

Le voyageur de commerce

1

Généré par Doxygen 1.7.1

Thu Jan 10 2013 11 :34 :31

Table des matières

1	Le problème du voyageur de commerce -- Projet d'algorithmique en langage C	1
1.1	problème	1
1.2	algorithmes implémenté	1
2	Index des classes	3
2.1	Liste des classes	3
3	Index des fichiers	5
3.1	Liste des fichiers	5
4	Documentation des classes	7
4.1	Référence de la structure Algo	7
4.1.1	Description détaillée	7
4.2	Référence de la structure Distance	7
4.2.1	Description détaillée	8
4.3	Référence de la structure Errors	8
4.3.1	Description détaillée	9
4.4	Référence de la structure Instance	9
4.4.1	Description détaillée	9
4.5	Référence de la structure Tour	10
4.5.1	Description détaillée	10
4.6	Référence de la structure Town	10
4.6.1	Description détaillée	10
5	Documentation des fichiers	11
5.1	Référence du fichier lib/bruteForce.h	11
5.1.1	Description détaillée	11
5.1.2	Documentation des fonctions	11
5.1.2.1	bruteForce_bestPath	11
5.2	Référence du fichier lib/distance.h	12

5.2.1	Description détaillée	12
5.2.2	Documentation des fonctions	12
5.2.2.1	distance_betweenTowns	12
5.2.2.2	distance_calculDistance	13
5.2.2.3	distance_new	13
5.2.2.4	distance_searchDistance	13
5.3	Référence du fichier lib/errors.h	13
5.3.1	Description détaillée	14
5.3.2	Documentation des fonctions	15
5.3.2.1	errors_displayErrorsMessage	15
5.3.2.2	errors_new	15
5.3.2.3	errors_setFileNotFound	15
5.3.2.4	errors_setMissingParameterGa	15
5.3.2.5	errors_setMissingParameterLsnr	15
5.3.2.6	errors_setMissingParameterLsr	15
5.3.2.7	errors_setNbArguments	16
5.3.2.8	errors_setNoAlgoSpecified	16
5.3.2.9	errors_setNoValidParameterLsnr	16
5.3.2.10	errors_setNoValidParameterLsr	16
5.3.2.11	errors_setTagFNotFound	16
5.4	Référence du fichier lib/instance.h	16
5.4.1	Description détaillée	17
5.4.2	Documentation des fonctions	17
5.4.2.1	instance_display	17
5.4.2.2	instance_displayLinearVector	18
5.4.2.3	instance_displayMatrix	18
5.4.2.4	instance_initializeDistancesMatrix	18
5.4.2.5	instance_new	18
5.4.2.6	instance_push	18
5.5	Référence du fichier lib/localSearch.h	19
5.5.1	Description détaillée	19
5.6	Référence du fichier lib/parsing.h	19
5.6.1	Description détaillée	20
5.6.2	Documentation du type de l'énumération	20
5.6.2.1	AlgoType	20
5.6.3	Documentation des fonctions	20

5.6.3.1	<code>parsing_algoType</code>	20
5.6.3.2	<code>parsing_parseFileName</code>	21
5.6.3.3	<code>parsing_parseVerboseMode</code>	21
5.7	Référence du fichier <code>lib/tour.h</code>	21
5.7.1	Description détaillée	22
5.7.2	Documentation des fonctions	22
5.7.2.1	<code>tour_2opt</code>	22
5.7.2.2	<code>tour_calculLength</code>	22
5.7.2.3	<code>tour_display</code>	23
5.7.2.4	<code>tour_new</code>	23
5.7.2.5	<code>tour_nextPermutation</code>	23
5.7.2.6	<code>tour_randomWalk</code>	23
5.8	Référence du fichier <code>lib/town.h</code>	23
5.8.1	Description détaillée	24
5.8.2	Documentation des fonctions	24
5.8.2.1	<code>town_new</code>	24
5.9	Référence du fichier <code>lib/util.h</code>	24
5.9.1	Description détaillée	25
5.9.2	Documentation des fonctions	25
5.9.2.1	<code>util_displayArray</code>	25
5.9.2.2	<code>util_rand</code>	25
5.9.2.3	<code>util_reverseArray</code>	26
5.9.2.4	<code>util_searchFirstOccurenceInArray</code>	26
5.9.2.5	<code>util_sousTabExist</code>	26
5.9.2.6	<code>util_sum</code>	27
5.9.2.7	<code>util_swap</code>	27
5.10	Référence du fichier <code>src/bruteForce.c</code>	27
5.10.1	Description détaillée	27
5.10.2	Documentation des fonctions	28
5.10.2.1	<code>bruteForce_bestPath</code>	28
5.11	Référence du fichier <code>src/distance.c</code>	28
5.11.1	Description détaillée	28
5.11.2	Documentation des fonctions	29
5.11.2.1	<code>distance_betweenTowns</code>	29
5.11.2.2	<code>distance_calculDistance</code>	29
5.11.2.3	<code>distance_new</code>	29

5.11.2.4	<code>distance_searchDistance</code>	29
5.12	Référence du fichier <code>src/genetic.c</code>	30
5.12.1	Description détaillée	30
5.13	Référence du fichier <code>src/instance.c</code>	30
5.13.1	Description détaillée	31
5.13.2	Documentation des fonctions	31
5.13.2.1	<code>instance_display</code>	31
5.13.2.2	<code>instance_displayLinearVector</code>	31
5.13.2.3	<code>instance_displayMatrix</code>	31
5.13.2.4	<code>instance_initializeDistancesMatrix</code>	31
5.13.2.5	<code>instance_new</code>	32
5.13.2.6	<code>instance_push</code>	32
5.14	Référence du fichier <code>src/localSearch.c</code>	32
5.14.1	Description détaillée	32
5.15	Référence du fichier <code>src/main.c</code>	33
5.15.1	Description détaillée	33
5.15.2	Documentation des fonctions	33
5.15.2.1	<code>main</code>	33
5.16	Référence du fichier <code>src/parsing.c</code>	33
5.16.1	Description détaillée	34
5.16.2	Documentation des fonctions	34
5.16.2.1	<code>parsing_algoType</code>	34
5.16.2.2	<code>parsing_parseFileName</code>	34
5.16.2.3	<code>parsing_parseVerboseMode</code>	35
5.17	Référence du fichier <code>src/tour.c</code>	35
5.17.1	Description détaillée	35
5.17.2	Documentation des fonctions	36
5.17.2.1	<code>tour_2opt</code>	36
5.17.2.2	<code>tour_calculLength</code>	36
5.17.2.3	<code>tour_display</code>	36
5.17.2.4	<code>tour_new</code>	36
5.17.2.5	<code>tour_nextPermutation</code>	36
5.17.2.6	<code>tour_randomWalk</code>	37
5.18	Référence du fichier <code>src/town.c</code>	37
5.18.1	Description détaillée	37
5.18.2	Documentation des fonctions	37

