

TSP Groupe 1

1

Généré par Doxygen 1.9.8

1 Documentation des structures de données	1
1.1 Référence de la structure City	1
1.1.1 Documentation des champs	1
1.2 Référence de la structure Infos	1
1.2.1 Documentation des champs	2
1.3 Référence de la structure Matrix	2
1.3.1 Documentation des champs	2
1.4 Référence de la structure Results	3
1.4.1 Documentation des champs	3
2 Documentation des fichiers	3
2.1 Référence du fichier genetique/ga.c	3
2.1.1 Documentation des fonctions	4
2.2 Référence du fichier genetique/ga.h	6
2.2.1 Documentation des fonctions	6
2.3 ga.h	6
2.4 Référence du fichier heuristiques/nearestneighbor.c	7
2.4.1 Documentation des fonctions	7
2.5 Référence du fichier heuristiques/nearestneighbor.h	7
2.5.1 Documentation des fonctions	8
2.6 nearestneighbor.h	8
2.7 Référence du fichier heuristiques/randomwalk.c	8
2.7.1 Documentation des fonctions	9
2.8 Référence du fichier heuristiques/randomwalk.h	10
2.8.1 Documentation des fonctions	10
2.9 randomwalk.h	11
2.10 Référence du fichier main.c	11
2.10.1 Documentation des fonctions	12
2.10.2 Documentation des variables	13
2.11 Référence du fichier tad/city.c	13
2.11.1 Documentation des fonctions	13
2.12 Référence du fichier tad/city.h	14
2.12.1 Documentation des fonctions	14
2.13 city.h	14
2.14 Référence du fichier tad/matrix.c	14
2.14.1 Documentation des fonctions	15
2.15 Référence du fichier tad/matrix.h	15
2.15.1 Documentation des fonctions	16
2.16 matrix.h	16
2.17 Référence du fichier tad/tsp.c	17
2.17.1 Documentation des macros	17
2.17.2 Documentation des définitions de type	18

2.17.3 Documentation des fonctions	18
2.18 Référence du fichier tad/tsp.h	19
2.18.1 Documentation des fonctions	20
2.19 tsp.h	21
Index	23

1 Documentation des structures de données

1.1 Référence de la structure City

```
#include <city.h>
```

Champs de données

- double **x**
- double **y**
- int **id**

1.1.1 Documentation des champs

id

```
int City::id
```

x

```
double City::x
```

y

```
double City::y
```

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

- tad/[city.h](#)

1.2 Référence de la structure Infos

```
#include <tsp.h>
```

Champs de données

- int `dimension`
- `City ** cityArray`
- char `edgeType [12]`

1.2.1 Documentation des champs

`cityArray`

```
City ** Infos::cityArray
```

`dimension`

```
int Infos::dimension
```

`edgeType`

```
char Infos::edgeType
```

La documentation de cette structure a été générée à partir des fichiers suivants :

- tad/[tsp.c](#)
- tad/[tsp.h](#)

1.3 Référence de la structure Matrix

```
#include <matrix.h>
```

Champs de données

- int `dimension`
- double ** `matrix`

1.3.1 Documentation des champs

`dimension`

```
int Matrix::dimension
```

matrix

```
double** Matrix::matrix
```

La documentation de cette structure a été générée à partir du fichier suivant :
— tad/[matrix.h](#)

1.4 Référence de la structure Results

```
#include <matrix.h>
```

Champs de données

- int [dimension](#)
- double [bestDistance](#)
- int * [bestPath](#)

1.4.1 Documentation des champs

bestDistance

```
double Results::bestDistance
```

bestPath

```
int* Results::bestPath
```

dimension

```
int Results::dimension
```

La documentation de cette structure a été générée à partir du fichier suivant :
— tad/[matrix.h](#)

