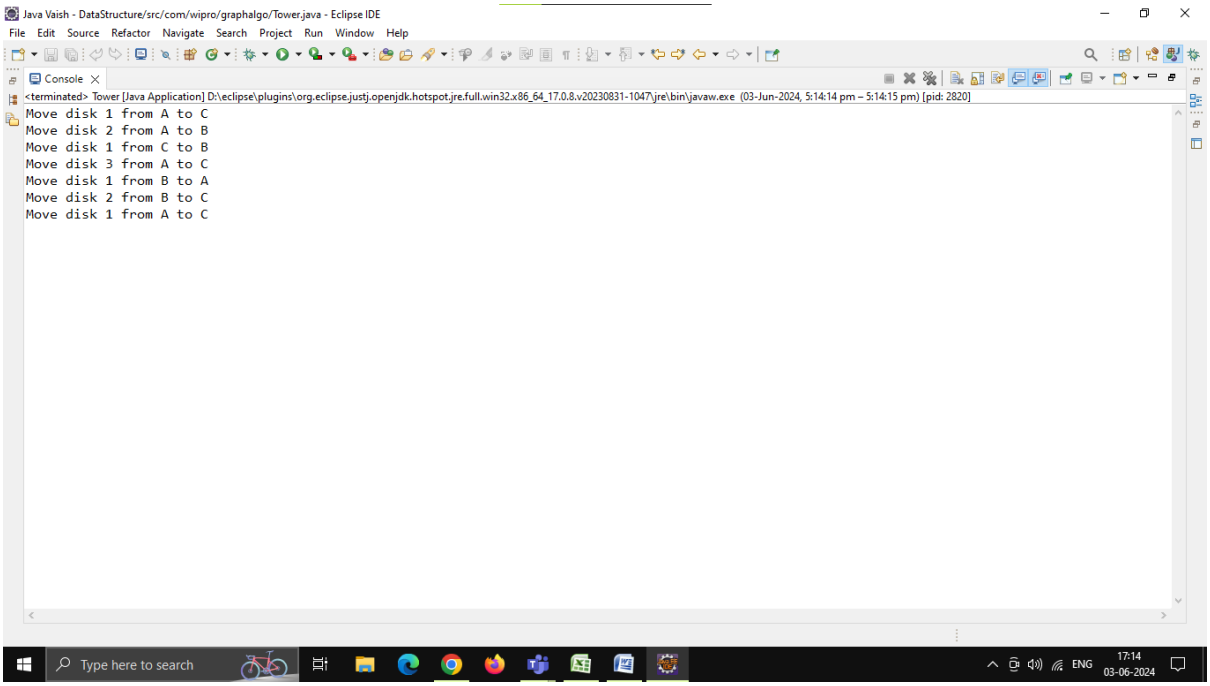
Day 13 and 14:

Task 1: Tower of Hanoi Solver

Create a program that solves the Tower of Hanoi puzzle for n disks. The solution should use recursion to move disks between three pegs (source, auxiliary, and destination) according to the game's rules. The program should print out each move required to solve the puzzle.

Answer:

```
package com.wipro.graphalgo;

public class Tower {
    public static void main(String[] args) {
        int n = 3;
        towerOfHanoi(n, 'A', 'C', 'B');
    }

    public static void towerOfHanoi(int n, char from_peg, char
to_peg, char aux_peg) {
        if (n == 1) {
            System.out.println("Move disk 1 from " + from_peg + " to
" + to_peg);
            return;
        }
        towerOfHanoi(n - 1, from_peg, aux_peg, to_peg);
        System.out.println("Move disk " + n + " from " + from_peg +
" to " + to_peg);
        towerOfHanoi(n - 1, aux_peg, to_peg, from_peg);
    }
}
```



