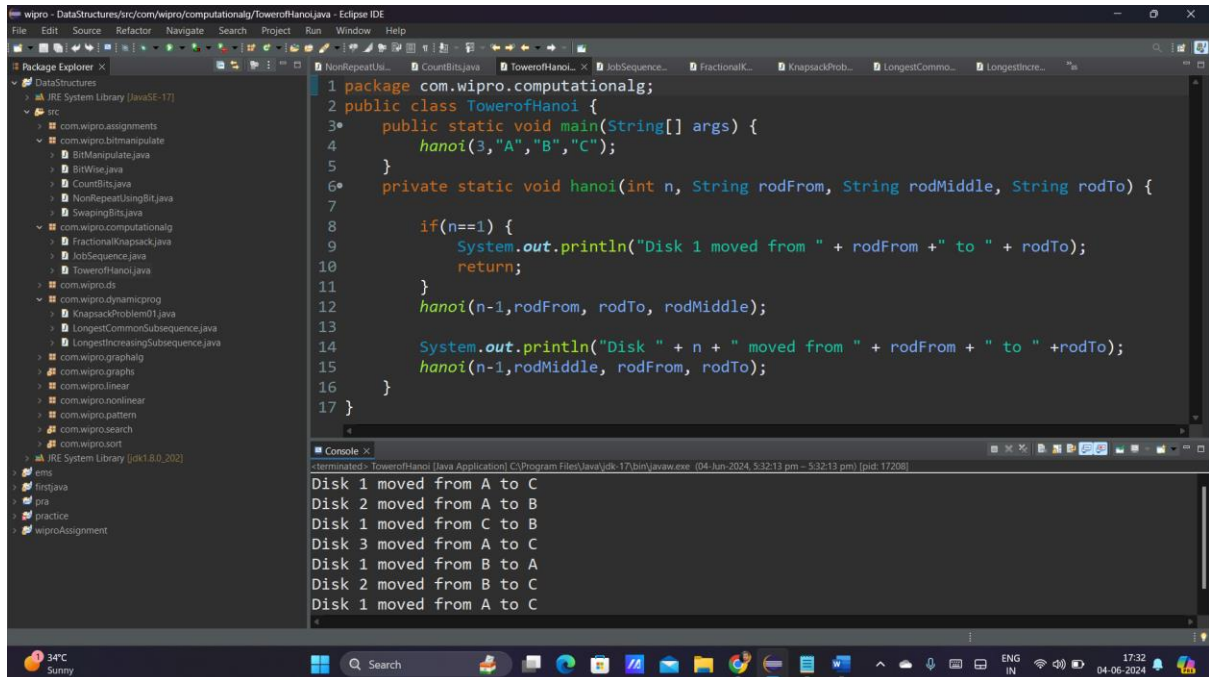


Assignment-Day13,14

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.



```
1 package com.wipro.computationalg;
2 public class TowerofHanoi {
3     public static void main(String[] args) {
4         hanoi(3, "A", "B", "C");
5     }
6     private static void hanoi(int n, String rodFrom, String rodMiddle, String rodTo) {
7
8         if(n==1) {
9             System.out.println("Disk 1 moved from " + rodFrom + " to " + rodTo);
10            return;
11        }
12        hanoi(n-1, rodFrom, rodTo, rodMiddle);
13
14        System.out.println("Disk " + n + " moved from " + rodFrom + " to " + rodTo);
15        hanoi(n-1, rodMiddle, rodFrom, rodTo);
16    }
17 }
```

Console Output:

```
<terminated> TowerofHanoi (Java Application) C:\Program Files\Java\jdk-17\bin\javaw.exe (04-Jun-2024, 5:32:13 pm - 5:32:13 pm) [pid: 17208]
Disk 1 moved from A to C
Disk 2 moved from A to B
Disk 1 moved from C to B
Disk 3 moved from A to C
Disk 1 moved from B to A
Disk 2 moved from B to C
Disk 1 moved from A to C
```

