Projet VDC

1.0

Généré par Doxygen 1.8.6

Mardi 15 Avril 2014 19 :43 :29

Table des matières

Chapitre 1

Index des structures de données

1.1 Structures de données

Liste des structures de données avec une brève description :

Chapitre 2

Index des fichiers

2.1 Liste des fichiers

Liste de tous les fichiers avec une brève description :

Chapitre 3

Documentation des structures de données

3.1 Référence de la structure arbrePlanaireGen

Champs de données

— Noeud racine
— ptr_affichage affiche

3.1.1 Documentation des champs

3.1.1.1 ptr_affichage affiche

3.1.1.2 Noeud racine

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/ArbrePlanaireGenerique.c

3.2 Référence de la structure arbrePlanaireInt

Champs de données

— ArbrePlanaireGen arbre

3.2.1 Documentation des champs

3.2.1.1 ArbrePlanaireGen arbre

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/ArbrePlanaireInt.c

3.3 Référence de la structure Arete

Arête entre la ville depart et la ville arrive.

Champs de données

— double cle

— int depart
— int arrive

3.3.1 Description détaillée

Arête entre la ville depart et la ville arrive.

3.3.2 Documentation des champs

3.3.2.1 int arrive

3.3.2.2 double cle

3.3.2.3 int depart

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Arete/Arete.c

3.4 Référence de la structure AreteHandle

Champs de données

— ElemHandle arete

3.4.1 Documentation des champs

3.4.1.1 ElemHandle arete

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Tas/TasArete.c

3.5 Référence de la structure elemHandle

Champs de données

— void * elem
— int handle

3.5.1 Documentation des champs

3.5.1.1 void* elem

3.5.1.2 int handle

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Tas/TasGenerique.c

3.6 Référence de la structure graphe

Crée un graphe aillant pour longueur taille et en paramètre des doubles contenus dans matrice.

Champs de données

— int taille
— double ∗∗ matrice

3.6.1 Description détaillée

Crée un graphe aillant pour longueur taille et en paramètre des doubles contenus dans matrice.

Crée un graphe aillant pour longueur len et contient des doubles contenus dans matrice.

Paramètres

mat	Tableau de tableau (tableau "2D") contenant des doubles.

3.6.2 Documentation des champs

3.6.2.1 double∗∗ matrice

3.6.2.2 int taille

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Graphe/Graphe.c

3.7 Référence de la structure input

Champs de données

— char ∗ nom_file
— char ∗ nom
— char ∗ type
— char ∗ commentaire
— int dimension
— char ∗ edge_weight_type
— char ∗ edge_weight_format
— char ∗ display_data_type
— double ∗∗ edge_weight_matrix
— double ∗∗ display_data

3.7.1 Documentation des champs

3.7.1.1 char∗ commentaire

3.7.1.2 int dimension

3.7.1.3 double∗∗ display_data

3.7.1.4 char∗ display_data_type

3.7.1.5 char∗ edge_weight_format

3.7.1.6 double∗∗ edge_weight_matrix

3.7.1.7 char∗ edge_weight_type

3.7.1.8 char∗ nom

3.7.1.9 char* nom_file

3.7.1.10 char* type

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Input/Input.c

3.8 Référence de la structure noeud

Champs de données

— void * elem
— Noeud pere
— Noeud premierFils
— Noeud frere

3.8.1 Documentation des champs

3.8.1.1 void* elem

3.8.1.2 Noeud frere

3.8.1.3 Noeud pere

3.8.1.4 Noeud premierFils

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/ArbrePlanaireGenerique.c

3.9 Référence de la structure TasArete

Champs de données

— TasMinGen tas

3.9.1 Documentation des champs

3.9.1.1 TasMinGen tas

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Tas/TasArete.c

3.10 Référence de la structure TasMinGen

Champs de données

— ElemHandle * sommets
— int taille
— int taille_tas
— ptr_compar comparaison
— ptr_compar comparaisonCle
— ptr_affichage affichage
— ptr_maj affecte

3.10.1 Documentation des champs

3.10.1.1 ptr_maj affecte

3.10.1.2 ptr_affichage affichage

3.10.1.3 ptr_compar comparaison

3.10.1.4 ptr_compar comparaisonCle

3.10.1.5 ElemHandle* sommets

3.10.1.6 int taille

3.10.1.7 int taille_tas

La documentation de cette structure a été générée à partir du fichier suivant :
— /net/cremi/twybrech/TSP-Project/src/Tas/TasGenerique.c

Chapitre 4

Documentation des fichiers

4.1 Référence du fichier /net/cremi/twybrech/TSP-Project/include/ArbrePlanaireGenerique.h

Module pour la création et les opérations utilisables sur les arbres planaires génériques.

```
#include <stdbool.h>
```

Définitions de type

- typedef void(∗ ptr_affichage)(void ∗)
- typedef struct noeud ∗ Noeud
- typedef struct arbrePlanaireGen ∗ ArbrePlanaireGen

Fonctions

- Noeud creerNoeud (void ∗element, Noeud parent, Noeud aine, Noeud cadet)
  Crée un noeud.
- void freeNoeud (Noeud this)
  Libère l'espace mémoire alloué à un noeud.
- ArbrePlanaireGen creerArbrePlanaireGen (ptr_affichage f)
  Crée un arbre planaire générique vide.
- void freeArbrePlanaireGen (ArbrePlanaireGen this)
  Libère l'espace mémoire alloué à un arbre planaire générique après avoir détruit l'arborescence de la racine.
- void ∗ getElement (Noeud this)
  Retourne l'élément stocké dans un noeud.
- Noeud getPremierFils (Noeud this)
  Retourne le premier fils d'un noeud.
- Noeud getPere (Noeud this)
  Retourne le père d'un noeud.
- Noeud getFrere (Noeud this)
  Retourne le frère d'un noeud.
- Noeud getRacine (ArbrePlanaireGen this)
  Retourne la racine d'un arbre planaire générique.
- void affichagePrefixe (ArbrePlanaireGen this)
  Affiche le parcours préfixe récursif de l'arbre.
- Noeud ajouterFils (ArbrePlanaireGen a, Noeud pere, void ∗elem)
  Ajoute un fils à un noeud en lui donnant l'élément elem.
- bool estFeuille (Noeud this)
  Retourne faux si le noeud entré en paramètre à un fils, vrai s'il n'en a pas.
- void supprimerNoeud (ArbrePlanaireGen a, Noeud this)
  Supprime un noeud d'un arbre planaire après avoir libéré l'espace mémoire à l'entier contenu dans le noeud.

4.1.1 Description détaillée

Module pour la création et les opérations utilisables sur les arbres planaires génériques.

Auteur

Delmas Rémi

4.1.2 Documentation des définitions de type

4.1.2.1 typedef struct arbrePlanaireGen* ArbrePlanaireGen

4.1.2.2 typedef struct noeud* Noeud

4.1.2.3 typedef void(* ptr_affichage)(void *)

4.1.3 Documentation des fonctions

4.1.3.1 void affichagePrefixe (ArbrePlanaireGen *this*)

Affiche le parcours préfixe récursif de l'arbre.

4.1.3.2 Noeud ajouterFils (ArbrePlanaireGen *a,* Noeud *pere,* void * *elem*)

Ajoute un fils à un noeud en lui donnant l'élément elem.

Précondition

Il faut mettre le père à NULL pour l'ajout en tant que racine.

4.1.3.3 ArbrePlanaireGen creerArbrePlanaireGen (ptr_affichage *f*)

Crée un arbre planaire générique vide.

Paramètres

Prend	un ptr_affichage qui est un pointeur générique en paramètre.

4.1.3.4 Noeud creerNoeud (void * *element,* Noeud *parent,* Noeud *aine,* Noeud *cadet*)

Crée un noeud.

Paramètres

Un	pointeur générique et trois noeuds pour déterminer la place du noeud crée par rapport à ceux déjà existant.

4.1.3.5 bool estFeuille (Noeud *this*)

Retourne faux si le noeud entré en paramètre à un fils, vrai s'il n'en a pas.

4.1.3.6 void freeArbrePlanaireGen (ArbrePlanaireGen *this*)

Libère l'espace mémoire alloué à un arbre planaire générique après avoir détruit l'arborescence de la racine.

4.1.3.7 void freeNoeud (Noeud *this*)

Libère l'espace mémoire alloué à un noeud.

4.1.3.8 void* getElement (Noeud *this*)

Retourne l'élément stocké dans un noeud.

4.1.3.9 Noeud getFrere (Noeud *this*)

Retourne le frère d'un noeud.

4.1.3.10 Noeud getPere (Noeud *this*)

Retourne le père d'un noeud.

4.1.3.11 Noeud getPremierFils (Noeud *this*)

Retourne le premier fils d'un noeud.

4.1.3.12 Noeud getRacine (ArbrePlanaireGen *this*)

Retourne la racine d'un arbre planaire générique.

4.1.3.13 void supprimerNoeud (ArbrePlanaireGen *a,* Noeud *this*)

Supprime un noeud d'un arbre planaire après avoir libéré l'espace mémoire à l'entier contenu dans le noeud.

Postcondition

le noeud supprimé est remplacé par son frere s'il était le premier fils de son père, sinon le frère de son frère devient le frère du sommet à détruire.

