TP Algo2 – session 7 – Algorithmes élémentaires sur les graphes

23 mai 2025

Objectif d'apprentissage : implémenter la structure de graphe par listes d'adjacences et les parcours en largeur et en profondeur. Comme on considère des graphes non orientés uniquement, il faudra bien penser à conserver la symétrie dans les listes d'adjacences. Il faudra utiliser le code suivant pour les définitions et implémenter chaque méthode :

```c
#ifndef GRAPH_H
#define GRAPH_H

#include <stddef.h>

enum color
{ WHITE, GRAY, BLACK };

typedef struct sgraph_vertex
{
  size_t degree;
  struct sgraph_vertex **neighbors;
  // Data for searching procedures
  enum color color;
  int d;
  int f;
  struct sgraph_vertex *parent;
} graph_vertex;

typedef struct
{
  size_t size;
  graph_vertex **vertices;
} graph;

/*
 * Create an empty graph
 */
void graph_create (graph * self);

/*
 * Destroy a graph
 */
void graph_destroy (graph * self);

/*
 * Get the number of vertices in the graph
 */
size_t graph_size (const graph * self);
```

```c
/*
 * Create a vertex, add it in the graph and return it
 */
graph_vertex *graph_add_vertex (graph * self);

/*
 * Add an edge between two vertices (does not check if edge already exists)
 */
void graph_add_edge (graph_vertex * source, graph_vertex * destination);

/*
 * Run breadth-first search to set color, d and parent starting with
 * the given source
 */
void graph_bfs (const graph * self, graph_vertex * source);

/*
 * Run depth-first search to set color, d, f and parent
 */
void graph_dfs (const graph * self);

#endif // GRAPH_H
```

Comme précédemment, lorsqu'une fonction ne peut par terminer son exécution (valeur absente ou argument à `NULL`), la fonction renverra une valeur par défaut : `0` si le type de retour est `int`, `NULL` si c'est un pointeur, `false` si c'est un booléen. On supposera cependant que si les pointeurs vers des structures `graph` sont définis (non nuls), alors les structures représentent des graphes valides et qu'il n'y a pas besoin de le vérifier dans `graph.c` (mais il faut s'assurer que les fonctions manipulent correctement ces structures dans `graph_tests.c`).

L'évaluation (facultative et à titre indicatif) consistera à soumettre deux fichiers : les implémentations (`graph.c`) et les tests unitaires (`graph_tests.c`). Si une fonction auxiliaire paraît utile pour les tests, c'est dans ce fichier de test qu'il faudra la rajouter.

L'évaluation de `graph.c` se fait sur la base du nombre d'avertissements générés par `gcc`, par `valgrind`, de la validité du code en utilisant les tests unitaires enseignants et sur le respect des conventions de codage avec `indent`.

Les deux fichiers doivent être soumis **tels quels** pour permettre le traitement et l'évaluation automatique (non compressés, non archivés, non renommés).