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 `i` to city `j`. The function should return the minimum cost to visit all cities and return to the starting city. Use dynamic programming for this solution.

```
wipro - wiproAssignment/src/day13/TravelingSalesman.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
Data Structures
  JRE System Library [JavaSE-17]
  src
    com.wipro.assignments
    com.wipro.backtracking
    com.wipro.bitmanipulate
      BitManipulate.java
      BitWise.java
      CountBits.java
      NonRepeatUsingBit.java
      SwappingBits.java
    com.wipro.computationalg
      FractionalKnapsack.java
      JobSequence.java
      TowerOfHanoi.java
    com.wipro.ds
    com.wipro.dynamicprog
      KnapsackProblem01.java
      LongestCommonSubsequence.java
      LongestIncreasingSubsequence.java
    com.wipro.graphalg
    com.wipro.graphs
    com.wipro.linear
    com.wipro.nonlinear
    com.wipro.pattern
    com.wipro.search
    com.wipro.sort

JRE System Library [jdk1.8.0_202]
ems
firstjava
pra
practice
wiproAssignment
  JRE System Library [JavaSE-17]
  src
    assignments.day20
    day13

1 package day13;
2 public class TravelingSalesman {
3     public static void main(String[] args) {
4         int[][] graph = {
5             {0, 10, 15, 20},
6             {10, 0, 35, 25},
7             {15, 35, 0, 30},
8             {20, 25, 30, 0}
9         };
10        System.out.println("The minimum cost to visit all cities and return to the starting point is:");
11    }
12    public static int findMinCost(int[][] graph) {
13        int n = graph.length;
14        int VISITED_ALL = (1 << n) - 1;
15        int[][] dp = new int[n][1 << n];
16        for (int i = 0; i < n; i++) {
17            for (int j = 0; j < (1 << n); j++) {
18                dp[i][j] = -1;
19            }
20        }
21        return tsp(0, 1, dp, graph, VISITED_ALL);
22    }
23    private static int tsp(int pos, int mask, int[][] dp, int[][] graph, int VISITED_ALL) {
24        if (mask == VISITED_ALL) {
25            return graph[pos][0];
26        }
27    }
```

