



Channabasaveshwara Institute of Technology

(An ISO 9001:2008 Certified Institution)

NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.



DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY

(15CSL47)

Prepared by

**Rashmi V and Anil Kumar
Assistant professors,
Department of ISE,
CIT Gubbi.**

Experiment No. 1

1. A) 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.

```
import java.util.Scanner;
public class student {
    String USN;
    String Name;
    String branch;
    int phone;
    void insertRecord(String reg,String name, String brnch,int ph)
    {
        USN=reg;
        Name=name;
        branch=brnch;
        phone=ph;
    }
    void displayRecord()
    {
        System.out.println(USN+" "+Name+" "+branch+" "+phone);
    }
    public static void main(String
args[]){ student s[]=new student [100];
Scanner sc=new Scanner(System.in);
System.out.println("enter the number
of students"); int n=sc.nextInt();
for(int i=0;i<n;i++) s[i]=new
student();
for(int j=0;j<n;j++)
{ System.out.println("enter the usn,name,branch,phone");
String USN=sc.next();
String Name=sc.next();
String branch=sc.next();
int phone=sc.nextInt();
s[j].insertRecord(USN,Name,branch,phone);
}
for( int m=0;m<n;m++){
s[m].displayRecord();
}
}
}
```

OUTPUT

enter the number
of students 2

enter the
usn,name,branch,phone 1

Rashm

i ISE

93411

enter the

usn,name,branch,phone 12

gowda

ISE

9785

students details are

1 Rashmi ISE

93411 12 gowda

ISE 9785

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

```
import java.util.Scanner;

public class stack {
    final int max=100;
    int s[]=new
int[max]; int top=-1;
    void push(int ele)
    {
        if(top>=max-1)
            System.out.println("stack
overflow"); else
            s[++top]=ele;
    }
    int pop()
    {
        int z=0;
        if(top== -1)
            System.out.println("stack underflow");
        else
            z=s[top--];

        return z;
    }
    void display()
    {
        if(top== -1)
            System.out.println("stack empty");
        else
        {
            for(int i=top; i>-1; i--)
                System.out.println(s[i]+ " ");
        }
    }
    public static void main(String args[])
    {
        int q=1;
        stack m = new stack();
        System.out.println("program to perform stack operations");
        Scanner sc=new Scanner(System.in);

        while(q!=0)
        {
            System.out.println("enter 1. push 2.pop 3. display
"); System.out.println("enter your choice");
```

```
        int ch=sc.nextInt();
    switch(ch)
    {
    case 1:System.out.println("enter the element to be
        pushed"); int ele=sc.nextInt();
        m.push(ele);
        break;
    case 2:int popele;
        popele=m.pop();
        System.out.println("the popped element
        is"); System.out.println(popele+ " ");
        break;
    case 3:System.out.println("elements in the
        stack are"); m.display();
        break;
    case 4:q=0;
    }
    }
    }
}
```

Output:

```
program to perform stack
operations enter 1. push 2.pop
3. display enter your choice
```

```
1
```

```
enter the element to
be pushed 10
```

```
enter 1. push 2.pop 3.
display enter your choice
```

```
1
```

```
enter the element to
be pushed 20
```

```
enter 1. push 2.pop 3.
display enter your choice
```

```
3
```

```
elements in the
stack are 20 10
```

```
enter 1. push 2.pop 3.
display enter your choice
```

```
1
```

```
enter the element to
be pushed 30
```

```
enter 1. push 2.pop 3.
display enter your choice
```

```
1
```

```
enter the element to
be pushed 40
```

```
enter 1. push 2.pop 3.
display enter your choice
```

