

1 Programme du premier semestre

Les séances de travaux pratiques du premier semestre poursuivent les objectifs suivants :

- consolider l'apprentissage de la programmation en langage Python qui a été entrepris dans les classes du lycée;
- mettre en place un environnement de travail;
- mettre en place une discipline de programmation : spécification précise des fonctions et programmes, annotations et commentaires, jeux de tests;
- introduire les premiers éléments de complexité des algorithmes : on ne présente que l'estimation asymptotique du coût dans le cas le pire;
- introduire des outils de validation : variants et invariants.

Le tableau ci-dessous présente les thèmes qui sont abordés lors de ces séances, et, en colonne de droite, une liste, sans aucun caractère impératif, d'exemples d'activités qui peuvent être proposées aux étudiants. L'ordre de ces thèmes n'est pas impératif.

Aucune connaissance relative aux modules éventuellement rencontrés lors de ces séances n'est exigible des étudiants.

Thèmes	Exemples d'activité, au choix du professeur et non exigibles des étudiants. Commentaires.
Recherche séquentielle dans un tableau unidimensionnel. Dictionnaire.	Recherche d'un élément. Recherche du maximum, du second maximum. Comptage des éléments d'un tableau à l'aide d'un dictionnaire. <i>Manipulations élémentaires d'un tableau unidimensionnel. Utilisation de dictionnaires en boîte noire. Notions de coût constant, de coût linéaire.</i>
Algorithmes opérant sur une structure séquentielle par boucles imbriquées.	Recherche d'un facteur dans un texte. Recherche des deux valeurs les plus proches dans un tableau. Tri à bulles. <i>Notion de complexité quadratique. On propose des outils pour valider la correction de l'algorithme.</i>
Utilisation de modules, de bibliothèques.	Lecture d'un fichier de données simples. Calculs statistiques sur ces données. Représentation graphique (histogrammes, etc.).
Algorithmes dichotomiques.	Recherche dichotomique dans un tableau trié. Exponentiation rapide. <i>On met en évidence une accélération entre complexité linéaire d'un algorithme naïf et complexité logarithmique d'un algorithme dichotomique. On met en œuvre des jeux de tests, des outils de validation.</i>
Fonctions récursives.	Version récursive d'algorithmes dichotomiques. Fonctions produisant à l'aide de <code>print</code> successifs des figures alphanumériques. Dessins de fractales. Énumération des sous-listes ou des permutations d'une liste. <i>On évite de se cantonner à des fonctions mathématiques (factorielle, suites récurrentes). On peut montrer le phénomène de dépassement de la taille de la pile.</i>
Algorithmes gloutons.	Rendu de monnaie. Allocation de salles pour des cours. Sélection d'activité. <i>On peut montrer par des exemples qu'un algorithme glouton ne fournit pas toujours une solution exacte ou optimale.</i>
Matrices de pixels et images.	Algorithmes de rotation, de réduction ou d'agrandissement. Modification d'une image par convolution : flou, détection de contour, etc. <i>Les images servent de support à la présentation de manipulations de tableaux à deux dimensions.</i>
Tris.	Algorithmes quadratiques : tri par insertion, par sélection. Tri par partition-fusion. Tri rapide. Tri par comptage. <i>On fait observer différentes caractéristiques (par exemple, stable ou non, en place ou non, comparatif ou non, etc).</i>