Programme de colle - Semaine 4 (du 24 septembre)

La démonstration des énoncés marqués d'une étoile est exigible

1 Rappels sur la récursivité et l'induction

Les récurrences et inductions devront toujours être proprement rédigées (propriété, initialisation, hérédité, conclusion).

- Preuves par récurrence simple
- Preuves par récurrence double
- Preuves par récurrence forte
- Ensemble défini par induction comme la plus petite partie contenant les axiomes et stable par les règles d'induction. **La définition version constructive est équivalente : $X = \bigcup_{n\in\mathbb{N}} X_n$ (*)**.
- Preuves par induction

2 Langages réguliers

- Mots : alphabet, mot sur un alphabet, concaténation, puissance d'un mot, structure de monoïde.

- Langages : opérations ensemblistes sur les langages (union, intersection, différence, complémentaire), concaténation de deux langages, étoile de Kleene d'un langage, **l'union d'une famille finie de langages réguliers est un langage régulier (*)**

- Langages réguliers (aussi appelés langages rationnels): définition inductive de la classe des langages réguliers, **tout langage fini est régulier (*)**

- Expressions régulières : définition inductive, langage dénoté par une expression régulière (notation $\mu(e)$)

- **Un langage est régulier si et seulement si il existe une expression régulière qui le dénote (*)** [NB : interroger sur un seul sens uniquement]

- Expressions régulières étendues : aucune connaissance spécifique n'est exigible mais les étudiants doivent savoir trouver une expression régulière équivalente pour chacune des opérations introduites.

3 Parcours de graphes

Tous les algorithmes ci-dessous doivent pouvoir être écrits aisément par l'élève en pseudo-code

- Définitions usuelles : graphe orienté, non orienté, sommet/nœud, arête/arc, voisins (entrants/sortants), degré, chemins, cycles, graphe pondéré, etc.

- Représentation par matrice d'adjacence ou listes d'adjacence.

- Généralités sur les parcours : notion de nœud ouvert (découvert mais non encore exploré), de nœud fermé (exploré, ce qui découvre ou redécouvre les voisins), arborescence d'un parcours

- Parcours en largeur : avec une file pour enregistrer les sommets ouverts

- Parcours en profondeur : avec une pile pour enregistrer les sommets ouverts, ou par récursivité

- Algorithme de Dijkstra : sur des graphes pondérés avec des **poids positifs**. **La valeur $g(n)$ attribuée à chaque nœud fermé est le poids d'un chemin optimal menant de la source à n (*)**. Algorithme exprimé à l'aide d'une file de priorité.

- Algorithme A* : heuristique, heuristique admissible, heuristique monotone; **une heuristique monotone est admissible(*)**. **Si l'heuristique est monotone alors A* construit des chemins optimaux(*)** (Preuve à l'aide Dijkstra : on introduit des poids alternatifs, on montre que les chemins optimaux sont les mêmes, enfin on montre que A* se comporte comme Dijkstra pour ces poids).