

Programme de colle - Semaine 6 (colle du 8 octobre)

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 admissible alors A^* construit des chemins optimaux. Si de plus l'heuristique est monotone, il n'est pas nécessaire de ré-ouvrir les sommets et la complexité est équivalente à Dijkstra.

4 Automates finis

Les étudiants doivent savoir appliquer les constructions et algorithmes sur les automates.

- Automate fini déterministe (afd) : définition et calcul d'un afd, notion de calcul dans un afd,
- Langage reconnu (ou accepté) par automate : calcul réussi, mots reconnus (ou acceptés). Classe des langages reconnaissables.
- Accessibilité et co-accessibilité; afd émondé
- Afd complet; complétion d'un afd
- **Automate complémentaire : Si L est reconnaissable alors \bar{L} l'est (*)**
- Automate fini non déterministe (afnd) : définition et calcul d'un afnd, mots reconnus, langage accepté
- Déterminisation d'un afnd

- Automate fini non déterministe avec transitions instantanées (ε -afnd) : définition, ε -clôture, calculs d'un ε -afnd, mots reconnus, langage accepté
- Détermination d'un ε -afnd (automate des parties)

NB : le lien entre langages réguliers et langages reconnaissables n'a pas encore été étudié