SAAD DAHLEB BLIDA 01 UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

2023-2024/S1

REPORT OF - PUBLIC TRANSPORT JOURNEY –PROJECT-PROLOG

Abdelatif Mekri

Intelligent Systems Engineering ,

Master's Student @USDB

11/27/2023

Table of Contents

INTRODUCTION :

The aim of the project is to develop a ProLog program capable of helping a user
identify the best route to get from one station to another, while respecting certain
certain conditions (departure time, arrival time, minimizing the number of connecting
number of connections or travel time, etc.).
The first questions in the statement are very detailed, while those towards the end are less
less exhaustive, to stimulate your creativity.

IMPLIMENTATION :

We took the provided predicates on the metro.pl file and added it to the main project file for easier manipulation and testing along the progress of writing the prolog code , aknowledging that we could have used the predefined predicate consult(name of file.pl) in our case consult(metro.pl) , to read the predicates like any other kind of developped programming languages .

A little demonstration about what has been implimented along the code :

EXO 01:

addh/3: Adds a number of minutes (N) to a given time ([X, Y]).

affiche/1: Displays a time in a specific format.

EXO 02:

membre/2: Checks if an element A is a member of a given list.

lig/3: Checks if there is a line (L) between two stations (A1 and A2).

rang/3: Finds the position (R) of a station (A) in the station list of a metro or bus line.

dernier/2: Finds the last element (N) in a list.

premier/2: Finds the first element (X) in a list.

directionligne/4: Determines the direction (D) of a metro or bus line (L) between two stations (A1 and A2).

diffh/3: Calculates the difference in minutes (M) between two time representations.

departtot/5: Calculates the departure time (A) from station A1 to A2 given the current time (H) and the line (L).

listedepart/4: Finds all possible departure times (R) from station A1 to A2 given the current time (H).

minimum/2: Finds the minimum element in a list of pairs (T, X).

premierdepart/4: Finds the first departure time from station A1 to A2 given the current time (H).

ligtot/3: Finds the line (L) and its corresponding departure time (H) from station A1 to A2 given the current time (H).

somme/2: Calculates the sum (T) of the elements in a list.

somme1/2: Calculates the sum (T) of the second elements in a list of pairs.

avan/3: Finds the sublist of elements in a list that occur before a given element.

fin/3: Finds the sublist of elements in a list that occur after a given element.

trajet/4: Finds the list of stations between two stations (A1 and A2) on a line (L).

tempstrajet/3: Calculates the total time (T) for a journey between two stations (A1 and A2) on a line (L).

arrivee/5: Calculates the arrival time (T) at station A2 given the departure time (H) from A1 and the line (L).

listearrivee/4: Finds all possible arrival times (R) at station A2 from A1 given the departure time (H).

dernierearrivee/4: Finds the last arrival time (X, T) at station A2 from A1 given the departure time (H).

ligtard/4: Finds the line (L) that arrives the latest at station A2 from A1 given the departure time (H).

EXO 03:

ajoute/3: Adds an element X to the beginning of a list Liste.

lister/2: Extracts the station names from a list of station-duration pairs.

lister_tt/3: Extracts station names from a list of lists of station-duration pairs.

li/3: Flattens a list of lists into a single list.

sup/4: Removes elements A and A2 from a list.

recup_der/2: Retrieves the last element from a list.

inter1/3: Finds the intersection of two lists.

ajout_fin/3: Adds an element X to the end of a list.

ajout_debut/3: Adds an element X to the beginning of a list.

renverse_liste/3: Reverses a list.

concatene/3: Concatenates two lists.

concatene2/3: Concatenates a list of lists.

fact/4: Appends a station-duration pair [A, N] to a list B.

renverse/3: Reverses a list.

en_arret/1: Finds all stations in the metro network.

ensemble_liste/1: Creates a list of all stations in the metro network.

parcours_sup/4: Finds a list of stations that can be reached from a given station.

liste_arret_conect/4: Finds a list of connected stations for a given list of stations.

liste_seif/4: Finds a list of stations that can be reached from A1 from station A given the current time H.

o_list/4: Finds the list of stations that can be reached from A1 from station A.

extra_par/5: Finds additional stations that can be reached from A1 from station A.

k_lig/3: Finds all lines connecting stations A and A1.

extra_list/3: Finds additional stations that can be reached from A1 from station A.

par/5: Finds pairs of stations and their connected stations in both directions.

parcours/3: Finds a list of connected stations between A and A1.

parcours_aux/3: Finds connected stations between A and A1 and adds them to a list.