3

elements in the stack
are 40 30 20 10

enter 1. push 2.pop 3.
display enter your choice

2

the popped
element is 40

enter 1. push 2.pop 3.
display enter your choice

2

the popped
element is 30

enter 1. push 2.pop 3.
display enter your choice

2

the popped
element is 20

enter 1. push 2.pop 3.
display enter your choice

2

the popped
element is 10

enter 1. push 2.pop 3.
display enter your choice

2

stack underflow

Experiment No. 2a

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.

```

class Staff {
int staffid,phone,salary;
String name;
public Staff(int id , int no, int sal,
String na){ staffid=id;
phone=no;
salary=sal;
name=na;
}
void display(){ System.out.println("-----
-----"); System.out.println("Staff ID:"+ " "+
staffid); System.out.println("Staff Phone number:" + "
"+ phone); System.out.println("Staff Salary:" + " "+
salary); System.out.println("Staff Name:" + " "+ name);

}
}
class Teaching extends
Staff { String domain;
int no_of_publications;
public Teaching(int id, int no, int sal, String na,String
d,int nop){ super(id,no,sal,na);
domain=d;
no_of_publications=nop;
}
void Tdisplay(){ System.out.println("-----
-----"); System.out.println("Teaching
Staff Details"); super.display();

System.out.println("Domain :"+ " "+domain);
System.out.println("No_of_publications:"+ " "+no_of_publications);
}
}
class Technical extends
Staff{ String skills;
public Technical(int id , int no, int sal, String
na,String sk){ super(id,no,sal,na);
skills=sk;
}
void Tdisplay(){ System.out.println("-----
-----"); System.out.println("Technical
Staff Details"); super.display();

System.out.println("Skills :"+ " "+skills);
}
}
class Contract extends
Staff{ int period;
public Contract(int id , int no, int sal, String
na,int pd){ super(id,no,sal,na);
period=pd;
}
}

```



```
}
void Cdisplay(){ System.out.println("-----
-----"); System.out.println("Contract Staff
Details"); super.display();

System.out.println("ContractPeriod:" + " "+period + "years");
}
}
public class Multilevel{
public static void main(String args[]){
Teaching t1=new Teaching(11,998765434,31000,"Anil","ISE",10);
Teaching t2=new Teaching(12,996655546,30000,"Anu","ISE",9);
Teaching t3=new
Teaching(13,999933442,32000,"Anusha","EEE",8); t1.Tdisplay();
t2.Tdisplay();
t3.Tdisplay();
Technicalte1=new Technical(21,994433221,22000,"Kumar","C");
Technicalte2=new Technical(22,998877665,28000,"Krisna","Java");
Technical te3=new Technical(23,991654321,33000,"Kiran","Java");
te1.Tedisplay();
te2.Tedisplay();
te3.Tedisplay();
Contract ct1=new Contract(31,998765434,35000,"Anil",3);
Contract ct2=new Contract(32,912345678,39000,"Meghana",2);
Contract ct3=new Contract(33,992233445,30000,"Uma",4);
ct1.Cdisplay();
ct2.Cdisplay();
ct3.Cdisplay();
}
}
```

Output:

Teaching Staff Details

Staff ID: 11
Staff Phone number: 998765434
Staff Salary: 31000
Staff Name: Anil
Domain : ISE
No_of_publications: 10

Teaching Staff Details

Staff ID: 12
Staff Phone number: 996655546
Staff Salary: 30000
Staff Name: Anu
Domain : ISE
No_of_publications: 9

Teaching Staff Details

Staff ID: 13
Staff Phone number: 999933442
Staff Salary: 32000
Staff Name: Anusha
Domain : EEE
No_of_publications: 8

Technical Staff Details

Staff ID: 21
Staff Phone number: 994433221
Staff Salary: 22000
Staff Name:
Kumar Skills : C

Technical Staff Details

Staff ID: 22
Staff Phone number: 998877665
Staff Salary: 28000
Staff Name: Krisna
Skills : Java

Technical Staff Details

Staff ID: 23
Staff Phone number: 991654321
Staff Salary: 33000
Staff Name: Kiran
Skills : Java

Contract Staff Details

Staff ID: 31
Staff Phone number: 998765434
Staff Salary: 35000
Staff Name: Anil
ContractPeriod: 3years

Contract Staff Details

Staff ID: 32
Staff Phone number: 912345678
Staff Salary: 39000
Staff Name: Meghana
ContractPeriod: 2years