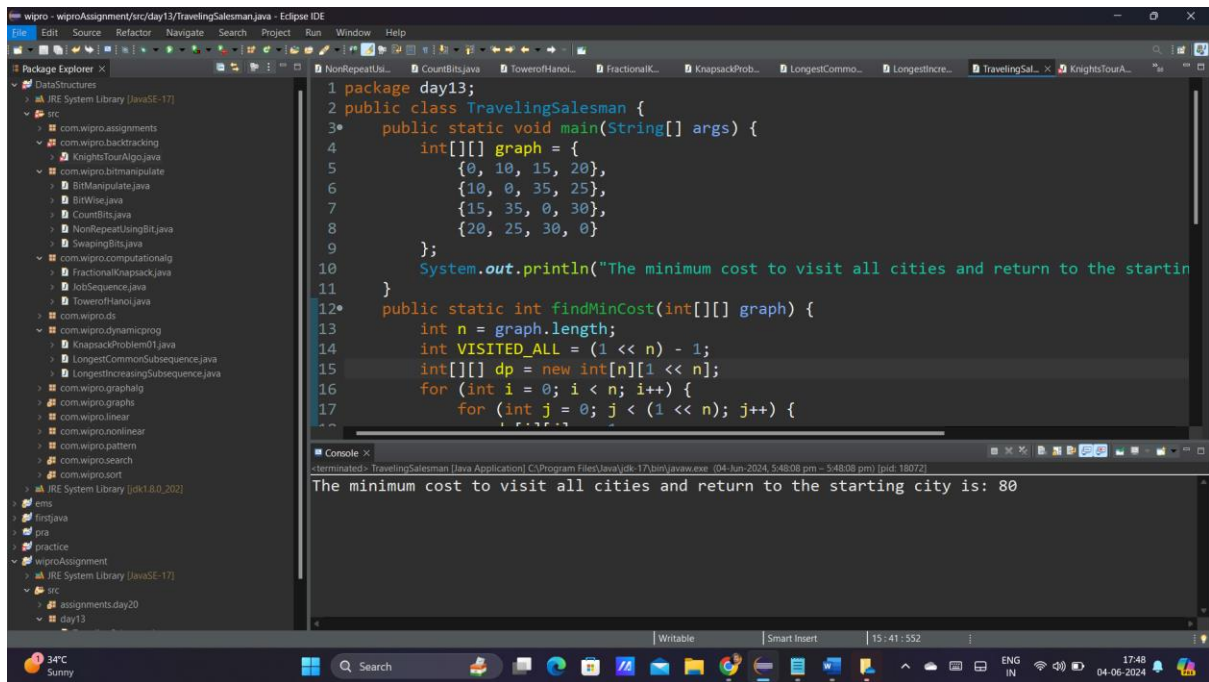
```
34°C Sunny
Search
ENG IN 17:46 04-06-2024

wipro - wiproAssignment/src/day13/TravelingSalesman.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer
Data Structures
  JRE System Library [JavaSE-17]
  src
    com.wipro.assignments
    com.wipro.backtracking
    com.wipro.bitmanipulate
      BitManipulate.java
      BitWise.java
      CountBits.java
      NonRepeatUsingBit.java
      SwappingBits.java
    com.wipro.computationalg
      FractionalKnapsack.java
      JobSequence.java
      TowerOfHanoi.java
    com.wipro.ds
    com.wipro.dynamicprog
      KnapsackProblem01.java
      LongestCommonSubsequence.java
      LongestIncreasingSubsequence.java
    com.wipro.graphalg
    com.wipro.graphs
    com.wipro.linear
    com.wipro.nonlinear
    com.wipro.pattern
    com.wipro.search
    com.wipro.sort

JRE System Library [jdk1.8.0_202]
ems
firstjava
pra
practice
wiproAssignment
  JRE System Library [JavaSE-17]
  src
    assignments.day20
    day13

16        for (int i = 0; i < n; i++) {
17            for (int j = 0; j < (1 << n); j++) {
18                dp[i][j] = -1;
19            }
20        }
21        return tsp(0, 1, dp, graph, VISITED_ALL);
22    }
23    private static int tsp(int pos, int mask, int[][] dp, int[][] graph, int VISITED_ALL) {
24        if (mask == VISITED_ALL) {
25            return graph[pos][0];
26        }
27        if (dp[pos][mask] != -1) {
28            return dp[pos][mask];
29        }
30        int minCost = Integer.MAX_VALUE;
31        for (int city = 0; city < graph.length; city++) {
32            if ((mask & (1 << city)) == 0) {
33                int newCost = graph[pos][city] + tsp(city, mask | (1 << city), dp, graph, VISITED_ALL);
34                minCost = Math.min(minCost, newCost);
35            }
36        }
37        dp[pos][mask] = minCost;
38        return minCost;
39    }
40    }
41    }
42 }
```