2 Documentation des fichiers

2.1 Référence du fichier genetique/ga.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <stdbool.h>
#include <float.h>
#include "../tad/city.h"
#include "../tad/matrix.h"
#include "../tad/tsp.h"
#include "../heuristiques/randomwalk.h"
#include "ga.h"
```

Fonctions

- void **mutate** (int *path, int n)
- bool **isEdgeInTour** (int u, int v, int *tour, int n)
- void **applyTwoOpt** (int *path, int n, **Matrix** *m)
- int * **crossover** (int *parent1, int *parent2, int n)

Effectue un croisement standard.

- int * **crossoverDPX** (int *parent1, int *parent2, int n, **Matrix** *m)

Croisement DPX.

- **Results** * **geneticAlgorithm** (**Matrix** *m, int pSize, int maxGen, double mutRate, int crossCount, bool useDPX)

Algorithme Génétique Principal.

2.1.1 Documentation des fonctions

applyTwoOpt()

```
void applyTwoOpt (
    int * path,
    int n,
    Matrix * m )
```

crossover()

```
int * crossover (
    int * parent1,
    int * parent2,
    int n )
```

Effectue un croisement standard.

Paramètres

<i>parent1</i>	Tableau du premier parent.
<i>parent2</i>	Tableau du second parent.
<i>n</i>	Dimension.

Renvoie

- Numéro de l'enfant.

crossoverDPX()

```
int * crossoverDPX (
    int * parent1,
```

```
int * parent2,
int n,
Matrix * m )
```

Croisement DPX.

Paramètres

<i>parent1</i>	Tableau du premier parent.
<i>parent2</i>	Tableau du second parent.
<i>n</i>	Dimension.
<i>m</i>	Matrice des distances.

Renvoie

- int* Nouvelle tournée.
- @complexity $O(N^2)$.

geneticAlgorithm()

```
Results * geneticAlgorithm (
    Matrix * m,
    int pSize,
    int maxGen,
    double mutRate,
    int crossCount,
    bool useDPX )
```

Algorithme Génétique Principal.

Paramètres

<i>m</i>	Matrice des distances.
<i>pSize</i>	Taille de la population.
<i>maxGen</i>	Nombre maximum de générations.
<i>mutRate</i>	Taux de mutation.
<i>crossCount</i>	Nombre de croisements par génération.
<i>useDPX</i>	Booléen pour activer le mode DPX optimisé.

Renvoie

- Results* La meilleure solution trouvée.

isEdgeInTour()

```
bool isEdgeInTour (
    int u,
    int v,
    int * tour,
    int n )
```

mutate()

```
void mutate (
    int * path,
    int n )
```

2.2 Référence du fichier genetique/ga.h

```
#include "../tad/matrix.h"
#include "../tad/tsp.h"
```

Fonctions

- [Results * geneticAlgorithm \(Matrix *m, int pSize, int maxGen, double mutRate, int crossCount, bool useDPX\)](#)

Algorithme Génétique Principal.

2.2.1 Documentation des fonctions

geneticAlgorithm()

```
Results * geneticAlgorithm (
    Matrix * m,
    int pSize,
    int maxGen,
    double mutRate,
    int crossCount,
    bool useDPX )
```

Algorithme Génétique Principal.

Paramètres

<i>m</i>	Matrice des distances.
<i>pSize</i>	Taille de la population.
<i>maxGen</i>	Nombre maximum de générations.
<i>mutRate</i>	Taux de mutation.
<i>crossCount</i>	Nombre de croisements par génération.
<i>useDPX</i>	Booléen pour activer le mode DPX optimisé.

Renvoie

- [Results* La meilleure solution trouvée.](#)

2.3 ga.h

[Aller à la documentation de ce fichier.](#)

```

00001 #ifndef GA_H
00002 #define GA_H
00003
00004 #include "../tad/matrix.h"
00005 #include "../tad/tsp.h"
00006
00007 // Fonction principale à appeler depuis le main
00008 Results* geneticAlgorithm(Matrix* m, int pSize, int maxGen, double mutRate, int crossCount, bool
useDPX);
00009
00010 #endif

```

2.4 Référence du fichier heuristiques/nearestneighbor.c