4.2 Référence du fichier /net/cremi/twybrech/TSP-Project/include/ArbrePlanaireInt.h

Module pour la création et les opérations utilisables sur les arbres planaires qui contiennent des int.

```
#include <ArbrePlanaireGenerique.h>
```

Définitions de type

- typedef struct arbrePlanaireInt * ArbrePlanaireInt

Fonctions

- ArbrePlanaireInt creerArbrePlanaireInt ()
  Crée un arbrePlanaireInt vide.
- void freeArbrePlanaireInt (ArbrePlanaireInt this)
  Libère l'espace mémoire alloué à un arbrePlanaireInt.
- int getInt (Noeud this)
  renvoie la valeur entière stockée dans le noeud entré en paramètre.
- void tableauArbreInt (ArbrePlanaireInt this, int *tab, int taille)
  remplit le tableau tab avec les valeurs contenues dans l'arbrePlanaireInt en parcours préfixe.
- void affichagePrefixeInt (ArbrePlanaireInt this)
  Affiche le parcours préfixe récursif de l'arbre.
- Noeud ajouterNoeudInt (ArbrePlanaireInt this, Noeud pere, int elem)
  Ajoute un noeud pere contenant l'entier elem à un arbrePlanaireInt.
- bool estUneFeuille (Noeud this)

Retourne vrai si le noeud entré en paramètre est une feuille.
- void supprimerNoeudInt (ArbrePlanaireInt a, Noeud this)
  Supprime un noeud d'un arbrePlanaireInt après avoir libéré la mémoire allouée à l'entier contenu dans le noeud.
- void afficheInt (void ∗elem)
  Affiche l'entier contenu dans un noeud.
- void freeInt (Noeud this)
  Libère l'espace mémoire alloué à l'entier stocké dans un noeud.

4.2.1 Description détaillée

Module pour la création et les opérations utilisables sur les arbres planaires qui contiennent des int.

Auteur

Delmas Rémi

4.2.2 Documentation des définitions de type

4.2.2.1 typedef struct arbrePlanaireInt∗ ArbrePlanaireInt

4.2.3 Documentation des fonctions

4.2.3.1 void affichagePrefixeInt (ArbrePlanaireInt *this*)

Affiche le parcours préfixe récursif de l'arbre.

4.2.3.2 void afficheInt (void ∗ *elem*)

Affiche l'entier contenu dans un noeud.

4.2.3.3 Noeud ajouterNoeudInt (ArbrePlanaireInt *this,* Noeud *pere,* int *elem*)

Ajoute un noeud pere contenant l'entier elem à un arbrePlanaireInt.

4.2.3.4 ArbrePlanaireInt creerArbrePlanaireInt ( )

Crée un arbrePlanaireInt vide.

4.2.3.5 bool estUneFeuille (Noeud *this*)

Retourne vrai si le noeud entré en paramètre est une feuille.

4.2.3.6 void freeArbrePlanaireInt (ArbrePlanaireInt *this*)

Libère l'espace mémoire alloué à un arbrePlanaireInt.

4.2.3.7 void freeInt (Noeud *this*)

Libère l'espace mémoire alloué à l'entier stocké dans un noeud.

4.2.3.8 int getInt (Noeud *this*)

renvoie la valeur entière stockée dans le noeud entré en paramètre.

4.2.3.9 void supprimerNoeudInt (ArbrePlanaireInt *a,* Noeud *this*)

Supprime un noeud d'un arbrePlanaireInt après avoir libéré la mémoire allouée à l'entier contenu dans le noeud.

Postcondition

le noeud supprimé est remplacé par son frere s'il était le premier fils de son père, sinon le frère de son frère devient le frère du sommet à détruire.

4.2.3.10 void tableauArbreInt (ArbrePlanaireInt *this,* int ∗ *tab,* int *taille*)

remplit le tableau tab avec les valeurs contenues dans l'arbrePlanaireInt en parcours préfixe.

Paramètres

tab	un tableau qui doit être déjà instancié.

4.3 Référence du fichier /net/cremi/twybrech/TSP-Project/include/Arete.h

Module pour la création d'arêtes et les opérations dessus.

Définitions de type

— typedef struct Arete ∗ Arete

Fonctions

— Arete creerArete (double cle, int depart, int arrive)
Crée une arête.
— int comparaisonArete (Arete a, Arete b)
Retourne -1 si la clé de l'arête à est infèrieure celle de l'arête b, 1 si elle lui est supérieure ou 0 si elles sont égales.
— int comparaisonAreteCle (Arete a, double cle)
Retourne -1 si la clé de l'arête à est supérieure à la clé entrée en paramètre, 1 si elle lui est supérieure ou 0 si elles sont égales.
— void freeArete (Arete a)
Libère l'espace mémoire alloué à l'arête entrée en paramètre.
— void afficheArete (Arete a)
Affiche les valeurs des données membres d'une arête sous la forme (Depart : villeDépart Arrive : villeArrivée Cle : valeurClé.
— double getCle (Arete a)
Retourne la clé de l'arête entrée en paramètre.
— int getDepart (Arete a)
Retourne le départ(une ville) de l'arête entrée en paramètre.
— int getArrive (Arete a)
Retourne l'arrivée(une ville) de l'arête entrée en paramètre.
— void setCle (double value, Arete a)
Donne une valeur (double) à une clé.
— void setDepart (int depart, Arete a)
Modifie la ville de départ de l'arête entrée en paramètre.
— void setArrive (int arrive, Arete a)
Modifie la ville d'arrivée de l'arête entrée en paramètre.

4.3.1 Description détaillée

Module pour la création d'arêtes et les opérations dessus.

Auteur

Delmas Rémi

4.3.2 Documentation des définitions de type

4.3.2.1 typedef struct Arete* Arete

4.3.3 Documentation des fonctions

4.3.3.1 void afficheArete (Arete *a*)

Affiche les valeurs des données membres d'une arête sous la forme (Depart : villeDépart Arrive : villeArrivée Cle : valeurClé.

Affiche les valeurs des données membres d'une arête sous la forme (Depart : villeDépart Arrive : villeArrivée Cle : valeurClé.

4.3.3.2 int comparaisonArete (Arete *a,* Arete *b*)

Retourne -1 si la clé de l'arête à est infèrieure celle de l'arête b, 1 si elle lui est supérieure ou 0 si elles sont égales.

Retourne -1 si la clé de l'arête à est infèrieure celle de l'arête b, 1 si elle lui est supérieure ou 0 si elles sont égales.

4.3.3.3 int comparaisonAreteCle (Arete *a,* double *cle*)

Retourne -1 si la clé de l'arête à est supérieure à la clé entrée en paramètre, 1 si elle lui est supérieure ou 0 si elles sont égales.

Retourne -1 si la clé de l'arête à est supérieure à la clé entrée en paramètre, 1 si elle lui est supérieure ou 0 si elles sont égales.

4.3.3.4 Arete creerArete (double *cle,* int *depart,* int *arrive*)

Crée une arête.

Paramètres

Un	double, et deux int représentant une ville de départ et une ville d'arrivée.

Crée une arête.

Paramètres

depart	Entier correspondant à la ville de départ.
arrive	Entier correspondant à la ville d'arrivée.

4.3.3.5 void freeArete (Arete *a*)

Libère l'espace mémoire alloué à l'arête entrée en paramètre.

Libère l'espace mémoire alloué à l'arête entrée en paramètre.

4.3.3.6 int getArrive (Arete *a*)

Retourne l'arrivée(une ville) de l'arête entrée en paramètre.

Retourne l'arrivée(une ville) de l'arête entrée en paramètre.

4.3.3.7 double getCle (Arete *a*)

Retourne la clé de l'arête entrée en paramètre.

Retourne la clé de l'arête entrée en paramètre.

4.3.3.8 int getDepart (Arete *a*)

Retourne le départ(une ville) de l'arête entrée en paramètre.

Retourne le départ(une ville) de l'arête entrée en paramètre.

4.3.3.9 void setArrive (int *arrive,* Arete *a*)

Modifie la ville d'arrivée de l'arête entrée en paramètre.

Modifie la ville d'arrivée de l'arête entrée en paramètre.

4.3.3.10 void setCle (double *value,* Arete *a*)

Donne une valeur (double) à une clé.

Donne une valeur (double) à une clé.

4.3.3.11 void setDepart (int *depart,* Arete *a*)

Modifie la ville de départ de l'arête entrée en paramètre.

Modifie la ville de départ de l'arête entrée en paramètre.

4.4 Référence du fichier /net/cremi/twybrech/TSP-Project/include/BruteForce.h

```
#include "Graphe.h"
#include <stdbool.h>
```

Fonctions

- double calculDistanceParcours (Graphe graph, int taille, int ∗parcours)
- int ∗ algorithmeBruteForce2 (Graphe graph)
- double parcoursSimple (Graphe graph)

4.4.1 Documentation des fonctions

4.4.1.1 int∗ algorithmeBruteForce2 (Graphe *graph*)

effectue l'algorithme force brute sur le graphe rentré en paramètre et renvoie le parcours optimal.

4.4.1.2 double calculDistanceParcours (Graphe *graph,* int *taille,* int ∗ *parcours*)

retourne la distance parcourue lors d'un parcours.

4.4.1.3 double parcoursSimple (Graphe *graph*)

retourne la distance parcourue lors du parcours [0,1,2,3, ... ,0]

4.5 Référence du fichier /net/cremi/twybrech/TSP-Project/include/BruteForceOpti.h

Module pour Branch and Bound.

```
#include <Graphe.h>
```

Fonctions

— int * BruteForceOpti (Graphe g, double *acc)
retourne un parcours de villes après avoir executé l'algorithme Branch and Bound sur un graphe.

4.5.1 Description détaillée

Module pour Branch and Bound.

Auteur

Delmas Rémi

4.5.2 Documentation des fonctions

4.5.2.1 int* BruteForceOpti (Graphe *g,* double * *acc*)

retourne un parcours de villes après avoir executé l'algorithme Branch and Bound sur un graphe.

Paramètres

g	graphe récupéré à partir d'un .tsp

Précondition

L'accumulateur entré en paramètre doit être initialisé à 0.

retourne un parcours de villes après avoir executé l'algorithme Branch and Bound sur un graphe.

