GEBZE TECHNICAL UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

2023-2024 FALL CSE341 PROGRAMMING LANGUAGES HOMEWORK-4 REPORT

AHMET YIGIT

200104004066

- ***Part-1***

    - Question Explanation
        - In this part of the homework, you are asked to write a simple expert system in Prolog for scheduling pickup and deliveries in a small college campus

    - Solving Approach
        - First of all, system control is object status on deliver or not

            - If object status on deliver
                - Return Person Id and Total Time to deliver

            - If object status not on deliver
                - Gets personals information's:
                    - Weight Capacity
                    - Load Carrying Hours
                    - Is On Job
                    - Location

                - Checking personals working on the current hour
                    - If True control of User is on Job
                        - If Not in Job, check personal can handle heavy.
                        - If Personal can print PersonX: TotalTime
                    - Otherwise print PersonX is not available or PersonX Cannot Handle It. It Is Too Heavy.

            - After evaluating all persons system says OBJECT TRANSFER by and gives his/her information. This person has already defined in the beginning of code.

            - For finding best path system uses dijikstra algorithm.

- Code Explanation
    - First 55 lines are used for definitions which is defined in homework pdf.

- `available_person_for_object(ObjectId, PersonId, TotalTime)`
    - This function declared twice.
        - First Declaration
            - Tries to find object is on transit.
            - Return Person Id and Total Time.

        - Second Declaration
            - Tries to find best path for each personal.
            - It uses dijikstra algorithm.
            - Control of :
                - Weight Capacity
                - Working Hours (in list each number is a load carrying times)
                - Is working
                - Location
            - Print all person time or excuses.
            - Returns Person Id and Total Time for object defined personal.

- `split_location_distance`
    - It gets, Location and returns (update) distance.
    - First declaration is base case.
    - Second declaration is recursive function.

- `find_distance`
    - This function helps to find the distance between two places it uses dijkstra algorithm.

- `neighbourhood`
    - Dijkstra algo Computes the shortest path from the Start node to the End node.

- `min_dist`
    - Finds the minimum distance between two nodes.

- `dijkstra`
    - First declaration is base case.
    - Second declaration is regular dijikstra algorithm.

- `choose_v`
    - choice of next vertex to expand.

```
diff
```

  - Removes vertices already in Closed from NB.

```
merge
```

  - First declaration is a base case.
  - Overall, this merge/4 function iterates through the first list of vertices and distances, updating the open list (NewOpen) based on certain conditions for each vertex-distance pair encountered in the list.

```
remove
```

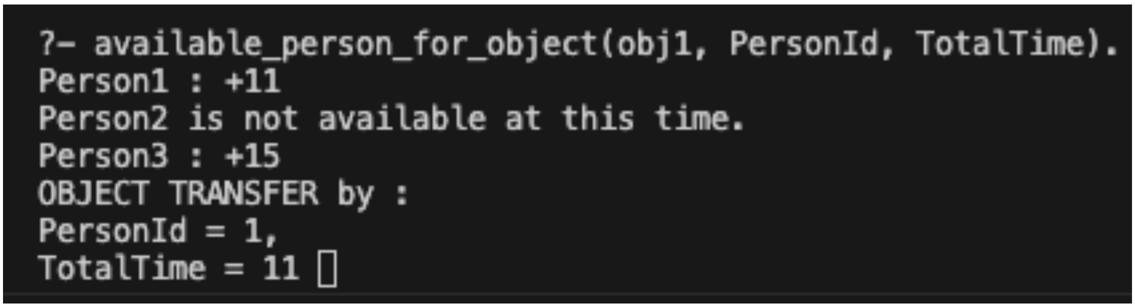
  - First declaration is a base case.
  - Removes X element to NT.

- Output
  - Test Personel:

```
delivery_personnel(1, 10, [4, 8, 12, 16, 20], none, adminOffice ).
% id, capacity, working hours, currentDeliveryJob, location
delivery_personnel(2, 10, [5, 9, 13, 17, 21], none, cafeteria ).
delivery_personnel(3, 10, [4, 8, 12, 16, 20], none, instituteY ).
```