Contract Staff Details

Staff ID: 33
Staff Phone number: 992233445
Staff Salary: 30000
Staff Name: Uma
ContractPeriod: 4years

Experiment No. 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;

class customer
{
    String name;
    String date;
    public void read()
    {
        Scanner input =new
        Scanner(System.in); name=input.next();
        date=input.next();
    }
    public void display()
    {
        System.out.print(name+",");
        String delims="/";
        StringTokenizer st=new StringTokenizer(date,delims);
        while(st.hasMoreElements()){
            System.out.print(st.nextElement()+",");
        }
        System.out.println();
    }

    public static void main(String[] args)
    {
        System.out.println("Enter the customer
        detail"); customer[] cus=new customer[30];
        Scanner sc =new Scanner(System.in);
```

```
        System.out.println("enter the number
of customer"); int n=sc.nextInt();

        for(int i=0;i<n;i++)
        {
                cus[i]=new customer();
                cus[i].read();
        }
        for(int i=0;i<n;i++)
                cus[i].display();
        }
}
```

Output:

```
Enter the customer
detail enter the number
of customer 2
Enter the customer name
and date Rashmi
12/2/2017
gowda
11/4/2017
Rashmi,12,2,2017,
gowda,11,4,2017,
```

Experiment No. 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 division
{
    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: Divide by zero
error"+e);
        }
    }
}
```

Output:

```
Input two
integers 6 2
Result=3
Input two
integers 3 0
```

```
exception caught: Divide by zero errorjava.lang.ArithmeticException: /
by zero
```

Experiment No. 3b

Write a Java program that implements a multi-thread application that hash tree 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.*;

class second implements Runnable
{
    public int x;
    public second (int x)
    {
        this.x=x;
    }
    public void run()
    {
        System.out.println("Second thread:Square of the number
is"+x*x);
    }
}
class third implements Runnable
{
    public int x;
    public third(int x)
    {
        this.x=x;
    }
    public void run()
    {
        System.out.println("third thread:Cube of the number is"+x*x*x);
    }
}
class first extends Thread
{
    public void run()
    {
        int num=0;
        Random r=new
        Random(); try
        {
            for(int i=0;i<5;i++)
            {
                num=r.nextInt(100);
                System.out.println("first thread generated number
is"+num);

                Thread t2=new Thread (new
                second(num)); t2.start();
                Thread t3=new Thread(new
                third(num)); t3.start();
                Thread.sleep(1000);
            }
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
        }
    }
}
```

```
        }
    }
}

public class multithread
{
    {
        first a=new
        first(); a.start();
    }
}
```

OUTPUT:

first thread generated number is65
Second thread:Square of the number is4225
third thread:Cube of the number is274625 first
thread generated number is33
Second thread:Square of the number
is1089 third thread:Cube of the number
is35937 first thread generated number is30
Second thread:Square of the number is900
third thread:Cube of the number is27000
first thread generated number is29 Second
thread:Square of the number is841 third
thread:Cube of the number is24389 first
thread generated number is93
Second thread:Square of the number is8649
third thread:Cube of the number is804357

Experiment No. 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.Random;
import java.util.Scanner;

public class quicksort {
    static int max=2000;
    int partition (int[] a, int low,int high)
    {
        int p,i,j,temp;
        p=a[low];
        i=low+1;
        j=high;
        while(low<high)
        {
            while(a[i]<=p&& i<high)
                i++;
            while(a[j]>p)
                j--;
            if(i<j)
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
            else
            {
                temp=a[low];
                a[low]=a[j];
                a[j]=temp;
                return j;
            }
        }
        return j;
    }
    void sort(int[] a,int low,int high)
    {
        if(low<high)
        {
            int s=partition(a,low,high);
            sort(a,low,s-1);
            sort(a,s+1,high);
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```