4.6 Référence du fichier /net/cremi/twybrech/TSP-Project/include/FonctionTest.h

Module contenant les fonctions de test pur les executables des tests.

```
#include <stdbool.h>
#include <Graphe.h>
```

Définitions de type

— typedef int *(* HeuristiqueAvecDepart)(Graphe g, int depart, double *acc)
— typedef int *(* HeuristiqueSansDepart)(Graphe g, double *acc)

Fonctions

- void testBaseHeuristiqueAD (char ∗nomFichier, int depart, HeuristiqueAvecDepart H)
  affiche OK si le test d'un heuristique qui prend une ville de départ en paramètre renvoie une solution valide, ECHEC sinon.
- void testBaseHeuristiqueSD (char ∗nomFichier, HeuristiqueSansDepart H)
  affiche OK si le test d'un heuristique qui ne prend pas de ville de départ en paramètre renvoie une solution valide, ECHEC sinon.
- bool estDimensionValide (double dim, double meilleureDistance)
  Renvoie la validité de la dimension du parcours, si le parcours fait une distance inférieure au plus court chemin il est invalidé.
- bool estCycleValide (int ∗tabCycle, int taille)
  retourne la validité du parcours entré en paramètre.
- void freeHeurisque (Graphe g, int ∗tabCycle)
  Libère la mémoire allouée au Graphe g et celle allouée au tableau contenant le parcours obtenu suite à l'execution d'un heuristique.
- Graphe getGraphe (char ∗nomFichier)
  Retourne le Graphe contenu dans un fichier TSP.

4.6.1 Description détaillée

Module contenant les fonctions de test pur les executables des tests.

Auteur

Delmas Rémi

4.6.2 Documentation des définitions de type

4.6.2.1 typedef int∗(∗ HeuristiqueAvecDepart)(Graphe g, int depart, double ∗acc)

4.6.2.2 typedef int∗(∗ HeuristiqueSansDepart)(Graphe g, double ∗acc)

4.6.3 Documentation des fonctions

4.6.3.1 bool estCycleValide (int ∗ *tabCycle,* int *taille*)

retourne la validité du parcours entré en paramètre.

Paramètres

tabCycle	un tableau d'entiers représentant un parcours de villes.

retourne la validité du parcours entré en paramètre.

Paramètres

tabCyble	Tableau contenant des entiers correspondant aux villes.
taille	Entier correspondant à la taille du tableau tabCycle.

4.6.3.2 bool estDimensionValide (double *dim,* double *meilleureDistance*)

Renvoie la validité de la dimension du parcours, si le parcours fait une distance inférieure au plus court chemin il est invalidé.

Renvoie la validité de la dimension du parcours, si le parcours fait une distance inférieure au plus court chemin il est invalidé.

Paramètres

dim	Double contenant une distance.
meilleure-Distance	Double contenant la distance du chemin min. d'un graphe.

4.6.3.3 void freeHeurisque (Graphe *g*, int * *tabCycle*)

Libère la mémoire allouée au Graphe g et celle allouée au tableau contenant le parcours obtenu suite à l'execution d'un heuristique.

4.6.3.4 Graphe getGraphe (char * *nomFichier*)

Retourne le Graphe contenu dans un fichier TSP.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

4.6.3.5 void testBaseHeuristiqueAD (char * *nomFichier*, int *depart*, HeuristiqueAvecDepart *H*)

affiche OK si le test d'un heuristique qui prend une ville de départ en paramètre renvoie une solution valide, ECHEC sinon.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

4.6.3.6 void testBaseHeuristiqueSD (char * *nomFichier*, HeuristiqueSansDepart *H*)

affiche OK si le test d'un heuristique qui ne prend pas de ville de départ en paramètre renvoie une solution valide, ECHEC sinon.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

4.7 Référence du fichier /net/cremi/twybrech/TSP-Project/include/Graphe.h

Module de manipulation de la matrice de distance.

Définitions de type

— typedef struct graphe * Graphe

Fonctions

— Graphe cree_graphe (int len, double **mat)
Fonction de création d'une instance de Graphe.
— void free_graphe (Graphe graphe)
Fonction de destruction d'une instance Graphe.
— void afficher_graphe (Graphe g)
Affiche une contenu instance de type Graphe sous la forme : Affichage de la structure graphe : Taille : <len> Matrice : <numVille>- <distance00>- <distance01>- ... (deux décimale affichées uniquement.)

- double distance_ville (Graphe graphe, int s1, int s2)
  Retourne la distance entre deux villes d'indice s1 et s2 dans graphe. L'ordre n'a pas d'importance.
- int get_taille (Graphe graphe)
  Retourne le nombre de ville dans l'instance graphe.
- double get_double (Graphe g, int i, int j)
  Fonction retournant le double se situant à la position[i][j] .

4.7.1 Description détaillée

Module de manipulation de la matrice de distance.

Auteur

Delmas Rémi

4.7.2 Documentation des définitions de type

4.7.2.1 typedef struct graphe* Graphe

4.7.3 Documentation des fonctions

4.7.3.1 void afficher_graphe (Graphe *g*)

Affiche une contenu instance de type Graphe sous la forme : Affichage de la structure graphe : Taille : <len> Matrice : <numVille>- <distance00>- <distance01>- ... (deux décimale affichées uniquement.)

Paramètres

input	Instance de type Graphe à afficher.

Affiche une contenu instance de type Graphe sous la forme : Affichage de la structure graphe : Taille : <len> Matrice : <numVille>- <distance00>- <distance01>- ... (deux décimale affichées uniquement.)

4.7.3.2 Graphe cree_graphe (int *len*, double ** *mat*)

Fonction de création d'une instance de Graphe.

Paramètres

len	Dimension de la matrice de distance, strictement positive.
mat	Matrice 2D contenant les distances entre chaque villes.

Postcondition

L'instance doit etre free avec free_graphe

4.7.3.3 double distance_ville (Graphe *graphe*, int *s1*, int *s2*)

Retourne la distance entre deux villes d'indice s1 et s2 dans graphe. L'ordre n'a pas d'importance.

Retourne la distance entre deux villes d'indice s1 et s2 dans graphe. L'ordre n'a pas d'importance.

4.7.3.4 void free_graphe (Graphe *graphe*)

Fonction de destruction d'une instance Graphe.

Fonction de destruction d'une instance Graphe.

4.7.3.5 double get_double (Graphe *g*, int *i*, int *j*)

Fonction retournant le double se situant à la position[i][j] .

4.7.3.6 int get_taille (Graphe *graphe*)

Retourne le nombre de ville dans l'instance graphe.

Retourne le nombre de ville dans l'instance graphe.

4.8 Référence du fichier /net/cremi/twybrech/TSP-Project/include/Input.h

Module pour le parsing de fichier TSP.

Définitions de type

— typedef struct input * Input

Fonctions

— void print_input_data (Input input)
Affiche une instance de type Input.
— Input open_TSP_file (char *nom_file)
Fonction de parsing d'un fichier TSP.
— void free_input (Input input)
Fonction de destruction d'une instance Input.
— char * get_nom_file (Input input)
Retourne le nom du fichier .tsp qui a servi à générer l'instance d'Input passée en paramètre.
— char * get_nom (Input input)
Retourne le champs nom du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— char * get_type (Input input)
Retourne le champs type (ex : TSP) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— char * get_commentaire (Input input)
— int get_dimension (Input input)
Retourne la dimension (Pas de borne sup) de la matrice du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— char * get_edge_weight_type (Input input)
Retourne le champs edge_weight_type (ex : EXPLICIT) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— char * get_edge_weight_format (Input input)
— char * get_display_data_type (Input input)
Retourne le champs display_data_type (ex : TWOD_DISPLAY) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— double ** get_edge_weight_matrix (Input input)
Retourne la matrice de travail du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
— double ** get_display_data (Input input)
Retourne la matrice d'affichage du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.1 Description détaillée

Module pour le parsing de fichier TSP.

Auteur

Delmas Rémi

4.8.2 Documentation des définitions de type

4.8.2.1 typedef struct input* Input

4.8.3 Documentation des fonctions

4.8.3.1 void free_input (Input *input*)

Fonction de destruction d'une instance Input.

4.8.3.2 char* get_commentaire (Input *input*)

Retourne le champs commentaire du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

si ce champs est vide ou non présent, la valeur renvoyée est "NoComm".

4.8.3.3 int get_dimension (Input *input*)

Retourne la dimension (Pas de borne sup) de la matrice du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

la dimension doit etre strictement supérieure a 0.

4.8.3.4 double** get_display_data (Input *input*)

Retourne la matrice d'affichage du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.3.5 char* get_display_data_type (Input *input*)

Retourne le champs display_data_type (ex : TWOD_DISPLAY) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.3.6 char* get_edge_weight_format (Input *input*)

Retourne le champs edge_weight_format (ex : FULL_MATRIX) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.3.7 double** get_edge_weight_matrix (Input *input*)

Retourne la matrice de travail du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

les valeurs contenues dans la matrice doivent être strictements supérieures a 0.

4.8.3.8 char* get_edge_weight_type (Input *input*)

Retourne le champs edge_weight_type (ex : EXPLICIT) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.3.9 char* get_nom (Input *input*)

Retourne le champs nom du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

si ce champs est vide ou non présent, la valeur renvoyée est "NoName".

4.8.3.10 char* get_nom_file (Input *input*)

Retourne le nom du fichier .tsp qui a servi à générer l'instance d'Input passée en paramètre.

4.8.3.11 char* get_type (Input *input*)

Retourne le champs type (ex : TSP) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.8.3.12 Input open_TSP_file (char * *nom_file*)

Fonction de parsing d'un fichier TSP.

Paramètres

nom_file	chemin du fichier .tsp à parser.

Précondition

Présentement, il n'y a que les instances de type FULL_MATRIX qui sont gérées un message d'erreur est affiché et la mémoire est libéré en cas de format incorrect. La matrice de donnée doit etre symétrique et contenir des valeurs strictement positives

Postcondition

l'instance Input doit etre free via free_input.

4.8.3.13 void print_input_data (Input *input*)

Affiche une instance de type Input.

Paramètres

input	instance de type Input à afficher.

4.9 Référence du fichier /net/cremi/twybrech/TSP-Project/include/NearestNeighbour.h

```
#include <stdbool.h>
#include "Graphe.h"
```

Fonctions

- int plusProcheVoisin (int sommet, bool *tab_dispo, Graphe graph, double *acc)
  Fonction permettant de retourner le voisin ayant la distance la plus proche de celui où l'on est.
- int * HeuristiquePlusProcheVoisin (Graphe graph, double *distanceAcc, int departChemin)
  Fonction permettant de retourner un tableau de voisin ayant la distance la plus proche avec son précédent.