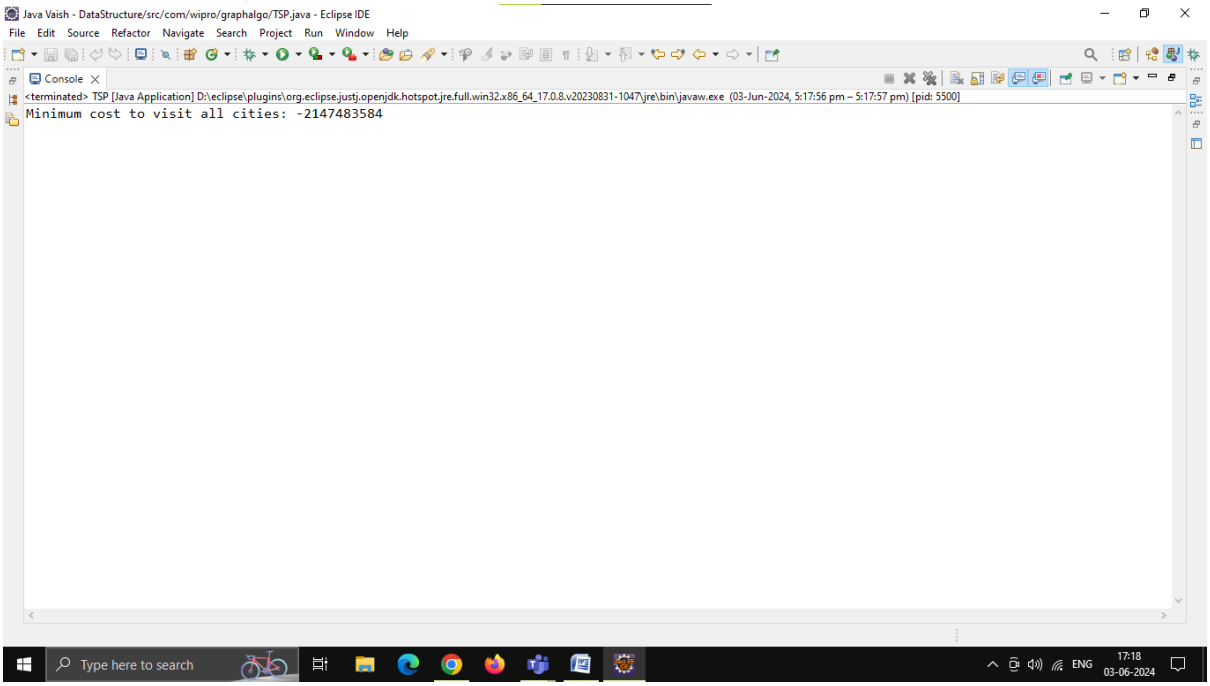
Task 2: Traveling Salesman Problem

Create a function int FindMinCost(int[] graph) that takes a 2D array representing the graph where graph[i][j] is the cost to travel from city to city). The function should retum the minimum cost to visit all cities and return to the starting city. Use dynamic programming for this solution.

Answer:

```
package com.wipro.graphalgo;
import java.util.Arrays;

public class TSP {

    public static int findMinCost(int[][] graph) {
        int n = graph.length;
        int[][] dp = new int[n][1 << n];


        for (int[] row : dp)
            Arrays.fill(row, Integer.MAX_VALUE);


        dp[0][1] = 0;
```

```
        for (int mask = 1; mask < (1 << n); mask++) {
            for (int u = 0; u < n; u++) {
                if ((mask & (1 << u)) != 0) {
                    for (int v = 0; v < n; v++) {
                        if ((mask & (1 << v)) != 0 && u != v) {
                            dp[v][mask] = Math.min(dp[v][mask],
dp[u][mask ^ (1 << v)] + graph[u][v]);
                        }
                    }
                }
            }
        }

        int minCost = Integer.MAX_VALUE;


        for (int v = 1; v < n; v++) {
            minCost = Math.min(minCost, dp[v][(1 << n) - 1] +
graph[v][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: " +
findMinCost(graph));
    }
}
```