5.18.2.1	town_new	37
5.19	Référence du fichier src/util.c	37
5.19.1	Description détaillée	38
5.19.2	Documentation des fonctions	38
5.19.2.1	util_displayArray	38
5.19.2.2	util_rand	38
5.19.2.3	util_reverseArray	39
5.19.2.4	util_searchFirstOccurenceInArray	39
5.19.2.5	util_sousTabExist	39
5.19.2.6	util_sum	40
5.19.2.7	util_swap	40

Chapitre 1

Le problème du voyageur de commerce -- Projet d'algorithmique en langage C

Auteur

L2 Antoine de Roquemaurel (G1.1)

1.1 problème

Étant donné n points (des "villes") et les distances les séparant, trouver un chemin de longueur totale qui passe exactement une fois par chaque point et reviennent au point de départ (une tournée).

Ce problème peut servir tel quel à l'optimisation de trajectoires de machines-outils : par exemple, pour minimiser le temps total que met une fraiseuse à commande numérique pour percer n points dans une plaque de tôle ou pour percer les trous des composants d'un circuit électronique comme dans le cas qui nous intéresse.

Ce problème est plus compliqué qu'il n'y paraît et on ne connaît pas de méthode de résolution permettant d'obtenir des solutions exactes en un temps raisonnable pour de grandes instances (grand nombre de villes) du problème. Pour ces grandes instances, on devra donc souvent se contenter de solutions approchées, car on se retrouve face à une explosion combinatoire : le nombre de chemins possibles passant par 69 villes est déjà un nombre d'une longueur de 100 chiffres. Pour comparaison, un nombre d'une longueur de 80 chiffres permettrait déjà de représenter le nombre d'atomes dans tout l'univers connu.

Le problème du "voyageur de commerce" a été étudié depuis longtemps et on dispose d'une grande variété d'algorithmes donnant le plus souvent des solutions approchées mais calculables en un temps raisonnable.

1.2 algorithmes implémentés

Ce problème sera implémenté via différents algorithmes :

- Brute force
- Recherche locale aléatoire
- Recherche locale systématique
- Algorithme génétique

Chapitre 2

Index des classes

2.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

Algo (Enumération d'un algorithme)	7
Distance (Objet des distances)	7
Errors (Objet des erreurs)	8
Instance (Objet des instances)	9
Tour (Objet d'une tournée)	10
Town (Objet des ville)	10

Chapitre 3

Index des fichiers

3.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

lib/ bruteForce.h (Fonctions de brute force)	11
lib/ distance.h (Fonctions des distances)	12
lib/ errors.h (Fonctions sur les erreurs)	13
lib/ genetic.h	??
lib/ instance.h (Fonctions sur les instances)	16
lib/ localSearch.h (Fonctions utiles)	19
lib/ parsing.h (Fonctions de parsing des arguments)	19
lib/ tour.h (Fonctions des tournées)	21
lib/ town.h (Fonctions des villes)	23
lib/ util.h (Fonctions utiles)	24
src/ bruteForce.c (Fonctions de brute force)	27
src/ distance.c (Fonctions des distances)	28
src/ genetic.c (Fonctions d'algorithmes génétiques)	30
src/ instance.c (Fonctions utiles)	30
src/ localSearch.c (Fonctions de recherche locale)	32
src/ main.c (Fonction main)	33
src/ parsing.c (Fonctions de parsing des arguments)	33
src/ tour.c (Fonctions des tournées)	35
src/ town.c (Fonctions des villes)	37
src/ util.c (Fonctions utiles)	37

Chapitre 4

Documentation des classes

4.1 Référence de la structure Algo

Enumération d'un algorithme.

```
#include <parsing.h>
```

Attributs publics

- [AlgoType](#) type
Le type de l'algorithme.
- int [firstParameter](#)
Premier paramètre de l'algorithme.
- int [secondParameter](#)
Second paramètre.

4.1.1 Description détaillée

Enumération d'un algorithme. Énumération d'un algorithme, contient le type de l'algorithme avec les eventuels paramètres

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[parsing.h](#)

4.2 Référence de la structure Distance

Objet des distances.

```
#include <distance.h>
```

Attributs publics

- [Town](#) firstTown

Première ville.

- [Town secondTown](#)

Seconde ville.

- double [distance](#)

Distance entre les deux villes.

4.2.1 Description détaillée

Objet des distances. [Distance](#) entre deux villes

Voir également

[Town](#)

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[distance.h](#)

4.3 Référence de la structure Errors

Objet des erreurs.

```
#include <errors.h>
```

Attributs publics

- char * [errorNbArguments](#)

Nombre d'argument incorrect.

- char * [errorTagFNotFound](#)

Tag -f non trouvé.

- char * [errorFileNotFound](#)

Fichier non trouvé.

- char * [errorNoAlgoSpecified](#)

Algorithme non spécifié.

- char * [errorMissingParameterLsr](#)

Paramètre après -lsr manquant.

- char * [errorMissingParameterLsnr](#)

Paramètre après -lsnr manquant.

- char * [errorMissingParameterGa](#)

Paramètre après -ga manquant.

- char * [errorNoValidParameterLsr](#)

Paramètre après -lsr non valide.

- char * [errorNoValidParameterLsnr](#)

Paramètre après -lsnr non valide.

- char * [errorNoValidParameterGa](#)
Paramètre après -ga non valide.
- int [nbErrors](#)
Nombre d'erreurs.

4.3.1 Description détaillée

Objet des erreurs. Toutes les chaînes de caractères des erreurs. Si une variable vaut NULL, l'erreur n'est pas présente, sinon elle sera affichée. nbErrors est le nombre d'erreur, si celui-ci =0 alors le programme peut fonctionner correctement

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[errors.h](#)

4.4 Référence de la structure Instance

Objet des instances.

```
#include <instance.h>
```

Attributs publics

- [Town towns](#) [N]
Tableau des villes([Town](#)) classés par ID.
- [Distance distances](#) [N]
Tableau linéaire contenant toutes les distances entre les villes.
- int [nbTowns](#)
Nombre de ville de l' [Instance](#).
- char * [name](#)
Nom de l' [Instance](#).
- char * [type](#)
Type de l' [Instance](#).

4.4.1 Description détaillée

Objet des instances.

Voir également

[Town](#)
[Distance](#)

Une instance contient toutes les villes ([Town](#)) classés par ID, les calculs des algorithmes utilisent une instance afin d'en retourner la meilleure tournée.

