EXPERIMENT DETAILS

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 n Student objects and print the USN, Name, Branch, and Phone of these objects with suitable headings.

……..

```
import java.util.Scanner;
class Student
{
	String USN;
	String name;
	String branch;
	String phone;
	public void getInfo()
	{
		Scanner in=new Scanner(System.in);
		System.out.println("Enter the USN:");
		USN=in.next();
		System.out.println("Enter the Name:");
		name=in.next();
		System.out.println("Enter the Branch:");
		branch=in.next();
		System.out.println("Enter the Phone No:");
		phone=in.next();
	}
	public void display()
	{
		System.out.println(USN+"\t"+name+"\t"+branch+"\t\t"+phone);
	}
}
public class Prog1a
{
	public static void main(String[] args)
	{
		int i;
		Student[] S=new Student[100];
```

```
            Scanner in=new Scanner(System.in);
            System.out.println("Enter the number of students");
            int n=in.nextInt();
            System.out.println("Enter the Student details");
            for(i=0;i<n;i++)
            {
                  S[i]=new Student();
                  S[i].getInfo();
            }
            System.out.println("USN\t\tName\tBranch\tPhone");
            System.out.println(".....……");
            for(i=0;i<n;i++)
            {
                  S[i].display();
            }
      }
}
```

OUTPUT:

```
Enter the number of students
3
Enter the Student details
Enter the USN:
4SF16CS010
Enter the Name:
Anu
Enter the Branch:
CS
Enter the Phone No:
999999999
Enter the USN:
4SF16EC010
Enter the Name:
Kishan
Enter the Branch:
EC
Enter the Phone No:
9997766666
Enter the USN:
4SF16ME001
```

```
Enter the Name:
Rahul
Enter the Branch:
ME
Enter the Phone No:
6666666666
USN          Name          Branch          Phone
.....
4SF16CS010        Anu          CS          999999999
4SF16EC010        Kishan       EC          9997766666
4SF16ME001        Rahul        ME          6666666666
```

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;
class Stack
{
    int top;
    int max;
    int[] stk;
    Stack()
    {
        top=-1;
        max=5;
        stk  = new int[max];
    }
    public  void push()
    {
    int elem;
    if(top==max-1)
        System.out.println("Stack overflow");
    else
    {
        System.out.println("Enter the element");
        Scanner in = new Scanner(System.in);
        elem = in.nextInt();
        top = top+1;
        stk[top]=elem;
```

```
		}
	}
	public void pop()
	{
		int a;
		if(top==-1)
			System.out.println("Stack Underflow");
		else
		{
			a=stk[top];
			top=top-1;
			System.out.println("Popped element is "+a);
		}
	}
	public void display()
	{
		if(top==-1)
			System.out.println("Stack is empty");
		else
		{
			System.out.println("Stack elements are");
			for(int i=top; i>=0; i--)
				System.out.println(stk[i]);
		}
	}
}
public class program1b
{
	public static void main(String[] args)
	{
		Scanner in = new Scanner(System.in);
		Stack obj=new Stack();
		while(true)
		{
			System.out.println("-----------Menu-----------");
			System.out.println("1.Push\t2.Pop\t3.Display");
			System.out.println("--------------------------");
			System.out.println("Enter your choice");
			int ch = in.nextInt();
			switch(ch)
```

```
                {
                        case 1:obj.push();
                                break;
                        case 2:obj.pop();
                                break;
                        case 3:obj.display();
                                break;
                        default:System.out.println("Invalid choice");
                                return;

                }
        }
}
}
```

OUTPUT:

```
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
1
Enter the element
1
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
1
Enter the element
2
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
1
Enter the element
3
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
```

```
Enter your choice
1
Enter the element
4
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
1
Enter the element
5
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
1
Stack overflow
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
3
Stack elements are
5
4
3
2
1
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Popped element is 5

----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Popped element is 4
```

```
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Popped element is 3
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Popped element is 2
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Popped element is 1
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
2
Stack Underflow
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
3
Stack is empty
----------Menu----------
1.Push        2.Pop   3.Display
--------------------------
Enter your choice
4
Invalid choice
```

2. a. Design a superclass 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;
	String name;
	String phn;
	int salary;
	staff(String sid,String Name,String Phn,int sal)
	{
		staffid=sid;
		name=Name;
		phn=Phn;
		salary=sal;
	}
	void display()
	{
		System.out.print(staffid+"\t"+name+"\t"+phn+"\t"+salary+"\t");
	}
}
class teaching extends staff
{
	String domain,publication;
	teaching(String staffid,String name,String phn,int salary,String Domain,String pub)
	{
		super(staffid,name,phn,salary);
		domain=Domain;
		publication=pub;
	}
	void display()
	{
		System.out.print("\nTeaching:\t");
		super.display();
		System.out.print(domain+"\t"+publication+"\t\t****\t****");
	}
}
```

```
class technical extends staff
{
	String skill;
	technical(String staffid,String name,String phn,int salary,String Skill)
	 {
		super(staffid,name,phn,salary);
		skill=Skill;
	}
	void display()
	{
		System.out.print("\nTechnical:\t");
		super.display();
		System.out.print("****\t****\t\t"+skill+"\t****");
	}
}
class contract extends staff
{
	String period;
	contract(String staffid,String name,String phn,int salary,String Period)
	{
		super(staffid,name,phn,salary);
		period=Period;
	}
	void display()
	{
		System.out.print("\nContract:\t");
		super.display();
		System.out.print("****\t****\t\t****\t"+period);
	}
}
public class prg2
{
	public static void main(String[] args)
	{
		staff[] s=new staff[3];
		s[0]=new teaching("1111","Ram","9999999999",50000,"CSE","Manthan");
		s[1]=new technical("2222","Raj","8888888888",15000,"Java");
		s[2]=new contract("3333","Adithi","7777777777",10000,"10");
		System.out.print("\t\tStaffId\tName\tPhoneNo\tSalary\tDomain\tPublication\tSkills\t
	Period");
```

```
            System.out.print("\n----------------------------------------------------------------------------
      ----------");
            for(int i=0;i<3;i++)
            {
                  s[i].display();
                  System.out.println();
            }
      }

}
```