```

        int[]
a; int i;
    System.out.println("Enter the array size");
    Scanner sc =new Scanner(System.in);
    int n=sc.nextInt();
    a= new int[max];
    Random generator=new Random();
    for( i=0;i<n;i++)
    a[i]=generator.nextInt(20);
    System.out.println("Array before sorting");
    for( i=0;i<n;i++)
        System.out.println(a[i]+" ");
    long startTime=System.nanoTime();

    quicksort m=new quicksort();
    m.sort(a,0,n-1);
    long stopTime=System.nanoTime();
    long elapseTime=(stopTime-startTime);
    System.out.println("Time taken to sort array is:"+elapseTime+"nano
seconds");
    System.out.println("Sorted
array is"); for(i=0;i<n;i++)
        System.out.println(a[i]);

    }
}

```

OUTPUT:

```

Enter the
array size 10
Array before
sorting 17 17 12 2
10 3 18 15 15 17

```

```

Time taken to sort array is:16980
nano seconds Sorted array is
2
3
10
12
15
15
17
17
17
17
18

```

Experiment No. 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 divideand- 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 mergesort {
    static int max=10000;
    void merge( int[] array,int low, int mid,int high)
    {
        int i=low; int
        j=mid+1; int
        k=low;
        int[]resarray;
        resarray=new int[max];

        while(i<=mid&& j<=high)
        {
            if(array[i]<array[j])
            {
                resarray[k]=array[i];
                i++;
                k++;
            }
            else
            {
                resarray[k]=array[j];
                j++;
                k++;
            }
        }
        while(i<=mid)
            resarray[k++]=array[i++];
        while(j<=high)
            resarray[k++]=array[j++];
        for(int m=low;m<=high;m++)
            array[m]=resarray[m];
    }

    void sort( int[] array,int low,int high)
    {
        if(low<high)
```

```

        {
            int mid=(low+high)/2;
            sort(array,low,mid);
            sort(array,mid+1,high);
            merge(array,low,mid,high);
        }
    }
    public static void main(String[]
    args) { int[] array;
    int i;
        System.out.println("Enter the array
        size"); Scanner sc =new
        Scanner(System.in); int n=sc.nextInt();
        array= new int[n];
        Random generator=new
        Random(); for( i=0;i<n;i++)
        array[i]=generator.nextInt(20);
        System.out.println("Array
        before sorting"); for( i=0;i<n;i++)
            System.out.println(array[i]+" ");
        long startTime=System.nanoTime();
        mergesort m=new mergesort();
        m.sort(array,0,n-1);
        long stopTime=System.nanoTime(); long
        elapseTime=(stopTime-startTime);

seconds");
        System.out.println("Sorted
        array is"); for(i=0;i<n;i++)
            System.out.println(array[i]);
    }
}

```

Output:

```

Enter the
array size 10
Array before
sorting 13 9 13
16 13 3 0 6 4 5

```

```

Time taken to sort array
is:171277nano seconds Sorted array is
0
3
4
5

```

6
9
13
13
13
16

Experiment No. 6

Implement in Java, the **0/1 Knapsack** problem using

- (a) Dynamic Programming method (b) Greedy method.

(a) Dynamic Programming method

```

import java.util.Scanner;

public class knapsackDP {

    /**
     * @param
     * args */
    public void solve(int[] wt, int[] val, int W, int N)
    {
        int i,j;

        int[][] sol = new int[N + 1][W + 1];

        for ( i = 0; i <= N; i++)
        {
            for ( j = 0; j <= W; j++)
            {
                if(i==0||j==0)
                    sol[i][j]=0;
                else if(wt[i]>j)
                    sol[i][j]=sol[i-1][j];
                else
                    sol[i][j]=Math.max((sol[i-1][j]), (sol[i - 1][j - wt[i]] + val[i]));

            }
        }
        System.out.println("The optimal solution is"+sol[N][W]);

        int[] selected = new int[N + 1];
        for(i=0;i<N+1;i++)
            selected[i]=0;
        i=N;
        j=W;
        while (i>0&& j>0)
        {
            if (sol[i][j] !=sol[i-1][j])
            {
                selected[i] = 1;
                j = j - wt[i];
            }
            i--;
        }

        System.out.println("\nItems selected : ");
        for ( i = 1; i < N + 1; i++)
            if (selected[i] == 1)
                System.out.print(i + " ");
    }
}

