

## CC PHP : Session 1 - Durée 1h30 - Sur machine

Téléchargez et décompressez l'archive déposée sur Moodle pour cet examen. Le dossier résultant contient différents fichiers à réutiliser ou à compléter. Vous testerez vos scripts en ligne de commande ou avec le conteneur Docker et Firefox selon les exercices. A l'issue du temps imparti, archivez votre dossier et déposez l'archive sur Moodle.

Une carte cognitive (alias **map** ou **cmap**) est un graphe orienté (alias **digraphe**) dont les nœuds sont étiquetés par des concepts et dont les arcs sont étiquetés par des valeurs qualifiant "l'influence" entre concepts. Le domaine d'influences est laissé au choix : domaine symbolique (+, -, 0, ?), entiers, pourcentages, etc. Par exemple, la carte illustrée en Figure 1 est étiquetée sur  $[-1, 1]$  et celle en Figure 2 est étiquetée sur  $\{0, ?, -, +\}$ .

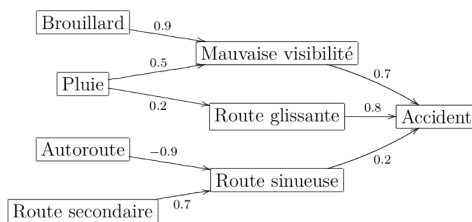


FIGURE 1 – Cmap étiquetée sur  $[-1, 1]$

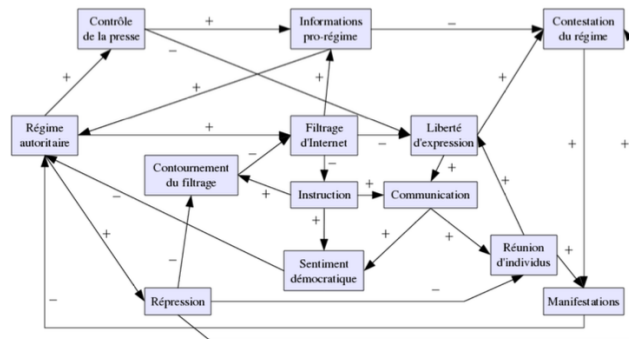


FIGURE 2 – Cmap étiquetée sur  $\{0, ?, -, +\}$ .

Les exercices proposés sont indépendants et portent sur le traitement de cartes cognitives sous différents angles : objets, base de données, XML et sessions. Le dossier décompressé contient un dossier par exercice et les éléments suivants :

- **utils** : dossier contenant un script PHP de connexion à base de données et un script JS qu'il utilise.
- **style.css** : feuille de styles utilisée dans certains exercices.

**Exercice 1. [Objets]**

Le dossier de cet exercice contient les fichiers suivants :

- **ADigraph.php** : fichier implémentant la classe abstraite **ADigraph** modélisant un digraphe.
- **Digraph.php** : fichier implémentant la sous-classe **Digraph** de **ADigraph** pour construire un digraphe.
- **docs/index.html** : documentation HTML des classes (générée avec DocBlock).
- **test.php** : script de tests de la classe **Digraph**.
- **data** : un dossier de cartes cognitives au format JSON.

Lisez la section introductive de la classe **AGraph** qui fournit les définitions et les principes de l'implémentation d'un digraphe  $G = (V, E, f)$ . Implémentez ensuite les méthodes manquantes dans la classe **Digraph** en répondant aux questions qui suivent et en respectant le cadrage donné en commentaires des méthodes. Le fichier **test.php** vous permettra de tester l'implémentation une fois complétée sur les jeux de données.

1. Les noeuds sont implémentés par un tableau indexé  $[1, \dots, |V|]$ . Implémentez la méthode **checkNodeId**.
2. Les identifiants d'arcs sont implémentés par un tableau indexé  $[1, \dots, |E|]$ . Implémentez la méthode **checkArcId**.
3. Tout arc est implémenté par un tableau de 2 éléments d'indices 0 et 1 et de valeurs respectives le noeud queue de l'arc et le noeud tête de l'arc. Implémentez la méthode **checkArc**.
4. L'association par la fonction  $f$  des identifiants d'arcs à leurs paires ordonnées de noeuds est implémentée par un tableau indexé associant l'identifiant d'un arc à sa paire de noeuds. Implémentez les méthodes **getIncomingArcs** et **getOutgoingArcs**.
5. Un noeud  $y$  est atteignable à partir d'un noeud  $x$  s'il existe un chemin de  $x$  à  $y$  dans  $G$  (une séquence d'arcs bien orientés démarrant de  $x$  et joignant  $y$ ). Implémentez la méthode **getReachableNodes** en utilisant la méthode **getDipaths**.
6. Un digraphe est cyclique s'il existe deux noeuds  $x$  et  $y$  tels que  $x$  est atteignable à partir de  $y$  et  $y$  à partir de  $x$ . Implémentez la méthode **isCyclic**.