OUTPUT:

	StaffId	Name	Phone No	Salary	Domain	Publication	Skills	Period
Teaching:	1111	Ram	9999999999	50000	CSE	Manthan	****	****
Technical:	2222	Raj	8888888888	15000	****	****	Java ****	
Contract:	3333	Adithi	7777777777	10000	****	****	****	10

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.Scanner;
import java.util.StringTokenizer;
class Customer
{
	String cname;
	String DOB;
	void readData()
	{
		Scanner in=new Scanner(System.in);
		System.out.println("Enter the customer name");
		cname=in.nextLine();
		System.out.println("Enter the Date of Birth");
		DOB=in.next();
	}
	void display()
	{
		StringTokenizer st=new StringTokenizer(DOB,"/");
```

```
                System.out.print(cname);
                while(st.hasMoreTokens())
                {
                        System.out.print(", "+st.nextToken());
                }
        }
}
public class Prog2b
{
        public static void main(String[] args)
        {
                Customer c=new Customer();
                c.readData();
                c.display();

        }

}
```

OUTPUT:

```
Enter the customer name
Preetham
Enter the Date of Birth

07/03/1992
Preetham, 07, 03, 1992
```

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.Scanner;
public class prog3a
{
        public static void main(String[] args)
        {
                Scanner in=new Scanner(System.in);
                System.out.println("Enter two numbers");
                int a=in.nextInt();
                int b=in.nextInt();
                try
                {
                        if(b==0)
```

```
                    throw new ArithmeticException("Devide by 0 error");
                int result=a/b;
                System.out.println(a + "/" + b + "=" + result);
            }
            catch(ArithmeticException e)
            {
                System.out.println("Raised exception is " + e);
            }
            finally
            {
                System.out.println("Execution Finished");
            }
        }

}
```

OUTPUT:

```
Enter two numbers
8
0
Raised exception is java.lang.ArithmeticException: Devide by 0 error
Execution Finished
```

……

```
Enter two numbers
9
3
9/3=3
Execution Finished
```

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 RandomNumThread extends Thread
{
    Random rand = new Random();
    public void run()
    {
```

```
            try
            {
                for(int i=0;i<10;i++)
                {
                    System.out.println("Number generated is " + rand.nextInt(100)+" ");
                    Thread.sleep(1000);
                }
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
}
class SquareNum extends Thread
{
        int num;

        SquareNum(int a)
        {
            num = a;
        }
        public void run()
        {
            for(int i=1;i<num;i++)
                System.out.println("Square of "+ i + " is:" + i*i);
        }
}
class CubeNum extends Thread
{
        int n;
        CubeNum(int a)
        {
            n = a;
        }
        public void run()
        {
            for(int i=1;i<10;i++)
                System.out.println("Cube of "+ i +" is:" + i*i*i);
        }
```

```
}
public class prog3b
{
	public static void main(String[] args)
	{
		Thread thread1 = new RandomNumThread();
		thread1.start();
		Thread thread2 = new SquareNum(10);
		thread2.start();
		Thread thread3 = new CubeNum(10);
		thread3.start();
	}
}
```

OUTPUT:

```
Number generated is 27
Square of 1 is:1
Square of 2 is:4
Square of 3 is:9
Square of 4 is:16
Square of 5 is:25
Square of 6 is:36
Square of 7 is:49
Square of 8 is:64
Square of 9 is:81
Cube of 1 is:1
Cube of 2 is:8
Cube of 3 is:27
Cube of 4 is:64
Cube of 5 is:125
Cube of 6 is:216
Cube of 7 is:343
Cube of 8 is:512
Cube of 9 is:729
Number generated is 67
Number generated is 30
Number generated is 54
Number generated is 79
Number generated is 67
Number generated is 0
Number generated is 34
Number generated is 90
Number generated is 37
```

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;
class QuickSort
{
	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 pos=partition(a,low,high);
                        sort(a,low,pos-1);
                        sort(a,pos+1,high);
                }
        }
}
public class prg4
{
        public static void main(String[] args)
        {
                int i,n=0;
                Scanner in = new Scanner(System.in);
                int[] a=new int[10000];
                QuickSort obj=new QuickSort();
                System.out.println("Press 1 to read numbers from the keyboard\nPress 2 to generate
                random numbers\nEnter your choice:");
                int choice = in.nextInt();
                switch(choice)
                {
                        case 1:System.out.println("Enter the number of elements to be sorted");
                                n = in.nextInt();
                                System.out.println("Enter "+n+" numbers");
                                for(i=0;i<n;i++)
                                {
                                        a[i]=in.nextInt();
                                }
                                break;
                        case 2:System.out.println("Enter the number of elements to be sorted");
                                n = in.nextInt();
                                Random rand= new Random();
                                for(i=0;i<n;i++)
                                {
                                        a[i]=rand.nextInt(1000);
                                }
```

```
                        break;
                default:System.out.println("Wrong Choice");
                                System.exit(0);
        }
        System.out.println("\nArray elements to be sorted are");
        for(i=0; i<n; i++)
                System.out.print(a[i]+" ");

        long start=System.nanoTime();
        obj.sort(a,0,n-1);
        long end=System.nanoTime();

        System.out.println("\n\nThe sorted elements are");
        for(i=0; i<n; i++)
                System.out.print(a[i]+" ");
        System.out.println("\n\nThe time taken to sort elemets by quick sort method is
        "+(end-start)+" nano seconds");
    }
}
```