```

```
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        knapsackDP ks = new knapsackDP();

        System.out.println("Enter number of
elements "); int n = scan.nextInt();
        int[] wt = new int[n + 1];
        int[] val = new int[n + 1];

        System.out.println("\nEnter weight for "+ n +
elements");
        for (int i = 1; i <= n; i++)
            wt[i] = scan.nextInt();
        System.out.println("\nEnter value for "+ n +
elements");
        for (int i = 1; i <= n; i++)
            val[i] = scan.nextInt();

        System.out.println("\nEnter knapsack weight
"); int W = scan.nextInt();
        ks.solve(wt, val, W, n);
    }
}
```

Output:

```
Enter number of
elements 4
```

```
Enter weight for 4
elements 2 1 3 2
```

```
Enter value for 4
elements 12 10 20 15
```

```
Enter knapsack
weight 5
The optimal solution
is37 Items selected :
1 2 4
```

(b) Greedy method.

```

import java.util.Scanner;

public class knapsacgreedy {

    /**
     * @param
     * args */
    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
items");

        for(i=0;i<n;i++)
            array[1][i]=sc.nextInt();

        System.out.println("Enter maximum volume of
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 items 12 10 20 15
```

```
Enter maximum volume of  
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
```


Experiment No. 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.Scanner;

public class Dijkstra {

    /**
     * @param
     * args */
    int d[]=new int[10];
    int p[]=new int[10];
    int visited[]=new int[10];

    public void dijk(int[][]a, int s, int n)
    {
        int u=-1,v,i,j,min;
        for(v=0;v<n;v++)
        {
            d[v]=99;
            p[v]=-1;
        }
        d[s]=0;
        for(i=0;i<n;i++){
            min=99;
            for(j=0;j<n;j++){
                if(d[j]<min&& visited[j]==0)
                {
                    min= d[j];
                    u=j;
                }
            }
            visited[u]=1;
            for(v=0;v<n;v++){
                if((d[u]+a[u][v]<d[v]) && (u!=v) &&visited[v]==0)
                {
                    d[v]=d[u]+a[u][v];
                    p[v]=u;
                }
            }
        }
    }

    void path(int v,int s)
    {
        if(p[v]!=-1)
            path(p[v],s);
        if(v!=s)
            System.out.print("<->"<v+" ");
    }

    void display(int s,int n){ int i;
        for(i=0;i<n;i++)
        {
            if(i!=s){
                System.out.print(s+"
                "); path(i,s);
            }
        }
    }
}
```

```
    }

    if(i!=s)
        System.out.print("="+d[i]+"
"); System.out.println();
    }
}

    public static void main(String[] args) {

        int a[][]=new
int[10][10]; int i,j,n,s;
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=0;i<n;i++)
for(j=0;j<n;j++)
    a[i][j]=sc.nextInt();
System.out.println("enter the
source vertex"); s=sc.nextInt();
Dijkstra tr=new Dijkstra();
tr.dijk(a,s,n);
System.out.println("the shortest path between
source"+s+"to remaining vertices are");
tr.display(s,n);
sc.close();
    }

}
```

Output:

```
enter the number
of vertices 5
enter the weighted
matrix 0 3 99 7 99 3 0
4 2 99 99 4 0 5 6 5 2 5
0 4 99 99 6 4 0
```

```
enter the
source vertex 0
the shortest path between source0to remaining vertices are
```

```
0 ->1 =3
0 ->1 ->2 =7
0 ->1 ->3 =5
0 ->1 ->3 ->4 =9
```

Experiment No. 8

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

