Explication de l'algorithme

Antoine Gennart & Marie-Pierre van Oldeneel

28 novembre 2015

Ce document est destiné à décrire tout l'algoritme du code du fichier code.oz. Il n'est pas destiné à un quelconque proffesseur, mais juste à éclaircir la compréhension du code et des environnements contextuels des différentes fonctions et sous fonctions.

Pour rappel, l'objectif du code est de gérer un jeu de "Qui-est ce ?" dans le très répandu language informatique OZ

Table des matières

1 BuildDecisionTreeWithQ

Description : Crée un arbre de décision à partir de du base de donnée de personnes et d'une liste de personnes.

— Gardener
— BestQuestion

1.1 MakeListOfNames L

Description : A partir d'une liste de person(NOM Q :true ..... false), crée une liste contenant uniquement les NOM.

— MakeListAcc

Complexité temporelle : $O(n)$ avec n la taille de liste L.

1.2 Gardener DB ListT ListF ActualQuestion ListOfQuestions

Description : C'est réellement le jardinier du programme, c'est lui qui s'occupe de créer l'arbre de manière récursive.

— BestQuestion
— MakeListOfNames

A chaque itération, Gardener se sépare en deux dimiuant toujours la taille de la DB.

Complexité temporelle : La fonction se rappelle a chaque itération deux fois (typiquement $\Theta(n^2)$, avec n la taille de la liste de questions), en changeant ses argument avec l'aide de la fonction BestQuestion ($O(n_1 \dot{n}_2)$).

On peut donc définir la complexité temporelle de cette fonction comme $\Theta(n_1 \cdot n_2^2)$ ou n_2 est la taille de la liste de question, n_1 la taille de la DB.

2 BestQuestion DB ListOfQuestions

Description : Choisis dans une liste de questions laquelle conviendra le mieux pour éliminer le plus de personnes possible d'une certaine base de donnée.

— DiffTrueFalseList
— MinList
— RemoveList

Connaissant la complexitées temporelles des sous fonction, on peut calculer la complexité temporelle de BestQuestion qui appelle chacune des fonctions séparément.

Complexité Temporelle : $O(n_1 \cdot n_2) + O(n_2) + O(n_2)$ ou n_1 est la taille de DB et n_2 la taille de ListOfQuestions

2.1 RemoveList L Nth

Description : Retire le Nième élément d'une liste

— RemoveListAcc

Complexité temporelle : $O(n)$ ou n est la longueur de la liste

2.2 MinList L

Description : Retourne l'indice de l'emplacement du plus petit élément de la liste

— MinListACc

Complexité temporelle : $O(n)$ ou n est la longueur de la liste L

2.3 DiffTrueFalseList DB ListOfQuestions

Description : Crée une liste dont chaque élément est la valeur absolue de la différence des réponses true et false par les personnes à laquelle on soustrait le nombre de personne qui ont répondu a la question.

— DiffTrueFalseListAcc

Complexité temporelle : $O(n1 \cdot n2)$ ou n1 est la taille de la DB et n2 la taille de ListOfQuestions

3 BuildDecisionTree

Description : Crée un arbre de décision juste à partir de la base de donnée. C'est cette fonction que l'on doit donner au player pour lancer l'interface graphique.

— MakeListOfQuestions
— BuildDecisionTreeWithQ

4 MakeListOfQuestions DB

Description : Fait une liste des questions a partir de toutes celle qui apparaisent dans une base de donnée.

— MakeListOfQuestionsAcc
— RemoveDouble

Complexité temporelle : $\Theta(n_1 \cdot n_2)$ avec n_1 la taille de la base de donnee et n_2 le nombre total de questions (de tous les joueurs).

4.1 RemoveDouble List

Description : Retire toutes les valeurs qui apparaissent deux fois dans une liste.

— RemoveDoubleAcc
— IsIn

Complexité temporelle : $O(n)$ ou n est la taille de la List

4.2 IsIn X L

Description : Fonction Boolean qui regarde si un élément appartient à une liste

Complexité temporelle : $\Theta(n)$, $\Omega(1)$ ou n est la taille de la liste L

5 GameDriver

Description : Pilote du jeux qui va se ballader dans l'arbre pour poser les bonnes questions à l'utilisateur. Si l'utilisateur est con et qu'il ne sait pas, il doit recreer un arbre.

— GameDriverAcc

5.1 NewDataBase DB

Description : Met à jour la base de donnée en retirant les personnes qui n'ont pas la même réponses que celle décidée par l'utilisateur.

— NewDBAcc

Complexité temporelle : $O(n)$ ou n est la taille de la DB

5.2 NewListOfQuestions ActualQ LQ

Description : Retire la dernière questions posée de la liste des questions.

— RemoveList

Complexité temporelle : $O(n)$ ou n est la taille de la liste de questions

5.3 GameDriverAcc

Description : Fonction récursive du GameDriver.

— NewListOfQuestions
— NewDataBase
— BuildDecisionTreeWithQ
— GameDriverAcc

Complexité temporelle : $\Theta(n_1 * O(BuildDecisoinTree))$

Le pire cas est lorque l'utilisateur ne connait pas la réponse, il faut donc faire appel n fois à la fonction BuildDecisionTreeWithQ