Averif/4: Verifies if there are additional stations that can be reached from A1 from station A.

parcours_metro/3: Finds connected stations between A and A1 in the metro network.

parcours_metro22/5: Appends additional stations that can be reached from A1 from station A.

parcours_metro2/3: Finds additional stations that can be reached from A1 from station A in the metro network.

plutot/4: Finds the earliest connection between two stations given the current time.

itintot/4: Finds the earliest connection between two stations given the current time.

plutard2/3: Finds the latest connection between two stations given the current time.

plutard/3: Finds the latest connection between two stations given the current time.

itintard/4: Finds the latest connection between two stations given the current time.

EXO 04:

affiche_resultat_tot/3: Displays the result for the earliest schedule, including the line and stations.

affiche_resultat_tard/3: Displays the result for the latest schedule, including the line and stations.

affiche_station1/5: Displays stations for a given route with the minimum number of stations.

affiche_station2/6: If Aux is between the positions RA and RB, includes it in the list.

station/3: Displays the list of stations for a given route.

affichage_station/2: Displays the station list.

addh/3: Adds minutes to the given time.

addh1/3: Returns the time when adding M minutes to the given time.

ligar/7: Connects stations A and B through the metro line M.

ligaller/3: Checks if it's an outbound journey.

ligaller1/5: Same as ligaller, but also returns the positions of stations A and B.

ligallerR/5: Same as ligaller1, but without checking the outbound direction.

calc/4: Calculates the position of station A in metro X relative to its initial position.

truealler/2: Checks if it's an outbound journey based on station numbers.

addNbstat/4: Adds minutes to the given time based on the number of stations to the departure station.

accumultard/5: Multiplies the time by the minutes Acc until it exceeds the given time T.

accumultot/5: Multiplies the time by the minutes Acc until it exceeds T.

compareMin/2: True if the schedule exceeds the given time.

plustot/2: Returns the smallest time in the list.

plustot1/3: Finds the smallest time among the list.

plustard/2: Returns the largest time in the list.

plustard1/3: Finds the largest time among the list.

time_n_ligne/3: Creates an element containing the metro name with the nearest schedule for that metro.

ligtot1/6: Finds the earliest schedule considering the minimum number of stations.

ligtard1/6: Finds the latest schedule considering the minimum number of stations.

ligar1/7: Connects stations A and B through the metro line M, considering the minimum number of stations.

nbstation/4: Counts the number of stations.

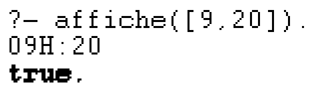
time_n_ligne1/4: Adds the line, the number of stations, and the time to a variable Z.

moinstat/2: Finds the line with the fewest stations.

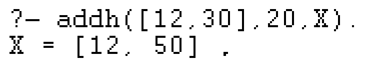
moinstat1/3: Finds the line with the fewest stations among the list.

PRACTICE :

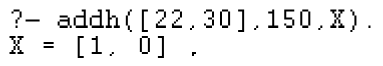
Some test examples :



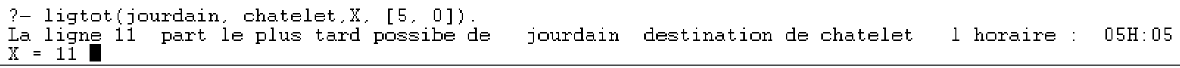
```
?- affiche([9,20]).
09H:20
true.
```



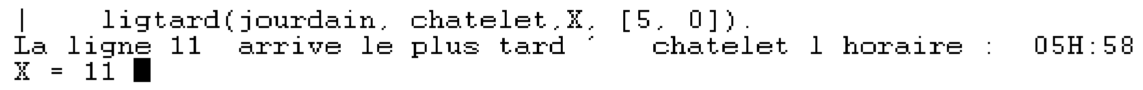
```
?- addh([12,30],20,X).
X = [12, 50] .
```