OUTPUT:

```
Press 1 to read numbers from the keyboard
Press 2 to generate random numbers
Enter your choice:
2
Enter the number of elements to be sorted
100

Array elements to be sorted are
53 937 820 235 912 615 535 716 146 648 134 11 326 692 938 981 740 158 754 431 106 922 530
511 85 279 835 517 941 492 627 829 686 843 413 707 139 447 356 563 245 555 594 949 993 376
677 786 359 234 320 525 886 376 423 625 654 662 787 810 933 472 166 2 59 350 249 612 401
191 345 765 396 627 889 747 24 921 496 477 23 433 307 789 513 127 381 526 269 612 410 342
949 923 251 9 463 773 44 247

The sorted elements are
2 9 11 23 24 44 53 59 85 106 127 134 139 146 158 166 191 234 235 245 247 249 251 269 279
307 320 326 342 345 350 356 359 376 376 381 396 401 410 413 423 431 433 447 463 472 477
492 496 511 513 517 525 526 530 535 555 563 594 612 612 615 625 627 627 648 654 662 677
```

686 692 707 716 740 747 754 765 773 786 787 789 810 820 829 835 843 886 889 912 921 922 923 933 937 938 941 949 949 981 993

The time taken to sort elemets by quick sort method is 107859 nano seconds

……

Press 1 to read numbers from the keyboard
Press 2 to generate random numbers
Enter your choice:
2
Enter the number of elements to be sorted
200

Array elements to be sorted are
141 260 354 892 177 570 88 642 292 194 659 126 65 912 47 99 407 441 572 981 81 464 288 370 463 145 568 811 569 184 223 59 756 741 657 228 381 38 545 168 235 912 295 602 409 708 469 753 713 546 176 478 364 685 988 739 861 843 400 256 749 926 633 547 481 860 659 907 729 688 658 202 306 362 782 116 898 90 124 616 371 76 169 650 344 352 481 842 981 322 758 37 300 809 173 789 130 453 203 776 976 706 277 82 836 292 500 8 664 270 550 82 338 882 249 940 923 87 507 402 468 650 286 717 321 171 951 102 393 966 637 957 534 617 902 390 287 408 173 833 718 443 512 189 51 553 190 697 830 124 486 222 76 936 426 572 593 617 459 288 708 40 365 31 304 537 483 889 886 677 582 348 752 79 502 92 689 204 875 909 792 772 109 717 211 352 816 320 975 87 764 984 246 441 112 201 546 223 958 455

The sorted elements are
8 31 37 38 40 47 51 59 65 76 76 79 81 82 82 87 87 88 90 92 99 102 109 112 116 124 124 126 130 141 145 168 169 171 173 173 176 177 184 189 190 194 201 202 203 204 211 222 223 223 228 235 246 249 256 260 270 277 286 287 288 288 292 292 295 300 304 306 320 321 322 338 344 348 352 352 354 362 364 365 370 371 381 390 393 400 402 407 408 409 426 441 441 443 453 455 459 463 464 468 469 478 481 481 483 486 500 502 507 512 534 537 545 546 546 547 550 553 568 569 570 572 572 582 593 602 616 617 617 633 637 642 650 650 657 658 659 659 664 677 685 688 689 697 706 708 708 713 717 717 718 729 739 741 749 752 753 756 758 764 772 776 782 789 792 809 811 816 830 833 836 842 843 860 861 875 882 886 889 892 898 902 907 909 912 912 923 926 936 940 951 957 958 966 975 976 981 981 984 988

The time taken to sort elemets by quick sort method is 155871 nano seconds

……

Press 1 to read numbers from the keyboard

```
Press 2 to generate random numbers
Enter your choice:
1
Enter the number of elements to be sorted
5
Enter 5 numbers
56
23
78
10
45

Array elements to be sorted are
56 23 78 10 45

The sorted elements are
10 23 45 56 78

The time taken to sort elemets by quick sort method is 25319 nano seconds
```

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;
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=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 mergeSort( int[] array,int low,int high)
    {
        if(low<high)
        {
            int mid=(low+high)/2;
            mergeSort(array,low,mid);
            mergeSort(array,mid+1,high);
            merge(array,low,mid,high);
        }
    }
}
public class prog5
```

```
{
	public static void main(String[] args)
	{
		int i,n=0;
		Scanner in = new Scanner(System.in);
		MergeSort obj=new MergeSort();
		int[] a=new int[10000];
		System.out.println("Press 1 to read numbers from the keyboard\nPress 2 to
		generate random numbers\nEnter your choice:");
		int choice = in.nextInt();
		switch(choice)
		{
			case 1:System.out.println("Enter the number of elements to be sorted");
				n = in.nextInt();
				System.out.println("Enter "+n+" numbers");
				for(i=0;i<n;i++)
				{
					a[i]=in.nextInt();
				}
				break;
			case 2:System.out.println("Enter the number of elements to be sorted");
			      n = in.nextInt();
				Random rand= new Random();
				for(i=0;i<n;i++)
				{
					a[i]=rand.nextInt(1000);
				}
				break;
			default:System.out.println("Wrong Choice");
					System.exit(0);
		}
		System.out.println("\nArray elements to be sorted are");
		for(i=0; i<n; i++)
			System.out.print(a[i]+" ");

		long start=System.nanoTime();

		obj.mergeSort(a,0,n-1);
		long end=System.nanoTime();
```

```
            System.out.println("\n\nThe sorted elements are");
            for(i=0; i<n; i++)
                  System.out.print(a[i]+" ");
            System.out.println("\n\nThe time taken to sort elements by merge sort method is
            "+(end-start)+" nano seconds");
      }
}
```

OUTPUT:

```
Press 1 to read numbers from the keyboard
Press 2 to generate random numbers
Enter your choice:
1
Enter the number of elements to be sorted
5
Enter 5 numbers
23
90
67
45
99

Array elements to be sorted are
23 90 67 45 99

The sorted elements are
23 45 67 90 99

The time taken to sort elements by merge sort method is 93159 nano seconds

……

Press 1 to read numbers from the keyboard
Press 2 to generate random numbers
Enter your choice:
2
Enter the number of elements to be sorted
100

Array elements to be sorted are
```

516 419 45 992 652 413 302 537 382 62 932 58 199 764 482 590 395 888 441 592 663 242 573 790 867 266 846 563 903 814 301 167 630 245 407 470 952 62 766 794 233 350 58 957 7 100 166 175 361 907 183 176 34 852 536 730 469 596 912 910 131 905 397 586 15 461 850 268 259 485 828 454 443 632 226 846 815 743 714 840 463 102 97 242 383 860 266 524 465 812 288 754 313 752 71 623 452 719 876 391

The sorted elements are
7 15 34 45 58 58 62 62 71 97 100 102 131 166 167 175 176 183 199 226 233 242 242 245 259 266 266 268 288 301 302 313 350 361 382 383 391 395 397 407 413 419 441 443 452 454 461 463 465 469 470 482 485 516 524 536 537 563 573 586 590 592 596 623 630 632 652 663 714 719 730 743 752 754 764 766 790 794 812 814 815 828 840 846 846 850 852 860 867 876 888 903 905 907 910 912 932 952 957 992

The time taken to sort elements by merge sort method is 2005755 nano seconds

……

Press 1 to read numbers from the keyboard
Press 2 to generate random numbers
Enter your choice:
2
Enter the number of elements to be sorted
200

Array elements to be sorted are
418 994 658 962 29 175 500 43 209 592 673 532 279 350 977 938 542 218 416 781 964 40 176 284 339 852 762 527 542 211 16 825 682 689 774 832 954 10 47 233 634 55 476 93 197 216 487 762 410 32 814 639 497 953 402 697 812 501 951 923 696 222 638 317 755 35 438 755 839 407 578 200 988 335 236 192 45 784 782 855 712 11 841 169 385 934 391 159 912 929 834 426 142 113 216 8 495 676 638 240 537 58 833 867 516 948 259 449 505 267 906 237 596 904 845 535 430 988 16 772 915 747 88 811 339 881 472 62 68 337 520 915 956 926 591 455 532 376 815 830 424 734 55 481 91 898 111 916 727 629 28 442 154 527 947 984 532 451 682 396 981 749 242 48 683 473 553 150 161 9 324 465 44 547 225 925 271 213 674 469 798 70 326 191 431 802 113 590 228 105 804 524 525 908 631 154 643 725 307 85

The sorted elements are
8 9 10 11 16 16 28 29 32 35 40 43 44 45 47 48 55 55 58 62 68 70 85 88 91 93 105 111 113 113 142 150 154 154 159 161 169 175 176 191 192 197 200 209 211 213 216 216 218 222 225 228 233 236 237 240 242 259 267 271 279 284 307 317 324 326 335 337 339 339 350 376 385 391 396 402 407 410 416 418 424 426 430 431 438 442 449 451 455 465 469 472 473 476 481 487

495 497 500 501 505 516 520 524 525 527 527 532 532 532 535 537 542 542 547 553 578 590 591 592 596 629 631 634 638 638 639 643 658 673 674 676 682 682 683 689 696 697 712 725 727 734 747 749 755 755 762 762 772 774 781 782 784 798 802 804 811 812 814 815 825 830 832 833 834 839 841 845 852 855 867 881 898 904 906 908 912 915 915 916 923 925 926 929 934 938 947 948 951 953 954 956 962 964 977 981 984 988 988 994

The time taken to sort elements by merge sort method is 4176019 nano seconds

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

……

```
import java.util.Scanner;
class knapsackDP
{
	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 class prog6a
{
        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 of "+ n +" elements");
                for (int i = 1; i <= n; i++)
                        wt[i] = scan.nextInt();
                System.out.println("\nEnter values of "+ n +" elements");
                for (int i = 1; i <= n; i++)
                        val[i] = scan.nextInt();
                System.out.println("\nEnter knapsack capacity ");
                int W = scan.nextInt();
                ks.solve(wt, val, W, n);
        }
}
```

OUTPUT:

```
Enter number of elements
4

Enter weight of 4 elements
2 1 3 2

Enter values of 4 elements
12 10 20 15

Enter knapsack capacity
5
The optimal solution is 37

Items selected :
1 2 4
```

(b) Greedy method.

……

```
import java.util.Scanner;
public class prg6b
{
	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
The total profit is 10.0
Quantity of item number: 4 added is 2
The total profit is 25.0
Quantity of item number: 3 added is 2
The total profit is 38.333332
```

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.*;
class Dijkstra
{
	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=1;v<=n;v++)
		{
			d[v]=99;
			p[v]=-1;
		}
		d[s]=0;
		for(i=1;i<=n;i++)
		{
			min=99;
			for(j=1;j<=n;j++)
			{
				if(d[j]<min&& visited[j]==0)
				{
					min=d[j];
					u=j;
				}

			}
			visited[u]=1;
			for(v=1;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=1;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 class prog7
{
        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=1;i<=n;i++)
                for(j=1;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);
        }
}
```

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
7    2    5     0     4
99   99   6     4    0

enter the source vertex
1
the shortest path between source 1 to remaining vertices are

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

8. Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program.