```

#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>
#include <signal.h>
#include "../tad/matrix.h"
#include "nearestneighbor.h"
#include <float.h>

```

Fonctions

- **Results * nearestNeighbour (Matrix *m, int k)**

Resolution du TSP par l'heuristique du Plus Proche Voisin.

2.4.1 Documentation des fonctions

nearestNeighbour()

```

Results * nearestNeighbour (
    Matrix * m,
    int k )

```

Resolution du TSP par l'heuristique du Plus Proche Voisin.

Paramètres

<i>m</i>	Pointeur vers la matrice des distances.
<i>k</i>	Indice de la ville de départ.

Renvoie

Results* Structure contenant le chemin trouvé et sa distance.

- @complexity O(N²).

2.5 Référence du fichier heuristiques/nearestneighbor.h

```

#include "../tad/matrix.h"
#include <stdlib.h>

```

```
#include <string.h>
```

Fonctions

- **Results * nearestNeighbour (Matrix *m, int k)**

Resolution du TSP par l'heuristique du Plus Proche Voisin.

2.5.1 Documentation des fonctions

nearestNeighbour()

```
Results * nearestNeighbour (
    Matrix * m,
    int k )
```

Resolution du TSP par l'heuristique du Plus Proche Voisin.

Paramètres

<i>m</i>	Pointeur vers la matrice des distances.
<i>k</i>	Indice de la ville de départ.

Renvoie

Results* Structure contenant le chemin trouvé et sa distance.

- @complexity $O(N^2)$.

2.6 nearestneighbor.h

[Aller à la documentation de ce fichier.](#)

```
00001 #ifndef NEARESTNEIGHBOR_H
00002 #define NEARESTNEIGHBOR_H
00003
00004 #include "../tad/matrix.h"
00005 #include <stdlib.h>
00006 #include <string.h>
00007
00008 Results* nearestNeighbour(Matrix* m,int k);
00009
0010 #endif
```

2.7 Référence du fichier heuristiques/randomwalk.c

```
#include <stdlib.h>
#include <time.h>
#include "../tad/matrix.h"
#include "../tad/tsp.h"
#include "randomwalk.h"
#include <stdbool.h>
#include <float.h>
```

Fonctions

- double **totalPathDistance2** (**Matrix** *m, const int *chemin, int n)
- **Results** * **randomWalk** (**Matrix** *m)

Resolution du TSP par l'heuristique de Random Walk.

- **Results** * **twoOptrw** (**Matrix** *mat, **Results** *res)

Resolution du TSP par 2-OPT du RW.

2.7.1 Documentation des fonctions

randomWalk()

```
Results * randomWalk (
    Matrix * m )
```

Resolution du TSP par l'heuristique de Random Walk.

Paramètres

<i>m</i>	Pointeur vers la matrice des distances.
----------	---

Renvoie

- **Results*** la tournée trouvé et sa distance associée.
- @complexity O(N).

totalPathDistance2()

```
double totalPathDistance2 (
    Matrix * m,
    const int * chemin,
    int n )
```

twoOptrw()

```
Results * twoOptrw (
    Matrix * mat,
    Results * res )
```

Resolution du TSP par 2-OPT du RW.

Paramètres

<i>mat</i>	Pointeur vers la matrice des distances.
<i>res</i>	Structure contenant la tournée à optimiser.

Renvoie

- Results* tournée optimisé avec sa distance .
- @complexity O(N²).

2.8 Référence du fichier heuristiques/randomwalk.h

```
#include <stdlib.h>
#include <string.h>
#include "../tad/matrixx.h"
```

Fonctions

— Results * randomWalk (Matrix *m)

Resolution du TSP par l'heuristique de Random Walk.

— Results * twoOptrw (Matrix *mat, Results *res)

Resolution du TSP par 2-OPT du RW.

— double totalPathDistance2 (Matrix *m, const int *chemin, int n)

2.8.1 Documentation des fonctions

randomWalk()

