Projet du pôle

Logique Computationnelle – Prolog
Complexité– Décidabilité

Thème du projet

Calcul d'itinéraires dans un réseau ferroviaire

1 Contexte et problématique

L'objectif est d'utiliser la logique des prédicats, les notions de complexité et Prolog pour gérer un réseau ferré.

La carte schématique donnée en appendice A représente le réseau ferré entre 19 villes du Canada.

Les données correspondantes sont à l'appendice B.

On suppose que nous disposons de deux types de trains :

- des trains de passagers dont la vitesse maximale par trajet est précisée dans le fichier des données, et
- des trains des marchandises avec une vitesse maximale de 120 km/h.

On suppose que les trains roulent à leur vitesse maximale.

2 Base de données

Les données se trouvent en appendice B.

Les données concernent :

- les liaisons entre deux villes, la vitesse maximale que les trains de passagers puissent atteindre entre ces deux villes et aussi la distance qui les sépare.
- le temps d'arrêt des trains passagers et des trains marchandises pour chaque ville.

Il est demandé de stocker ces données dans autant de bases de données que nécessaire mais d'une façon parcimonieuse.

Il est à noter que la structure des bases de données dépend de leur utilisation. On peut privilégier la vitesse de traitement ou l'espace disque occupé. Il est judicieux de pouvoir faire un compromis raisonné.

Livrable : un rapport présentant la structure de(s) fichier(s) et la justification de cette structure.

3 Interrogation de(s) base(s) de données

On doit pouvoir poser les questions suivantes :

1. Liaisons entre deux villes. La réponse doit préciser :
   – toutes les liaisons possibles ;
   – les villes par lesquelles on passe ;
   – le temps total du trajet, en y incluant le temps d'arrêt dans les gares intermédiares, et
   – la longueur total du trajet.

   Exemple : `? liaison(bradford, hamilton).`
   La réponse est
   1.- `bradford - hamilton, liaison directe, 19 mn, 40 km`
   2.- `bradford - hamilton, via kitchener, toronto, 92 mn, 190 km`
   Vérifier, en utilisant l'arbre de résolution SLD, que la réponse obtenue à l'exemple précédent est celle qui était attendue.
2. Les villes qui sont en liaison directe avec une ville précise. La réponse doit fournir
   – le nom de la ville
   – le temps du trajet, et
   – la longueur du trajet.

   Exemple : `? liasonDirecte(kingston).`
   La réponse est
   1.- `brocville, 20 mn, 80 km`
   2.- `toronto, 63 mn, 260 km`
3. La ville qui est la plus proche d'une ville donnée. La réponse doit fournir
   – le nom de la ville
   – le temps du trajet, et
   – la longueur du trajet.

   Dans le cas où il y a plus d'une ville à égale distance, il faut donner les noms de toutes les villes.
   Exemple : `? liasonPlusCourte(chatman).`
   La réponse est
   1.- `windsor, 25 mn, 70 km`
   2.- `sarnia, 27 mn, 70 km`
   Vérifier que les réponses obtenues pour cet exemple, font partie du modèle minimal de Herbrand.

Livrable : un rapport qui contient

1. les réponses aux deux questions de logique, et
2. le cahier d'analyse des programmes que vous avez écrit pour répondre aux trois questions.

4 Recherche d'un trajet

L'objectif de cette section est de trouver des trajets selon différents critères.

1. Étant données deux villes on cherche tous les trajets qu'il est possible d'effectuer pour aller de l'une à l'autre, sans passer deux fois par la même ville. On écrira le prédicat :
   `trajet(Origine, Destination, TypeTrain, Trajet, Longueur, Duree).`
   Exemple `? trajet(sarnia, chatman, 1, Trajet, Longueur, Duree).`
   avec réponses :
   1.- `Trajet = [sarnia, chatman], Longueur = 70, Duree = 27 ;`
   2.- `Trajet =[sarnia, london, chatman], Longeur = 200, duree = 79.`
   N.B. On suppose que les trains sont de deux types, avec type 1 : train de passagers, et type 2 : train des marchandises.
2. Étant données deux villes on cherche tous les trajets qu'il est possible d'effectuer pour aller de l'une à l'autre, sans passer deux fois par la même ville et avec une durée au plus égale à `DureeMax`. On écrira le prédicat :
   `trajetLimite(Origine, Destination, TypeTrain, DureeMax, Trajet, Longueur, Duree).`
   Exemple `? trajetLimite(sarnia, chatman, 1, 70, Trajet, Longueur, Duree).`
   avec réponses :
   `Trajet = [sarnia, chatman], Longueur = 70, Duree = 27 ;`
3. Étant données deux villes on cherche le plus court trajet (en distance) qu'il soit possible d'effectuer pour aller de l'une à l'autre. On écrira le prédicat :
   `trajetOptim(Origine, Destination, TypeTrain, Trajet, Longueur, Duree).`
   Exemple `? trajetOptim(sarnia, chatman, 1, 70, Trajet, Longueur, Duree).`
   avec réponses :
   `Trajet = [sarnia, chatman], Longueur = 70, Duree = 27 ;`

