Yann DEBAIN

Antoine MILCENT

**Rapport de préconception : Projet : Problème du Voyageur de Commerce**

**Introduction**

Sommaire :

I)Types de données utilisées

II)Modules

III) Fonctions

IV)Tests prévus

V)Répartition des tâches

**I)Types de données utilisées**

La carte du problème est représenté par un graphe, ce graphe est implémenté par un tableau de sommets.

Un arc est un quadruplet :

* numéro de la ville de départ
* numéro de la ville d'arrivée
* distance entre les deux
* quantité de phéromones

Chacun des sommets contient les informations suivantes :

* Le numéro du sommet (numéro de la ville)
* Le nom du sommet (nom de la ville)
* les coordonnées du sommet
* La liste de ses voisins

la liste des voisins est la liste d'adjacence associée à ce sommet, c'est une liste chaînée répertoriant les pointeurs sur les arcs qui partent de ce sommet.

Les fourmis virtuelles sont des entités qui parcourent l’intégralité du graphe en mémorisant leur parcours (appelé solution de la fourmi).

A chaque fois que la fourmis part d'une ville, elle décide dans quelle ville elle se va se rendre. Cette décision est aléatoire mais influencée par la distance entre les villes considérées et la quantité de phéromone déposée sur le chemin vers la ville.

Chaque fourmi est représentée par :

* Sa solution, c'est à dire l'ensemble des pointeurs d'arcs entre les villes déjà visitées qui sera une “file”.
* Le numéro de la ville de laquelle elle est partie
* Le numéro de la ville dans laquelle elle se trouve actuellement

|  |
| --- |
| //  // structure.h  // Graphe LPC  //  // Created by Yann Debain on 28/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  #ifndef structure\_h  #define structure\_h  typedef struct arc\* ELEMENT;  struct cellule {  ELEMENT val;  struct cellule \* suiv;} ;  typedef struct cellule \* Liste;  //definition de la structure d'un arc  struct arc {  unsigned int depart;  unsigned int arrive;  double cout;  double pheromone;  };  //definition de la structure d'un sommet  struct sommet {  char nom;  unsigned int numero;  //unsigned int x;  //unsigned int y;  Liste voisin;  };  typedef Liste File;  struct fourmis {  unsigned int ville\_depart;  unsigned int ville\_courante;  File solution;  };  typedef struct arc ARC;  typedef struct arc\* ELEMENT;  typedef struct sommet SOMMET;  typedef struct sommet\* GRAPHE;  typedef struct fourmis FOURMIS;  #endif /\* structure\_h \*/ |

**II)Modules**

On découpe le problème en différent modules pour pouvoir se répartir le travail et avoir une meilleure organisation du code.

* structures liées à l'algorithme
* chargement des graphes (fichiers)
* construction des graphes
* construction des listes
* gestion des fourmis
* tests
* Makefile

**III)Fonctions**

Les prototypes des fonctions se trouvent dans les “.h”

|  |
| --- |
| //  // graphe.h  // Graphe LPC  //  // Created by Yann Debain on 15/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  #ifndef graphe\_h  #define graphe\_h  #include <stdio.h>  #include "structure.h"  GRAPHE creer\_graphe(unsigned int taille); //retourne un graphe vide  int est\_voisin (GRAPHE g, SOMMET x, SOMMET y); //retourne un booleen: 1 si x et y sont voisins 0 sinon  Liste voisins(GRAPHE g, SOMMET x); //retourne la liste des voisins de x  GRAPHE ajout\_arc(GRAPHE g, SOMMET x, SOMMET y); //ajout d'un arc entre x et y si absent  GRAPHE supp\_arc (GRAPHE g, SOMMET x, SOMMET y); //supprime l'arc entre x et y si present  Liste supprime\_numero\_sommet(int numero\_sommet\_arrive, Liste L);  int valeur\_arc(GRAPHE g, SOMMET x, SOMMET y); //retourne le cout de l'arc entre x et y si present, +inf sinon  void visualiser\_graphe(GRAPHE g, unsigned int taille); //visualise le graphe x  #endif /\* graphe\_h \*/ |
| //  // liste.h  // Graphe LPC  //  // Created by Yann Debain on 15/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  #ifndef \_LISTE  #define \_LISTE  #include <stdio.h>  #include <string.h>  #include "structure.h"  Liste creer\_liste(void);  int est\_vide(Liste L);  Liste rech(ELEMENT e, Liste L);  Liste ajout\_tete(ELEMENT e, Liste L);  void visualiser\_liste(Liste L);  #endif |
| //  // fourmis.h  // Graphe LPC  //  // Created by Yann Debain on 28/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  #ifndef fourmis\_h  #define fourmis\_h  #include <stdio.h>  #include "structure.h"  //definition des différentes variables pour “guider” la fourmis  #define NOMBRE\_DE\_FOURMIS 128 //optimisation en 2^m  #define COEFFICIENT\_EVAPORATION\_PHEROMONE 0,5  #define IMPORTANCE\_PHEROMONE 1  #define IMPORTANCE VISIBILITEE 2  #define VALEUR\_INITIALE\_PHEROMONE 1E-5  #define QUANTITE\_PHEROMONE\_A DEPOSEE 1  #define MAX\_CYCLE 50  SOMMET choix\_prochaine\_ville(SOMMET x); //determine dans quelle ville la fourmis va y aller  double mise\_a\_jour\_des\_pheromones(double phero); //recalcule la quantite de pheromones sur un arc  File parcours(FOURMIS f); //File qui contient le parcours de la fourmis f  #endif /\* fourmis\_h \*/ |
| #ifndef \_FILE  #define \_FILE  #include <stdio.h>  #include "structure.h"  //  // file.h  // Graphe LPC  //  // Created by Yann Debain on 15/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  File creer\_file(void); //Créer une file vide  int file\_vide(File f); //Teste si la file est vide  void visualiser\_file(File f);  File enfiler(SOMMET x, File f); //Ajoute un sommet à la file par la queue  #endif /\* \_FILE \*/ |
| //  // fichiers.h  // Graphe LPC  //  // Created by Yann Debain on 28/04/2017.  // Copyright © 2017 Debain-Milcent. All rights reserved.  //  #ifndef fichiers\_h  #define fichiers\_h  #include <stdio.h>  #include "structure.h"  int chargeFichierGraphe(GRAPHE\* pGraphe, char\* nomDuFichier);  #endif /\* fichiers\_h \*/ |

**IV)Tests prévus**

Les tests seront faits en deux parties. Tout d'abord celui qui écrit le programme fera des tests simples pour vérifier si le programme est bien écrit, compile et s’exécute en donnant ce qu’il attend. Puis celui qui n’a pas écrit le programme cherchera à “casser” le code pour voir si le programme prend en compte toutes les possibilités.

On testera donc en premiers lieux le chargement de fichiers, la gestion des graphes puis après la gestion des fourmis et enfin l’ensemble des fichiers pour répondre à la problématique du sujet

**V)Répartition des tâches**

Yann DEBAIN :

* Définition des structures
* Gestion des graphes
* Gestion des fourmis
* Tests

Antoine MILCENT :

* Définition des structures
* Gestion des fichiers
* Makefile
* Tests

Les fonctions pour les listes et les files ont déjà été écrites lors des séances précédentes donc on les réutilise avec quelques adaptations pour correspondre au sujet.