```
import java.util.Scanner;

public class kruskal {

    int parent[]=new int[10];
    int find(int m)
    {
        int p=m;
        while(parent[p]!=0)
            p=parent[p];
        return p;
    }
    void union(int i,int j)
    {
        if(i<j)
            parent[i]=j;
        else
            parent[j]=i;
    }
    void krkl(int[][]a, int n)
    {
        int u=0,v=0,min,k=0,i,j,sum=0;
        while(k<n-1)
        {
            min=99;
            for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                    if(a[i][j]<min&&i!=j)
                    {
                        min=a[i][j];
                        u=i;
                        v=j;
                    }
            i=find(u);
            j=find(v);
            if(i!=j)
            {
                union(i,j);
                System.out.println("(" +u+", "+v+")"+"="+a[u][v]);
                sum=sum+a[u][v];
                k++;
            }
            a[u][v]=a[v][u]=99;
        }
        System.out.println("The cost of minimum spanning tree = "+sum);
    }

    public static void main(String[]
args) { int a[][]=new int[10][10];
int i,j;
System.out.println("Enter the number of vertices of
the graph"); Scanner sc=new Scanner(System.in);
```

```
int n;
n=sc.nextInt();
System.out.println("Enter the
wiegthed matrix"); for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
        a[i][j]=sc.nextInt();
kruskal k=new kruskal();
k.krkl(a,n); sc.close();

}

}
```

Output:

```
Enter the number of vertices of
the graph 6
Enter the wieghted
matrix 0 3 99 99 6 5 3
0 1 99 99 4 99 1 0 6 99
4 99 99 6 0 8 5 6 99 99
8 0 2 5 4 4 5 2 0
(2,3)=1 (5,6)=2 (1,2)=3
(2,6)=4 (4,6)=5
```

The cost of minimum spanning tree = 15

9. Find Minimum Cost Spanning Tree of a given undirected graph using **Prim's algorithm**. Implement the program in Java language.

```
import java.util.Scanner;

public class prims {

    public static void main(String[]
args) { int w[][]=new int[10][10];
int n,i,j,s,k=0;
int min;
int sum=0;
int u=0,v=0;
int flag=0;
int sol[]=new int[10];
System.out.println("Enter the number of
vertices"); Scanner sc=new Scanner(System.in);
n=sc.nextInt();
for(i=1;i<=n;i++)
    sol[i]=0;
System.out.println("Enter the
weighted graph"); for(i=1;i<=n;i++)
    for(j=1;j<=n;j++)
        w[i][j]=sc.nextInt();
System.out.println("Enter the
source vertex"); s=sc.nextInt();
sol[s]=1;
k=1;
while (k<=n-1)
{
    min=99;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            if(sol[i]==1&&sol[j]==0)
                if(i!=j&&min>w[i][j])
                {
                    min=w[i][j];
                    u=i;
                    v=j;
                }
            sol[v]=1;
            sum=sum+min;
            k++; System.out.println(u+"-
">"+v+"="+min);
        }
    for(i=1;i<=n;i++)
        if(sol[i]==0)
            flag=1;
    if(flag==1)
        System.out.println("No spanning tree");
    else
        System.out.println("The cost of minimum spanning tree is"+sum);
sc.close();
}
}
```


Output:

```
Enter the number of
vertices 6
Enter the weighted
graph 0 3 99 99 6 5 3 0
1 99 99 4 99 1 0 6 99 4
99 99 6 0 8 5 6 99 99 8
0 2 5 4 4 5 2 0
```

```
Enter the source
vertex 1 1->2=3 2->3=1
2->6=4 6->5=2 6->4=5
```

The cost of minimum spanning tree is 15

Experiment No. 10 A)

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

```
import java.util.Scanner;

public class floyd {

    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();
floyd f=new floyd();
f.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();

}

}
```

Output:

```
enter the number
of vertices 4
Enter the weighted
matrix 0 99 3 99 2 0 99
99 99 7 0 1 6 99 99 0
```

```
The shortest path matrix
is 0 10 3 4 2 0 5 6 7 7
0 1 6 16 9 0
```


Experiment No. 10.B

Write Java programs to Implement All-Pairs Shortest Paths problem using Travelling Sales Person problem using Dynamic programming.

