# TD6 : Révisions

**Exercice 1** (Composantes connexes d'un graphe)**.**
Soit $G$ un graphe dont les sommets sont représentés par les entiers $0, \ldots, n-1$. On veut calculer les composantes connexes de $G$, sous la forme d'un tableau `composante` de taille $n$, tel que `composante[v]` $= i$ si le sommet $v$ appartient à la $i$-ème composante connexe de $G$.

Pour simplifier, on suppose que l'on dispose de la fonction `parcoursProf(G,v)` qui renvoie l'arbre enraciné en $v$ représentant le parcours en profondeur de la composante $C$ contenant le sommet $v$ dans $G$. Sa complexité est linéaire en le nombre d'arêtes incidentes à $C$ dans $G$.

1. Écrire un algorithme `composantesConnexes(G)` qui construit le tableau `composante`. Quelle est sa complexité dans le pire cas (en fonction du nombre de sommets $n$, du nombre d'arêtes $m$, et du nombre de composantes $r$ de $G$) ?
2. Si besoin, adapter l'algorithme précédent pour que sa complexité soit $O(m)$.

**Exercice 2** (File à l'aide de deux piles)**.**
On souhaite implémenter une structure de file dans un langage ne disposant que de la structure de pile. Nous allons pour cela procéder à l'aide de deux piles, une représentant une première partie de la file avec le dernier élément ajouté en tête, et l'autre représentant le reste de la file avec le prochain élément à extraire en tête.

```
struct file :
    debut : pile
    fin : pile
```

1. Implanter la fonction `enfile(F,x)` qui insère un nouvel élément $x$ à une file $F$.
2. Implanter la fonction `défile(F)` qui supprime l'élément le plus ancien d'une file F, et le renvoie.
3. Justifier que la complexité amortie de ces deux opérations est $O(1)$.

**Exercice 3** (Structure de dictionnaire)**.**
Un *dictionnaire* est une structure de données permettant de stocker des valeurs associées à un ensemble de *clés*, à condition qu'il existe un ordre naturel sur les clés. Par exemple, un tableau de taille $n$ peut être vu comme un dictionnaire dont les clés sont les entiers $0, \ldots, n-1$.

Pour un dictionnaire $D$, on a accès au moins aux primitives suivantes :
- `add(D,k,v)` : affecte à la clé `k` une nouvelle valeur `v`, en écrasant l'ancienne valeur s'il en existait une ;
- `get(D,k)` : retourne la valeur associée à la clé `k`, ou `None` si la clé est absente de `D` ;
- `delete(D,k)` : retire la clé `k` et la valeur associée de `D`.

1. Expliquer comment on peut utiliser la structure d'arbre binaire de recherche équilibré (AVL) pour représenter un dictionnaire.
2. Expliquer le fonctionnement de chacune des primitives d'un dictionnaire représenté à l'aide d'un AVL, et calculer leur complexité.

**Exercice 4** (Enveloppe convexe)**.**
On souhaite calculer l'enveloppe convexe d'un nuage de points $\mathscr{Q}$ dans le plan Euclidien. Pour rappel, il s'agit du plus petit polygone convexe qui contient $\mathscr{Q}$. Pour cela, nous commençons par trouver un point $q_0 \in \mathscr{Q}$ qui servira de pivot. On choisit $q_0$ comme étant le point le plus en bas à gauche (parmi les points d'ordonnée minimale, il s'agit de celui d'abscisse minimale). Soit $P$ l'enveloppe convexe de $\mathscr{Q}$.

1. Justifier que $q_0 \in P$.
2. Nous allons appliquer le principe Diviser pour Régner. Comment faire pour trouver un point $q_1 \in \mathscr{Q}$ tel que la droite $(q_0q_1)$ sépare $\mathscr{Q}$ en deux nuages de points $\mathscr{Q}_0, \mathscr{Q}_1$ de cardinalité presque égale (différence de cardinal $\leq 1$) ?
3. On se donne deux polygones convexes $P_0, P_1$, chacun représenté par un tableau `t0` ou `t1` contenant les coordonnées des sommets qui le composent, de sorte que deux sommets consécutifs dans le tableau forment un de ses côtés. On nous donne la fonction `memeCôté(A,B,p,q)` qui détermine en temps $O(1)$ si les points $p$ et $q$ sont du même côté de la droite $(AB)$. Implanter une fonction `estTangente(t0,t1,i,j)` de complexité $O(1)$ qui détermine si la droite (`t0[i]t1[j]`) est tangente aux polygones $P_0$ et $P_1$ (cf. Figure 1).

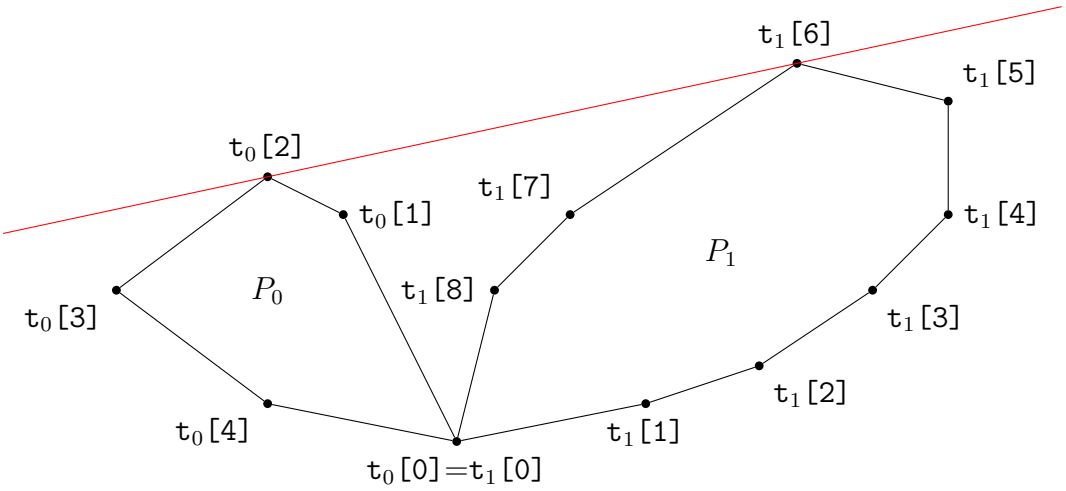


FIGURE 1 – La droite (`t0[2]t1[6]`) est tangente aux polygones $P_0$ et $P_1$.

4. Pour $i \in \{0, 1\}$, soit $P_i$ l'enveloppe convexe de $\mathscr{Q}_i$ représentée par un tableau $\mathtt{t}_i$. Implanter une fonction `trouveTangente(t0,t1)` qui renvoie un couple $(i, j)$ (avec $i > 0$ et $j > 0$) tel que la droite (`t0[i]t1[j]`) est tangente aux polygones $P_0$ et $P_1$. Quelle est sa complexité ?
5. Implanter la fonction `fusion(t0,t1)` qui renvoie l'enveloppe convexe de $\mathscr{Q}$, étant donné que $\mathtt{t}_i$ représente l'enveloppe convexe de $\mathscr{Q}_i$, pour $i \in \{0, 1\}$.
6. Implanter la fonction `enveloppeConvexe(t)` qui retourne l'enveloppe convexe du nuage de points contenus dans le tableau `t`. Calculer sa complexité à l'aide du Master Theorem.