---

**Exercice 2. [Requêtage et formulaires]**

Importez le script **map.sql** qui génère la base de données **13\_cc\_23\_php\_map** (attention : le script créera lui-même la base de données, pas besoin de la créer vous-même à l'avance). L'objectif de cet exercice est de récupérer une carte cognitive en utilisant l'API php PDO.

1. Développez la fonction **select\_map()** pour récupérer une carte cognitives de la table **MAP**.
2. Développez la fonction **select\_nodes()** pour récupérer les noeuds pour un identifiant de carte donné.
3. Développez la fonction **select\_arcs()** pour récupérer les arcs pour un identifiant de carte donné. Utilisez la requête fournie dans le code.

---

**Exercice 3. [XML vers base de données]**

Le fichier **exercice3.php** comporte un formulaire pour récupérer le nom d'un fichier XML qui contient une carte cognitive. L'objectif de l'exercice est de développer les fonctions nécessaires pour importer le fichier puis ajouter les valeurs obtenues dans la base de données. Développez vos fonctions dans les fichiers **xml.php** et **bdd\_insert.php** dans le répertoire **./exercice3**.

1. Développez une fonction **importer\_xml()** qui prend le nom d'un fichier XML comme paramètre puis l'importe et retourne un tableau associatif.
2. Développez la fonction **insert\_map()** pour ajouter un enregistrement dans la table **MAP**. La fonction prend comme paramètres : un nom de map, le nombre de nœuds et le nombre d'arcs d'un map.
3. Développez la fonction **insert\_nodes()** qui prend un ensemble de nœuds pour les ajouter dans la table **NODE**. Utilisez des requêtes préparées.

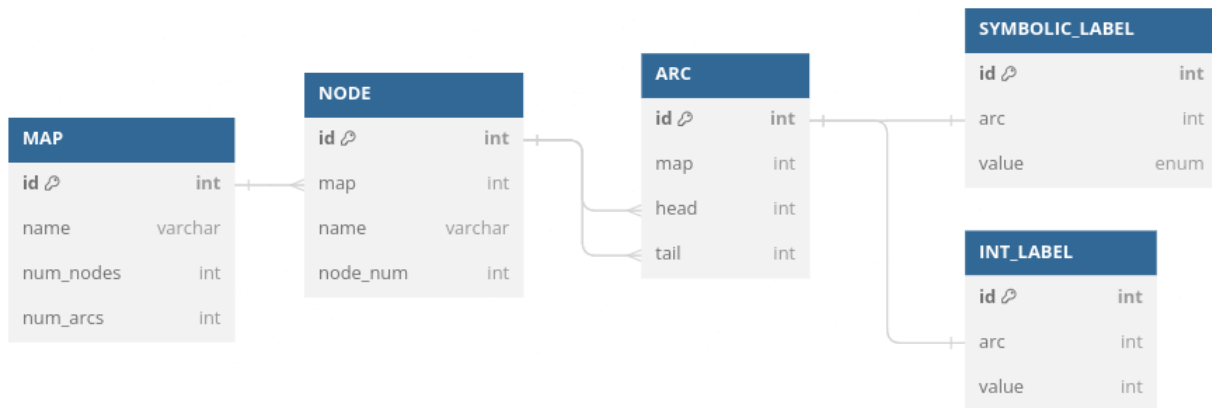


FIGURE 3 – Schéma de la base de données 13\_cc\_23\_php\_map

4. Similairement à la fonction précédente, développez la fonction `insert_arcs()`.
5. Complétez le fichier **exercice3.php** pour obtenir le comportement souhaité, c'est-à-dire, l'import du XML et l'insertion dans la base de données.

#### Exercice 4. [Sessions/cookies]

1. Le service d'affichage de cartes suit une stratégie "*freemium*" qui permet d'afficher au maximum 3 fois la même carte ; cette restriction est à durée illimitée. Implémentez ce comportement en utilisant cookies ou variables de session. Codez votre solution dans le fichier `./exercice4/freemium.php`. Ajoutez un message d'erreur si une carte a été demandée au-delà du seuil autorisé (voir Figure 4).

### Service d'affichage de cartes cognitives

Choisissez une carte cognitive:

Vous ne pouvez plus afficher cette map (Error: Nombre maximum d'affichages atteint).

FIGURE 4 – Résultat en cas de dépassement du nombre d'affichage autorisé par carte