```
Results * randomWalk (
    Matrix * m )
```

Resolution du TSP par l'heuristique de Random Walk.

Paramètres

<i>m</i>	Pointeur vers la matrice des distances.
----------	---

Renvoie

- Results* la tournée trouvé et sa distance associée.
- @complexity O(N).

totalPathDistance2()

```
double totalPathDistance2 (
    Matrix * m,
    const int * chemin,
    int n )
```

twoOptrw()

```
Results * twoOptrw (
    Matrix * mat,
    Results * res )
```

Resolution du TSP par 2-OPT du RW.

Paramètres

<i>mat</i>	Pointeur vers la matrice des distances.
<i>res</i>	Structure contenant la tournée à optimiser.

Renvoie

- *Results** tournée optimisé avec sa distance .
- @complexity $O(N^2)$.

2.9 randomwalk.h

[Aller à la documentation de ce fichier.](#)

```
00001 #ifndef RANDOMWALK_H
00002 #define RANDOMWALK_H
00003 #include <stdlib.h>
00004 #include <string.h>
00005 #include "../tad/matrix.h"
00006
00007
00008
00009 /* Renvoie une tournée aléatoire et on calcule longueur en utilisant la matrice */
00010 Results* randomWalk(Matrix* m);
00011 Results* twoOptrw(Matrix* mat, Results* res);
00012 double totalPathDistance2(Matrix *m, const int *chemin, int n);
00013
00014 #endif
```

2.10 Référence du fichier main.c

```
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <limits.h>
#include <signal.h>
#include <time.h>
#include <getopt.h>
#include "tad/city.h"
#include "tad/matrix.h"
#include "tad/tsp.h"
#include "heuristiques/nearestneighbor.h"
#include "heuristiques/randomwalk.h"
#include "brutforce/brutforce.h"
#include "genetique/ga.h"
```

Fonctions

- void **INThandler** (int sig)
- void **printOutputFile** (char *file_name, **Results** *results, **Infos** *infos, char *fn, char *method, double cpu_time_used)
- void **printOutput** (**Results** *results, **Infos** *infos, char *fn, char *method, double cpu_time_used)
- int **main** (int argc, char *argv[])

Variables

- char * **methode**
- double **temps_cpu**
- double **longueur**
- int **ij**

2.10.1 Documentation des fonctions

INThandler()

```
void INThandler (
    int sig )
```

main()

```
int main (
    int argc,
    char * argv[ ] )
```

printOutput()

```
void printOutput (
    Results * results,
    Infos * infos,
    char * fn,
    char * method,
    double cpu_time_used )
```

printOutputFile()

```
void printOutputFile (
    char * file_name,
    Results * results,
    Infos * infos,
    char * fn,
    char * method,
    double cpu_time_used )
```

2.10.2 Documentation des variables

ij

```
int ij
```

longueur

```
double longueur
```

methode

```
char* methode
```

temps_cpu

```
double temps_cpu
```

2.11 Référence du fichier tad/city.c

```
#include "city.h"
#include <stdio.h>
#include <stdlib.h>
```

Fonctions

- `City * createCity (int id, double xCoor, double yCoor)`
- `void freeCity (City *city)`

2.11.1 Documentation des fonctions

createCity()

```
City * createCity (
    int id,
    double xCoor,
    double yCoor )
```

freeCity()

```
void freeCity (
    City * city )
```

2.12 Référence du fichier tad/city.h

Structures de données

- struct `City`

Fonctions

- `City * createCity (int id, double xCoor, double yCoor)`
- `void freeCity (City *city)`

2.12.1 Documentation des fonctions

`createCity()`

```
City * createCity (
    int id,
    double xCoor,
    double yCoor )
```

`freeCity()`

```
void freeCity (
    City * city )
```

2.13 city.h

[Aller à la documentation de ce fichier.](#)

