TP1et2 : Graphes eulériens et semi-eulériens (2 séances)

Un graphe est eulérien (resp. semi-eulérien), ou encore peut être dessiné sans lever le crayon ni passer deux fois sur le même trait, s'il est connexe et tous ses sommets sont de degré pair (il a exactement deux sommets de degré impair). Dans la première partie, nous allons seulement caractériser les graphes eulériens (resp. semi-eulériens). Dans la deuxième partie vous calculerez un chemin eulérien (resp. semi-eulérien).

Pendant le premier TP, vous avez pu vous familliariser avec `CamlGraph`. Pensez à définir quelques exemples de graphes pour tester vos algorithmes.

## 1. TP1 : Caractérisation des graphes eulériens (resp. semi-eulérien)

a) Programmez la fonction *est_ connexe*, qui prend un graphe en entrée et renvoie un booléen indiquant si le graphe est connexe. Vous pouvez utiliser l'algorithme de Demoucron.
b) Programmez une fonction *est_ degre_ pair* qui prend un graphe et le sommet, et renvoie un booléen indiquant si un sommet est de degré pair.
c) A l'aide des fonctions précédentes, programmer les fonctions *est_ eulerien* et *est_ semi_ eulerien* qui vérifient si un graphe est eulérien et semi-eulérien.

## 2. TP2 : Calcul d'un cycle ou chemin eulérien

Pour un graphe eulérien (resp. semi-eulérien), on souhaite trouver un cycle (resp. chemin) eulérien. Ecrivez une fonction qui prend en entrée un graphe eulérien (resp. semi-eulérien) et retourne une liste de sommets correspondant à un parcours eulérien (resp. semi-eulérien).

Pour trouver une chaîne eulérienne, ou un cycle, on choisit une chaîne initiale, ou un cycle initial, puis on ajoute des cycles connectés, jusqu'à qu'on ait visité toutes les arêtes.

Quelques remarques :
— on pourra faire une copie du graphe, et éplucher ses arêtes au fur et à mesure qu'on les parcourt,
— le *backtracking* n'est pas nécessaire puisqu'on est sûr de trouver un parcours en avançant sur une arête (toutes les arêtes doivent être visitées).