```

import java.util.Scanner;

class TSPExp {

    int weight [][] , n , tour [] , finalCost;

    final int INF=1000;

    TSPExp ()
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter no. of
nodes:=>"); n=s.nextInt();
        weight=new int[n][n];
        tour=new int[n-1];
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                if(i!=j)
                {
                    System.out.print("Enter weight of
"+(i+1)+" to "+(j+1)+":=>");
                    weight[i][j]=s.nextInt();
                }
            }
        }

        System.out.println();
        System.out.println("Starting node assumed to be
node 1."); eval();
    }

    public int COST(int currentNode,int inputSet[],int setSize)
    {
        if(setSize==0)
            return weight[currentNode][0];
        int min=INF;
        int setToBePassedOnToNextCallOfCOST[]=new
int[n-1]; for(int i=0;i<setSize;i++)
        {
            int k=0;//initialise new set
            for(int j=0;j<setSize;j++)
            {
                if(inputSet[i]!=inputSet[j])

                setToBePassedOnToNextCallOfCOST[k++]=inputSet[j];
            }
            int
temp=COST(inputSet[i],setToBePassedOnToNextCallOfCOST,setSize-1);
            if((weight[currentNode][inputSet[i]]+temp) <
min)
            {
                min=weight[currentNode][inputSet[i]]+temp;
            }
        }
        return min;
    }
}

```



```

public int MIN(int currentNode,int inputSet[],int setSize)
{
    if(setSize==0)
        return weight[currentNode][0];
    int min=INF,minindex=0;
    int setToBePassedOnToNextCallOfCOST[]= new int[n-1]; for(int
    i=0;i<setSize;i++)//considers each node of inputSet
    {
        int k=0;
        for(int j=0;j<setSize;j++)
        {
            if(inputSet[i]!=inputSet[j])

setToBePassedOnToNextCallOfCOST[k++]=inputSet[j];
        }
        int
temp=COST(inputSet[i],setToBePassedOnToNextCallOfCOST,setSize-1);
        if((weight[currentNode][inputSet[i]]+temp) < min)
        {
            min=weight[currentNode][inputSet[i]]+temp;
            minindex=inputSet[i];
        }
    }
    return minindex;
}

public void eval()
{
    int dummySet[]=new int[n-
    1]; for(int i=1;i<n;i++)
        dummySet[i-1]=i;
    finalCost=COST(0,dummySet,n-
    1); constructTour();
}

public void constructTour()
{
    int previousSet[]=new int[n-1]; int
    nextSet[]=new int[n-2]; for(int
    i=1;i<n;i++) previousSet[i-1]=i; int
    setSize=n-1;
    tour[0]=MIN(0,previousSet,setSize);
    for(int i=1;i<n-1;i++)
    {
        int k=0;
        for(int j=0;j<setSize;j++)
        {
            if(tour[i-1]!=previousSet[j])
            nextSet[k++]=previousSet[j];
        }
        --setSize; tour[i]=MIN(tour[i-
        1],nextSet,setSize); for(int
        j=0;j<setSize;j++)
        previousSet[j]=nextSet[j];
    }
    display();
}

public void display()

```

```
{
System.out.println();
System.out.print("The tour is 1-
"); for(int i=0;i<n-1;i++)
System.out.print((tour[i]+1)+"-
"); System.out.print("1");
System.out.println();
System.out.println("The final cost is "+finalCost);
}

}

class TSP
{
public static void main(String args[])
{
TSPExp obj=new TSPExp();
}
}
}
```

Output:

```
Enter no. of
nodes:=> 4
Enter weight of 1 to 2:=>2
Enter weight of 1 to 3:=>5
Enter weight of 1 to 4:=>7
Enter weight of 2 to 1:=>2
Enter weight of 2 to 3:=>8
Enter weight of 2 to 4:=>3
Enter weight of 3 to 1:=>5
Enter weight of 3 to 2:=>8
Enter weight of 3 to 4:=>1
Enter weight of 4 to 1:=>7
Enter weight of 4 to 2:=> 3
Enter weight of 4 to 3:=>1

Starting node assumed to be node 1.