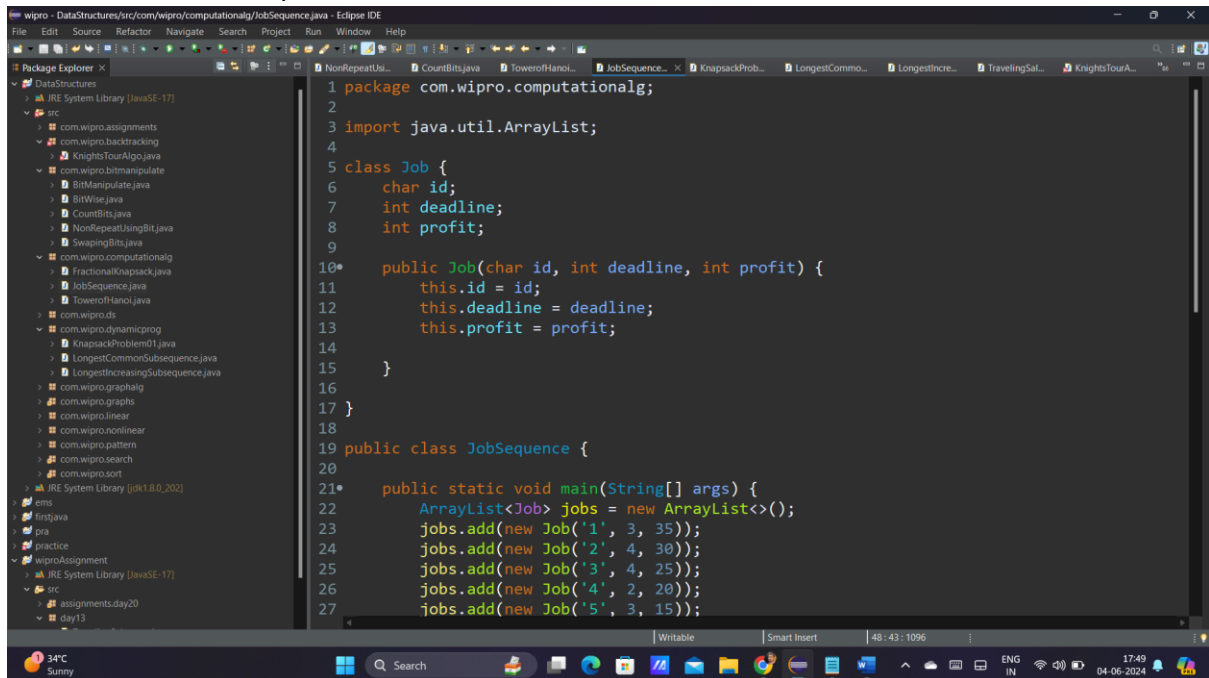


```
1 package day13;
2 public class TravelingSalesman {
3     public static void main(String[] args) {
4         int[][] graph = {
5             {0, 10, 15, 20},
6             {10, 0, 35, 25},
7             {15, 35, 0, 30},
8             {20, 25, 30, 0}
9         };
10        System.out.println("The minimum cost to visit all cities and return to the starting city is: 80");
11    }
12    public static int findMinCost(int[][] graph) {
13        int n = graph.length;
14        int VISITED_ALL = (1 << n) - 1;
15        int[][] dp = new int[n][1 << n];
16        for (int i = 0; i < n; i++) {
17            for (int j = 0; j < (1 << n); j++) {
```

The minimum cost to visit all cities and return to the starting city is: 80

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.



```
1 package com.wipro.computationalg;
2
3 import java.util.ArrayList;
4
5 class Job {
6     char id;
7     int deadline;
8     int profit;
9 }
10 public Job(char id, int deadline, int profit) {
11     this.id = id;
12     this.deadline = deadline;
13     this.profit = profit;
14 }
15 }
16
17 public class JobSequence {
18
19     public static void main(String[] args) {
20         ArrayList<Job> jobs = new ArrayList<>();
21         jobs.add(new Job('1', 3, 35));
22         jobs.add(new Job('2', 4, 30));
23         jobs.add(new Job('3', 4, 25));
24         jobs.add(new Job('4', 2, 20));
25         jobs.add(new Job('5', 3, 15));
```

The screenshot displays the Eclipse IDE interface with the `JobSequence.java` file open. The Package Explorer on the left shows the project structure, including the `src` folder and various sub-packages like `com.wipro.assignments`, `com.wipro.backtracking`, and `com.wipro.dynamicprog`.

The main editor shows the following Java code:

```
46 jobs.add(new Job(4, 2, 20));
47 jobs.add(new Job('5', 3, 15));
48 jobs.add(new Job('6', 1, 12));
49
50 doJobSequence(jobs);
51 }
52
53 private static void doJobSequence(ArrayList<Job> jobs) {
54     jobs.sort((a, b) -> b.profit - a.profit);
55
56     int maxDeadline = Integer.MIN_VALUE;
57     for (Job job : jobs) {
58         maxDeadline = Math.max(maxDeadline, job.deadline);
59     }
60
61     boolean[] filledSlots = new boolean[maxDeadline];
62
63     char[] results = new char[maxDeadline];
64     int totalProfit=0;
65     for (Job job : jobs) {
66         for (int i = job.deadline - 1; i >= 0; i--) {
67             if (!filledSlots[i]) {
68                 filledSlots[i] = true;
69                 results[i] = job.id;
70                 totalProfit += job.profit;
71                 break;
72             }
73         }
74     }
75
76     System.out.println("Total profit after sequencing:"+totalProfit);
77     for(char id: results) {
78         System.out.println(id + " ");
79     }
80 }
```

The console output at the bottom shows the execution results:

```
terminated- JobSequence [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (04-Jun-2024, 5:50:21 pm - 5:50:24 pm) [pid: 13780]
Total profit after sequencing:110
4
3
1
2
```