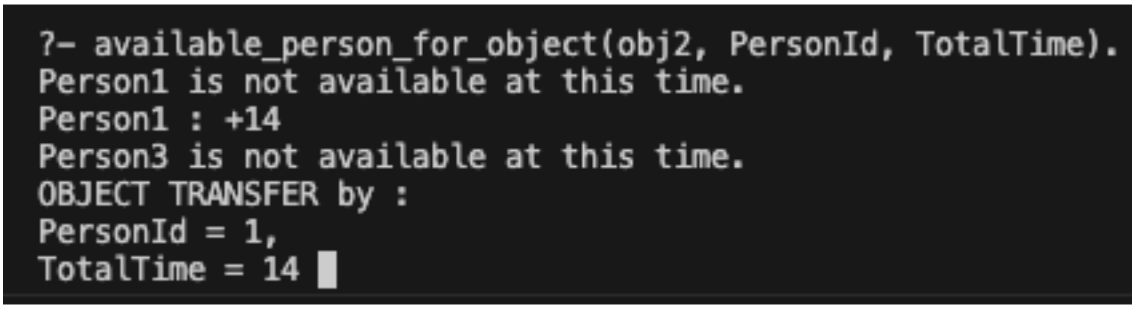
    - Test Object

```
% objects to be delivered
object(obj1, 8, adminOffice, instituteX, low, 1). % id, weight,
source, destination, priority, delivery_personnel
object(obj2, 5, cafeteria, instituteX, medium, 1).
object(obj3, 5, socialSciencesBuilding, instituteY, low,
in_transit(2)).
object(obj4, 5, library, instituteX, high, 1).
object(obj5, 5, engineeringBuilding, instituteY, high, 1).
```



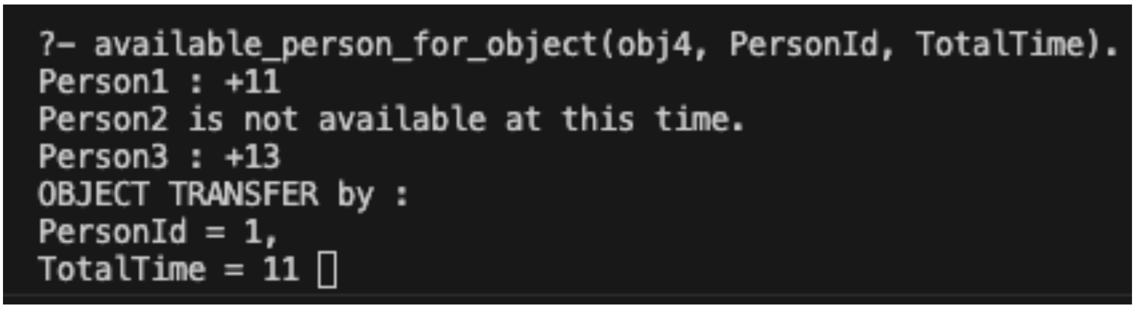
```
?- available_person_for_object(obj1, PersonId, TotalTime).
Person1 : +11
Person2 is not available at this time.
Person3 : +15
OBJECT TRANSFER by :
PersonId = 1,
TotalTime = 11
```

  - Time = 8
  - Person2 is working on 5,9,13,17,21 so it cannot available.



```
?- available_person_for_object(obj2, PersonId, TotalTime).
Person1 is not available at this time.
Person1 : +14
Person3 is not available at this time.
OBJECT TRANSFER by :
PersonId = 1,
TotalTime = 14
```

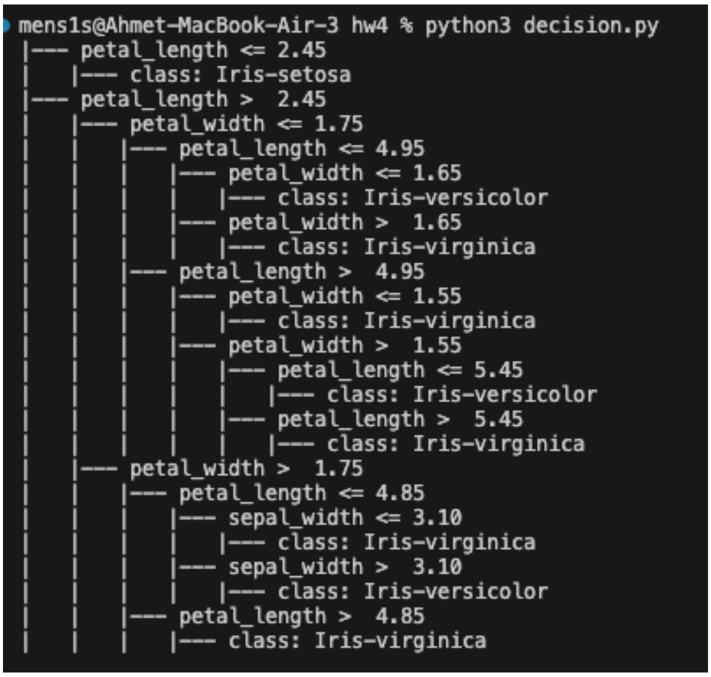
- Time = 9
- Person1-3 are working 4,8,12,16,2