The tour is 1-2-4-3-1
The final cost is 11
```

Experiment No. 11

Design and implement in Java to find a **subset** of a given set $S = \{S_1, S_2, \dots, S_n\}$ 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.Scanner;
import static java.lang.Math.pow;

public class subSet {

    /**
     * @param
     * args */
    void subset(int num,int n, int x[])
    {
        int i;
        for(i=1;i<=n;i++)
            x[i]=0;
        for(i=n;num!=0;i--)
        {
            x[i]=num%2;
            num=num/2;
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated
        method stub int a[]=new int[10];
        int x[]=new int[10]; int
        n,d,sum,present=0;
        int j;
        System.out.println("enter the number of elements
        of set"); Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
        System.out.println("enter the elements
        of set"); for(int i=1;i<=n;i++)
        a[i]=sc.nextInt();
        System.out.println("enter the positive
        integer sum"); d=sc.nextInt();
        if(d>0)
        {
            for(int i=1;i<=Math.pow(2,n)-1;i++)
            {
                subSet s=new subSet();
                s.subset(i,n,x);
                sum=0;
                for(j=1;j<=n;j++)
                if(x[j]==1)
```

```
        sum=sum+a[j];
    if(d==sum)
    {
        System.out.print("Subset={");
        present=1;
        for(j=1;j<=n;j++)
            if(x[j]==1)
                System.out.print(a[j]+",");

        System.out.print("}="+d);
        System.out.println();
    }
}

}
if(present==0)
    System.out.println("Solution does not exists");

}

}
```

Output:

```
enter the number of elements
of set 5
enter the elements
of set 1 2 5 6 8
enter the positive
integer sum 9
Subset={1, 8, }=9
Subset={1, 2, 6, }=9
```

Experiment No. 12

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

```
import java.util.*;

class Hamiltoniancycle
{
    private int adj[][] , x[] , n;
    public Hamiltoniancycle()
    {
        Scanner src = new Scanner(System.in);
        System.out.println("Enter the number
of nodes"); n=src.nextInt();
        x=new int[n];
        x[0]=0;
        for (int i=1;i<n;
            i++) x[i]=-1;
        adj=new int[n][n];
        System.out.println("Enter the adjacency
matrix"); for (int i=0;i<n; i++)
            for (int j=0; j<n; j++)
                adj[i][j]=src.nextInt();
    }

    public void nextValue (int k)
    {
        int i=0;
        while(true)
        {
            x[k]=x[k]+1;
            if (x[k]==n)
                x[k]=-1; if
                (x[k]==-1)
                    return;
            if (adj[x[k-1]][ x[k]]==1)
                for (i=0; i<k; i++)
                    if (x[i]==x[k])
                        break;
            if (i==k)
                if (k<n-1 || k==n-1 &&
                    adj[x[n-1]][0]==1) return;
        }
    }

    public void getHCycle(int k)
    {
        while(true)
        {
            nextValue(k);
            if (x[k]==-1)
                return;
            if (k==n-1)
            {
                System.out.println("\nSolution :
"); for (int i=0; i<n; i++)
                    System.out.print((x[i]+1)+"
"); System.out.println(1);
            }
            else getHCycle(k+1);
        }
    }
}
```

```
    }  
    }  
    }  
    class HamiltoniancycleExp  
    {  
        public static void main(String args[])  
        {  
            Hamiltoniancycle obj=new  
            Hamiltoniancycle(); obj.getHCycle(1);  
        }  
    }  
}
```

Output:

```
Enter the number  
of nodes 6  
Enter the adjacency  
matrix 0 1 1 1 0 0 1 0 1  
0 0 1 1 1 0 1 1 0 1 0 1  
0 1 0 0 0 1 1 0 1 0 1 0  
0 1 0
```

Solution :

1 2 6 5 3 4 1

Solution :

1 2 6 5 4 3 1

Solution :

1 3 2 6 5 4 1

Solution :

1 3 4 5 6 2 1

Solution :

1 4 3 5 6 2 1

Solution :

1 4 5 6 2 3 1