);
            return 1;
        }


void printSolution(int path[])
        {
            System.out.println("Solution Exists: Following" +
                               " is one Hamiltonian Cycle");
            for (int i = 0; i < V; i++)
                System.out.print(" " + path[i] + " ");

            System.out.println(" " + path[0] + " ");
        }


public static void main(String args[])
        {
            HamiltonianCycle hamiltonian =new HamiltonianCycle();

    int graph1[][] = {{0, 1, 0, 1, 0},
                {1, 0, 1, 1, 1},
                {0, 1, 0, 0, 1},
                {1, 1, 0, 0, 1},
                {0, 1, 1, 1, 0},
            };


hamiltonian.hamCycle(graph1);


int graph2[][] = {{0, 1, 0, 1, 0},
                {1, 0, 1, 1, 1},
                {0, 1, 0, 0, 1},
                {1, 1, 0, 0, 0},
                {0, 1, 1, 0, 0},
            };


hamiltonian.hamCycle(graph2);
        }
       }
```

Output:

```
Solution Exists: Following is one Hamiltonian Cycle
 0 1 2 4 3 0

Solution does not exist
```