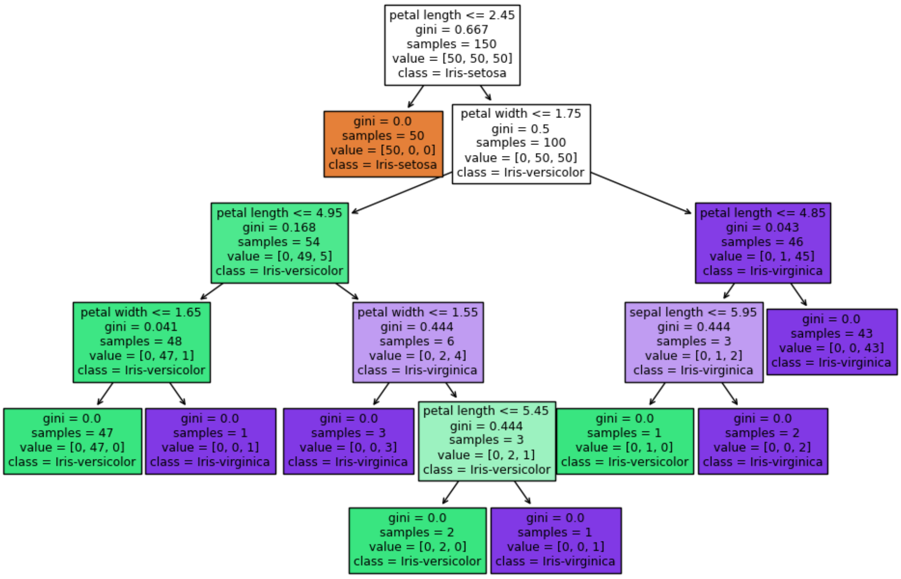
```
?- available_person_for_object(obj4, PersonId, TotalTime).
Person1 : +11
Person2 is not available at this time.
Person3 : +13
OBJECT TRANSFER by :
PersonId = 1,
TotalTime = 11
```

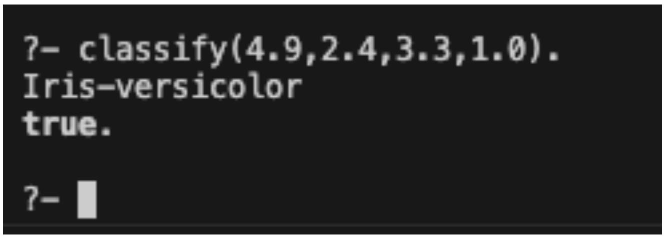
- Time = 8
- Person1-3 are working 4,8,12,16,2

- Part – 2
  - Solving approach
    - Data loading:
      - Load the Iris dataset using pandas.
    - Data preparation:
      - Separate features X and target y and split the data info training and testing sets.
    - Model Creation and Training:
      - Create a decision tree classifier, train it using the training data.
    - Generate Decision Rule:
      - Print the rules and copy to txt.
    - Export this rules to Prolog:
      - Using printed rules, defined all information about the iris data and test it.

- Output:
  - 

```
mens1s@Ahmet-MacBook-Air-3 hw4 % python3 decision.py
|--- petal_length <= 2.45
|   |--- class: Iris-setosa
|--- petal_length >  2.45
|   |--- petal_width <= 1.75
|   |   |--- petal_length <= 4.95
|   |   |   |--- petal_width <= 1.65
|   |   |   |   |--- class: Iris-versicolor
|   |   |   |--- petal_width >  1.65
|   |   |   |   |--- class: Iris-virginica
|   |   |--- petal_length >  4.95
|   |   |   |--- petal_width <= 1.55
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- petal_width >  1.55
|   |   |   |   |--- petal_length <= 5.45
|   |   |   |   |   |--- class: Iris-versicolor
|   |   |   |   |--- petal_length >  5.45
|   |   |   |   |   |--- class: Iris-virginica
|   |--- petal_width >  1.75
|   |   |--- petal_length <= 4.85
|   |   |   |--- sepal_width <= 3.10
|   |   |   |   |--- class: Iris-virginica
|   |   |   |--- sepal_width >  3.10
|   |   |   |   |--- class: Iris-versicolor
|   |   |--- petal_length >  4.85
|   |   |   |--- class: Iris-virginica
```

  - This output can be change because of random statement.
  - 

  - 

```
?- classify(4.9,2.4,3.3,1.0).
Iris-versicolor
true.

?- 
```