……

```
import java.util.Scanner;
class Krushkal
{
	int kruskals(int n,  int cost[][])
	{
		int k = 1, a = 0, u = 0, b = 0, v = 0, min, mincost = 0;
		int parent[] = new int[10];
		while (k < n)
		{
			min = 999;
			for (int i = 1; i <= n; i++)
			{
				for (int j = 1; j <= n; j++)
				{
```

```
                                        if (cost[i][j] < min)
                                        {
                                                min = cost[i][j];
                                                a = u = i;
                                                b = v = j;
                                        }
                                }
                        }
                        while (parent[u] > 0)
                                u = parent[u];
                        while (parent[v] > 0)
                                v = parent[v];
                        if (u != v)
                        {
                                System.out.println((k++) + ">minimum edge is :");
                                System.out.println("(" + a + "," + b +
                                                ") and its cost is:" + min);
                                mincost += min;
                                parent[v] = u;
                        }
                        cost[a][b] = cost[b][a] = 999;
                }
                return mincost;
        }
}
public class prg8
{
        public static void main(String[] args)
        {
                int cost[][] = new int[10][10];
                int i, j, mincost = 0;
                Scanner in = new Scanner(System.in);
                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++)
                        {
                                cost[i][j] = in.nextInt();
```

```
                }
            }
            System.out.println("The entered cost matrix is");
            for (i = 1; i <= n; i++)
            {
                for (j = 1; j <= n; j++)
                {
                    System.out.print(cost[i][j] + "\t");
                }
                System.out.println();
            }
            Krushkal k=new Krushkal();
            mincost = k.kruskals(n, cost);
            System.out.println("The minimum spanning tree cost is:");
            System.out.println(mincost);
        }
}
```

OUTPUT:

```
Enter the number of nodes:
7
Enter the cost matrix
0    28   999  999  999  10   999
28   0    16   999  999  999  14
999  16   999  12   999  999  999
999  999  12   0    22   999  18
999  999  999  22   0    25   24
10   999  999  999  25   0    999
999  14   999  18   24   999  0
The entered cost matrix is
0    28   999  999  999  10   999
28   0    16   999  999  999  14
999  16   999  12   999  999  999
999  999  12   0    22   999  18
999  999  999  22   0    25   24
10   999  999  999  25   0    999
999  14   999  18   24   999  0

1>minimum edge is :
(1,6) and its cost is:10
2>minimum edge is :
```

(3,4) and its cost is:12
3>minimum edge is :
(2,7) and its cost is:14
4>minimum edge is :
(2,3) and its cost is:16
5>minimum edge is :
(4,5) and its cost is:22
6>minimum edge is :
(5,6) and its cost is:25
The minimum spanning tree cost is:
99

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

……

```
import java.util.Scanner;
public class prg9
{
	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];
			Scanner sc=new Scanner(System.in);
			System.out.println("Enter the number of vertices");
			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
7
Enter the weighted graph
0    28    99    99    99    10    99
28   0     16    99    99    99    14
99   16    99    12    99    99    99
99   99    12    0     22    99    18
99   99    99    22    0     25    24
10   99    99    99    25    0     99
99   14    99    18    24    99    0
Enter the source vertex
1
1->6=10
6->5=25
5->4=22
4->3=12
3->2=16
2->7=14
The cost of minimum spanning tree is 99
```

10. Write Java programs to

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

……

```
import java.util.Scanner;
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 class prg10a
{
	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("Enterthe 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
7
Enterthe weighted matrix
0    28    99    99    99    10    99
28   0     16    99    99    99    14
99   16    99    12    99    99    99
99   99    12    0     22    99    18
99   99    99    22    0     25    24
10   99    99    99    25    0     99
99   14    99    18    24    99    0
The shortest path matrix is
0 28 44 56 35 10 42
28 0 16 28 38 38 14
44 16 24 12 34 54 30
56 28 12 0 22 47 18
```

35 38 34 22 0 25 24
10 38 54 47 25 0 49
42 14 30 18 24 49 0

b. Implement Travelling Sales Person problem using Dynamic programming.