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[instance.h](#)

4.5 Référence de la structure Tour

Objet d'une tournée.

```
#include <tour.h>
```

Attributs publics

- [Town towns](#) [N]
Tableau de ville. Les villes sont triés dans l'ordre de la tournée.
- int [nbTowns](#)
Nombre de ville de la tournée.
- double [length](#)
Longueur de la tournée.
- [Distance](#) * [distances](#)
Matrice de distances.

4.5.1 Description détaillée

Objet d'une tournée. Informations concernant une tournée.

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[tour.h](#)

4.6 Référence de la structure Town

Objet des ville.

```
#include <town.h>
```

Attributs publics

- int [x](#)
Abscisse de la ville.
- int [y](#)
Ordonnée de la ville.
- int [id](#)
Id de la ville.

4.6.1 Description détaillée

Objet des ville. Structure représentant une ville

La documentation de cette structure a été générée à partir du fichier suivant :

- lib/[town.h](#)

Chapitre 5

Documentation des fichiers

5.1 Référence du fichier lib/bruteForce.h

Fonctions de brute force.

```
#include "instance.h"
#include "tour.h"
#include "util.h"
```

Fonctions

- [Tour bruteForce_bestPath](#) ([Instance](#) pInstance)
Permet d'obtenir le meilleur chemin d'une instance via la force brute.

5.1.1 Description détaillée

Fonctions de brute force.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 17 :58 :36

Entêtes des fonctions servant à la force brute. Ces fonctions ne sont appelés que depuis le main

5.1.2 Documentation des fonctions

5.1.2.1 Tour bruteForce_bestPath (Instance *pInstance*)

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

Paramètres

pInstance L'instance pour laquelle on doit calculer le plus court chemin

Renvoi

La meilleur tournée.

5.2 Référence du fichier lib/distance.h

Fonctions des distances.

```
#include <math.h>
#include "town.h"
```

Classes

- struct [Distance](#)
Objet des distances.

Fonctions

- [Distance](#) [distance_new](#) ([Town](#) *pFirstTown, [Town](#) *pSecondTown)
Créer une nouvelle distance.
- double [distance_calculDistance](#) (const [Town](#) pTown1, const [Town](#) pTown2)
Calcul la distance entre deux villes.
- double [distance_betweenTowns](#) ([Distance](#) *pDistances, int i, int j)
Calcul la distance entre deux ID de villes.
- [Distance](#) [distance_searchDistance](#) ([Distance](#) *pDistances, const int pFirst, const int pSecond)
Recherche la distance entre deux villes dans le tableau de distance.

5.2.1 Description détaillée

Fonctions des distances.

Auteur

Antoine de Roquemaurel

Date

01/12/2012 20 :33 :44

Entêtes des fonctions se rapportant aux distances

5.2.2 Documentation des fonctions

5.2.2.1 double [distance_betweenTowns](#) ([Distance](#) * *pDistances*, int *i*, int *j*)

Calcul la distance entre deux ID de villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

5.2.2.2 double distance_calculDistance (const Town *pTown1*, const Town *pTown2*)

Calcul la distance entre deux villes.

Paramètres

pTown1 Première ville

pTown2 Seconde ville

Renvoie

La distance

5.2.2.3 Distance distance_new (Town * *pFirstTown*, Town * *pSecondTown*)

Créer une nouvelle distance.

Paramètres

pFirstTown Première ville

pSecondTown Seconde ville

Renvoie

[Distance](#) entre les deux villes

5.2.2.4 Distance distance_searchDistance (Distance * *pDistances*, const int *pFirst*, const int *pSecond*)

Recherche la distance entre deux villes dans le tableau de distance.

Paramètres

pDistances La matrice de distance

pFirst La première ville

pSecond la seconde ville

Renvoie

La distance

5.3 Référence du fichier lib/errors.h

Fonctions sur les erreurs.

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

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

Classes

- struct `Errors`
Objet des erreurs.

Fonctions

- `Errors errors_new ()`
Initialisation de l'objet.
- void `errors_displayErrorMessage (const Errors pErrors)`
Affiches toutes les erreurs de pErrors.
- void `errors_setNbArguments (Errors *pErrors)`
Signale que le nombre d'argument est incorrect.
- void `errors_setTagFNotFound (Errors *pErrors)`
Signale que la balise -f n'a pas été trouvée.
- void `errors_setFileNotFound (Errors *pErrors, char *fileName)`
Signale que le fichier fileName n'existe pas.
- void `errors_setNoAlgoSpecified (Errors *pErrors)`
Signale qu'aucun algorithme n'a été spécifié.
- void `errors_setNoValidParameterLsnr (Errors *pErrors)`
Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).
- void `errors_setNoValidParameterLsr (Errors *pErrors)`
Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).
- void `errors_setMissingParameterGa (Errors *pErrors)`
Signale qu'aucun paramètre n'a été spécifié derrière -ga.
- void `errors_setMissingParameterLsnr (Errors *pErrors)`
Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.
- void `errors_setMissingParameterLsr (Errors *pErrors)`
Signale qu'aucun paramètre n'a été spécifié derrière -lsr.

5.3.1 Description détaillée

Fonctions sur les erreurs.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :42 :37

Entêtes des fonctions concernant les erreurs du programme.

5.3.2 Documentation des fonctions

5.3.2.1 void errors_displayErrorsMessage (const Errors *pErrors*)

Affiches toutes les erreurs de *pErrors*.

Paramètres

pErrors L'objet pour laquelle on doit afficher les erreurs

5.3.2.2 Errors errors_new ()

Initialisation de l'objet.

Renvoie

Une instance de Erreur initialisée

5.3.2.3 void errors_setFileNotFound (Errors * *pErrors*, char * *fileName*)

Signale que le fichier *fileName* n'existe pas.

Paramètres

pErrors L'objet des erreurs

fileName Le nom de fichier non trouvé

5.3.2.4 void errors_setMissingParameterGa (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -ga.

Paramètres

pErrors L'objet des erreurs

5.3.2.5 void errors_setMissingParameterLsnr (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -lsnr.

Paramètres

pErrors L'objet des erreurs

5.3.2.6 void errors_setMissingParameterLsr (Errors * *pErrors*)

Signale qu'aucun paramètre n'a été spécifié derrière -lsr.

Paramètres

pErrors L'objet des erreurs

5.3.2.7 void errors_setNbArguments (Errors * *pErrors*)

Signale que le nombre d'argument est incorrect.

Paramètres

pErrors L'objet des erreurs

5.3.2.8 void errors_setNoAlgoSpecified (Errors * *pErrors*)

Signale qu'aucun algorithme n'a été spécifié.

Paramètres

pErrors L'objet des erreurs

5.3.2.9 void errors_setNoValidParameterLsnr (Errors * *pErrors*)

Signale que les paramètres derrière -lsnr ne sont pas valides (non entier).