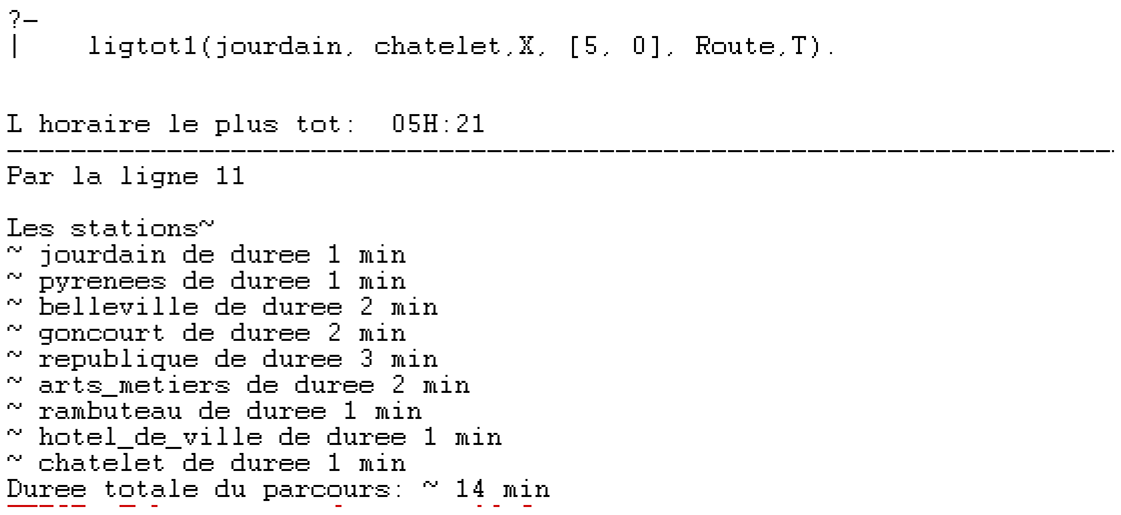
```
?- addh([22,30],150,X).
X = [1, 0] .
```



```
?- ligtot(jourdain, chatelet,X, [5, 0]).
La ligne 11  part le plus tard possibe de   jourdain  destination de chatelet   l horaire :  05H:05
X = 11 █
```



```
 |    ligtard(jourdain, chatelet,X, [5, 0]).
La ligne 11  arrive le plus tard ´   chatelet l horaire :  05H:58
X = 11 █
```



```
?-
|    ligtot1(jourdain, chatelet,X, [5, 0], Route,T).


L horaire le plus tot:  05H:21
----------------------------------------------------------------------
Par la ligne 11

Les stations~
~ jourdain de duree 1 min
~ pyrenees de duree 1 min
~ belleville de duree 2 min
~ goncourt de duree 2 min
~ republique de duree 3 min
~ arts_metiers de duree 2 min
~ rambuteau de duree 1 min
~ hotel_de_ville de duree 1 min
~ chatelet de duree 1 min
Duree totale du parcours: ~ 14 min
```

```
?- ligtard1(jourdain, chatelet,X, [5, 0], Route,T).
iciPas d itineraire pour le moyen de transport demande
```

REFRENCES :

SITES :

https://www.swi-prolog.org/pack/list?p=date_time

https://www.swi-prolog.org/pldoc/man?section=timedate

https://stackoverflow.com/questions/37375133/prolog-compare-elements-of-list

OpenAI. (2022). GPT-3.5 "ChatGPT." Retrieved [20/27-11-2023] from https://www.openai.com/

Université Paris 13. (n.d.). TP Prolog Source for Students. Retrieved [date] from https://lipn.univ-paris13.fr/~pagani/TP_Prolog/src_etudiants/

PAPERS & THESISES :

BOUMERTIT, K., & KHELFAOUI, M. (2010, May 22). RAPPORT PROGRAMMATION LOGIQUE. Institut Galilée.

PHUNG, W., & ALI, I. (2010, May 21). Projet de Programmation Logique: Rapport - Calcul d'un trajet de transports publics. L3 INFO,Uiversite du Paris Nord