# TP11 : graphes

Algorithmique et projet de programmation — M1 mathématiques

13 décembre 2016

Dans ce TP, on s'intéresse aux graphes finis simples, c'est-à-dire aux couples $(V, E)$ avec $V$ un ensemble fini et $E \subseteq V \times V$ une relation binaire anti-réflexive et symétrique. On nomme les éléments de $V$ des *sommets* et ceux de $E$ des *arêtes*.

Pour représenter un graphe $(V, E)$ de sommets $v_0, \ldots, v_{n-1}$, on peut utiliser

- une matrice $A = (a_{i,j})_{0 \leq i,j \leq n-1}$, dite *matrice d'adjacence*, définie par :

$$a_{i,j} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{sinon} \end{cases}$$

- ou un vecteur $t = (t_i)_{0 \leq i \leq n-1}$ d'ensembles finis défini par :

$$t_i = \{j : (v_i, v_j) \in E\}$$

Pour représenter les ensembles finis du vecteur d'adjacence, on va utiliser une structure inspirée des listes chainées.

**Exercice 1.** Écrire une petite bibliothèque (création, ajout, suppression, impression, etc.) portant sur les *listes circulaires doublement chainées*, i.e.

```
typedef struct node_s* cdll; // circular doubly linked list

struct node_s {
  int key;
  cdll left, right;
};
```

**Exercice 2.** Écrire une fonction `cdll* mat_to_vec (int** adj, int n)` qui prend en paramètres une matrice d'adjacence d'un graphe et son nombre `n` de sommets, et qui retourne le vecteur d'adjacence de ce même graphe.

On va implémenter l'agorithme d'Hierholzer qui détermine un *circuit eulérien* dans un graphe quand il existe. On rappelle qu'un circuit eulérien est un chemin du graphe partant d'un sommet $v$, finissant au sommet $v$ et passant par chaque arête une et une seule fois.

**Exercice 3.** Montrer qu'un graphe fini simple admet un circuit eulérien si et seulement si il est connexe et le degré de tous ses sommets est pair.

L'algorithme d'Hierholzer fonctionne sur un graphe connecté avec sommets de degré pair comme suit :

- choisir un sommet $v$ quelconque et trouver un circuit simple basé en $v$, i.e. un chemin de $v$ à $v$ utilisant une arête au plus une fois,
- tant que le circuit construit n'est pas eulérien, choisir un sommet $u$ le long du circuit ayant des arêtes adjacentes non parcourue et trouver un circuit simple basé en $u$ n'utilisant que des arêtes alors non parcourue ; puis combiner ce circuit au circuit basé en $v$.

Le circuit devient ultimement eulérien, et l'algorithme s'arête alors.

On représentera en C un circuit d'un graphe comme la liste circulaire doublement chainée (`cdll`) des sommets parcourus.

**Exercice 4.** Écrire une fonction

```
cdll cycle_from (int i, int n, cdll* unused, cdll rvertex);
```

qui trouve un circuit simple basé en $i \in \{0, \ldots, n-1\}$ dans un graphe à $n$ sommets. Le tableau `unused` contient en case $j$ les listes des arêtes restantes à emprunter depuis $j$ : toute arête choisie doit en particulier en être retirée. On maintiendra également la liste `rvertex` des sommets ayant encore des arêtes adjacentes non visitées.

**Exercice 5.** En déduire une implémentation de l'algorithme de Hierholzer :

```
cdll euler_cycle (cdll* graph, int n);
```