Paramètres

pErrors L'objet des erreurs

5.3.2.10 void errors_setNoValidParameterLsr (Errors * *pErrors*)

Signale que le paramètre derrière -lsr n'est pas valide (n'est pas un entier).

Paramètres

pErrors L'objet des erreurs

5.3.2.11 void errors_setTagFNotFound (Errors * *pErrors*)

Signale que la balise -f n'a pas été trouvée.

Paramètres

pErrors L'objet des erreurs

5.4 Référence du fichier lib/instance.h

Fonctions sur les instances.

```
#include <stdbool.h>
#include <string.h>
#include "town.h"
#include "distance.h"
```


Classes

- struct [Instance](#)
Objet des instances.

Macros

- #define [N](#) 1024
Taille maximale des tableaux.

Fonctions

- void [instance_display](#) (const [Instance](#) pInstance)
Affiche une [Instance](#).
- [Instance](#) [instance_new](#) (FILE *pFile)
Créer une nouvelle [Instance](#) à partir d'un fichier.
- void [instance_push](#) ([Instance](#) *pInstance, const [Town](#) pTown)
Ajoute une nouvelle ville dans une [Instance](#).
- void [instance_initializeDistancesMatrix](#) ([Instance](#) *pInstance)
Initialise la matrice des distances.
- void [instance_displayLinearVector](#) ([Instance](#) pInstance)
Affiche la matrice des distances sous forme linéaire.
- void [instance_displayMatrix](#) ([Instance](#) pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.4.1 Description détaillée

Fonctions sur les instances.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :03 :34

Toutes les entêtes des fonctions se rapportant à une instance.

5.4.2 Documentation des fonctions

5.4.2.1 void [instance_display](#) (const [Instance](#) *pInstance*)

Affiche une [Instance](#).

Paramètres

pInstance l' [Instance](#) à afficher

5.4.2.2 void instance_displayLinearVector (Instance *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.4.2.3 void instance_displayMatrix (Instance *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.4.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

[Distance](#)

5.4.2.5 Instance instance_new (FILE * *pFile*)

Créer une nouvelle [Instance](#) à partir d'un fichier.

Paramètres

pFile Le pointeur sur fichier contenant les informations de l'instance

Renvoie

la nouvelle [Instance](#)

5.4.2.6 void instance_push (Instance * *pInstance*, const Town *pTown*)

Ajoute une nouvelle ville dans une [Instance](#).

Paramètres

pInstance L'instance à modifier

pTown La ville à ajouter

5.5 Référence du fichier lib/localSearch.h

Fonctions utiles.

```
#include "tour.h"
#include <time.h>
#include <stdlib.h>
```

Fonctions

- [Tour](#) `localSearch_randomBestPath` ([Instance](#) pInstance, int pTryNb)
- [Tour](#) `localSearch_systematicBestPath` ([Instance](#) pInstance, int pTryNb)

5.5.1 Description détaillée

Fonctions utiles. Fonctions de recherche locale.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :30

Entêtes des fonctions pouvant être utiles dans tout le projet. Ces fonctions ne sont appelés que depuis le main

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :18

Entêtes des fonctions ayant rapport avec les algorithmes de recherches locales.

5.6 Référence du fichier lib/parsing.h

Fonctions de parsing des arguments.

```
#include <stdbool.h>
#include "util.h"
#include "errors.h"
```

Classes

- struct [Algo](#)
Enumération d'un algorithme.

Énumérations

- enum [AlgoType](#) {
 BRUTEFORCE, LOCALSEARCH_RANDOM, LOCALSEARCH_SYSTEMATIC, GENETIC,
 END }
 [parsing.h](#)

Fonctions

- bool [parsing_parseVerboseMode](#) (char **pTab, const int pSize)
 Cherche si le mode verbeux à été spécifié ou non.
- char * [parsing_parseFileName](#) (char **pTab, const int pSize, [Errors](#) *pErrors)
 Cherche le nom du fichier de l'Instance.
- void [parsing_algoType](#) (char **pTab, const int pSize, [Errors](#) *pErrors, [Algo](#) *algos)

5.6.1 Description détaillée

Fonctions de parsing des arguments.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :17 :24

Entêtes des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

5.6.2 Documentation du type de l'énumération

5.6.2.1 enum AlgoType

[parsing.h](#)

Énumération des types d'algorithmes Les différents types d'algorithmes qui peuvent être appelés.

Valeurs énumérées :

BRUTEFORCE L'algorithme de brute force.
LOCALSEARCH_RANDOM L'algorithme de recherche locale aléatoire.
LOCALSEARCH_SYSTEMATIC L'algorithme de recherche locale systématique.
GENETIC L'algorithme génétique.
END Correspond au marqueur de fin des algorithmes.

5.6.3 Documentation des fonctions

5.6.3.1 void parsing_algoType (char ** pTab, const int pSize, Errors * pErrors, Algo * algos)

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

algos Tableau d'algorithmes, ceci au cas où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

[Algo](#)

5.6.3.2 char* parsing_parseFileName (char ** *pTab*, const int *pSize*, Errors * *pErrors*)

Cherche le nom du fichier de l'Instance.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

Renvoie

Le nom du fichier

Voir également

[Instance](#)

5.6.3.3 bool parsing_parseVerboseMode (char ** *pTab*, const int *pSize*)

Cherche si le mode verbeux a été spécifié ou non.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

Renvoie

Vrai si mode verbeux, Faux sinon

5.7 Référence du fichier lib/tour.h

Fonctions des tournées.

```
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include "town.h"
#include "instance.h"
#include "distance.h"
```

Classes

- struct **Tour**
Objet d'une tournée.

Fonctions

- **Tour** **tour_new** (**Instance** pInstance)
Cr  er une nouvelle tournée initialis  e avec les donn  es d'une instance.
- bool **tour_nextPermutation** (**Tour** *pPermutation)
G  n  re la permutation de ville suivante d'une tournée.
- void **tour_calculLength** (**Tour** *pTour)
Calcul la longueur d'une tournée.
- void **tour_display** (const **Tour** pTour)
Affiche une tournée.
- **Tour** **tour_randomWalk** (const **Instance** pInstance)
G  n  re une tournée al  atoire.
- void **tour_2opt** (**Tour** *pTour, int pFirst, int pSecond)
Fait une 2opt.

5.7.1 Description d  taill  e

Fonctions des tourn  es.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :13

Ent  tes des fonctions se rapportant    une tournée. *

5.7.2 Documentation des fonctions

5.7.2.1 void **tour_2opt** (**Tour** * *pTour*, int *pFirst*, int *pSecond*)

Fait une 2opt.