……

```
import java.util.*;
class Sales
{
        int cost_opt;
        int a[][]=new int[100][100];
        int visit[]=new int[100];
        void mincost_opt(int city,int n)
```

```
    {
        int i,ncity;
        visit[city]=1;
        System.out.print(city+"-->");
        ncity=least_opt(city,n);
        if(ncity==999)
        {
            ncity=1;
            System.out.println(ncity);
            cost_opt+=a[city][ncity];
            return;
        }
        mincost_opt(ncity,n);
    }
    int least_opt(int c,int n)
    {
        int i,nc=999;
        int min=999,kmin=999;
        for(i=1;i<=n;i++)
        {
            if((a[c][i]!=0)&&visit[i]==0)
            if(a[c][i]<min)
            {
                min=a[i][1]+a[c][i];
                kmin=a[c][i];
                nc=i;
            }
        }
        if(min!=999)
            cost_opt+=kmin;
        return nc;
    }
}
public class prg10b
{
    public static void main(String[] args)
    {
        int i,j;
        Sales x=new Sales();
        Scanner sc=new Scanner(System.in);
        System.out.println("enter no of cities:");
```

```
            int n=sc.nextInt();
            System.out.println("enter the cost matrix");
            for(i=1;i<=n;i++)
            {
                  for(j=1;j<=n;j++)
                        x.a[i][j]=sc.nextInt();
                        x.visit[i]=0;
            }
            System.out.println("the cost line is");
            for(i=1;i<=n;i++)
            {
                  for(j=1;j<=n;j++)
                        System.out.print(x.a[i][j]+"\t");
                  System.out.println();
            }
            System.out.println("optimal solution");
            System.out.println("the path is:");
            x.mincost_opt(1,n);
            System.out.println("minimum cost is "+x.cost_opt);
      }

}
```

OUTPUT:

```
enter no of cities:
4
enter the cost matrix
0 1 3 6
1 0 2 3
3 2 0 1
6 3 1 0
the cost line is
0       1       3       6
1       0       2       3
3       2       0       1
6       3       1       0
optimal solution
the path is:
1-->2-->4-->3-->1
minimum cost is 8
```

11. 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.Scanner;
import static java.lang.Math.pow;
public class prg11
{
	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)
	{
		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();
		prg11 s=new prg11();
		if(d>0)
```

```
        {
            for(int i=1;i<=Math.pow(2,n)-1;i++)
            {
                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
```

……

```
enter the number of elements of set
3
```

```
enter the elements of set
2
3
5
enter the positive integer sum
11
Solution does not exists
```

12. Design and implement in Java to find all Hamiltonian Cycles in a connected undirected Graph G of n vertices using backtracking principle.

……

