1.A Towers of Hanoi

class TowerOfHanoi {

static void towerOfHanoi(int n, char fromRod, char toRod, char auxRod) {

if (n == 1) {

System.out.println("Move disk 1 from rod " + fromRod + " to rod " + toRod);

return;

}

towerOfHanoi(n - 1, fromRod, auxRod, toRod);

System.out.println("Move disk " + n + " from rod " + fromRod + " to rod " + toRod);

towerOfHanoi(n - 1, auxRod, toRod, fromRod);

}

public static void main(String[] args) {

int n = 4;

towerOfHanoi(n, 'A', 'B', 'C');

}

}

2.A Travelling Sales Problem

public static int findMinCost(int[][] graph) {

int n = graph.length; // Number of cities

int allCities = (1 << n) - 1; // Binary representation of all cities visited

int[][] dp = new int[n][1 << n];

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

Arrays.fill(dp[i], Integer.MAX\_VALUE);

dp[i][1 << i] = graph[i][0]; // Initialize base case: starting city to itself

}

for (int mask = 1; mask <= allCities; mask++) {

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

if ((mask & (1 << curCity)) != 0) { // If curCity is in the mask

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

if (prevCity != curCity && (mask & (1 << prevCity)) != 0) {

dp[curCity][mask] = Math.min(dp[curCity][mask],

dp[prevCity][mask ^ (1 << curCity)] + graph[curCity][prevCity]);

}

}

}

}

}

int minCost = Integer.MAX\_VALUE;

for (int i = 1; i < n; i++) {

minCost = Math.min(minCost, dp[i][allCities] + graph[i][0]);

}

return minCost;

}

public static void main(String[] args) {

int[][] graph = {

{0, 10, 15, 20},

{10, 0, 35, 25},

{15, 35, 0, 30},

{20, 25, 30, 0}

};

System.out.println("Minimum cost to visit all cities and return: " + findMinCost(graph));

}

}

3.A Job Sequence Problem

import java.util.\*;

class Job {

int id;

int deadline;

int profit;

public Job(int id, int deadline, int profit) {

this.id = id;

this.deadline = deadline;

this.profit = profit;

}

}

public class JobSequencing {

public static List<Job> jobSequencing(List<Job> jobs) {

Collections.sort(jobs, (a, b) -> b.profit - a.profit);

int maxDeadline = 0;

for (Job job : jobs) {

maxDeadline = Math.max(maxDeadline, job.deadline);

}

List<Job> result = new ArrayList<>();

boolean[] slotFilled = new boolean[maxDeadline];

for (Job job : jobs) {

for (int i = job.deadline - 1; i >= 0; i--) {

if (!slotFilled[i]) {

result.add(job);

slotFilled[i] = true;

break;

}

}

}

return result;

}

public static void main(String[] args) {

List<Job> jobs = new ArrayList<>();

jobs.add(new Job(1, 4, 20));

jobs.add(new Job(2, 1, 10));

jobs.add(new Job(3, 1, 40));

jobs.add(new Job(4, 1, 30));

List<Job> result = jobSequencing(jobs);

System.out.println("Maximum profit sequence of jobs:");

for (Job job : result) {

System.out.print(job.id + " ");

}

}

}

**Output:** For the given example, the output will be: 3 1