ÉPREUVE DE TIPE D'INFORMATIQUE 2016

ÉNS : CACHAN - LYON - PARIS

		ENS Cachan	ENS Lyon	ENS Paris
Coefficients :	**Concours MP :**	**2**	**1,5**	**8**
	Concours Info :	**3**	**1,5**	**1**

MEMBRES DU JURY : A. CARAYOL, F. PROST , M. SIGHIREANU

Le jury a évalué cette année 116 candidats présentant un dossier TIPE d'informatique. La proportion de candidates (six cette année) présentant un TIPE d'informatique reste toutefois anormalement faible au regard de celle des chercheuses en informatique (supérieure à 25%). Nous encourageons vivement les enseignants à diriger les candidates vers des sujets d'informatique.

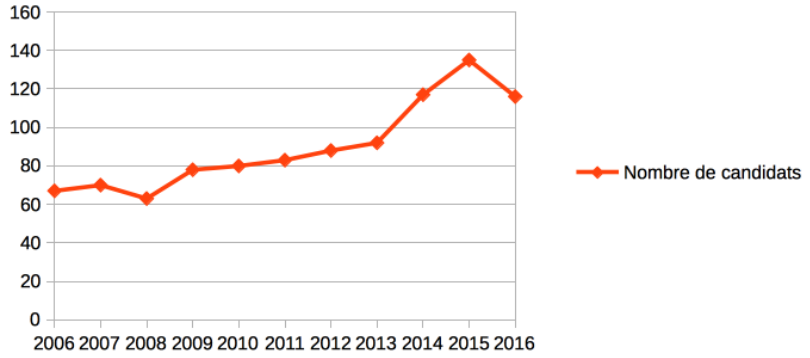

FIGURE 1 – Nombre de candidats en TIPE d'informatique, par année.

Comme les années précédentes, le jury a apprécié la qualité d'un nombre important de TIPE aussi bien dans le thème de cette année qu'en dehors. Les candidats ont été évalués sur leur maîtrise des concepts informatiques généraux, sur la qualité scientifique de leur travail, et surtout sur leur maîtrise et leur compréhension du sujet choisi. Le jury tient à souligner que le niveau des candidats auditionnés s'est significativement amélioré depuis 2007. De très nombreux oraux étaient de niveau très satisfaisant et plusieurs de niveau excellent ; de nombreux candidats se sont révélés lors des questions.

L'épreuve orale, d'une durée de 40 minutes se déroule ainsi. Le candidat a tout d'abord l'opportunité de présenter de manière synthétique son travail. Pour ce faire, le candidat peut, s'il le souhaite, s'aider d'un support visuel par ordinateur (diaporama en format PDF) prévu pour une présentation d'environ 10 minutes. Bien qu'optionnelle, le jury a constaté que cette présentation permettait au candidat de mieux valoriser son travail et de l'aider à structurer ses réponses aux questions. En effet, pour cette épreuve, le candidat sera amené à répondre aux nombreuses questions des membres du jury qui porteront soit directement sur le travail présenté (par exemple pour éclaircir des points techniques) ou sur des sujets connexes (par exemples, des approches alternatives ou des notions de base en informatique). De manière générale, le recul du candidat sur son sujet et sa culture générale en informatique sont des éléments appréciés par le jury. Ce dernier cherchera systématiquement à évaluer ces aspects au travers de questions sortant du cadre strict de l'étude présentée. Le candidat peut, à tout moment, consulter ses notes pour s'aider dans ses réponses aux questions du jury.

Voici une liste de situations typiques : le candidat utilise une notion complexe et le jury lui propose de la définir ; le candidat décrit un algorithme et le jury lui demande d'en évaluer la

complexité en temps ou bien de prouver son bon fonctionnement [1] ; le candidat utilise une structure de données classique et le jury l'interroge sur les algorithmes classiques sur cette structure de données ; un candidat expose sa solution au problème qu'il s'est posé, et le jury le guide vers des structures de données plus performantes pour résoudre son problème. Chaque fois que le sujet s'y prête, un travail expérimental et une réalisation logicielle sont attendus ; le candidat est alors amené à commenter ses programmes. Notons que le jury attend en terme d'implémentation plus que de simples appels à des bibliothèques préexistantes (solveur de programmation linéaire, simulateur d'automates cellulaires,...).

Le jury s'autorise à poser des questions sortant du cadre strict du travail réalisé. Par exemple, il pourra demander au candidat de réfléchir en direct à des variantes ou extensions du problème étudié. La réactivité et la capacité de proposition du candidat seront alors évaluées. Lorsque le candidat utilise des notions hors du programme des CPGE, il doit s'attendre à des questions sur celles-ci. L'épreuve des TIPE doit être vue comme **un oral à part entière portant sur un sujet choisi et spécialement préparé par le candidat**. Le choix du sujet est ainsi particulièrement important et doit permettre au candidat de mettre en valeur ses capacités créatives, la rigueur de son approche, son esprit critique et sa capacité à s'impliquer dans un projet qui le motive.