```
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 prg12
{
	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
```

VIVA QUESTIONS

1. What is an algorithm?

An algorithm is a sequence of unambiguous instructions for solving a problem. i.e., for obtaining a required output for any legitimate input in a finite amount of time.

2. What are important problem types? (or) Enumerate some important types of problems.

1. Sorting 2. Searching 3. Numerical problems 4. Geometric problems 5. Combinatorial Problem 6. Graph Problems 7. String processing Problems

3. Name some basic Efficiency classes

1. Constant 2. Logarithmic 3. Linear 4. Nlogn 5. Quadratic 6. Cubic 7. Exponential 8. Factorial

4. What are algorithm design techniques?

Algorithm design techniques (or strategies or paradigms) are general approaches to solving problems algorithmatically, applicable to a variety of problems from different areas of computing. General design techniques are: (i) Brute force (ii) divide and conquer (iii) decrease and conquer (iv) transform and concquer (v) greedy technique (vi) dynamic programming (vii) backtracking (viii) branch and bound

5. How is an algorithm's time efficiency measured?

Time efficiency indicates how fast the algorithm runs. An algorithm's time efficiency is measured as a function of its input size by counting the number of times its basic operation(running time) is executed. Basic operation is the most time consuming operation in the algorithm's innermost loop.

6. How is the efficiency of the algorithm defined?

The efficiency of an algorithm is defined with the components.

(i) Time efficiency -indicates how fast the algorithm runs

(ii) Space efficiency -indicates how much extra memory the algorithm needs.

7. What are the characteristics of an algorithm?

Every algorithm should have the following five characteristics

(i) Input

(ii) Output

(iii) Definiteness

(iv) Effectiveness

(v) Termination

Therefore, an algorithm can be defined as a sequence of definite and effective instructions, which terminates with the production of correct output from the given input.

8. Write general plan for analyzing non-recursive algorithms.

i. Decide on parameter indicating an input's size.

ii. Identify the algorithm's basic operation

iii. Checking the no.of times basic operation executed depends on size of input.

iv. Set up sum expressing the no.of times the basic operation is executed. depends on some additional property,then best,worst,avg.cases need to be investigated (establishing order of growth)

9. Define the terms: pseudocode, flow chart

A pseudocode is a mixture of a natural language and programming language like constructs. A pseudocode is usually more precise than natural language. A flowchart is a method of expressing an algorithm by a collection of connected geometric shapes containing descriptions of the algorithm's steps.

10. write general plan for analyzing recursive algorithms.

i. Decide on parameter indicating an input's size.

ii. Identify the algorithm's basic operation

iii. Checking the no.of times basic operation executed depends on size of input.if it depends on some additional property,then best,worst,avg.cases need to be investigated of times the basic operation is executed

iv. Set up the recurrence relation,with an appropriate initial condition,for the number

v. Solve recurrence (establishing order of growth)

11. Define the divide an conquer method.

Given a function to compute on 'n' inputs the divide-and-comquer strategy suggests splitting the inputs in to'k' distinct susbsets, 1<k <n, yielding 'k' subproblems. The subproblems must be solved recursively, and then a method must be found to combine subsolutions into a solution of the whole.

12. What is Merge sort?

Merge sort is an O (n log n) comparison-based sorting algorithm. Where the given array is divided into two equal parts. The left part of the array as well as the right part of the array is sorted recursively. Later, both the left sorted part and right sorted part are merged into a single sorted array.

13. What is general divide and conquer recurrence?

Time efficiency T(n)of many divide and conquer algorithms satisfies the equation T(n)=a.T(n/b)+f(n).This is the general recurrence relation.

14. Describe the recurrence relation for merge sort?

If the time for the merging operation is proportional to n, then the computing time of merge sort is described by the recurrence relationT(n) = a n = 1, a a constant 2T (n/2) + n n >1, c a constant

15. The relation between order of growth of functions

O(1) < O(log n) < O(n) < O(n * log n) < O(n2) < O(n3) < O(2n)

16. Asymptotic notations

Big Oh(Worst Case), Big Theta (Average Case), Big Omega(Best Case)

17) What is the time complexity of linear search?
Θ(n)

18) What is the time complexity of binary search?
Θ(log2n)

19) What is the major requirement for binary search?
The given list should be sorted.

20) What is binary search?
It is an efficient method of finding out a required item from a given list, provided the list is in order.
The process is:
1. First the middle item of the sorted list is found.
2. Compare the item with this element.
3. If they are equal search is complete.
4. If the middle element is greater than the item being searched, this process is repeated in the upper half of the list.
5. If the middle element is lesser than the item being searched, this process is repeated in the lower half of the list.

21) What is parental dominance?
The key at each node is greater than or equal to the keys at its children.

22) What is heap?
A heap can be defined as a binary tree with keys assigned to its nodes (one key per node) provided the following two conditions are met:
1 The tree's shape requirement – The binary tree is essentially complete (or simply complete), that is, all its levels are full except possibly the last level, where only some rightmost leaves may be missing.
2. The parental dominance requirement – The key at each node is greater than or equal to the keys at its children

23) What is height of Binary tree?
It is the longest path from the root to any leaf.

24) What is the time complexity of heap sort?
Θ(n logn)

25) Who invented Merge Sort?

John Von Neumann in 1945.

26) On which paradigm is it based?
Merge-sort is based on the divide-and-conquer paradigm.

27) What is the time complexity of merge sort?
n log n.

28) What is the space requirement of merge sort?
Space requirement: Θ(n) (not in-place).

30) When do you say an algorithm is stable and in place?
Stable – if it retains the relative ordering. In – place if it does not use extra memory.

31) Who invented Dijkstra's Algorithm?
Edsger Dijkstra invented this algorithm.

32) What is the other name for Dijkstra's Algorithm?
Single Source Shortest Path Algorithm.

33) Which technique the Dijkstra's algorithm is based on?
Greedy Technique.

34) What is the purpose of Dijkstra's Algorithm?
To find the shortest path from source vertex to all other remaining vertices

35) Name the different algorithms based on Greedy technique.
Prim's Algorithm, Kruskal's algorithm

36) What is the constraint on Dijkstra's Algorithm?
This algorithm is applicable to graphs with non-negative weights only.

37) What is a Weighted Graph?
A weighted graph is a graph with numbers assigned to its edges. These numbers are called weights or costs.

38) What is a Connected Graph?
A graph is said to be connected if for every pair of its vertices u and v there is a path from u to v.

39) What is the time complexity of Dijkstra's algorithm?
For adjacency matrix, It is O(IVI*IVI)
For adjacency list with edges stored as min heap, it is O(IEIlogIVI)

40) Differentiate b/w Traveling Salesman Problem(TSP) and Dijkstra's Algorithm.
In TSP, given n cities with known distances b/w each pair , find the shortest tour that passes through all the cities exactly once before returning to the starting city.
In Dijkstra's Algorithm, find the shortest path from source vertex to all other remaining vertices

41) Differentiate b/w Prim's Algorithm and Dijkstra's Algorithm?
Prim's algorithm is to find minimum cost spanning tree.
Dijkstra's algorithm is to find the shortest path from source vertex to all other remaining vertices.