Param  tres

pTour Le tour pour laquelle on veut faire une 2opt

pFirst L'id du d  but du premier trajet

pSecond L'id du d  but du second trajet

5.7.2.2 void **tour_calculLength** (**Tour** * *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pour laquelle on veut calculer la longueur

5.7.2.3 void tour_display (const Tour pTour)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.7.2.4 Tour tour_new (Instance pInstance)

Créer une nouvelle tournée initialisée avec les données d'une instance.

Paramètres

pInstance Instance servant à initialiser la tournée

Renvoie

la nouvelle tournée

5.7.2.5 bool tour_nextPermutation (Tour * pPermutation)

Génère la permutation de ville suivante d'une tournée.

Paramètres

pPermutation La tournée pour laquelle la permutation doit être générée

Renvoie

Vrai si une permutation a été générée faux s'il ne reste plus de permutation.

5.7.2.6 Tour tour_randomWalk (const Instance pInstance)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

5.8 Référence du fichier lib/town.h

Fonctions des villes.

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

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

Classes

- struct [Town](#)
Objet des ville.

Fonctions

- [Town town_new](#) (const int pId, const int pX, const int pY)
Création d'une nouvelle ville.

5.8.1 Description détaillée

Fonctions des villes.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :19

Entêtes des fonctions se rapportant à une ville.

5.8.2 Documentation des fonctions

5.8.2.1 Town town_new (const int *pId*, const int *pX*, const int *pY*)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.9 Référence du fichier lib/util.h

Fonctions utiles.

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include "tour.h"
```

Fonctions

- int [util_searchFirstOccurenceInArray](#) (char **pArray, const int pSize, char *pSearch)
Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.

- void `util_reverseArray` (`Town *pTab`, const int pBegin, const int pEnd)
Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.
- void `util_displayArray` (const int *pTab, const int pSize)
Affiche le contenu d'un tableau d'entiers.
- int `util_sum` (const int pBegin, const int pEnd)
Calcul la somme des éléments allant de pBegin à pEnd.
- void `util_swap` (int *a, int *b)
Échange deux variables.
- int `util_rand` (const int pMin, const int pMax)
Calcul une valeur aléatoire entre pMin et pMax.
- bool `util_sousTabExist` (`Tour pChild`, const int pBegin, const int pEnd, `Tour pParent`, bool pRecursvite)
Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Variables

- bool `gVerboseMode`
Mode verbose.

5.9.1 Description détaillée

Fonctions utiles.

Auteur

Antoine de Roquemaurel

Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.9.2 Documentation des fonctions

5.9.2.1 void util_displayArray (const int * pTab, const int pSize)

Affiche le contenu d'un tableau d'entiers.

Paramètres

pTab Le tableau à afficher

pSize La taille du tableau

5.9.2.2 int util_rand (const int pMin, const int pMax)

Calcul une valeur aléatoire entre pMin et pMax.

Paramètres

pMin Minimum

pMax Maximum

Renvoie**5.9.2.3 void util_reverseArray (Town * pTab, const int pBegin, const int pEnd)**

Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.

Paramètres

pTab Tableau à inverser

pBegin Début de la section à inverser

pEnd Fin de la section à inverser

5.9.2.4 int util_searchFirstOccurenceInArray (char ** pArray, const int pSize, char * pSearch)

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaînes de caractères.

Paramètres

pArray Le tableau de chaînes de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaîne de caractère à chercher

Renvoie

La position de la chaîne dans le tableau ou -1 si elle n'a pas été trouvée

5.9.2.5 bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursive)

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Paramètres

pChild Le tableau à chercher

pBegin Le début de la section à chercher

pEnd La fin de la section à chercher

pParent Le tableau dans lequel chercher

pRecursive Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

Renvoie

Vrai si le sous-tableau a été trouvé faux sinon

5.9.2.6 int util_sum (const int *pBegin*, const int *pEnd*)

Calcul la somme des éléments allant de *pBegin* à *pEnd*.

Paramètres

pBegin Début de la somme

pEnd Fin de la somme

Renvoie

La somme des éléments

5.9.2.7 void util_swap (int * *a*, int * *b*)

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

5.10 Référence du fichier src/bruteForce.c

Fonctions de brute force.

```
#include "bruteForce.h"
```

```
#include "util.h"
```

Fonctions

- [Tour bruteForce_bestPath](#) (Instance *pInstance*)
Permet d'obtenir le meilleur chemin d'une instance via la force brute.

5.10.1 Description détaillée

Fonctions de brute force. Fonctions sur les erreurs.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 17 :58 :29

Implémentation des fonctions utilisant la force brute. Ces fonctions ne sont appelés que depuis le main

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :42 :31

Implémentation des fonctions concernant les erreurs du programme.

5.10.2 Documentation des fonctions**5.10.2.1 Tour bruteForce_bestPath (Instance *pInstance*)**

Permet d'obtenir le meilleur chemin d'une instance via la force brute.

Il est conseillé de ne pas essayer avec des instances de plus de 8 villes.

Paramètres

pInstance L'instance pour laquelle on doit calculer le plus court chemin

Renvoie

La meilleur tournée.

5.11 Référence du fichier src/distance.c

Fonctions des distances.

```
#include "distance.h"
#include "util.h"
```

Fonctions

- [Distance distance_new](#) (Town *pFirstTown, Town *pSecondTown)
Créer une nouvelle distance.
- double [distance_calculDistance](#) (const Town pTown1, const Town pTown2)
Calcul la distance entre deux villes.
- [Distance distance_searchDistance](#) (Distance *pDistances, const int pFirst, const int pSecond)
Recherche la distance entre deux villes dans le tableau de distance.
- double [distance_betweenTowns](#) (Distance *pDistances, int i, int j)
Calcul la distance entre deux ID de villes.

5.11.1 Description détaillée

Fonctions des distances.

Auteur

Antoine de Roquemaurel

Date

01/12/2012 20 :33 :39

Entêtes des fonctions se rapportant aux distances

5.11.2 Documentation des fonctions

5.11.2.1 double distance_betweenTowns (Distance * *pDistances*, int *i*, int *j*)

Calcul la distance entre deux ID de villes.

Paramètres

pDistances La matrice de distances

i La première ville

j La seconde ville

Renvoie

La distance

5.11.2.2 double distance_calculDistance (const Town *pTown1*, const Town *pTown2*)

Calcul la distance entre deux villes.

Paramètres

pTown1 Première ville

pTown2 Seconde ville

Renvoie

La distance

5.11.2.3 Distance distance_new (Town * *pFirstTown*, Town * *pSecondTown*)

Créer une nouvelle distance.

Paramètres

pFirstTown Première ville

pSecondTown Seconde ville

Renvoie

[Distance](#) entre les deux villes

5.11.2.4 Distance distance_searchDistance (Distance * *pDistances*, const int *pFirst*, const int *pSecond*)

Recherche la distance entre deux villes dans le tableau de distance.

Paramètres

pDistances La matrice de distance

pFirst La première ville

pSecond la seconde ville

Renvoie

La distance

5.12 Référence du fichier src/genetic.c

Fonctions d'algorithmes génétiques.