4.9.1 Documentation des fonctions

4.9.1.1 int* HeuristiquePlusProcheVoisin (Graphe *graph*, double * *distanceAcc*, int *departChemin*)

Fonction permettant de retourner un tableau de voisin ayant la distance la plus proche avec son précédent.

Retourne un tableau désignant le chemin le plus court optenu par l'algorithme du plus proche voisin.

Paramètres

graphe	structure graphe(matrice2D double + taille matrice) contenant la matrice de l'instance TSP associée et sa dimension.
distanceAcc	/ !\ la valeur pointée avant appel de HeuristiquePlusProcheVoisin est écrassée, on affecte la distance du chemin via effet de bord au double pointé.
depart	entier en 0 et nombre_de_ville désignant la ville de depart du cycle.

Postcondition

le tableau doit etre free après traitement par le client.

Paramètres

distanceAcc	Tableau de booléens contenant les distances entre les villes.
departChemin	Entier correspondant à la ville de départ.
graph	Représente la matrice TSP.

4.9.1.2 int plusProcheVoisin (int *sommet*, bool * *tabDispo*, Graphe *graph*, double * *acc*)

Fonction permettant de retourner le voisin ayant la distance la plus proche de celui où l'on est.

Paramètres

sommet	Entier correspondant au numéro du sommet actuel dans le graphe.
tabDispo	Tableau de booléan servant à savoir la liste des villes qui n'ont pas été visitées.
graph	Représente la matrice TSP.
acc	Tableau de double contenant les distance des villes.

4.10 Référence du fichier /net/cremi/twybrech/TSP-Project/include/Prim.h

```
#include <Graphe.h>
```

Fonctions

— int * Prim (Graphe g, int depart, double *acc)
Algorithme de Prim servant à créer un arbre de poids minimum.

4.10.1 Documentation des fonctions

4.10.1.1 int* Prim (Graphe *g*, int *depart*, double * *acc*)

Algorithme de Prim servant à créer un arbre de poids minimum.

Retourne un tableau désignant le chemin le plus court optenu par l'algorithme de Prim (contruction d'un arbre de poid couvrant minimal).

Paramètres

g	graphe (matrice2D double + taille matrice) contenant la matrice de l'instance TSP associée et sa dimension.
depart	entier en 0 et nombre_de_ville désignant la ville de depart du cycle.
acc	/ !\ la valeur pointée avant appel de Prim est écrassée, on affecte la distance du chemin via effet de bord au double pointé.

Postcondition

le tableau doit etre free après traitement par le client.

Paramètres

g	Graphe représentant la matrice.
depart	Entier représentant la ville de départ du graphe g.
acc	Tableau de booléens contenant la distance des villes.

4.11 Référence du fichier /net/cremi/twybrech/TSP-Project/include/TasArete.h

```
#include <stdbool.h>
#include "Arete.h"
```

Définitions de type

— typedef struct TasArete ∗ TasMinArete
— typedef struct AreteHandle ∗ AreteHandle

Fonctions

— TasMinArete creerTasMinArete (int taille)
Crée un tas min contenant des arêtes.
— void freeTasArete (TasMinArete tasMin)
Libère l'espace mémoire alloué à un tas min contenant des arêtes.
— void freeAreteHandle (AreteHandle a)
Libère l'espace mémoire alloué à une areteHandle contenant des elemHandle (une structure contenant un pointeur générique et un int).
— AreteHandle ajouterAreteHandle (TasMinArete tasMin, Arete a)
Ajoute une areteHandle à un tas min d'arêtes, la fonction l'ajoute d'abord en tant que feuille puis la positionne à un endroit \ qui fait en sorte que le tas reste un tas min.
— bool estVide (TasMinArete tasMin)
Vérifie si un tas min d'arêtes est vide (le champ taille de la structure est à -1 dans le cas où le tas est vide).
— Arete extraireAreteMin (TasMinArete tasMin)
Retourne l'arete minimale contenue dans tasMIn.
— void affichageTasArete (TasMinArete tasMin)
Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des arêtes contenues dans tasMin.
— int indiceAreteHandle (AreteHandle areteH)
Retourne l'indice de la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.
— Arete getArete (AreteHandle areteH)
Retourne la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.
— void diminuerCleArete (TasMinArete tas_a, AreteHandle areteH, double newCle)
Cette fonction donne à la clé de l'arete de areteH la valeur de newCle dans le cas ou newCle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

4.11.1 Documentation des définitions de type

4.11.1.1 typedef struct AreteHandle∗ AreteHandle

4.11.1.2 typedef struct TasArete* TasMinArete

4.11.2 Documentation des fonctions

4.11.2.1 void affichageTasArete (TasMinArete *tasMin*)

Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des arêtes contenues dans tasMin.

4.11.2.2 AreteHandle ajouterAreteHandle (TasMinArete *tasMin,* Arete *a*)

Ajoute une areteHandle à un tas min d'arêtes, la fonction l'ajoute d'abord en tant que feuille puis la positionne à un endroit \ qui fait en sorte que le tas reste un tas min.

Postcondition

Il faut que le tas ne soit pas plein, la fonction le vérifie.

4.11.2.3 TasMinArete creerTasMinArete (int *taille*)

Crée un tas min contenant des arêtes.

4.11.2.4 void diminuerCleArete (TasMinArete *tas_a,* AreteHandle *areteH,* double *newCle*)

Cette fonction donne à la clé de l'arete de areteH la valeur de newCle dans le cas ou newCle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

Précondition

la fonction s'arrête si la clé entrée en paramètre n'est pas infèrieure à la clé contenue dans la donnée membre \ arete de areteH.

Postcondition

les deux sommets reliés par l'arete de areteH sont échangés si c'est nécessaire pour que le tas reste min.

4.11.2.5 bool estVide (TasMinArete *tasMin*)

Vérifie si un tas min d'arêtes est vide (le champ taille de la structure est à -1 dans le cas où le tas est vide).

4.11.2.6 Arete extraireAreteMin (TasMinArete *tasMin*)

Retourne l'arete minimale contenue dans tasMIn.

4.11.2.7 void freeAreteHandle (AreteHandle *a*)

Libère l'espace mémoire alloué à une areteHandle contenant des elemHandle (une structure contenant un pointeur générique et un int).

4.11.2.8 void freeTasArete (TasMinArete *tasMin*)

Libère l'espace mémoire alloué à un tas min contenant des arêtes.

4.11.2.9 Arete getArete (AreteHandle *areteH*)

Retourne la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.

4.11.2.10 int indiceAreteHandle (AreteHandle *areteH*)

Retourne l'indice de la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.

4.12 Référence du fichier /net/cremi/twybrech/TSP-Project/include/TasGenerique.h

Module de manipulation de la matrice de distance.

```
#include <stdbool.h>
```

Définitions de type

- typedef struct TasMinGen ∗ TasMinGen
- typedef struct elemHandle ∗ ElemHandle
- typedef int(∗ ptr_compar)(void ∗, void ∗)
- typedef void(∗ ptr_affichage)(void ∗)
- typedef void(∗ ptr_maj)(void ∗element, void ∗nouvelle_cle)

Fonctions

- int getTailleTas (TasMinGen tas)
- void setIndice (int indice, ElemHandle element)
- int getIndice (ElemHandle element)
- void ∗ getElem (ElemHandle e)
- void setElem (void ∗obj, ElemHandle element)
- ElemHandle creerElemHandle (void ∗element, int indice)
- void freeElemHandle (ElemHandle elem)
- TasMinGen creerTasMinGen (int taille, ptr_compar cmp, ptr_compar cmpCle, ptr_affichage affichage, ptr_maj majCle)
- void freeTasGen (TasMinGen tas)
- bool estvide (TasMinGen tas)
- ElemHandle ajouterSommet (TasMinGen tas, void ∗element)
- void entasserTas (TasMinGen tas, int indice)
  Réorganise le tas en tas min à partir du sommet de nom indice entré en paramètre lorsque c'est nécessaire pour \ préserver l'accés aux données.
- void ∗ extraireMin (TasMinGen tas)
- void diminuerCle (TasMinGen tas, ElemHandle element, void ∗cle)
  Cette fonction donne à la clé de elem de element la valeur de cle(entrée en paramètre) dans le cas ou cle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.
- void affichageTas (TasMinGen tas)
  Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des elements contenus dans tas.
- ElemHandle sommet (TasMinGen tas, int indice)
  Retourne l'ElemHandle situé à l'indice entré en paramètre de la donnée membre sommets[] de tas.

4.12.1 Description détaillée

Module de manipulation de la matrice de distance.

Auteur

Delmas Rémi

4.12.2 Documentation des définitions de type

4.12.2.1 typedef struct elemHandle* ElemHandle

4.12.2.2 typedef void(* ptr_affichage)(void *)

4.12.2.3 typedef int(* ptr_compar)(void *, void *)

4.12.2.4 typedef void(* ptr_maj)(void *element, void *nouvelle_cle)

4.12.2.5 typedef struct TasMinGen* TasMinGen

4.12.3 Documentation des fonctions

4.12.3.1 void affichageTas (TasMinGen *tas*)

Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des elements contenus dans tas.

4.12.3.2 ElemHandle ajouterSommet (TasMinGen *tas,* void * *element*)

ajoute un element à un tas min générique en vérifiant qu'il reste min.

Précondition

le tas min générique en parmètre ne doit pas être plein.

4.12.3.3 ElemHandle creerElemHandle (void * *element,* int *indice*)

Crée un ElemHandle et le renvoie.

4.12.3.4 TasMinGen creerTasMinGen (int *taille,* ptr_compar *cmp,* ptr_compar *cmpCle,* ptr_affichage *affichage,* ptr_maj *majCle*)

Crée un tas min générique vide.

Paramètres

4.12.3.5 void diminuerCle (TasMinGen *tas,* ElemHandle *element,* void * *cle*)

Cette fonction donne à la clé de elem de element la valeur de cle(entrée en paramètre) dans le cas ou cle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

Précondition

la fonction s'arrête si la clé entrée en paramètre n'est pas infèrieure à la clé contenue dans la donnée membre \ elem de element.