42) What are the applications of Dijkstra's Algorithm?
It finds its application even in our day to day life while travelling. They are helpful in routing applications.

43) Who was the inventor of the Quicksort?
C.A.R.HOARE, a British Scientist.

44) Which design strategy does Quicksort uses?
Divide and Conquer

45) What is Divide and Conquer Technique?
(I) Divide the instance of a problem into two or more smaller instances
(II) Solve the smaller instances recursively
(III) Obtain solution to original instances by combining these solutions

46) What is the another name for Quicksort?
Partition Exchange Sort

47)Is Quicksort Stable as well as In place?
Not Stable but In place.

48) When do you say that a Quick sort having best case complexity?
When the pivot exactly divides the array into equal half

49) When do you say that Quick sort having worst case complexity?
When any one of the subarray is empty

50) How many more comparisons are needed for average case compared to best case?
38% more

51) when do you say that the pivot element is in its final position?
When all the elements towards the left of pivot are <= pivot and all the elements towards the right of pivot are >= pivot.

52) What technique does kruskal's algorithm follow.
Greedy technique

53) What is the time complexity of kruskal algorithm.
With an efficient sorting algorithm, the time efficiency of an kruskal's algorithm will be inO(|E|log|E|).

54) Who is the inventor of kruskal's algorithm.
Joseph Kruskal.

55) Which technique is used to solve BFS & DFS problems?
Decrease and Conquer

56) What is DFS traversal?
Consider an arbitrary vertex v and mark it as visited on each iteration, proceed to an unvisited vertex w adjacent to v. we can explore this new vertex depending upon its adjacent information. This process continues until dead end is encountered.(no adjacent unvisited vertex is available). At the dead end the algorithm backs up by one edge and continues the process of visiting unvisited vertices. This process continues until we reach the starting vertex.

57) What are the various applications of BFS & DFS tree traversal technique?
Application of BFS
Checking connectivity and checking acyclicity of a graph.
Check whether there is only one root in the BFS forest or not.
If there is only one root in the BFS forest, then it is connected graph otherwise disconnected graph.
For checking cycle presence, we can take advantage of the graph's representation in the form of a BFS forest, if the latter vertex does not have cross edge, then the graph is acyclic.
Application of DFS
To check whether given graph is connected or not.
To check whether the graph is cyclic or not.
To find the spanning tree.
Topological sorting.

58) Which data structures are used in BFS & DFS tree traversal technique?
We use Stack data structure to traverse the graph in DFS traversal. We use queue data structure to traverse the graph in BFS traversal.

59) What is Dynamic Programming?
It is a technique for solving problems with overlapping sub problems.

60) What does Dynamic Programming have in common with Divideand-Conquer?
Both solve the problems by dividing the problem into sub problems. Using the solutions of sub problems, the solutions for larger instance of the problem can be obtained.

61) What is a principle difference between the two techniques?
Only one instance of the sub problem is computed & stored. If the same instance of sub problem is encountered, the solution is retrieved from the table and never recomputed. Very precisely re - computations are not preformed.

63) What is a spanning tree ?
A spanning tree of a weighted connected graph is a connected acyclic(no cycles) subgraph (i.e. a tree)that contains all the vertices of a graph and number of edges is one less than number of vertices.

64) What is a minimum spanning tree ?
Minimum spanning tree of a connected graph is its spanning tree of smallestweight.

65) Prim's algorithm is based on which technique.
Greedy technique.

66) Name some of the application greedy technique
They are helpful in routing applications: In the case of network where large number of computers are connected, to pass the data from one node to another node we can find the shortest distance easily by this algorithm. Communication Networks: Suppose there are 5 different cities and we want to establish computer connectivity among them, then we take into the account of cost factor for communication links. Building a road network that joins n cities with minimum cost.

67) Does Prim's Algorithm always yield a minimum spanning tree ?
Yes

68) What is the purose of Floyd's algoreithm?
- To find the all pair shortest path.

69) What is the time efficiency of floyd's algorithm?
- It is same as warshall's algorithm that is θ(n3).

70) What is shortest path?
- It is the path which is having shortest length among all possible paths.

71) warshall's algorithm is optimization problem or non-optimization problem ?
Non-optimization problem

72) For what purpose we use Warshall's algorithm?
Warshall's algorithm for computing the transitive closure of a directed graph

73) Warshall's algorithm depends on which property?
Transitive property

74) what is the time complexity of Warshall's algorithm?
Time complexity of warshall's algorithm is θ(n3).

75) what is state space tree?
Constructing a tree of choices being made and processing is known as state-spacetree. Root represents an initial state before the search for solution begins.

76) what are promising and non-promising state-space-tree?
Node which may leads to solution is called promising. Node which doesn't leads to solution is called non-promising.

77) What are the requirements that are needed for performing Backtracking?
Complex set of constraints. They are:
i. Explicit constraints.
ii. Implicit constraints.

GREEDY METHOD

1. Explain the greedy method.

Greedy method is the most important design technique, which makes a choice that looks best atthat moment. A given 'n' inputs are required us to obtain a subset that satisfies some constraints that is the feasible solution. A greedy method suggests that one candevice an algorithm that works in stages considering one input at a time.

2. Define feasible and optimal solution.

Given n inputs and we are required to form a subset such that it satisfies some given constraints then such a subset is called feasible solution. A feasible solution either maximizes or minimizes the given objective function is called as optimal solution.

3. What are the constraints of knapsack problem?

To maximize ∑pixi

The constraint is : ∑wixi ≥ m and 0 ≤ xi ≤ 1 1≤ i ≤ nwhere m is the bag capacity, n is the number of objects and for each object i wi and pi are the weight and profit of object respectively.

4. Specify the algorithms used for constructing Minimum cost spanning tree.

a) Prim's Algorithm

b) Kruskal's Algorithm

5. State single source shortest path algorithm (Dijkstra's algorithm).

For a given vertex called the source in a weigted connected graph,find shotrtest paths to all its other vertices.Dijikstra's algorithm applies to graph with non-negative weights only.

6. State efficiency of prim's algorithm.

O(|v|2) (WEIGHT MATRIX AND PRIORITY QUEUE AS UNORDERED ARRAY)

O(|E| LOG|V|) (ADJACENCY LIST AND PRIORITY QUEUE AS MIN-HEAP)

7. State Kruskal Algorithm.

The algorithm looks at a MST for a weighted connected graph as an acyclic subgraph with |v|-1 edges for which the sum of edge weights is the smallest.

8. State efficiency of Dijkstra's algorithm.

O(|v|2) (WEIGHT MATRIX AND PRIORITY QUEUE AS UNORDERED ARRAY)

O(|E| LOG|V|) (ADJACENCY LIST AND PRIORITY QUEUE AS MIN-HEAP)

DYNAMIC PROGRAMMING

1. Define multistage graph

A multistage graph G =(V,E) is a directed graph in which the vertices are partitioned into K>=2 disjoint sets Vi,1<=i<=k.The multi stage graph problem is to find a minimum cost paths from s(source) to t(sink)
Two approach(forward and backward)

2. Define All pair shortest path problem

Given a weighted connected graph, all pair shortest path problem asks to find the lengths of shortest paths from each vertex to all other vertices.

3.Define floyd's algorithm

To find all pair shortest path

4. State the time efficiency of floyd's algorithm

O(n3)
It is cubic