ÉPREUVE ORALE D'INFORMATIQUE
ENS : PARIS LYON CACHAN
MEMBRES DE JURY : S. HADDAD, O. SERRE, E. THIERRY

Coefficients :	**PARIS**	**MPI : 20**	**groupe I : 4**
	LYON	**MPI : 3**	**groupe I : 4**
	CACHAN	**MPI : 12**	**groupe I : 12**

Cette épreuve a concerné les candidats aux trois écoles normales supérieures du concours MP-option informatique et les candidats aux trois écoles normales supérieures du concours informatique. Le jury a examiné 159 candidats (181 l'année dernière). L'épreuve testait en 45 minutes sans préparation l'habileté des candidats à manipuler rigoureusement les objets fondamentaux de l'informatique (automates, langages formels, graphes, logique) et leur demandait une bonne intuition algorithmique. Les notes se sont étalées entre 1 et 20, avec une moyenne de 10,77 et un écart-type de 4,79 (voir l'histogramme en fin de document). Trente sujets originaux différents ont été proposés aux candidats. La majorité des sujets (18) comportait au moins une question portant sur les langages rationnels ou la reconnaissabilité par automates finis. Il est à noter que les exercices relevant des automates étaient non classiques afin d'évaluer au mieux la capacité du candidat à travailler autour de ce concept. Certains sujets faisaient travailler sur des objets classiques (graphes, ordres, variantes des automates) sortant du cadre du programme et dont il fallait dans un premier temps assimiler la définition. Il était demandé systématiquement la rédaction propre d'une preuve. Enfin, la majorité des sujets comportait également un volet algorithmique simple.

Le jury a particulièrement évalué :
– La capacité d'initiative du candidat, son inventivité, sa manière d'aborder les questions et sa capacité à assimiler rapidement des concepts étrangers.
– La capacité à formaliser un problème et à exprimer mathématiquement les propriétés à prouver.
– La capacité à écrire des preuves propres (de propriété mais aussi d'algorithmes).
– La bonne connaissance des notions (et des preuves) au programme.

La plupart des candidats réagissent bien face à des concepts qu'ils ne connaissent pas *a priori*. Les candidats ont pour la plupart de bonnes idées et, même lorsque celles-ci ne conduisent pas au résultat, ils ne se démontent pas, ce que le jury a apprécié. En effet, le jury recherche des candidats pouvant à terme être de bons chercheurs et l'inventivité est l'un des critères fondamental. Les preuves sont souvent proprement écrites, même si cela est à tempérer pour les preuves de terminaison et de correction d'algorithmes.

Les faiblesses les plus marquantes ont été les suivantes :

– Les candidats voient souvent mal l'intérêt du non déterminisme pour les automates finis. Il ne faut pas voir le non déterminisme comme un caractère fâcheux de ces machines mais plutôt comme un atout pour deviner des propriétés des mots lus en entrée. En revanche, il ne faut pas hésiter, surtout lorsque l'on ne se soucie pas des questions de complexité, à prendre des automates deterministes complets lorsque l'on veut prouver des propriétés particulières : autant se placer dans le cadre le plus simple.
  Les candidats n'ont que trop rarement le réflexe d'utiliser les états (en ajoutant une composante par exemple) des automates pour stocker de l'information à vérifier plus tard.
  Le langage $\frac{L}{2} = \{u \mid \exists v,\ |u| = |v|, \text{ tq. } uv \in L\}$ où L est un rationnel illustre bien les deux remarques précédentes : le non-déterminisme permettra de deviner v lors de la lecture de u tandis que l'on stockera l'état intermédiaire (deviné dès l'état initial) dans lequel se trouve un automate reconnaissant L (que l'on choisira déterministe pour simplifier l'écriture des preuves) après avoir lu u.
– Les notions classiques sur les automates sont connues assez diversement : si le lemme de l'étoile (et sa preuve) est très bien maîtrisé, il n'en va pas de même pour la preuve de la déterminisation ou la

construction de l'intersection (ou de l'union) de deux langages rationnels. C'est dommage à double titre : le jury est toujours négativement influencé par un candidat qui ne connaît pas l'un des classiques du cours, et un candidat qui ne connaît pas une telle construction aura du mal à s'en inspirer pour en déduire une construction pour un problème original.
Concernant le lemme de l'étoile, nous ne saurions que trop recommander à nos collègues enseignant en CPGE de présenter une version permettant de pomper soit en début de mot (prendre $u = \varepsilon$ dans ce qui suit) soit dans un facteur arbitraire : pour tout L rationnel, il existe $N \geq 0$ tel que pour tout mot $uvw \in L$ avec $|v| \geq N$, il existe une factorisation de v en $v = xyz$ avec $|y| > 0$ telle que pour tout $k \geq 0$, $uxy^k zv \in L$. En effet, cela simplifie pas mal de preuves (ne pas hésiter aussi à prendre $k = 0$ qui permet souvent de conclure facilement).

– Les candidats ont rarement le réflexe d'utiliser les propriétés de clôture pour prouver la rationalité d'un langage. Par exemple, si $L \subseteq A^*$ est rationnel, et si $Min(L) = \{u \in L \mid \forall v \text{ prefixe strict de } u,\ v \notin L\}$, il est facile de voir que $Min(L)$ est rationnel en écrivant simplement que $Min(L) = L \setminus LA^+$. Une preuve par automate est beaucoup plus fastidieuse.
– Nous ne saurions que trop encourager les candidats à considérer des exemples pour aborder les questions que nous posons. C'est un réflexe trop rare qui se révèle pourtant payant et qui est à la base du métier de chercheur. De même, rien n'interdit de faire un dessin pour raisonner sur les automates ou sur les graphes, même si cela ne dispensera pas d'une preuve propre au final.
– Concernant la correction et la terminaison des algorithmes, beaucoup de candidats ont du mal à donner un invariant et à faire une preuve propre (une récurrence vaut bien mieux qu'un raisonnement se terminant par ...). Une autre faiblesse concerne l'analyse de la complexité des algorithmes : si les candidats arrivent souvent bien à compter le nombre de passages dans une boucle, ils ont souvent du mal à déterminer le coût précis d'un passage dans la boucle et plus encore à dire quelles structures de données (listes, tableaux, arbres...) utiliser pour obtenir les complexités annoncées.
– Quelques notions d'algorithmique de base sur les graphes (parcours essentiellement) se révèlent très utiles (pas seulement pour les exercices sur les graphes mais aussi pour ceux portant sur les automates).