```
00001 #ifndef CITY_H
00002 #define CITY_H
00003
00004 typedef struct {
00005     double x;
00006     double y; // coordonnées GPS
00007     int id;
00008 }City ;
00009
00010 City* createCity(int id, double xCoor, double yCoor);
00011 void freeCity(City* city);
00012 #endif //CITY_H
```

2.14 Référence du fichier tad/matrix.c

```
#include "matrix.h"
#include "city.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

Fonctions

- `Matrix * MatrixCreate (int dimension)`
- `void freeMatrix (Matrix *m)`
- `void setDistance (Matrix *m, int i, int j, double distance)`
- `double getDistance (Matrix *m, int i, int j)`

2.14.1 Documentation des fonctions

`freeMatrix()`

```
void freeMatrix (
    Matrix * m )
```

`getDistance()`

```
double getDistance (
    Matrix * m,
    int i,
    int j )
```

`MatrixCreate()`

```
Matrix * MatrixCreate (
    int dimension )
```

`setDistance()`

```
void setDistance (
    Matrix * m,
    int i,
    int j,
    double distance )
```

2.15 Référence du fichier tad/matrix.h

```
#include "city.h"
```

Structures de données

- `struct Matrix`
- `struct Results`

Fonctions

- `Matrix * MatrixCreate (int dimension)`
- `void freeMatrix (Matrix *m)`
- `void setDistance (Matrix *m, int i, int j, double distance)`
- `double getDistance (Matrix *m, int i, int j)`

2.15.1 Documentation des fonctions

freeMatrix()

```
void freeMatrix (
    Matrix * m )
```

getDistance()

```
double getDistance (
    Matrix * m,
    int i,
    int j )
```

MatrixCreate()

```
Matrix * MatrixCreate (
    int dimension )
```

setDistance()

```
void setDistance (
    Matrix * m,
    int i,
    int j,
    double distance )
```

2.16 matrix.h

[Aller à la documentation de ce fichier.](#)

```
00001 #ifndef MATRIX_H
00002 #define MATRIX_H
00003
00004 #include "city.h"
00005
00006 typedef struct {
00007     int dimension;
00008     double **matrix;
00009 } Matrix;
00010
00011 typedef struct {
00012     int dimension;
00013     double bestDistance;
00014     int *bestPath;
00015 } Results;
00016
00017
00018 Matrix* MatrixCreate(int dimension);
00019 void freeMatrix(Matrix* m);
00020 void setDistance(Matrix* m, int i, int j, double distance);
00021 double getDistance(Matrix* m, int i, int j);
00022
00023 #endif //MATRIX_H
```

2.17 Référence du fichier tad/tsp.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "city.h"
#include "matrix.h"
```

Structures de données

— struct **Infos**

Macros

```
— #define _USE_MATH_DEFINES
— #define RRR 6378.388
— #define M_PI 3.1415926535
```

Définitions de type

— typedef double(* **fctd**) (**City** *, **City** *)

Fonctions

```
— double latitude (City *city)
— double longitude (City *city)
— double distanceGeo (City *cityA, City *cityB)
— double distanceEucl (City *cityA, City *cityB)
— double distanceAtt (City *cityA, City *cityB)
— void printMatrix (Matrix *m)
— double canonicalTourLength (Matrix *m)
— Infos * readTsp (FILE *f)
— Matrix * distanceMatrix (Infos *infos, fctd fctd)
```

2.17.1 Documentation des macros

_USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

M_PI

```
#define M_PI 3.1415926535
```

RRR

```
#define RRR 6378.388
```

2.17.2 Documentation des définitions de type**fctd**

```
typedef double(* fctd) (City *, City *)
```

2.17.3 Documentation des fonctions**canonicalTourLength()**

```
double canonicalTourLength (
    Matrix * m )
```

distanceAtt()

```
double distanceAtt (
    City * cityA,
    City * cityB )
```

distanceEucl()

```
double distanceEucl (
    City * cityA,
    City * cityB )
```

distanceGeo()