```
#include "genetic.h"
#include "tour.h"
```

Fonctions

- [Tour](#) `genetic_distancePreservingCrossover` ([Tour](#) pParent1, [Tour](#) pParent2)

5.12.1 Description détaillée

Fonctions d'algorithmes génétiques.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :25

Implémentation des fonctions servant aux algorithmes génétiques. Ces fonctions ne sont appelés que depuis le main

5.13 Référence du fichier src/instance.c

Fonctions utiles.

```
#include "instance.h"
```

Fonctions

- [Instance](#) `instance_new` (FILE *pFile)
Créer une nouvelle [Instance](#) à partir d'un fichier.
- void `instance_display` (const [Instance](#) pInstance)
Affiche une [Instance](#).
- void `instance_push` ([Instance](#) *pInstance, const [Town](#) pTown)
Ajoute une nouvelle ville dans une [Instance](#).
- void `instance_initializeDistancesMatrix` ([Instance](#) *pInstance)
Initialise la matrice des distances.
- void `instance_displayLinearVector` ([Instance](#) pInstance)
Affiche la matrice des distances sous forme linéaire.
- void `instance_displayMatrix` ([Instance](#) pInstance)
Affiche la matrice des distances sous forme de matrice symétrique.

5.13.1 Description détaillée

Fonctions utiles.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :03 :26

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.13.2 Documentation des fonctions

5.13.2.1 void instance_display (const Instance *pInstance*)

Affiche une [Instance](#).

Paramètres

pInstance l' [Instance](#) à afficher

5.13.2.2 void instance_displayLinearVector (Instance *pInstance*)

Affiche la matrice des distances sous forme linéaire.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.13.2.3 void instance_displayMatrix (Instance *pInstance*)

Affiche la matrice des distances sous forme de matrice symétrique.

Paramètres

pInstance L'instance à afficher

Voir également

[Distance](#)

5.13.2.4 void instance_initializeDistancesMatrix (Instance * *pInstance*)

Initialise la matrice des distances.

Paramètres

pInstance L'instance à modifier

Voir également

[Distance](#)

5.13.2.5 Instance `instance_new (FILE * pFile)`

Cr  er une nouvelle [Instance](#)    partir d'un fichier.

Param  tres

pFile Le pointeur sur fichier contenant les informations de l'instance

Renvoie

la nouvelle [Instance](#)

5.13.2.6 `void instance_push (Instance * pInstance, const Town pTown)`

Ajoute une nouvelle ville dans une [Instance](#).

Param  tres

pInstance L'instance    modifier

pTown La ville    ajouter

5.14 R  f  rence du fichier `src/localSearch.c`

Fonctions de recherche locale.

```
#include "util.h"
#include "localSearch.h"
#include "instance.h"
#include "tour.h"
```

Fonctions

- [Tour](#) `localSearch_randomBestPath (Instance pInstance, int pTryNb)`
- [Tour](#) `localSearch_systematicBestPath (Instance pInstance, int pTryNb)`

5.14.1 Description d  taill  e

Fonctions de recherche locale.

Auteur

Antoine de Roquemaurel

Date

27/12/2012 18 :00 :13

Ent  tes des fonctions ayant rapport avec les algorithmes de recherches locales.

5.15 Référence du fichier src/main.c

Fonction main.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include "parsing.h"
#include "errors.h"
#include "bruteForce.h"
#include "localSearch.h"
```

Fonctions

- int `main` (int argc, char **argv)
Fonction d'entrée du programme.

5.15.1 Description détaillée

Fonction main.

Auteur

Antoine de Roquemaurel

Date

19/11/2012 10 :42 :29

Point d'entrée du programme. Aucune fonction ne doit être déclarée Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.15.2 Documentation des fonctions

5.15.2.1 int main (int argc, char ** argv)

Fonction d'entrée du programme.

Paramètres

argc Nombre d'arguments du programme

argv Tableau contenant la liste des arguments du programme

Renvoie

Code d'erreur du programme

5.16 Référence du fichier src/parsing.c

Fonctions de parsing des arguments.

```
#include "parsing.h"
```

Fonctions

- bool [parsing_parseVerboseMode](#) (char **pTab, const int pSize)
Cherche si le mode verbeux à été spécifié ou non.
- char * [parsing_parseFileName](#) (char **pTab, const int pSize, [Errors](#) *pErrors)
Cherche le nom du fichier de l'Instance.
- void [parsing_algoType](#) (char **pTab, const int pSize, [Errors](#) *pErrors, [Algo](#) *algos)

5.16.1 Description détaillée

Fonctions de parsing des arguments.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 17 :17 :18

Implémentation des fonctions permettant de parser les arguments, et ainsi d'appeler les différents algorithmes demandés, d'utiliser le mode verbeux et de spécifier le fichier.

5.16.2 Documentation des fonctions

5.16.2.1 void [parsing_algoType](#) (char ** *pTab*, const int *pSize*, [Errors](#) * *pErrors*, [Algo](#) * *algos*)

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

algos Tableau d'algorithmes, ceci au cas où l'utilisateur entre plusieurs algorithmes. La fin du tableau est marqué par END

Voir également

[Algo](#)

5.16.2.2 char* [parsing_parseFileName](#) (char ** *pTab*, const int *pSize*, [Errors](#) * *pErrors*)

Cherche le nom du fichier de l'Instance.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

pErrors L'objet des erreurs, il est modifié si des erreurs interviennent

Renvoie

Le nom du fichier

Voir également

[Instance](#)

5.16.2.3 bool parsing_parseVerboseMode (char ** pTab, const int pSize)

Cherche si le mode verbeux à été spécifié ou non.

Paramètres

pTab Le tableau contenant les arguments

pSize Le nombre des arguments

Renvoie

Vrai si mode verbeux, Faux sinon

5.17 Référence du fichier src/tour.c

Fonctions des tournées.

```
#include "tour.h"
```

Fonctions

- [Tour](#) [tour_new](#) ([Instance](#) pInstance)
Créer une nouvelle tournée initialisée avec les données d'une instance.
- bool [tour_nextPermutation](#) ([Tour](#) *pPermutation)
Génère la permutation de ville suivante d'une tournée.
- void [tour_calculLength](#) ([Tour](#) *pTour)
Calcul la longueur d'une tournée.
- void [tour_display](#) (const [Tour](#) pTour)
Affiche une tournée.
- [Tour](#) [tour_randomWalk](#) (const [Instance](#) pInstance)
Génère une tournée aléatoire.
- void [tour_2opt](#) ([Tour](#) *pTour, int pFirst, int pSecond)
Fait une 2opt.

