Name Of the Student : Sujit Sachin Shaha

Class / Division : TY – D

PRN No. : 122B1B255

Roll No. : 122B1B255

**Assignment No. 06**

**Problem Statement** : Let there be N students and N clubs. Any student can be assigned to any club, incurring some cost that may vary depending on the student club assignment. It is required to allocate all clubs by assigning exactly one student to each club and exactly one club to each agent in such a way that the total cost of the assignment is minimized. Implement club assignment problem using Branch and bound.

**Code**

package Lab\_20Oct;

import java.util.\*;

public class Assign\_Club {

static final int *INF* = Integer.*MAX\_VALUE*;

static class Node {

int[] assigned;

boolean[] available;

int cost;

int level;

Node(int[] assigned, boolean[] available, int cost, int level) {

this.assigned = assigned.clone();

this.available = available.clone();

this.cost = cost;

this.level = level;

}

}

static int findMinCost(int[][] costMatrix, int N) {

PriorityQueue<Node> pq = new PriorityQueue<>(Comparator.*comparingInt*(a -> a.cost));

boolean[] available = new boolean[N];

Arrays.*fill*(available, true);

int[] assigned = new int[N];

Node root = new Node(assigned, available, 0, 0);

pq.add(root);

while (!pq.isEmpty()) {

Node node = pq.poll();

int level = node.level;

if (level == N) {

return node.cost;

}

for (int i = 0; i < N; i++) {

if (node.available[i]) {

node.assigned[level] = i;

node.available[i] = false;

int newCost = node.cost + costMatrix[level][i];

pq.add(new Node(node.assigned, node.available, newCost, level + 1));

node.available[i] = true;

}

}

}

return *INF*;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc = new Scanner(System.*in*);

int n;

System.*out*.println("Enter no. of clubs: ");

n = sc.nextInt();

int ip;

int[][] costMatrix = new int[n][n];

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

System.*out*.println("Enter Cost Matrix line by line ( end with -1): ");

ip = sc.nextInt();

int j=0;

while(ip != -1) {

costMatrix[i][j] = ip;

j++;

ip = sc.nextInt();

}

}

int N = costMatrix.length;

int minCost = *findMinCost*(costMatrix, N);

System.*out*.println("Minimum cost: " + minCost);

}

}

**Output**

Enter no. of clubs:

4

Enter Cost Matrix line by line ( end with -1):

6 2 8 7 -1

Enter Cost Matrix line by line ( end with -1):

3 4 6 1 -1

Enter Cost Matrix line by line ( end with -1):

5 6 7 2 -1

Enter Cost Matrix line by line ( end with -1):

3 6 4 5 -1

Minimum cost: 11