```
double distanceGeo (
    City * cityA,
    City * cityB )
```

distanceMatrix()

```
Matrix * distanceMatrix (
    Infos * infos,
    fctd fctd )
```

latitude()

```
double latitude (
    City * city )
```

longitude()

```
double longitude (
    City * city )
```

printMatrix()

```
void printMatrix (
    Matrix * m )
```

readTsp()

```
Infos * readTsp (
    FILE * f )
```

2.18 Référence du fichier tad/tsp.h

```
#include "city.h"
#include "matrix.h"
#include <stdio.h>
```

Structures de données

- struct Infos

Fonctions

- Infos * readTsp (FILE *f)
- Matrix * distanceMatrix (Infos *infos, double(*fctd)(City *, City *))
- void printMatrix (Matrix *m)
- double distanceAtt (City *cityA, City *cityB)
- double distanceEucl (City *cityA, City *cityB)
- double distanceGeo (City *cityA, City *cityB)
- double canonicalTourLength (Matrix *m)
- double latitude (City *city)
- double longitude (City *city)

2.18.1 Documentation des fonctions

canonicalTourLength()

```
double canonicalTourLength (
    Matrix * m )
```

distanceAtt()

```
double distanceAtt (
    City * cityA,
    City * cityB )
```

distanceEucl()

```
double distanceEucl (
    City * cityA,
    City * cityB )
```

distanceGeo()

```
double distanceGeo (
    City * cityA,
    City * cityB )
```

distanceMatrix()

```
Matrix * distanceMatrix (
    Infos * infos,
    double(*) (City *, City *) fctd )
```

latitude()

```
double latitude (
    City * city )
```

longitude()

```
double longitude (
    City * city )
```

printMatrix()

```
void printMatrix (
    Matrix * m )
```

readTsp()

```
Infos * readTsp (
    FILE * f )
```

2.19 tsp.h

[Aller à la documentation de ce fichier.](#)