5.17.1 Description détaillée

Fonctions des tournées.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :04 :08

Implémentation des fonctions se rapportant à une tournée.

5.17.2 Documentation des fonctions

5.17.2.1 void tour_2opt (Tour * *pTour*, int *pFirst*, int *pSecond*)

Fait une 2opt.

Paramètres

pTour Le tour pour laquelle on veut faire une 2opt

pFirst L'id du début du premier trajet

pSecond L'id du début du second trajet

5.17.2.2 void tour_calculLength (Tour * *pTour*)

Calcul la longueur d'une tournée.

Paramètres

pTour La tournée pour laquelle on veut calculer la longueur

5.17.2.3 void tour_display (const Tour *pTour*)

Affiche une tournée.

Paramètres

pTour La tournée à afficher

5.17.2.4 Tour tour_new (Instance *pInstance*)

Créer une nouvelle tournée initialisée avec les données d'une instance.

Paramètres

pInstance [Instance](#) servant à initialiser la tournée

Renvoie

la nouvelle tournée

5.17.2.5 bool tour_nextPermutation (Tour * *pPermutation*)

Génère la permutation de ville suivante d'une tournée.

Paramètres

pPermutation La tournée pour laquelle la permutation doit être générée

Renvoie

Vrai si une permutation a été générée faux s'il ne reste plus de permutation.

5.17.2.6 Tour `tour_randomWalk` (`const Instance pInstance`)

Génère une tournée aléatoire.

Paramètres

pInstance L'instance pour laquelle générer un random walk

Renvoie

La tournée aléatoire

5.18 Référence du fichier src/town.c

Fonctions des villes.

```
#include "town.h"
```

Fonctions

- [Town `town_new`](#) (const int pId, const int pX, const int pY)
Création d'une nouvelle ville.

5.18.1 Description détaillée

Fonctions des villes.

Auteur

Antoine de Roquemaurel

Date

21/11/2012 22 :35 :14

implémentation des fonctions se rapportant à une ville.

5.18.2 Documentation des fonctions

5.18.2.1 Town `town_new` (`const int pId`, `const int pX`, `const int pY`)

Création d'une nouvelle ville.

Paramètres

pId Id de la ville

pX Abscisse

pY Ordonnée

5.19 Référence du fichier src/util.c

Fonctions utiles.

```
#include "tour.h"
#include "util.h"
```

Fonctions

- int [util_searchFirstOccurenceInArray](#) (char **pArray, const int pSize, char *pSearch)
Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaîne caractère.
- void [util_reverseArray](#) (Town *pTab, const int pBegin, const int pEnd)
Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.
- void [util_displayArray](#) (const int *pTab, const int pSize)
Affiche le contenu d'un tableau d'entiers.
- int [util_sum](#) (const int pBegin, const int pEnd)
Calcul la somme des éléments allant de pBegin à pEnd.
- void [util_swap](#) (int *a, int *b)
Échange deux variables.
- int [util_rand](#) (const int pMin, const int pMax)
Calcul une valeur aléatoire entre pMin et pMax.
- bool [util_sousTabExist](#) (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecursvite)
Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

5.19.1 Description détaillée

Fonctions utiles.

Auteur

Antoine de Roquemaurel

Date

19/11/2012 16 :27 :39

Entêtes des fonctions pouvant être utiles dans tout le projet. Ce sont des fonctions simples, qui doivent être indépendantes du projet.

5.19.2 Documentation des fonctions

5.19.2.1 void [util_displayArray](#) (const int * *pTab*, const int *pSize*)

Affiche le contenu d'un tableau d'entiers.

Paramètres

pTab Le tableau à afficher

pSize La taille du tableau

5.19.2.2 int [util_rand](#) (const int *pMin*, const int *pMax*)

Calcul une valeur aléatoire entre pMin et pMax.

Paramètres

pMin Minimum

pMax Maximum

Renvoie**5.19.2.3 void util_reverseArray (Town * pTab, const int pBegin, const int pEnd)**

Inverse les éléments d'un tableau pTab entre les cases pBegin et pEnd.

Paramètres

pTab Tableau à inverser

pBegin Début de la section à inverser

pEnd Fin de la section à inverser

5.19.2.4 int util_searchFirstOccurenceInArray (char ** pArray, const int pSize, char * pSearch)

Cherche la première occurrence d'une chaîne de caractère dans un tableau de chaînes de caractères.

Paramètres

pArray Le tableau de chaînes de caractères dans lequel chercher

pSize La taille du tableau

pSearch La chaîne de caractère à chercher

Renvoie

La position de la chaîne dans le tableau ou -1 si elle n'a pas été trouvée

5.19.2.5 bool util_sousTabExist (Tour pChild, const int pBegin, const int pEnd, Tour pParent, bool pRecurvite)

Indique si un sous-tableau d'un tableau est présent dans un autre tableau.

Paramètres

pChild Le tableau à chercher

pBegin Le début de la section à chercher

pEnd La fin de la section à chercher

pParent Le tableau dans lequel chercher

pRecurvite Pour simplifier vis-à-vis de la récursivité, doit toujours être à false

Renvoie

Vrai si le sous-tableau a été trouvé faux sinon

5.19.2.6 int util_sum (const int *pBegin*, const int *pEnd*)

Calcul la somme des éléments allant de pBegin à pEnd.

Paramètres

pBegin Début de la somme

pEnd Fin de la somme

Renvoie

La somme des éléments

5.19.2.7 void util_swap (int * *a*, int * *b*)

Échange deux variables.