Postcondition

les deux sommets reliés par l'arete contenue dans element sont échangés si c'est nécessaire pour que le tas reste min.

4.12.3.6 void entasserTas (TasMinGen *tas,* int *indice*)

Réorganise le tas en tas min à partir du sommet de nom indice entré en paramètre lorsque c'est nécessaire pour \ préserver l'accés aux données.

4.12.3.7 bool estvide (TasMinGen *tas*)

Teste si un tas min générique est vide.

4.12.3.8 void* extraireMin (TasMinGen *tas*)

Retourne le plus petit élément du tas min générique.

Postcondition

Supprime l'élément qui est retourné et réorganise le tas en tas min.

4.12.3.9 void freeElemHandle (ElemHandle *elem*)

Libère la mémoire allouée à un ElemHandle.

4.12.3.10 void freeTasGen (TasMinGen *tas*)

Libère l'espace mémoire alloué à un tas min générique.

4.12.3.11 void* getElem (ElemHandle *e*)

Retourne l'élément contenu dans la structure ElemHandle passé en paramètre.

Précondition

element objet du tas dont on veut l'element.

4.12.3.12 int getIndice (ElemHandle *element*)

Retourne l'indice de l'ElemHandle passé en paramètre.

4.12.3.13 int getTailleTas (TasMinGen *tas*)

Retourne le nombre d'élément contenu dans le tas min

4.12.3.14 void setElem (void * *obj,* ElemHandle *element*)

Affete un ElemHandle à un pointeur générique.

4.12.3.15 void setIndice (int *indice,* ElemHandle *element*)

Affecte un indice a un element du tas.

Paramètres

indice	entier a affecté a l'ElemHandle correspondant a sa position dans la structure interne du tas
element	cible de la modification d'indice.

Précondition

l'indice doit etre supérieur ou égal a 0.

4.12.3.16 ElemHandle sommet (TasMinGen *tas,* int *indice*)

Retourne l'ElemHandle situé à l'indice entré en paramètre de la donnée membre sommets[] de tas.

4.13 Référence du fichier /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/ArbrePlanaireGenerique.c

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <ArbrePlanaireGenerique.h>
```

Structures de données

- struct noeud
- struct arbrePlanaireGen

Fonctions

- Noeud creerNoeud (void ∗element, Noeud parent, Noeud aine, Noeud cadet)
  Crée un noeud.
- void freeNoeud (Noeud this)
  Libère l'espace mémoire alloué à un noeud.
- ArbrePlanaireGen creerArbrePlanaireGen (ptr_affichage f)
  Crée un arbre planaire générique vide.
- void freeArbrePlanaireGen (ArbrePlanaireGen this)
  Libère l'espace mémoire alloué à un arbre planaire générique après avoir détruit l'arborescence de la racine.
- void ∗ getElement (Noeud this)
  Retourne l'élément stocké dans un noeud.
- Noeud getPremierFils (Noeud this)
  Retourne le premier fils d'un noeud.
- Noeud getPere (Noeud this)
  Retourne le père d'un noeud.
- Noeud getFrere (Noeud this)
  Retourne le frère d'un noeud.
- Noeud getRacine (ArbrePlanaireGen this)
  Retourne la racine d'un arbre planaire générique.
- void affichagePrefixe (ArbrePlanaireGen this)
  Affiche le parcours préfixe récursif de l'arbre.
- Noeud ajouterFils (ArbrePlanaireGen a, Noeud pere, void ∗elem)
  Ajoute un fils à un noeud en lui donnant l'élément elem.
- bool estFeuille (Noeud this)
  Retourne faux si le noeud entré en paramètre à un fils, vrai s'il n'en a pas.
- void supprimerNoeud (ArbrePlanaireGen a, Noeud this)
  Supprime un noeud d'un arbre planaire après avoir libéré l'espace mémoire à l'entier contenu dans le noeud.

4.13.1 Documentation des fonctions

4.13.1.1 void affichagePrefixe (ArbrePlanaireGen *this*)

Affiche le parcours préfixe récursif de l'arbre.

4.13.1.2 Noeud ajouterFils (ArbrePlanaireGen *a,* Noeud *pere,* void ∗ *elem*)

Ajoute un fils à un noeud en lui donnant l'élément elem.

Précondition

Il faut mettre le père à NULL pour l'ajout en tant que racine.

4.13.1.3 ArbrePlanaireGen creerArbrePlanaireGen (ptr_affichage *f*)

Crée un arbre planaire générique vide.

Paramètres

Prend	un ptr_affichage qui est un pointeur générique en paramètre.

4.13.1.4 Noeud creerNoeud (void * *element,* Noeud *parent,* Noeud *aine,* Noeud *cadet*)

Crée un noeud.

Paramètres

Un	pointeur générique et trois noeuds pour déterminer la place du noeud crée par rapport à ceux déjà existant.

4.13.1.5 bool estFeuille (Noeud *this*)

Retourne faux si le noeud entré en paramètre à un fils, vrai s'il n'en a pas.

4.13.1.6 void freeArbrePlanaireGen (ArbrePlanaireGen *this*)

Libère l'espace mémoire alloué à un arbre planaire générique après avoir détruit l'arborescence de la racine.

4.13.1.7 void freeNoeud (Noeud *this*)

Libère l'espace mémoire alloué à un noeud.

4.13.1.8 void* getElement (Noeud *this*)

Retourne l'élément stocké dans un noeud.

4.13.1.9 Noeud getFrere (Noeud *this*)

Retourne le frère d'un noeud.

4.13.1.10 Noeud getPere (Noeud *this*)

Retourne le père d'un noeud.

4.13.1.11 Noeud getPremierFils (Noeud *this*)

Retourne le premier fils d'un noeud.

4.13.1.12 Noeud getRacine (ArbrePlanaireGen *this*)

Retourne la racine d'un arbre planaire générique.

4.13.1.13 void supprimerNoeud (ArbrePlanaireGen *a*, Noeud *this*)

Supprime un noeud d'un arbre planaire après avoir libéré l'espace mémoire à l'entier contenu dans le noeud.

Postcondition

le noeud supprimé est remplacé par son frere s'il était le premier fils de son père, sinon le frère de son frère devient le frère du sommet à détruire.

4.14 Référence du fichier /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/ArbrePlanaireInt.c

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <ArbrePlanaireInt.h>
```

Structures de données

— struct arbrePlanaireInt

Fonctions

— ArbrePlanaireInt creerArbrePlanaireInt ()
Crée un arbrePlanaireInt vide.
— void freeArbrePlanaireInt (ArbrePlanaireInt this)
Libère l'espace mémoire alloué à un arbrePlanaireInt.
— void freeInt (Noeud this)
Libère l'espace mémoire alloué à l'entier stocké dans un noeud.
— int getInt (Noeud this)
renvoie la valeur entière stockée dans le noeud entré en paramètre.
— void affichagePrefixeInt (ArbrePlanaireInt this)
Affiche le parcours préfixe récursif de l'arbre.
— Noeud ajouterNoeudInt (ArbrePlanaireInt this, Noeud pere, int elem)
Ajoute un noeud pere contenant l'entier elem à un arbrePlanaireInt.
— bool estUneFeuille (Noeud this)
Retourne vrai si le noeud entré en paramètre est une feuille.
— void supprimerNoeudInt (ArbrePlanaireInt a, Noeud this)
Supprime un noeud d'un arbrePlanaireInt après avoir libéré la mémoire allouée à l'entier contenu dans le noeud.
— void afficheInt (void ∗elem)
Affiche l'entier contenu dans un noeud.
— void tableauArbreInt (ArbrePlanaireInt this, int ∗tab, int taille)
remplit le tableau tab avec les valeurs contenues dans l'arbrePlanaireInt en parcours préfixe.

4.14.1 Documentation des fonctions

4.14.1.1 void affichagePrefixeInt (ArbrePlanaireInt *this*)

Affiche le parcours préfixe récursif de l'arbre.

4.14.1.2 void afficheInt (void ∗ *elem*)

Affiche l'entier contenu dans un noeud.

4.14.1.3 Noeud ajouterNoeudInt (ArbrePlanaireInt *this*, Noeud *pere*, int *elem*)

Ajoute un noeud pere contenant l'entier elem à un arbrePlanaireInt.

4.14.1.4 ArbrePlanaireInt creerArbrePlanaireInt ( )

Crée un arbrePlanaireInt vide.

4.14.1.5 bool estUneFeuille (Noeud *this*)

Retourne vrai si le noeud entré en paramètre est une feuille.

4.14.1.6 void freeArbrePlanaireInt (ArbrePlanaireInt *this*)

Libère l'espace mémoire alloué à un arbrePlanaireInt.

4.14.1.7 void freeInt (Noeud *this*)

Libère l'espace mémoire alloué à l'entier stocké dans un noeud.

4.14.1.8 int getInt (Noeud *this*)

renvoie la valeur entière stockée dans le noeud entré en paramètre.

4.14.1.9 void supprimerNoeudInt (ArbrePlanaireInt *a*, Noeud *this*)

Supprime un noeud d'un arbrePlanaireInt après avoir libéré la mémoire allouée à l'entier contenu dans le noeud.

Postcondition

le noeud supprimé est remplacé par son frere s'il était le premier fils de son père, sinon le frère de son frère devient le frère du sommet à détruire.

4.14.1.10 void tableauArbreInt (ArbrePlanaireInt *this*, int * *tab*, int *taille*)

remplit le tableau tab avec les valeurs contenues dans l'arbrePlanaireInt en parcours préfixe.

Paramètres

tab	un tableau qui doit être déjà instancié.

4.15 Référence du fichier /net/cremi/twybrech/TSP-Project/src/ArbrePlanaire/TestArbre.c

```
#include <ArbrePlanaireGenerique.h>
#include <ArbrePlanaireInt.h>
#include <stdlib.h>
#include <stdio.h>
```

Fonctions

— int main ()

4.15.1 Documentation des fonctions

4.15.1.1 int main ( )

4.16 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Arete/Arete.c

Programme générant des arêtes.

```
#include <stdlib.h>
#include <stdio.h>
#include <Arete.h>
```

Structures de données

- struct Arete
  Arête entre la ville depart et la ville arrive.

