**SOURCE CODE:**

/\*

Data Structure & Graph Algorithm using User Define Function.

\*/

package algo\_practice;

import java.util.\*;

public class Algo\_practice {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("-----------------------\n"

+ " Algorithm \n"

+ "-----------------------\n"

+ " 1. Data structure \n"

+ " 2. Graph algorithm");

System.out.println();

System.out.print("Enter your choice: ");

int choice = s.nextInt();

System.out.println();

if (choice == 1) {

System.out.println(" algorithm \n"

+ "---------------------\n"

+ "1. binary search\n"

+ "2. linear search\n"

+ "3. Bubble sort\n"

+ "4. Insertion sort\n"

+ "5. Selection sort\n"

+ "6. Quick sort\n"

+ "7. Merge sort\n"

+ "8. Heap sort\n"

+ "9. Radix sort\n"

+ "10. Bucket sort\n");

System.out.println("Please enter your choice: ");

int n1 = s.nextInt();

switch(n1){

case 1:

binarySearch bi = new binarySearch();

bi.searching();

break;

case 2:

linearSearch li = new linearSearch();

li.searching();

break;

case 3:

bubbleSort bu = new bubbleSort();

bu.sorting();

break;

case 4:

insertionSort ins = new insertionSort();

ins.sorting();

break;

case 5:

selectionSort si = new selectionSort();

si.sorting();

break;

case 6:

quickSort q = new quickSort();

q.sorting();

break;

case 7:

break;

case 8:

break;

case 9:

break;

case 10:

break;

default:

System.out.println("Please enter the correct number.");

break;

}

} else if (choice == 2) {

System.out.println(" Graph \n"

+ "---------------------\n"

+ "1. BFS\n"

+ "2. DFS\n");

System.out.println("Please enter your choice: ");

int n1 = s.nextInt();

switch(n1){

case 1:

bfs g = new bfs(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

System.out.println("Following is Breadth First Traversal " + "(starting from vertex 2)");

g.BFS(2);

break;

case 2:

dfs d = new dfs(4);

d.addEdge(0, 1);

d.addEdge(0, 2);

d.addEdge(1, 2);

d.addEdge(2, 3);

System.out.println("Following is Depth First Traversal");

d.DFS(2);

break;

case 3:

break;

case 4:

break;

default:

System.out.println("Please enter the correct number.");

break;

}

}

}

}

**BFS:**

package algo\_practice;

import java.util.\*;

public class bfs {

private int V;

private LinkedList<Integer> adj[];

// Create a graph

bfs(int v) {

V = v;

adj = new LinkedList[v];

for (int i = 0; i < v; ++i)

adj[i] = new LinkedList();

}

// Add edges to the graph

void addEdge(int v, int w) {

adj[v].add(w);

}

// BFS algorithm

void BFS(int s) {

boolean visited[] = new boolean[V];

LinkedList<Integer> queue = new LinkedList();

visited[s] = true;

queue.add(s);

while (queue.size() != 0) {

s = queue.poll();

System.out.print(s + " ");

Iterator<Integer> i = adj[s].listIterator();

while (i.hasNext()) {

int n = i.next();

if (!visited[n]) {

visited[n] = true;

queue.add(n);

}

}

}

}

public static void main(String args[]) {

bfs g = new bfs(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

System.out.println("Following is Breadth First Traversal " + "(starting from vertex 2)");

g.BFS(2);

}

}

**Binary\_Search:**

package algo\_practice;

import java.util.Scanner;

public class binarySearch {

public void searching() {

int c, first, last, middle, n, search, array[];

Scanner in = new Scanner(System.in);

System.out.println("Enter number of elements: ");

n = in.nextInt();

array = new int[n];

System.out.println("Enter " + n + " integers");

for (c = 0; c < n; c++)

array[c] = in.nextInt();

System.out.println("Enter value to find");

search = in.nextInt();

first = 0;

last = n - 1;

middle = (first + last)/2;

while( first <= last )

{

if ( array[middle] < search )

first = middle + 1;

else if ( array[middle] == search )

{

System.out.println(search + " found at location " + (middle + 1) + ".");

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

System.out.println(search + " isn't present in the list.\n");

}

}

**DFS:**

package algo\_practice;

import java.util.\*;

class dfs {

private LinkedList<Integer> adjLists[];

private boolean visited[];

// Graph creation

dfs(int vertices) {

adjLists = new LinkedList[vertices];

visited = new boolean[vertices];

for (int i = 0; i < vertices; i++)

adjLists[i] = new LinkedList<Integer>();