Recommandation sur le choix du sujet et la préparation du rapport. Chaque TIPE doit impérativement répondre à un questionnement informatique, avec une formalisation explicite des problèmes étudiés. Ainsi, un TIPE portant sur la simulation informatique d'un phénomène physique au comportement sophistiqué, mais où la part informatique est réduite à l'écriture d'un programme simple et ne fait pas appel à des structures de données ou des algorithmes évolués, met d'entrée de jeu le candidat dans une situation difficile. De même, la part informatique ne peut se réduire à une optimisation *ad hoc* d'un code sur un processeur donné, à moins que les méthodes proposées ne soient présentées dans une problématique générale (celle de la compilation dans cet exemple). De manière générale, le choix de présenter le TIPE en informatique ne peut pas se baser uniquement sur le fait qu'il y a eu un travail de programmation : nous attendons des candidats un réel questionnement sur les algorithmes et les structures de données utilisés.

Tous les rapports ont bien respecté la recommandation de ne pas dépasser les 6 pages (hors annexes et figures). Cependant, deux points restent à améliorer. Premièrement, il est important que tous les candidats incluent **en annexe** l'intégralité de leur code. Il n'est pas nécessaire de venir avec une version imprimée du rapport (les membres du jury ont la version électronique). Deuxièmement, nous demandons aux candidats d'indiquer **systématiquement** lorsque le travail de TIPE est issu d'une collaboration avec d'autres étudiants (ce qui est parfaitement acceptable). **Dans ce cas, le rapport tout comme les transparents doivent être un travail personnel**. Le candidat doit en effet être capable de refaire tout ce qui figure dans son rapport : de nombreux candidats se sont retrouvés en difficulté car ils avaient survolé des points essentiels dans les parties *préliminaires* de leur travail (peut-être faites par un binôme) ; embourbés dans une maitrise approximative de ces points préliminaires, ils n'ont pas pu présenter le cœur de leurs travaux.

Nous avons apprécié l'originalité de certains sujets traités ou la démarche créative de certains travaux. Il n'est pas acceptable que certains candidats se contentent de restituer des connaissances acquises dans un livre ou sur Internet. **Une réflexion personnelle doit s'élaborer autour de ces connaissances, avec un esprit critique.** De même, un TIPE ne peut pas se résumer à la présentation d'un TP ou d'un devoir à la maison.

À l'inverse, l'absence de recherche bibliographique a conduit certains candidats à un travail d'une grande naïveté : il est nécessaire de connaître les approches efficaces du problème considéré, les connaissances minimales attendues étant celles qui figurent dans une encyclopédie (comme par exemple, les pages wikipedia consacrées à ce sujet).

1. La preuve du bon fonctionnement de tout algorithme présenté doit pouvoir être donnée par le candidat (au moins dans les grandes lignes si celle-ci est difficile ou fait appel à des notions hors programme).

Certains sujets semblent maintenant trop balisés pour permettre des développements suffisants, en voici une courte liste. L'écriture d'un simple algorithme de *backtracking* (pour résoudre un problème d'optimisation) ne peut être l'unique objet d'un TIPE, à moins bien sûr que celui-ci requiert la mise en place de structures de données sophistiquées, ou une induction compliquée, ou une preuve de terminaison originale, ou bien encore une analyse intéressante de sa complexité en temps. Ainsi un TIPE ne peut se résumer à de la programmation uniquement. On pourra aussi s'intéresser à en améliorer les performances par des méthodes de type *branch-and-bound* où le choix des bornes serait un des objets d'étude du TIPE. Un candidat qui fait le choix de résoudre un problème en utilisant des méthodes à base de méta-heuristiques (algorithmes génétiques, colonies de fourmis...) ou des techniques d'apprentissage automatique (réseaux de neurones...) ne doit pas se limiter à la mise eu œuvre de la méthode. Dans le cas d'un algorithme d'approximation, il est important de prouver des garanties en terme de qualité de la solution (facteur d'approximation) et de performance. Les algorithmes génétiques, par exemple, sont un paradigme de programmation comme un autre, où le vrai programme est codé dans le choix des fonctions de *fitness*, de mutation et de croisement. C'est évidemment sur le choix de ces trois fonctions que le débat doit porter, en relation avec le problème à résoudre. **Il est également important que le candidat s'interroge préalablement sur l'existence d'algorithmes classiques efficaces (et exacts) pour résoudre le problème considéré** (c'est par exemple le cas pour la recherche de chemin dans un graphe...) et soit capable de comparer les performances obtenues par les approches heuristiques (par exemple des algorithmes de colonies de fourmis) avec celles des autres paradigmes plus classiques. Le jury attend une analyse fine des différentes options et de leurs impacts sur les performances et non la simple écriture d'un simulateur inadapté au problème étudié. De même, lors de la présentation d'un algorithme d'exploration de type α-β, le jury attend du candidat une compréhension précise du sens de ces valeurs, de leur rôle et de leur utilisation.

Nous regrettons également que les candidats ne pensent pas systématiquement à l'encodage par des entiers (peut-être même à des fonctions de *hash*) pour les tests d'égalité répétitifs d'objets structurés. Nous encourageons les candidats à approfondir l'analyse de la complexité des algorithmes présentés. Assez souvent, la complexité annoncée ne correspond pas à celle effectivement implémentée du fait d'un mauvais choix des structures de données. Nous recommandons aux candidats de faire également des estimations du temps de calcul de leur algorithme en partant sur une base de $\sim$1.000.000.000 d'opérations par secondes (correspondant grosso-modo au microprocesseur actuel de fréquence 1GHz), cela leur permettra de mieux comprendre les parties de leur programme qui pourraient être à l'origine des mauvaises performances.