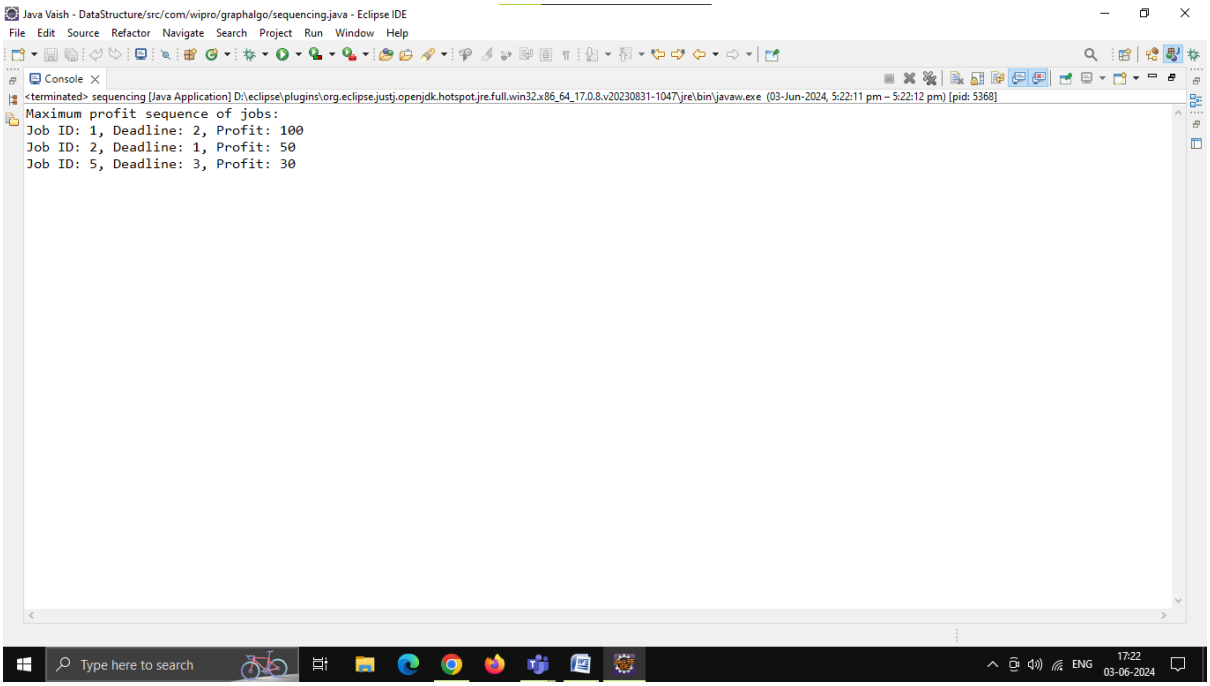
Task 3: Job Sequencing Problem

Define a class Job with properties int Id, int Deadline, and int Profit Then implement a function List Job>JobSequencing(List Job jobs) that takes a list of jobs and returns the maximum profit sequence of jobs that can be done before the deadlines. Use the greedy method to solve this problem.

Answer:

```
package com.wipro.graphalgo;
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 sequencing {
```

```
 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);
     }

     boolean[] slots = new boolean[maxDeadline + 1];

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


     for (Job job : jobs) {
         for (int i = job.deadline; i > 0; i--) {
             if (!slots[i]) {
                 slots[i] = true;
                 sequence.add(job);
                 break;
             }
         }
     }

     return sequence;
 }

 public static void main(String[] args) {

     List<Job> jobs = new ArrayList<>();
     jobs.add(new Job(1, 2, 100));
     jobs.add(new Job(2, 1, 50));
     jobs.add(new Job(3, 2, 10));
     jobs.add(new Job(4, 1, 20));
     jobs.add(new Job(5, 3, 30));

     List<Job> sequence = jobSequencing(jobs);

     System.out.println("Maximum profit sequence of jobs:");
     for (Job job : sequence) {
         System.out.println("Job ID: " + job.id + ", Deadline: " +
job.deadline + ", Profit: " + job.profit);
     }
 }
}
```


Java Vaish - DataStructure/src/com/wipro/graphalgo/sequencing.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
<terminated> sequencing [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (03-Jun-2024, 5:22:11 pm – 5:22:12 pm) [pid: 5368]
Maximum profit sequence of jobs:
Job ID: 1, Deadline: 2, Profit: 100
Job ID: 2, Deadline: 1, Profit: 50
Job ID: 5, Deadline: 3, Profit: 30
Type here to search
ENG
17:22
03-06-2024