Programme de l'option mathématiques et informatique en classe Spéciale MP

Préambule

L'informatique sera considérée ici comme une science opérant sur des représentations rigoureuses de concepts bien définis. Le programme doit donc permettre de présenter les principes de la programmation ainsi que les bases de l'algorithmique, de la théorie des automates et de la logique. Le programme qui suit se veut à la fois ambitieux et cohérent, tout en évitant d'aborder les concepts trop difficiles ou trop techniques, qui relèvent des études ultérieures en École.
Par ailleurs, un enseignement d'informatique doit être confronté à un «principe de réalité» : les élèves doivent donc avoir à mettre en œuvre les outils conceptuels étudiés, en programmant dans un langage de programmation, sous la forme de programmes clairs, courts et précis. La liste des langages autorisés sera publiée et actualisée par une circulaire. Les candidats au concours ne devront pas être évalués sur leur virtuosité à manipuler tel ou tel langage et il est souhaitable que les épreuves de concours ne favorisent pas les utilisateurs d'un langage particulier.

Méthodes de programmation

On dégagera la méthode d'analyse descendante (c'est-à-dire par raffinements successifs). Même si l'on ne prouve pas systématiquement tous les algorithmes, on dégagera l'idée qu'un algorithme doit se prouver, à l'instar d'un théorème de mathématiques. On étudiera systématiquement la complexité des algorithmes du programme et, sur certains exemples, le lien entre cette complexité et les structures de données. On s'attachera à obtenir des étudiants une documentation aussi complète que possible de leurs algorithmes (conditions d'entrée dans un module, conditions de sortie, invariants dans les boucles ou les appels récursifs). Toutes ces notions seront dégagées à partir des algorithmes au programme, sans aucune théorie sur les prédicats ou les invariants de boucles.

Itération

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Récursivité

On donnera de nouveaux exemples de structures récursives en liaison avec les notions introduites aux paragraphes suivants.

On insistera sur l'équivalence des différentes formes d'une expression algébrique (arbre, notation préfixée, expression parenthésée).

Diviser pour régner

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Éléments de complexité des algorithmes

Étude des récurrences usuelles :

$T(n) = T(n-1) + a.n$	par exemple pour le tri par insertion.
$T(n) = a.T(n/2) + b$	par exemple pour la recherche dichotomique.
$T(n) = 2.T(n/2) + f(n)$	notamment pour la méthode «diviser pour régner»

Utilisation de la notation O.

Notion de taille de données, évaluation du nombre d'opérations (calculs, comparaisons...) nécessaires à l'exécution et de l'encombrement mémoire.
Notion et exemples de complexité logarithmique, polynomiale, exponentielle (comparaison de temps d'exécution).

On se limitera à dénombrer les opérations nécessaires pour évaluer le temps d'exécution.
Tout autre considération est exclue du programme.

Exemples de calculs de complexité dans le cas le plus défavorable, dans lé cas moyen (tris, recherches dans un tableau ou dans un arbre, calcul de PGCD).

Pour le cas moyen, on traitera un ou deux exemples où l'on peut mener simplement un calcul explicite grâce à l'analyse combinatoire et dans certains cas on se contentera de notions intuitives sur la probabilité de répartitions des données.

Comparaisons de complexités de divers algorithmes pour résoudre un même problème : calcul des puissances, évaluation de polynômes, suites récurrentes (méthode récursive simple, méthode récursive avec stockage des résultats intermédiaires, méthode itérative), recherche dans un tableau ordonné (recherche linéaire, recherche dichotomique).

En particulier, on mettra en évidence sur des exemples le compromis souvent nécessaire entre temps de calcul et occupation de la mémoire. On montrera comment le stockage de résultats intermédiaires peut diminuer la complexité d'un algorithme.

Structures de données et algorithmes

Il s'agit de montrer l'influence des structures de données sur les algorithmes et les méthodes de programmation. Notamment, on mettra en parallèle les structures récursives des types de données et des programmes qui les manipulent. Les algorithmes seront présentés au tableau, en étudiant, dans la mesure du possible, leur complexité (dans le cas le pire et/ou en moyenne). Certains de ces algorithmes feront l'objet d'une programmation effective : les programmes correspondants devront rester clairs, courts et précis.

Listes et Piles

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Arbres

Définition récursive du type arbre binaire.

On peut utiliser la notation :

$$\text{Arbre} = \text{nil} + \text{Arbre} \times \text{Élément} \times \text{Arbre}.$$

Nœuds, feuilles. Arbres n-aires. Hauteur. Arbres équilibrés, et hauteur en $\log n$ dans le cas d'un tel arbre de n feuilles. Relation entre nombre de nœuds et nombre de feuilles dans le cas des arbres binaires.

On ne parlera pas des arbres AVL, 2-3, 2-3-4 ou bicolores.

Calculs récursifs du nombre de nœuds, de la hauteur, du nombre de feuilles. Insertion et suppression dans un arbre. Arbres de recherche.

On insistera sur la simplicité de l'écriture récursive.

Représentation des expressions arithmétiques par des arbres. Parcours préfixes, infixes et postfixes d'arbres. Application à l'impression préfixe, infixe ou postfixe d'expressions arithmétiques.

Cette partie interagit avec le dernier paragraphe du chapitre logique.

Automates finis

Il s'agit ici de présenter un modèle élémentaire de calculateur. On insistera sur cet aspect en présentant les automates comme des systèmes simples réagissant à des événements extérieurs.

Automates finis déterministes et non-déterministes.	On donnera une représentation graphique des automates.
Langage reconnu par un automate. Algorithme de déterminisation.	On présentera à ce propos les définitions de base : alphabet, mot, langage.
Expressions rationnelles. Langage associé.	On précisera bien la différence entre expressions rationnelles et langages.
Propriétés de fermeture des langages reconnus par automate : complémentation, intersection, union, concaténation et itération.	On insistera évidemment sur l'aspect constructif de ces propriétés de fermeture.
Lemme de l'étoile. Existence de langages non reconnus par un automate.	

Notions de logique

Le but de cette partie est de familiariser progressivement les élèves à la différence entre syntaxe et sémantique, ceci à travers l'étude des expressions logiques et arithmétiques. L'étude du calcul des prédicats et les théorèmes généraux de la logique du premier ordre ne sont pas au programme.

Éléments de base du calcul propositionnel

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Fonctions booléennes

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Circuits élémentaires

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Exemples de manipulation formelle de termes et de formules sans quantificateur

Différence entre syntaxe abstraite (ou arborescente) et valeur d'une expression arithmétique. Évaluation dans $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$ d'une expression arithmétique (avec les opérateurs $+$, $-$, $\times$, $/$). Dérivation formelle. Exemples de formules sans quantificateur écrites avec des symboles de prédicat unaires ou binaires ; interprétation.	Le but de cette partie est de montrer la différence entre syntaxe et sémantique, expression formelle et interprétation. Elle est l'occasion d'une familiarisation avec du calcul formel tout à fait élémentaire et elle fournit des exemples de formules logiques qui peuvent servir dans des preuves de programmes simples (par exemple le tri).