Fonctions

- Arete creerArete (double cle, int depart, int arrive)
  Fonction créant une arête entre la ville de départ et la ville d'arrivée associée à une clé.
- int comparaisonArete (Arete a, Arete b)
  Fonction comparant deux arêtes en vérifiant si elles ont la même clé.
- int comparaisonAreteCle (Arete a, double cle)
  Fonction comparant une arête et une clé.
- void freeArete (Arete a)
  Fonction libérant l'arête passée en paramètre.
- void afficheArete (Arete a)
  Fonction affichant l'arête passée en paramètre.
- double getCle (Arete a)
  Fonction retournant la clé de l'arête passée en paramètre.
- int getDepart (Arete a)
  Fonction retournant la ville de départ de l'arête passée en paramètre.
- int getArrive (Arete a)
  Fonction retournant la ville d'arrivée de l'arête passée en paramètre.
- void setCle (double value, Arete a)
  Fonction donnant une clé à l'arête passée en paramètre.
- void setDepart (int depart, Arete a)
  Fonction donnant une ville de départ à l'arête passée en paramètre.
- void setArrive (int arrive, Arete a)
  Fonction donnant une ville d'arrivée à l'arête passée en paramètre.

4.16.1 Description détaillée

Programme générant des arêtes.

4.16.2 Documentation des fonctions

4.16.2.1 void afficheArete (Arete *a*)

Fonction affichant l'arête passée en paramètre.

Affiche les valeurs des données membres d'une arête sous la forme (Depart : villeDépart Arrive : villeArrivée Cle : valeurClé.

4.16.2.2 int comparaisonArete (Arete *a*, Arete *b*)

Fonction comparant deux arêtes en vérifiant si elles ont la même clé.

Retourne -1 si la clé de l'arête à est infèrieure celle de l'arête b, 1 si elle lui est supérieure ou 0 si elles sont égales.

4.16.2.3 int comparaisonAreteCle (Arete *a*, double *cle*)

Fonction comparant une arête et une clé.

Retourne -1 si la clé de l'arête à est supérieure à la clé entrée en paramètre, 1 si elle lui est supérieure ou 0 si elles sont égales.

4.16.2.4 Arete creerArete (double *cle*, int *depart*, int *arrive*)

Fonction créant une arête entre la ville de départ et la ville d'arrivée associée à une clé.

Crée une arête.

Paramètres

depart	Entier correspondant à la ville de départ.
arrive	Entier correspondant à la ville d'arrivée.

4.16.2.5 void freeArete (Arete *a*)

Fonction libérant l'arête passée en paramètre.

Libère l'espace mémoire alloué à l'arête entrée en paramètre.

4.16.2.6 int getArrive (Arete *a*)

Fonction retournant la ville d'arrivée de l'arête passée en paramètre.

Retourne l'arrivée(une ville) de l'arête entrée en paramètre.

4.16.2.7 double getCle (Arete *a*)

Fonction retournant la clé de l'arête passée en paramètre.

Retourne la clé de l'arête entrée en paramètre.

4.16.2.8 int getDepart (Arete *a*)

Fonction retournant la ville de départ de l'arête passée en paramètre.

Retourne le départ(une ville) de l'arête entrée en paramètre.

4.16.2.9 void setArrive (int *arrive*, Arete *a*)

Fonction donnant une ville d'arrivée à l'arête passée en paramètre.

Modifie la ville d'arrivée de l'arête entrée en paramètre.

4.16.2.10 void setCle (double *value*, Arete *a*)

Fonction donnant une clé à l'arête passée en paramètre.

Donne une valeur (double) à une clé.

4.16.2.11 void setDepart (int *depart*, Arete *a*)

Fonction donnant une ville de départ à l'arête passée en paramètre.

Modifie la ville de départ de l'arête entrée en paramètre.

4.17 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Arete/TestArete.c

Programme de tests.

```
#include <stdlib.h>
#include <stdio.h>
#include <Arete.h>
```

Fonctions

— int main ()
Fonction permettant d'effectuer des test pour vérifier les fonctions du fichier Arete.c.

4.17.1 Description détaillée

Programme de tests.

4.17.2 Documentation des fonctions

4.17.2.1 int main ( )

Fonction permettant d'effectuer des test pour vérifier les fonctions du fichier Arete.c.

4.18 Référence du fichier /net/cremi/twybrech/TSP-Project/src/BruteForce/BruteForce2.c

```
#include "BruteForce.h"
#include "Graphe.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdbool.h>
#include "ArbrePlanaireInt.h"
#include "ArbrePlanaireGenerique.h"
```

Fonctions

— void BruteForce2 (Graphe graph, int nbVilles, int ∗parcours, bool ∗libre, double distanceParcours, int ∗parcoursFinal)

- bool * creer_tab_dispo (int nombreSommets)
- int sommet_suivant (bool *tab_dispo, Graphe graph)
- double parcoursSimple (Graphe graph)
- double calculDistanceParcours (Graphe graph, int taille, int *parcours)
- int * algorithmeBruteForce2 (Graphe graph)

4.18.1 Documentation des fonctions

4.18.1.1 int* algorithmeBruteForce2 (Graphe *graph*)

effectue l'algorithme force brute sur le graphe rentré en paramètre et renvoie le parcours optimal.

4.18.1.2 void BruteForce2 (Graphe *graph*, int *nbVilles*, int * *parcours*, bool * *libre*, double *distanceParcours*, int * *parcoursFinal*)

4.18.1.3 double calculDistanceParcours (Graphe *graph*, int *taille*, int * *parcours*)

retourne la distance parcourue lors d'un parcours.

4.18.1.4 bool * creer_tab_dispo (int *nombreSommets*)

4.18.1.5 double parcoursSimple (Graphe *graph*)

retourne la distance parcourue lors du parcours [0,1,2,3, ... ,0]

4.18.1.6 int sommet_suivant (bool * *tab_dispo*, Graphe *graph*)

4.19 Référence du fichier /net/cremi/twybrech/TSP-Project/src/BruteForce/testBruteForce.c

```
#include "BruteForce.h"
#include "Graphe.h"
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>
#include <assert.h>
#include "Input.h"
```

Fonctions

- void afficher (int *t, int taille)
- int Unicite (int *t, Graphe graph)
- int main ()

4.19.1 Documentation des fonctions

4.19.1.1 void afficher (int * *t*, int *taille*)

4.19.1.2 int main ( )

4.19.1.3 int Unicite (int * *t,* Graphe *graph*)

4.20 Référence du fichier /net/cremi/twybrech/TSP-Project/src/BruteForceOpti/BruteForceOpti.c

Programme mettant en place un heuristique de Bruteforce optimisé.

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <assert.h>
#include <stdbool.h>
#include <Graphe.h>
#include <NearestNeighbour.h>
#include <Input.h>
#include <BruteForceOpti.h>
```

Fonctions

— int * BruteForceOpti (Graphe g, double *acc)
Fonction appliquant le brute force optimisé. Cette fonction appelle la fonction BruteForceOptimise.qui va chercher tous les chemins possible en ne testant que les chemins aillant la distance parcourue la plus courte et ne poursuit pas un chemin si sa distance est supérieur ou égale à la distance passé en paramètre.

4.20.1 Description détaillée

Programme mettant en place un heuristique de Bruteforce optimisé.

4.20.2 Documentation des fonctions

4.20.2.1 int* BruteForceOpti (Graphe *g,* double * *acc*)

Fonction appliquant le brute force optimisé. Cette fonction appelle la fonction BruteForceOptimise.qui va chercher tous les chemins possible en ne testant que les chemins aillant la distance parcourue la plus courte et ne poursuit pas un chemin si sa distance est supérieur ou égale à la distance passé en paramètre.

retourne un parcours de villes après avoir executé l'algorithme Branch and Bound sur un graphe.

4.21 Référence du fichier /net/cremi/twybrech/TSP-Project/src/BruteForceOpti/TestBrute1.c

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include <stdbool.h>
#include <Graphe.h>
#include <NearestNeighbour.h>
#include <Input.h>
#include <BruteForceOpti.h>
#include <FonctionTest.h>
```

Fonctions

— int main ()
Fonction permettant de tester des fonctions implémenté dans le fichier BruteForceOpti.c.

4.21.1 Documentation des fonctions

4.21.1.1 int main ( )

Fonction permettant de tester des fonctions implémenté dans le fichier BruteForceOpti.c.

4.22 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Graphe/Graphe.c

Programme mettant en place un graphe correspondant à une matrice.

```
#include <Graphe.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <assert.h>
```

Structures de données

— struct graphe
Crée un graphe aillant pour longueur taille et en paramètre des doubles contenus dans matrice.

Fonctions

— Graphe cree_graphe (int len, double ∗∗mat)
Fonction de création d'une instance de Graphe.
— void free_graphe (Graphe graphe)
Fonction permettant la suppression d'un graphe.
— void afficher_graphe (Graphe g)
Fonction permettant d'afficher le graphe passé en paramètre.
— double get_double (Graphe g, int i, int j)
Fonction retournant le double se situant à la position[i][j] .
— double distance_ville (Graphe graphe, int s1, int s2)
Fonction retournant la distance se situant à la position[s1][s2].
— int get_taille (Graphe graphe)
Fonction retournant la taille du graphe passé en paramètre.

4.22.1 Description détaillée

Programme mettant en place un graphe correspondant à une matrice.

4.22.2 Documentation des fonctions

4.22.2.1 void afficher_graphe (Graphe *g*)

Fonction permettant d'afficher le graphe passé en paramètre.

Affiche une contenu instance de type Graphe sous la forme : Affichage de la structure graphe : Taille : <len> Matrice : <numVille>- <distance00>- <distance01>- ... (deux décimale affichées uniquement.)

4.22.2.2 Graphe cree_graphe (int *len*, double ** *mat*)

Fonction de création d'une instance de Graphe.