}

// Add edges

void addEdge(int src, int dest) {

adjLists[src].add(dest);

}

// DFS algorithm

void DFS(int vertex) {

visited[vertex] = true;

System.out.print(vertex + " ");

Iterator<Integer> ite = adjLists[vertex].listIterator();

while (ite.hasNext()) {

int adj = ite.next();

if (!visited[adj])

DFS(adj);

}

}

public static void main(String args[]) {

dfs d = new dfs(4);

d.addEdge(0, 1);

d.addEdge(0, 2);

d.addEdge(1, 2);

d.addEdge(2, 3);

System.out.println("Following is Depth First Traversal");

d.DFS(2);

}

}

**Insertion\_sort:**

package algo\_practice;

import java.util.Scanner;

public class insertionSort{

public void sorting(){

int size, i, j, temp;

int arr[] = new int[50];

Scanner scan = new Scanner(System.in);

System.out.println("Enter number of elements: ");

size = scan.nextInt();

System.out.print("Enter " + size + " integers: ");

for(i=0; i<size; i++)

{

arr[i] = scan.nextInt();

}

for(i=1; i<size; i++)

{

temp = arr[i];

j = i - 1;

while((temp < arr[j]) && (j >= 0))

{

arr[j+1] = arr[j];

j = j - 1;

}

arr[j+1] = temp;

}

System.out.print("Sorted list of numbers: ");

for(i=0; i<size; i++)

{

System.out.print(arr[i] + " ");

}

}

}

**Quick\_sort:**

package algo\_practice;

import java.util.Scanner;

public class quickSort

{

public static int sorting(int a[],int l,int h)

{

int i=l+1 ,j=h,c=l,temp;

for(; i<=j ;)

{

while(i<=h && a[i]<a[c] )

i++;

while(a[j]>a[c] && j>l)

j--;

if(i<j)

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

else

break;

}

temp=a[c];

a[c]=a[j];

a[j]=temp;

return j;

}

public static void Sort(int a[],int l,int h)

{

if(l<h)

{

int m=partition(a,l,h);

Sort(a,l,m-1);

Sort(a,m+1,h);

}

}

public static void printarray(int a[])

{

for(int i=0; i < a.length; i++)

{

System.out.print(a[i]+" ");

}

}

public static void main(String[] args)

{

int n, res,i;

Scanner s = new Scanner(System.in);

System.out.print("Enter number of elements in the array:");

n = s.nextInt();

int a[] = new int[n];

System.out.println("Enter "+n+" elements ");

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

{

a[i] = s.nextInt();

}

System.out.println( "elements in array ");

printarray(a);

Sort(a,0,n-1);

System.out.println( "\nelements after sorting");

printarray(a);

}

}

**Linear\_Search:**

package algo\_practice;

import java.util.Scanner;

public class linearSearch {

public void searching() {

int i, len, key, array[];

Scanner input = new Scanner(System.in);

System.out.println("Enter number of elements: ");

len = input.nextInt();

array = new int[len];

System.out.println("Enter " + len + " elements");

for (i = 0; i < len; i++)

{

array[i] = input.nextInt();

}

System.out.println("Enter the search key value:");

key = input.nextInt();

for (i = 0; i < len; i++)

{

if (array[i]== key)

{

System.out.println(key+" is present at location "+(i+1));

break;

}

}

if (i == len)

System.out.println(key + " doesn't exist in array.");

}

}

**Scanner:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package algo\_practice;

/\*\*

\*

\* @author Rahat Kabir Dhrubo

\*/

public class scanner {

}

**Selection\_Sort:**

package algo\_practice;

import java.util.Scanner;

public class selectionSort

{

public void sorting()

{

int size, i, j, temp, small, index = 0, count=0;

int arr[] = new int[50];

Scanner scan = new Scanner(System.in);

System.out.print("Enter number of elements: ");

size = scan.nextInt();

System.out.print("Enter " + size + " integers: ");

for(i=0; i<size; i++)

{

arr[i] = scan.nextInt();

}

for(i=0; i<(size-1); i++)

{

small = arr[i];

for(j=(i+1); j<size; j++)

{

if(small>arr[j])

{

small = arr[j];

count++;

index = j;

}

}

if(count!=0)

{

temp = arr[i];

arr[i] = small;

arr[index] = temp;

}

count=0;

}

System.out.print("Now the Array after Sorting is :\n");

for(i=0; i<size; i++)

{

System.out.print(arr[i]+ " ");

}

}

}