# Programme d'informatique

| | COMMENTAIRES |
|---|---|
| ALGORITHMIQUE<br><br>1) Algorithmes gloutons<br>2) Suites récurrentes<br>3) Diviser pour régner<br>4) Algorithmes d'exploration avec retour en arrière<br>5) Variant et invariant<br>6) Spécification | 1) Changement de base (voir le Thème Architecture), rendu de monnaie. Le candidat doit pouvoir justifier la correction d'un algorithme de ce type.<br>2) On se limite au calcul de suite d'ordre k fixé en utilisant une version itérative et une version récursive. Il pourra être demandé dans le second cas d'évaluer le nombre d'appels. Les éléments des suites peuvent être des tableaux ou des listes.<br>3) La recherche dichotomique dans un tableau trié et le tri fusion serviront d'exemples parmi d'autres. Il est demandé de pouvoir appliquer le théorème sur les suites récurrentes de la forme $T(n) = a.T(n/b)+n^c$ afin d'évaluer un algorithme de ce type.<br>4) Par exemple, le problème des n Reines et DPLL. Il n'est pas exigible de fournir une évaluation du gain par rapport à une exploration exhaustive. Le candidat doit pouvoir formuler des critères permettant de revenir en arrière et pouvoir évaluer une proposition de nouveau critère.<br>5) Le candidat doit pouvoir spécifier un invariant et un variant pour chaque boucle utilisée. Il n'est pas demandé d'introduire un calcul de plus faibles préconditions, mais une justification informelle pourra être demandée.<br>6) Les fonctions doivent être spécifiées par un contrat liant leur résultat à leurs arguments. Dans le cas de fonctions traitant des tableaux, il pourra être demandé une formulation logique précise. |
| ANALYSE QUANTITATIVE D'ALGORITHMES<br><br>1) Définition de la complexité en temps et en espace. Notations O, Ω, Θ<br><br>2) Calculer la complexité en moyenne et dans le pire cas d'un programme impératif<br><br>3) Calcul de la complexité dans le pire cas d'une fonction récursive : Méthode par développement, par substitution.<br><br>4) Équations récurrentes linéaires d'ordre k.<br><br>5) Récurrences par division : linéarisation par changement de variable, preuve du théorème pour les récurrences de la forme $T(n) = a \cdot T(n/b)+n^c$ | 1) Les Machines de Turing ne sont pas au programme. Les définitions se basent sur des exemples de programmes déjà étudiés.<br>2) Pour la complexité moyenne en présence de boucles, seuls les problèmes dans lesquels la probabilité de sortie de boucle est constante et aisément trouvable ou fournie dans l'énoncé pourront être posés aux candidats.<br>3) Dans le cas de résolution par substitution, il sera demandé au candidat de fournir la solution et de prouver sa correction.<br>4) Limitation à des équations avec coefficients constants. Lorsqu'il n'y a pas de racines évidentes, limitation au cas $k \leq 2$. On traite aussi le cas où la partie non-homogène est une somme de puissances multipliées par des polynômes. La résolution d'une équation sans justification algorithmique pourra être demandée à un candidat.<br>5) Le théorème sera en particulier utilisé pour évaluer rapidement des algorithmes de type "diviser pour régner". |

| | COMMENTAIRES |
|---|---|
| ARCHITECTURE<br><br>1) Codage des entiers<br>2) Codage des nombres à virgule flottante<br>3) Tableaux de Karnaugh | 1) Entiers signés et non-signés, nombre de bits. Ce point pourra être traité dans le Thème Algorithmique.<br>2) En particulier il est demandé de traiter la comparaison de nombres à virgules avec une précision relative donnée. La perte de précision en cas d'enchaînement de calculs n'est pas au programme.<br>3) Simplification de formules booléennes. Limitation à moins de 4 variables. Ce point pourra être traité dans le Thème Logique. |
| GRAPHES<br><br>1) Relations binaires sur un ensemble fini et | 1) On en profitera pour introduire les relations d'ordre, d'ordre bien fondé, et d'équivalence.<br>2) Le choix de la représentation la mieux adaptée pour une |

| | |
|---|---|
| propriétés<br>2) Graphes et leurs représentations : matrice d'adjacence, dictionnaire. Calcul de clôtures. Connexité et connexité forte.<br>3) Parcours en largeur, en profondeur, tri topologique.<br>4) Recherche des plus courts chemins d'un sommet aux autres. | opération de clôture réflexive, symétrique, transitive sera laissé au candidat<br>3) Le calcul des composantes fortement connexes à partir du tri topologique est à connaître.<br>4) Les algorithmes de Bellman, cas particulier des graphes sans circuits (avec utilisation du tri topologique), algorithme de Dijsktra doivent pouvoir être appliqués sur un graphe donné. Le choix de l'algorithme utilisé doit correspondre au mieux au graphe donné. |
| STRUCTURES DE DONNÉES<br>1) Spécification par un type abstrait.<br>2) Structures linéaires : piles, files, listes.<br>3) Listes de priorité.<br>4) Tables de hachage.<br>5) Arbres binaires de recherche, Arbres de recherche équilibrés | 1) Les exemples de ce thème seront tous introduits par un type abstrait indiquant les fonctions implémentées et leurs propriétés. Les preuves déductives ou inductives sur les types abstraits ne sont pas au programme.<br>2) Analyse de la complexité de l'insertion, de la suppression, et de la recherche d'un élément.<br>3) Analyse de la complexité de l'insertion, de la suppression, et de la recherche d'un élément. La justification de la complexité amortie des tas binomiaux et de Fibonacci n'est pas au programme, mais la connaissance de ces résultats peut être demandée pour justifier de l'utilisation d'une file de priorité.<br>4) Analyse de la complexité de l'insertion, de la suppression, et de la recherche d'un élément.<br>5) Analyse de la complexité de l'insertion, de la suppression, et de la recherche d'un élément dans les deux cas. |

| | **COMMENTAIRES** |
|---|---|
| LOGIQUE<br>1) Langage de la logique propositionnelle<br>2) Évaluation d'une formule<br>3) Normalisation d'une formule<br>4) Techniques d'évaluation de formules : Tables de vérité, résolution, DPLL (voir le Thème Algorithmique) | 1) Formules bien formées, opérations, variables, constantes.<br>2) Tautologie, satisfiabilité, contradiction.<br>3) Simplifier une formule (voir le Thème Architecture).Mettre une formule propositionnelle en forme normale conjonctive ou disjonctive.<br>4) Une technique particulière pourra être demandée pour évaluer une formule propositionnelle donnée. |
| LANGAGES<br>1) Mots, concaténation, préfixe, suffixe, longueur, langage.<br>2) Codes, ambiguïté, critères de non-ambiguïté, construction de l'automate associé.<br>3) Recherche d'un motif dans un texte.<br>4) Automate fini, langage reconnu, langage régulier, propriétés de clôture (union, intersection, clôture transitive).<br>5) Lemme de pompage.<br>6) Déterminisation d'un automate fini, reconnaissance du complémentaire d'un langage régulier.<br>7) Expressions régulières, langage reconnu, équivalence avec les langages réguliers | Dans tous les cas une construction explicite peut être demandée. Le lemme de pompage sera utilisé dans des problèmes consistant à décider si un langage est régulier. |