LIVRABLE : un rapport qui contient le cahier d'analyse des programmes que vous avez écrit pour répondre aux questions.

5 Recherche de circuits

Étant donnée une ville on cherche tous les circuits qui commencent et se terminent à cette ville, avec la contrainte de passer au plus une fois par chaque ville.

1. Proposer un algorithme en pseudo-code pour la recherche d'un circuit dans un graphe.
2. En utilisant cet algorithme, on écrira le prédicat :
   `circuit(Ville, Trajet, Longueur, Duree).`
   Exemple : `? circuit(bradford, Trajet, Longueur, Duree).`
   avec réponses :
   1.- `Trajet = [bradford, kitchener, london, bradford], Longueur = 205, Duree = 87 ;`
   2.- `Trajet = [bradford, london, kitchener, bradford], Longueur = 205, Duree = 87 ;`
   3.- `Trajet = [bradford, hamilton, toronto, kitchener, bradford], Longueur = 200, Duree = 116 ;`
   4.- `Trajet = [bradford, kitchener, toronto, hamilton, bradford], Longueur = 200, Duree = 116 ;`
   5.- `Trajet = [bradford, london, kitchener, toronto, hamilton, bradford], Longueur = 285, Duree = 162 ;`
   5.- `Trajet = [bradford, hamilton, toronto, kitchener, london, bradford], Longueur = 285, Duree = 162 ;`

   Exemple : `? circuit(sundbury, Trajet, Longueur, Duree).`
   avec réponse :
   `Trajet = [], Longueur = 0, Duree = 0 ;`

3. Trouver la classe de complexité à laquelle appartient cet algorithme. Vous devez justifier complètement votre réponse.

Livrable : un rapport qui contient

1. l'algorithme de la recherche du circuit ;
2. l'analyse de la complexité de l'algorithme, et
3. le cahier d'analyse du programme que vous avez écrit pour la recherche du circuit.

6 Programmes

LIVRABLE : Tous les programmes Prolog dans un seul fichier avec l'extension `*.pl`, accompagnés de la notice d'utilisation.

Tous les rapports et le fichier des programmes seront packagés dans un seul fichier zippé et envoyés en utilisant le bouton correspondant sur ***http ://sifoci.eisti.fr***, rubrique **Logique computationnelle**.

Le nom du fichier sera XYY.zip où

– X est votre groupe de TD (A, B, C pour Cergy, F pour Pau), et

– YY est votre numéro de groupe.

