**PRACTICAL NO: 8**

**AIM:** Write a program to implement greedy set cover algorithm to solve set covering problem.

PROGRAM:

package practicalno8;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.LinkedHashSet;

import java.util.List;

import java.util.Scanner;

import java.util.Set;

public class Practicalno8

{

interface Filter<T>

{

boolean matches(T t);

}

private static <T> Set<T> shortestCombination(Filter<Set<T>> filter,List<T> listOfSets)

{

final int size = listOfSets.size();

if (size > 20)

throw new IllegalArgumentException("Too many combinations");

int combinations = 1 << size;

List<Set<T>> possibleSolutions = new ArrayList<Set<T>>();

for (int l = 0; l < combinations; l++)

{

Set<T> combination = new LinkedHashSet<T>();

for (int j = 0; j < size; j++)

{

if (((l >> j) & 1) != 0)

combination.add(listOfSets.get(j));

}

possibleSolutions.add(combination);

}

Collections.sort(possibleSolutions, new Comparator<Set<T>>()

{

public int compare(Set<T> o1, Set<T> o2)

{

return o1.size() - o2.size();

}

}

);

for (Set<T> possibleSolution : possibleSolutions)

{

if (filter.matches(possibleSolution))

return possibleSolution;

}

return null;

}

public static void main(String[] args){

System.out.println("Enter 2D array size : ");

Scanner sc=new Scanner(System.in);

System.out.println("Enter rows size of array : ");

int rows=sc.nextInt();

System.out.println("Enter columns size of array : ");

int columns=sc.nextInt();

System.out.println("Enter array elements : ");

Integer arrayOfSets[][]=new Integer[rows][columns];

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

{

for(int j=0; j<columns;j++)

{

arrayOfSets[i][j]=sc.nextInt();

}

}

Integer[] solution = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

List<Set<Integer>> listOfSets = new ArrayList<Set<Integer>>();

for (Integer[] array : arrayOfSets)listOfSets.add(new LinkedHashSet<Integer>(Arrays.asList(array)));

final Set<Integer> solutionSet = new LinkedHashSet<Integer>(Arrays.asList(solution));

Filter<Set<Set<Integer>>> filter = new Filter<Set<Set<Integer>>>()

{

public boolean matches(Set<Set<Integer>> integers)

{

Set<Integer> union = new LinkedHashSet<Integer>();

for (Set<Integer> ints : integers)

union.addAll(ints);

return union.equals(solutionSet);

}

};

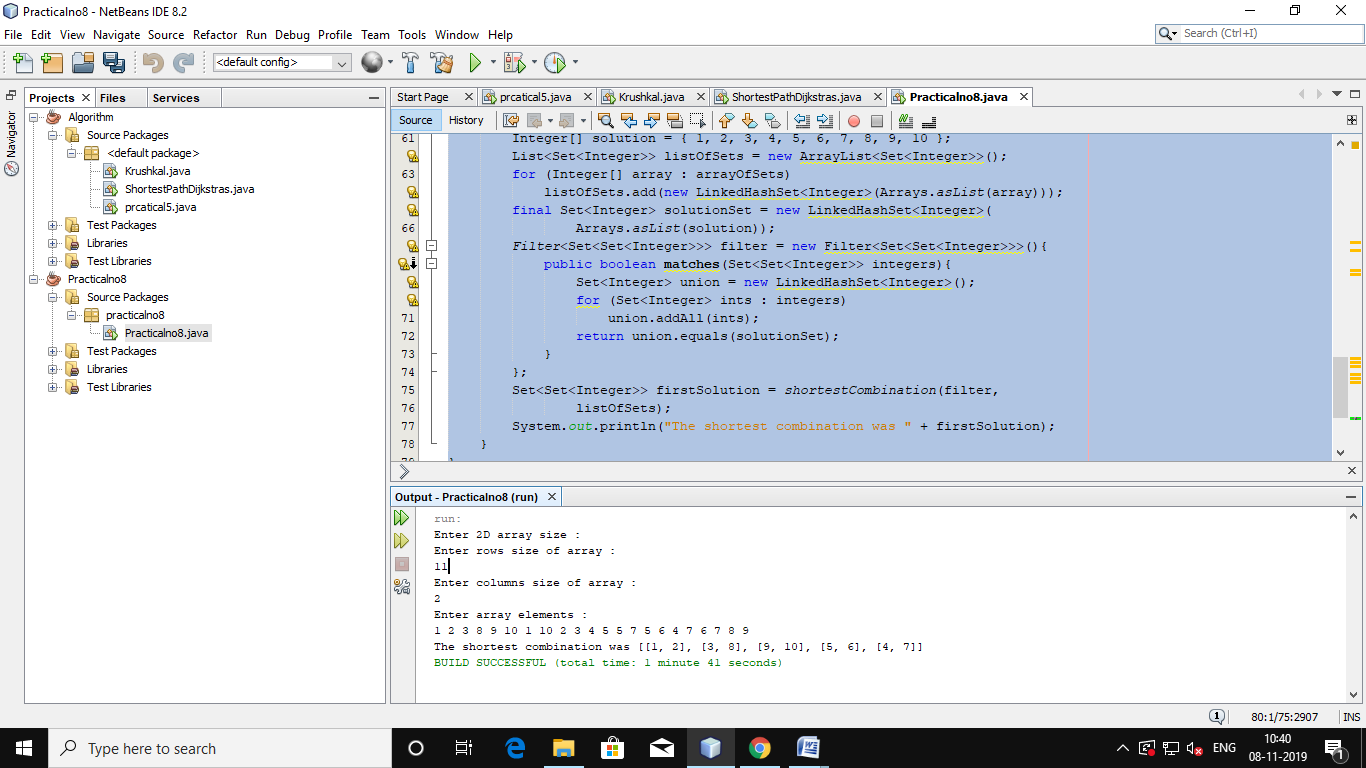
Set<Set<Integer>> firstSolution = shortestCombination(filter,listOfSets);

System.out.println("The shortest combination was " + firstSolution);

}

}

**OUTPUT:**

****