```
00001 #ifndef TSP_H
00002 #define TSP_H
00003
00004 #include "city.h"
00005 #include "matrix.h"
00006 #include <stdio.h>
00007
00008 /* Structure servant au retour et au stockage des informations lors de la lecture */
00009 typedef struct {
00010     int dimension;
00011     City** cityArray;
00012     char edgeType[12]; // Type de distance (EUCL_2D, ATT, GEO, etc.)
00013 } Infos;
00014
00015 /* Fonctions de lecture et de traitement */
00016 Infos* readTsp(FILE *f);
00017 Matrix* distanceMatrix(Infos* infos, double (*fctd)(City*, City*));
00018 void printMatrix(Matrix *m);
00019
00020 /* Fonctions de distance */
00021 double distanceAtt(City* cityA, City* cityB);
00022 double distanceEucl(City* cityA, City* cityB);
00023 double distanceGeo(City* cityA, City* cityB);
00024 double canonicalTourLength(Matrix* m);
00025
00026 /* Fonctions utilitaires pour le mode GEO */
00027 double latitude(City* city);
00028 double longitude(City* city);
00029
00030 #endif
```


Index

_USE_MATH_DEFINES
 tsp.c, 17

applyTwoOpt
 ga.c, 4

bestDistance
 Results, 3

bestPath
 Results, 3

canonicalTourLength
 tsp.c, 18
 tsp.h, 20

City, 1
 id, 1
 x, 1
 y, 1

city.c
 createCity, 13
 freeCity, 13

city.h
 createCity, 14
 freeCity, 14

cityArray
 Infos, 2

createCity
 city.c, 13
 city.h, 14

crossover
 ga.c, 4

crossoverDPX
 ga.c, 4

dimension
 Infos, 2
 Matrix, 2
 Results, 3

distanceAtt
 tsp.c, 18
 tsp.h, 20

distanceEucl
 tsp.c, 18
 tsp.h, 20

distanceGeo
 tsp.c, 18
 tsp.h, 20

distanceMatrix
 tsp.c, 18
 tsp.h, 20

edgeType
 Infos, 2

fctd
 tsp.c, 18

freeCity

 city.c, 13
 city.h, 14

 matrix.c, 15
 matrix.h, 16

 ga.c
 applyTwoOpt, 4
 crossover, 4
 crossoverDPX, 4
 geneticAlgorithm, 5
 isEdgeInTour, 5
 mutate, 5

 ga.h
 geneticAlgorithm, 6

geneticAlgorithm
 ga.c, 5
 ga.h, 6

genetique/ga.c, 3

genetique/ga.h, 6

getDistance
 matrix.c, 15
 matrix.h, 16

heuristiques/nearestneighbor.c, 7

heuristiques/nearestneighbor.h, 7, 8

heuristiques/randomwalk.c, 8

heuristiques/randomwalk.h, 10, 11

id
 City, 1

ij
 main.c, 13

Infos, 1
 cityArray, 2
 dimension, 2
 edgeType, 2

INTHandler
 main.c, 12

isEdgeInTour
 ga.c, 5

latitude
 tsp.c, 18
 tsp.h, 20

longitude
 tsp.c, 19
 tsp.h, 20

longueur
 main.c, 13

M_PI
 tsp.c, 17

main
 main.c, 12

 main.c, 11

ij, 13
 INTHandler, 12
 longueur, 13
 main, 12
 methode, 13
 printOutput, 12
 printOutputFile, 12
 temps_cpu, 13
 Matrix, 2
 dimension, 2
 matrix, 2
 matrix
 Matrix, 2
 matrix.c
 freeMatrix, 15
 getDistance, 15
 MatrixCreate, 15
 setDistance, 15
 matrix.h
 freeMatrix, 16
 getDistance, 16
 MatrixCreate, 16
 setDistance, 16
 MatrixCreate
 matrix.c, 15
 matrix.h, 16
 methode
 main.c, 13
 mutate
 ga.c, 5
 nearestneighbor.c
 nearestNeighbour, 7
 nearestneighbor.h
 nearestNeighbour, 8
 nearestNeighbour
 nearestneighbor.c, 7
 nearestneighbor.h, 8
 printMatrix
 tsp.c, 19
 tsp.h, 20
 printOutput
 main.c, 12
 printOutputFile
 main.c, 12
 randomWalk
 randomwalk.c, 9
 randomwalk.h, 10
 randomwalk.c
 randomWalk, 9
 totalPathDistance2, 9
 twoOptrw, 9
 randomwalk.h
 randomWalk, 10
 totalPathDistance2, 10
 twoOptrw, 10
 readTsp

tsp.c, 19
 tsp.h, 20
 Results, 3
 bestDistance, 3
 bestPath, 3
 dimension, 3
 RRR
 tsp.c, 18
 setDistance
 matrix.c, 15
 matrix.h, 16
 tad/city.c, 13
 tad/city.h, 14
 tad/matrix.c, 14
 tad/matrix.h, 15, 16
 tad/tsp.c, 17
 tad/tsp.h, 19, 21
 temps_cpu
 main.c, 13
 totalPathDistance2
 randomwalk.c, 9
 randomwalk.h, 10
 tsp.c
 _USE_MATH_DEFINES, 17
 canonicalTourLength, 18
 distanceAtt, 18
 distanceEucl, 18
 distanceGeo, 18
 distanceMatrix, 18
 fctd, 18
 latitude, 18
 longitude, 19
 M_PI, 17
 printMatrix, 19
 readTsp, 19
 RRR, 18
 tsp.h
 canonicalTourLength, 20
 distanceAtt, 20
 distanceEucl, 20
 distanceGeo, 20
 distanceMatrix, 20
 latitude, 20
 longitude, 20
 printMatrix, 20
 readTsp, 20
 twoOptrw
 randomwalk.c, 9
 randomwalk.h, 10

x
 City, 1

y
 City, 1