Paramètres

len	Dimension de la matrice de distance, strictement positive.
mat	Matrice 2D contenant les distances entre chaque villes.

Postcondition

L'instance doit etre free avec free_graphe

4.22.2.3 double distance_ville (Graphe *graphe*, int *s1*, int *s2*)

Fonction retournant la distance se situant à la position[s1][s2].

Retourne la distance entre deux villes d'indice s1 et s2 dans graphe. L'ordre n'a pas d'importance.

4.22.2.4 void free_graphe (Graphe *graphe*)

Fonction permettant la suppression d'un graphe.

Fonction de destruction d'une instance Graphe.

4.22.2.5 double get_double (Graphe *g*, int *i*, int *j*)

Fonction retournant le double se situant à la position[i][j] .

4.22.2.6 int get_taille (Graphe *graphe*)

Fonction retournant la taille du graphe passé en paramètre.

Retourne le nombre de ville dans l'instance graphe.

4.23 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Input/Input.c

```
#include <Input.h>
#include <malloc.h>
#include <error.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
```

Structures de données

— struct input

Macros

— #define _GNU_SOURCE

Fonctions

- char * alloc_chaine (ssize_t taille_ligne, int taille_pattern, char *ligne_lue)
- char * get_nom_file (Input input)
  Retourne le nom du fichier .tsp qui a servi à générer l'instance d'Input passée en paramètre.
- char * get_nom (Input input)
  Retourne le champs nom du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- char * get_type (Input input)
  Retourne le champs type (ex : TSP) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- char * get_commentaire (Input input)
- int get_dimension (Input input)
  Retourne la dimension (Pas de borne sup) de la matrice du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- char * get_edge_weight_type (Input input)
  Retourne le champs edge_weight_type (ex : EXPLICIT) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- char * get_edge_weight_format (Input input)
- char * get_display_data_type (Input input)
  Retourne le champs display_data_type (ex : TWOD_DISPLAY) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- double ** get_edge_weight_matrix (Input input)
  Retourne la matrice de travail du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- double ** get_display_data (Input input)
  Retourne la matrice d'affichage du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.
- void print_input_data (Input input)
  Affiche une instance de type Input.
- Input open_TSP_file (char *nom_file)
  Fonction de parsing d'un fichier TSP.
- void free_input (Input input)
  Fonction de destruction d'une instance Input.

4.23.1 Documentation des macros

4.23.1.1 #define _GNU_SOURCE

4.23.2 Documentation des fonctions

4.23.2.1 char * alloc_chaine (ssize_t *taille_ligne*, int *taille_pattern*, char * *ligne_lue*)

4.23.2.2 void free_input (Input *input*)

Fonction de destruction d'une instance Input.

4.23.2.3 char* get_commentaire (Input *input*)

Retourne le champs commentaire du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

si ce champs est vide ou non présent, la valeur renvoyée est "NoComm".

4.23.2.4 int get_dimension (Input *input*)

Retourne la dimension (Pas de borne sup) de la matrice du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

la dimension doit etre strictement supérieure a 0.

4.23.2.5 double** get_display_data (Input *input*)

Retourne la matrice d'affichage du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.23.2.6 char* get_display_data_type (Input *input*)

Retourne le champs display_data_type (ex : TWOD_DISPLAY) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.23.2.7 char* get_edge_weight_format (Input *input*)

Retourne le champs edge_weight_format (ex : FULL_MATRIX) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.23.2.8 double** get_edge_weight_matrix (Input *input*)

Retourne la matrice de travail du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

les valeurs contenues dans la matrice doivent être strictements supérieures a 0.

4.23.2.9 char* get_edge_weight_type (Input *input*)

Retourne le champs edge_weight_type (ex : EXPLICIT) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.23.2.10 char* get_nom (Input *input*)

Retourne le champs nom du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

Précondition

si ce champs est vide ou non présent, la valeur renvoyée est "NoName".

4.23.2.11 char* get_nom_file (Input *input*)

Retourne le nom du fichier .tsp qui a servi à générer l'instance d'Input passée en paramètre.

4.23.2.12 char* get_type (Input *input*)

Retourne le champs type (ex : TSP) du fichier .tsp qui a servi à générer l'instance d'Input en paramètre.

4.23.2.13 Input open_TSP_file (char * *nom_file*)

Fonction de parsing d'un fichier TSP.

Paramètres

nom_file	chemin du fichier .tsp à parser.

Précondition

Présentement, il n'y a que les instances de type FULL_MATRIX qui sont gérées un message d'erreur est affiché et la mémoire est libéré en cas de format incorrect. La matrice de donnée doit etre symétrique et contenir des valeurs strictement positives

Postcondition

l'instance Input doit etre free via free_input.

4.23.2.14 void print_input_data (Input *input*)

Affiche une instance de type Input.

Paramètres

input	instance de type Input à afficher.

4.24 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Input/TestInput1.c

```
#include <stdlib.h>
#include <stdio.h>
#include <Graphe.h>
#include <Input.h>
```

Fonctions

— int main (int argc, char ∗argv[])

4.24.1 Documentation des fonctions

4.24.1.1 int main (int *argc,* char ∗ *argv[]*)

4.25 Référence du fichier /net/cremi/twybrech/TSP-Project/src/NearestNeighbour/NearestNeighbour.c

Programme mettant en place l'heuristique de NearestNeighbour. Programme créant l'heuristique de NearestNeighbour.

```
#include <NearestNeighbour.h>
#include <Graphe.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>
#include <assert.h>
```

Fonctions

— int plusProcheVoisin (int sommet, bool ∗tabDispo, Graphe graph, double ∗acc)
Fonction permettant de retourner le voisin ayant la distance la plus proche de celui où l'on est.
— int ∗ HeuristiquePlusProcheVoisin (Graphe graph, double ∗distanceAcc, int departChemin)
Fonction permettant de retourner un tableau de voisin ayant la distance la plus proche avec son précédent.

4.25.1 Description détaillée

Programme mettant en place l'heuristique de NearestNeighbour. Programme créant l'heuristique de Nearest-Neighbour.

4.25.2 Documentation des fonctions

4.25.2.1 int∗ HeuristiquePlusProcheVoisin (Graphe *graph,* double ∗ *distanceAcc,* int *departChemin*)

Fonction permettant de retourner un tableau de voisin ayant la distance la plus proche avec son précédent.

Paramètres

distanceAcc	Tableau de booléens contenant les distances entre les villes.
departChemin	Entier correspondant à la ville de départ.
graph	Représente la matrice TSP.

4.25.2.2 int plusProcheVoisin (int *sommet,* bool ∗ *tabDispo,* Graphe *graph,* double ∗ *acc*)

Fonction permettant de retourner le voisin ayant la distance la plus proche de celui où l'on est.

Paramètres

sommet	Entier correspondant au numéro du sommet actuel dans le graphe.
tabDispo	Tableau de booléan servant à savoir la liste des villes qui n'ont pas été visitées.
graph	Représente la matrice TSP.
acc	Tableau de double contenant les distance des villes.

4.26 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Prim-MST/Prim.c

Programme implémentant l'algorithme de Prim.

```
#include <stdlib.h>
#include <malloc.h>
#include <assert.h>
#include <stdio.h>
#include <ArbrePlanaireInt.h>
#include <Arete.h>
#include <TasArete.h>
#include <Graphe.h>
#include <Prim.h>
```

Fonctions

— int ∗ Prim (Graphe g, int depart, double ∗acc)
Algorithme de Prim servant à créer un arbre de poids minimum.

4.26.1 Description détaillée

Programme implémentant l'algorithme de Prim.

4.26.2 Documentation des fonctions

4.26.2.1 int* Prim (Graphe *g*, int *depart*, double * *acc*)

Algorithme de Prim servant à créer un arbre de poids minimum.

Paramètres

g	Graphe représentant la matrice.
depart	Entier représentant la ville de départ du graphe g.
acc	Tableau de booléens contenant la distance des villes.

4.27 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Tas/TasArete.c

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <TasGenerique.h>
#include <TasArete.h>
```

Structures de données

- struct TasArete
- struct AreteHandle

Fonctions

- TasMinArete creerTasMinArete (int taille)
  Crée un tas min contenant des arêtes.
- void freeTasArete (TasMinArete tasMin)
  Libère l'espace mémoire alloué à un tas min contenant des arêtes.
- void freeAreteHandle (AreteHandle a)
  Libère l'espace mémoire alloué à une areteHandle contenant des elemHandle (une structure contenant un pointeur générique et un int).
- bool estVide (TasMinArete tasMin)
  Vérifie si un tas min d'arêtes est vide (le champ taille de la structure est à -1 dans le cas où le tas est vide).
- AreteHandle ajouterAreteHandle (TasMinArete tasMin, Arete a)
  Ajoute une areteHandle à un tas min d'arêtes, la fonction l'ajoute d'abord en tant que feuille puis la positionne à un endroit \ qui fait en sorte que le tas reste un tas min.
- int indiceAreteHandle (AreteHandle areteH)
  Retourne l'indice de la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.
- Arete extraireAreteMin (TasMinArete tasMin)
  Retourne l'arete minimale contenue dans tasMIn.
- Arete getArete (AreteHandle areteH)
  Retourne la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.
- void affichageTasArete (TasMinArete tasMin)
  Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des arêtes contenues dans tasMin.
- void diminuerCleArete (TasMinArete tas_a, AreteHandle areteH, double newCle)
  Cette fonction donne à la clé de l'arete de areteH la valeur de newCle dans le cas ou newCle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

4.27.1 Documentation des fonctions

4.27.1.1 void affichageTasArete (TasMinArete *tasMin*)

Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des arêtes contenues dans tasMin.

4.27.1.2 AreteHandle ajouterAreteHandle (TasMinArete *tasMin,* Arete *a*)

Ajoute une areteHandle à un tas min d'arêtes, la fonction l'ajoute d'abord en tant que feuille puis la positionne à un endroit \ qui fait en sorte que le tas reste un tas min.

Postcondition

Il faut que le tas ne soit pas plein, la fonction le vérifie.

4.27.1.3 TasMinArete creerTasMinArete (int *taille*)

Crée un tas min contenant des arêtes.

4.27.1.4 void diminuerCleArete (TasMinArete *tas_a,* AreteHandle *areteH,* double *newCle*)

Cette fonction donne à la clé de l'arete de areteH la valeur de newCle dans le cas ou newCle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

Précondition

la fonction s'arrête si la clé entrée en paramètre n'est pas infèrieure à la clé contenue dans la donnée membre \ arete de areteH.

Postcondition

les deux sommets reliés par l'arete de areteH sont échangés si c'est nécessaire pour que le tas reste min.

4.27.1.5 bool estVide (TasMinArete *tasMin*)

Vérifie si un tas min d'arêtes est vide (le champ taille de la structure est à -1 dans le cas où le tas est vide).

4.27.1.6 Arete extraireAreteMin (TasMinArete *tasMin*)

Retourne l'arete minimale contenue dans tasMIn.

4.27.1.7 void freeAreteHandle (AreteHandle *a*)

Libère l'espace mémoire alloué à une areteHandle contenant des elemHandle (une structure contenant un pointeur générique et un int).

4.27.1.8 void freeTasArete (TasMinArete *tasMin*)

Libère l'espace mémoire alloué à un tas min contenant des arêtes.

4.27.1.9 Arete getArete (AreteHandle *areteH*)

Retourne la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.

4.27.1.10 int indiceAreteHandle (AreteHandle *areteH*)

Retourne l'indice de la donnée membre arete contenue dans l'AreteHandle areteH passée en paramètre.

4.28 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Tas/TasGenerique.c

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <assert.h>
#include <TasGenerique.h>
#include <Arete.h>
```