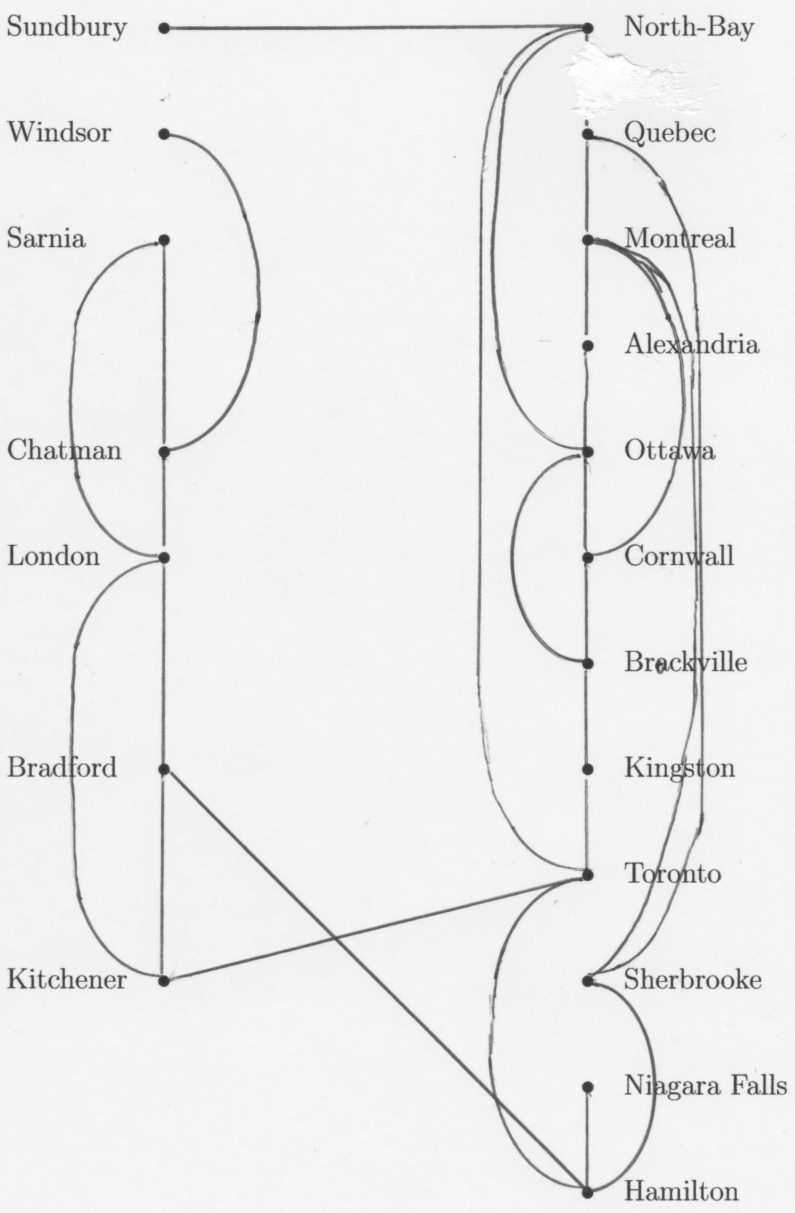
Les rapports doivent impérativement être rédigés en LaTeX.

Date de livraison : XX.XX.XXXX

NB.- Si l'un des livrables est absent, la note sera de 0 (zéro) pour l'*ensemble du projet*.
Nous vous rappelons aussi que le plagiat sera sanctionné.

APPENDICES

A Carte schématique du réseau ferré



B Données

```
%Line format:  City 1, City 2, Speed Limit (km/h), Length (km)

Montreal, Sherbrooke, 150, 145
Sherbrooke, Quebec City, 140, 215
Quebec City, Montreal, 200, 270
Montreal, Alexandria, 180, 100
Alexandria, Ottawa, 180, 100
Ottawa, Cornwall, 120, 80
Ottawa, Brockville, 120, 110
Ottawa, North Bay, 150, 350
North Bay, Sudbury, 90, 150
North Bay, Toronto, 150, 350
Montreal, Cornwall, 200, 100
Cornwall, Brockville, 210, 100
Brockville, Kingston, 230, 80
Kingston, Toronto, 250, 260
Toronto, Kitchener, 200, 100
Kitchener, London, 200, 95
London, Chatham, 200, 105
Chatham, Windsor, 170, 70
Chatham, Sarnia, 150, 70
Sarnia, London, 150, 95
London, Brantford, 180, 80
Brantford, Kitchener, 100, 30
Brantford, Hamilton, 130, 40
Hamilton, Niagara Falls, 120, 70
Hamilton, Toronto, 150, 60

%Line format:  City, Minimum freight train stop time (min), Minimum
%passenger train stop time (min).

Montreal,       20,     15
Sherbrooke,     0,      3
Quebec City,    10,     10
Alexandria,     0,      3
Ottawa,         10,     10
Cornwall,       0,      3
Brockville,     0,      3
North Bay,      0,      5
Sudbury,        0,      5
Toronto,        20,     15
Kingston,       10,     10
Kitchener,      0,      5
London,         0,      8
Chatham,        0,      3
Windsor,        10,     8
Sarnia,         10,     8
Brantford,      0,      3
Hamilton,       0,      5
Niagara Falls,  0,      3
```