Paramètres

a Première variable à échanger

b Seconde variable

Index

- Algo, [7](#)
- AlgoType
 - [parsing.h](#), [20](#)
- BRUTEFORCE
 - [parsing.h](#), [20](#)
- bruteForce.c
 - [bruteForce_bestPath](#), [28](#)
- bruteForce.h
 - [bruteForce_bestPath](#), [11](#)
- bruteForce_bestPath
 - [bruteForce.c](#), [28](#)
 - [bruteForce.h](#), [11](#)
- Distance, [7](#)
- distance.c
 - [distance_betweenTowns](#), [29](#)
 - [distance_calculDistance](#), [29](#)
 - [distance_new](#), [29](#)
 - [distance_searchDistance](#), [29](#)
- distance.h
 - [distance_betweenTowns](#), [12](#)
 - [distance_calculDistance](#), [13](#)
 - [distance_new](#), [13](#)
 - [distance_searchDistance](#), [13](#)
- distance_betweenTowns
 - [distance.c](#), [29](#)
 - [distance.h](#), [12](#)
- distance_calculDistance
 - [distance.c](#), [29](#)
 - [distance.h](#), [13](#)
- distance_new
 - [distance.c](#), [29](#)
 - [distance.h](#), [13](#)
- distance_searchDistance
 - [distance.c](#), [29](#)
 - [distance.h](#), [13](#)
- END
 - [parsing.h](#), [20](#)
- Errors, [8](#)
- errors.h
 - [errors_displayErrorsMessage](#), [15](#)
 - [errors_new](#), [15](#)
 - [errors_setFileNotFound](#), [15](#)
 - [errors_setMissingParameterGa](#), [15](#)
 - [errors_setMissingParameterLsnr](#), [15](#)
 - [errors_setMissingParameterLsr](#), [15](#)
 - [errors_setNbArguments](#), [15](#)
 - [errors_setNoAlgoSpecified](#), [16](#)
 - [errors_setNoValidParameterLsnr](#), [16](#)
 - [errors_setNoValidParameterLsr](#), [16](#)
 - [errors_setTagFNotFound](#), [16](#)
- errors_displayErrorsMessage
 - [errors.h](#), [15](#)
- errors_new
 - [errors.h](#), [15](#)
- errors_setFileNotFound
 - [errors.h](#), [15](#)
- errors_setMissingParameterGa
 - [errors.h](#), [15](#)
- errors_setMissingParameterLsnr
 - [errors.h](#), [15](#)
- errors_setMissingParameterLsr
 - [errors.h](#), [15](#)
- errors_setNbArguments
 - [errors.h](#), [15](#)
- errors_setNoAlgoSpecified
 - [errors.h](#), [16](#)
- errors_setNoValidParameterLsnr
 - [errors.h](#), [16](#)
- errors_setNoValidParameterLsr
 - [errors.h](#), [16](#)
- errors_setTagFNotFound
 - [errors.h](#), [16](#)
- GENETIC
 - [parsing.h](#), [20](#)
- Instance, [9](#)
- instance.c
 - [instance_display](#), [31](#)
 - [instance_displayLinearVector](#), [31](#)
 - [instance_displayMatrix](#), [31](#)
 - [instance_initializeDistancesMatrix](#), [31](#)
 - [instance_new](#), [32](#)
 - [instance_push](#), [32](#)
- instance.h
 - [instance_display](#), [17](#)
 - [instance_displayLinearVector](#), [17](#)

- instance_displayMatrix, 18
- instance_initializeDistancesMatrix, 18
- instance_new, 18
- instance_push, 18
- instance_display
 - instance.c, 31
 - instance.h, 17
- instance_displayLinearVector
 - instance.c, 31
 - instance.h, 17
- instance_displayMatrix
 - instance.c, 31
 - instance.h, 18
- instance_initializeDistancesMatrix
 - instance.c, 31
 - instance.h, 18
- instance_new
 - instance.c, 32
 - instance.h, 18
- instance_push
 - instance.c, 32
 - instance.h, 18
- lib/bruteForce.h, 11
- lib/distance.h, 12
- lib/errors.h, 13
- lib/instance.h, 16
- lib/localSearch.h, 19
- lib/parsing.h, 19
- lib/tour.h, 21
- lib/town.h, 23
- lib/util.h, 24
- LOCALSEARCH_RANDOM
 - parsing.h, 20
- LOCALSEARCH_SYSTEMATIC
 - parsing.h, 20
- main
 - main.c, 33
- main.c
 - main, 33
- parsing.c
 - parsing_algoType, 34
 - parsing_parseFileName, 34
 - parsing_parseVerboseMode, 34
- parsing.h
 - AlgoType, 20
 - BRUTEFORCE, 20
 - END, 20
 - GENETIC, 20
 - LOCALSEARCH_RANDOM, 20
 - LOCALSEARCH_SYSTEMATIC, 20
 - parsing_algoType, 20
 - parsing_parseFileName, 21
 - parsing_parseVerboseMode, 21
- parsing_algoType
 - parsing.c, 34
 - parsing.h, 20
- parsing_parseFileName
 - parsing.c, 34
 - parsing.h, 21
- parsing_parseVerboseMode
 - parsing.c, 34
 - parsing.h, 21
- src/bruteForce.c, 27
- src/distance.c, 28
- src/genetic.c, 30
- src/instance.c, 30
- src/localSearch.c, 32
- src/main.c, 33
- src/parsing.c, 33
- src/tour.c, 35
- src/town.c, 37
- src/util.c, 37
- Tour, 10
- tour.c
 - tour_2opt, 36
 - tour_calculLength, 36
 - tour_display, 36
 - tour_new, 36
 - tour_nextPermutation, 36
 - tour_randomWalk, 36
- tour.h
 - tour_2opt, 22
 - tour_calculLength, 22
 - tour_display, 23
 - tour_new, 23
 - tour_nextPermutation, 23
 - tour_randomWalk, 23
- tour_2opt
 - tour.c, 36
 - tour.h, 22
- tour_calculLength
 - tour.c, 36
 - tour.h, 22
- tour_display
 - tour.c, 36
 - tour.h, 23
- tour_new
 - tour.c, 36
 - tour.h, 23
- tour_nextPermutation
 - tour.c, 36
 - tour.h, 23
- tour_randomWalk

- tour.c, [36](#)
 - tour.h, [23](#)
- Town, [10](#)
- town.c
 - town_new, [37](#)
- town.h
 - town_new, [24](#)
- town_new
 - town.c, [37](#)
 - town.h, [24](#)
- util.c
 - util_displayArray, [38](#)
 - util_rand, [38](#)
 - util_reverseArray, [39](#)
 - util_searchFirstOccurenceInArray, [39](#)
 - util_sousTabExist, [39](#)
 - util_sum, [39](#)
 - util_swap, [40](#)
- util.h
 - util_displayArray, [25](#)
 - util_rand, [25](#)
 - util_reverseArray, [26](#)
 - util_searchFirstOccurenceInArray, [26](#)
 - util_sousTabExist, [26](#)
 - util_sum, [26](#)
 - util_swap, [27](#)
- util_displayArray
 - util.c, [38](#)
 - util.h, [25](#)
- util_rand
 - util.c, [38](#)
 - util.h, [25](#)
- util_reverseArray
 - util.c, [39](#)
 - util.h, [26](#)
- util_searchFirstOccurenceInArray
 - util.c, [39](#)
 - util.h, [26](#)
- util_sousTabExist
 - util.c, [39](#)
 - util.h, [26](#)
- util_sum
 - util.c, [39](#)
 - util.h, [26](#)
- util_swap
 - util.c, [40](#)
 - util.h, [27](#)