Structures de données

- struct elemHandle
- struct TasMinGen

Fonctions

- int getTailleTas (TasMinGen tas)
- TasMinGen creerTasMinGen (int taille, ptr_compar cmp, ptr_compar cmpCle, ptr_affichage affichage, ptr_maj majCle)
- ElemHandle creerElemHandle (void ∗element, int indice)
- void freeElemHandle (ElemHandle elem)
- void freeTasGen (TasMinGen tas)
- void affichageTas (TasMinGen tas)
  Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des elements contenus dans tas.
- bool estvide (TasMinGen tas)
- ElemHandle ajouterSommet (TasMinGen tas, void ∗element)
- void entasserTas (TasMinGen tas, int indice)
  Réorganise le tas en tas min à partir du sommet de nom indice entré en paramètre lorsque c'est nécessaire pour \ préserver l'accés aux données.
- void ∗ extraireMin (TasMinGen tas)
- void diminuerCle (TasMinGen tas, ElemHandle element, void ∗cle)
  Cette fonction donne à la clé de elem de element la valeur de cle(entrée en paramètre) dans le cas ou cle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.
- ElemHandle sommet (TasMinGen tas, int indice)
  Retourne l'ElemHandle situé à l'indice entré en paramètre de la donnée membre sommets[] de tas.
- void setIndice (int indice, ElemHandle element)
- int getIndice (ElemHandle element)
- void ∗ getElem (ElemHandle e)
- void setElem (void ∗obj, ElemHandle element)

4.28.1 Documentation des fonctions

4.28.1.1 void affichageTas (TasMinGen *tas*)

Affiche 'CONTENU DU TAS MIN' puis en dessous, une ligne de la liste des elements contenus dans tas.

4.28.1.2 ElemHandle ajouterSommet (TasMinGen *tas*, void ∗ *element*)

ajoute un element à un tas min générique en vérifiant qu'il reste min.

Précondition

le tas min générique en parmètre ne doit pas être plein.

4.28.1.3 ElemHandle creerElemHandle (void * *element,* int *indice*)

Crée un ElemHandle et le renvoie.

4.28.1.4 TasMinGen creerTasMinGen (int *taille,* ptr_compar *cmp,* ptr_compar *cmpCle,* ptr_affichage *affichage,* ptr_maj *majCle*)

Crée un tas min générique vide.

Paramètres

4.28.1.5 void diminuerCle (TasMinGen *tas,* ElemHandle *element,* void * *cle*)

Cette fonction donne à la clé de elem de element la valeur de cle(entrée en paramètre) dans le cas ou cle est inférieure et \ elle fait en sorte que le tas reste un tas min en échangeant les sommets reliés par l'arete si besoin.

Précondition

la fonction s'arrête si la clé entrée en paramètre n'est pas infèrieure à la clé contenue dans la donnée membre \ elem de element.

Postcondition

les deux sommets reliés par l'arete contenue dans element sont échangés si c'est nécessaire pour que le tas reste min.

4.28.1.6 void entasserTas (TasMinGen *tas,* int *indice*)

Réorganise le tas en tas min à partir du sommet de nom indice entré en paramètre lorsque c'est nécessaire pour \ préserver l'accés aux données.

4.28.1.7 bool estvide (TasMinGen *tas*)

Teste si un tas min générique est vide.

4.28.1.8 void* extraireMin (TasMinGen *tas*)

Retourne le plus petit élément du tas min générique.

Postcondition

Supprime l'élément qui est retourné et réorganise le tas en tas min.

4.28.1.9 void freeElemHandle (ElemHandle *elem*)

Libère la mémoire allouée à un ElemHandle.

4.28.1.10 void freeTasGen (TasMinGen *tas*)

Libère l'espace mémoire alloué à un tas min générique.

4.28.1.11 void* getElem (ElemHandle *e*)

Retourne l'élément contenu dans la structure ElemHandle passé en paramètre.

Précondition

element objet du tas dont on veut l'element.

4.28.1.12 int getIndice (ElemHandle *element*)

Retourne l'indice de l'ElemHandle passé en paramètre.

4.28.1.13 int getTailleTas (TasMinGen *tas*)

Retourne le nombre d'élément contenu dans le tas min

4.28.1.14 void setElem (void * *obj,* ElemHandle *element*)

Affete un ElemHandle à un pointeur générique.

4.28.1.15 void setIndice (int *indice,* ElemHandle *element*)

Affecte un indice a un element du tas.

Paramètres

indice	entier a affecté a l'ElemHandle correspondant a sa position dans la structure interne du tas
element	cible de la modification d'indice.

Précondition

l'indice doit etre supérieur ou égal a 0.

4.28.1.16 ElemHandle sommet (TasMinGen *tas,* int *indice*)

Retourne l'ElemHandle situé à l'indice entré en paramètre de la donnée membre sommets[] de tas.

4.29 Référence du fichier /net/cremi/twybrech/TSP-Project/src/Test/FonctionTest.c

Programme de tests.

```
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <FonctionTest.h>
#include <Graphe.h>
#include <Input.h>
```

Fonctions

— void testBaseHeuristiqueAD (char *nomFichier, int depart, HeuristiqueAvecDepart H)
affiche OK si le test d'un heuristique qui prend une ville de départ en paramètre renvoie une solution valide, ECHEC sinon.

- void testBaseHeuristiqueSD (char *nomFichier, HeuristiqueSansDepart H)
  affiche OK si le test d'un heuristique qui ne prend pas de ville de départ en paramètre renvoie une solution valide, ECHEC sinon.
- bool estCycleValide (int *tabCycle, int taille)
  Fonction vérifiant si la solution est un cycle n'ayant qu'une seule fois chaque ville (sauf la première et la dernière qui doivent être identique).
- bool estDimensionValide (double dim, double meilleureDistance)
  Fonction vérifiant si la solution renvoie un résultat possible.
- void afficheCycle (int *tabCycle, int taille, double valeur)
  Fonction qui affiche le tableau(cycle) en paramètre avec sa valeur.
- Graphe getGraphe (char *nomFichier)
  Retourne le Graphe contenu dans un fichier TSP.
- void freeHeurisque (Graphe g, int *tabCycle)
  Libère la mémoire allouée au Graphe g et celle allouée au tableau contenant le parcours obtenu suite à l'execution d'un heuristique.

4.29.1 Description détaillée

Programme de tests. Programme de test qui vérifie les solutions des algorithmes.

4.29.2 Documentation des fonctions

4.29.2.1 void afficheCycle (int * *tabCycle*, int *taille*, double *valeur*)

Fonction qui affiche le tableau(cycle) en paramètre avec sa valeur.

4.29.2.2 bool estCycleValide (int * *tabCycle*, int *taille*)

Fonction vérifiant si la solution est un cycle n'ayant qu'une seule fois chaque ville (sauf la première et la dernière qui doivent être identique).

retourne la validité du parcours entré en paramètre.

Paramètres

tabCyble	Tableau contenant des entiers correspondant aux villes.
taille	Entier correspondant à la taille du tableau tabCycle.

4.29.2.3 bool estDimensionValide (double *dim*, double *meilleureDistance*)

Fonction vérifiant si la solution renvoie un résultat possible.

Renvoie la validité de la dimension du parcours, si le parcours fait une distance inférieure au plus court chemin il est invalidé.

Paramètres

dim	Double contenant une distance.
meilleure-Distance	Double contenant la distance du chemin min. d'un graphe.

4.29.2.4 void freeHeurisque (Graphe *g*, int * *tabCycle*)

Libère la mémoire allouée au Graphe g et celle allouée au tableau contenant le parcours obtenu suite à l'execution d'un heuristique.

4.29.2.5 Graphe getGraphe (char * *nomFichier*)

Retourne le Graphe contenu dans un fichier TSP.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

4.29.2.6 void testBaseHeuristiqueAD (char * *nomFichier,* int *depart,* HeuristiqueAvecDepart *H*)

affiche OK si le test d'un heuristique qui prend une ville de départ en paramètre renvoie une solution valide, ECHEC sinon.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

4.29.2.7 void testBaseHeuristiqueSD (char * *nomFichier,* HeuristiqueSansDepart *H*)

affiche OK si le test d'un heuristique qui ne prend pas de ville de départ en paramètre renvoie une solution valide, ECHEC sinon.

Paramètres

nomFichier	chaîne de caractères représentant le nom d'un fichier TSP contenant un graphe.

Index