RAPPORT DE L'ÉPREUVE ÉCRITE D'INFORMATIQUE-MATHÉMATIQUES
FILIÈRE MP – CONCOURS INFO – SESSION 2017
ÉCOLES CONCERNÉES : ENS DE CACHAN, ENS DE LYON, ENS DE PARIS, ENS DE RENNES

Coefficients (en pourcentage du total d'admission) :	**PARIS**	**groupe I : 13,3%**
	LYON	**groupe I : 12,7%**
	CACHAN	**groupe I : 13,2%**
	RENNES	**groupe I : 8,6%**

MEMBRES DE JURY : N. AUBRUN, D. BAELDE & A. SAURIN

L'épreuve écrite d'informatique-mathématiques concerne les candidats aux quatre Écoles Normales Supérieures sur le concours INFO. Le nombre de candidats ayant composé était de 344 pour la session 2017 (pour 429 inscrits) contre 318 en 2016. Les notes se sont échelonnées de 0 à 20 avec une moyenne de 9,6 et un écart-type de 3,67.

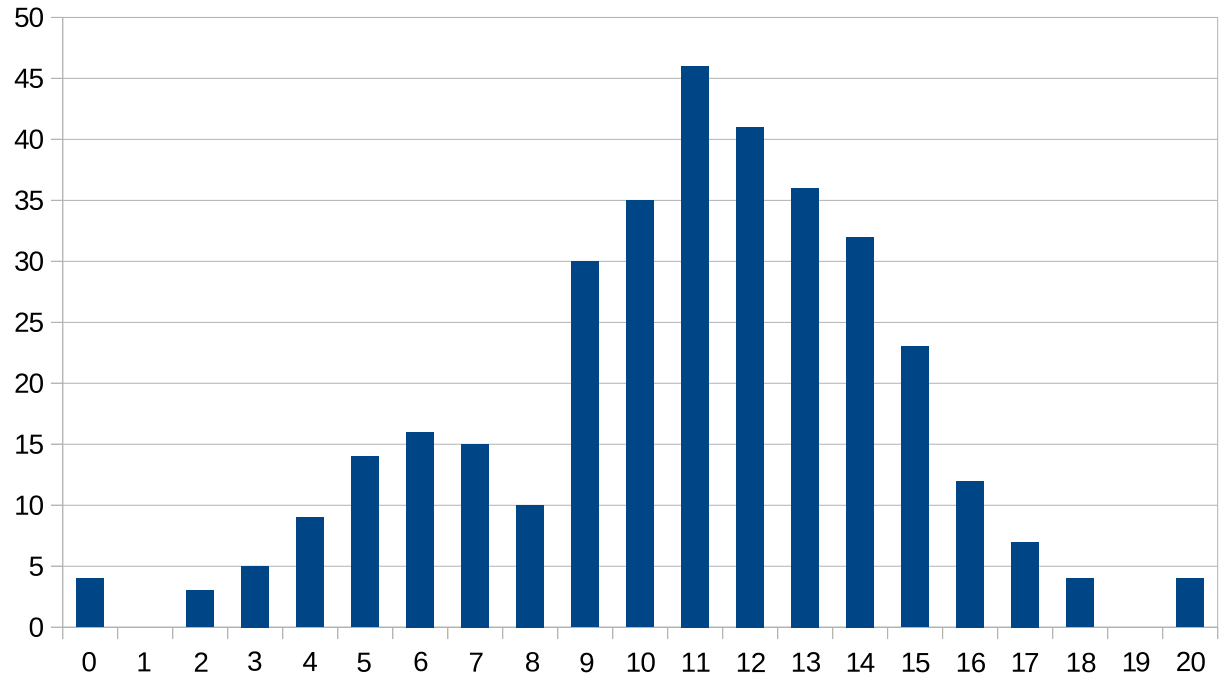

FIGURE 1 – Histogramme des notes obtenues par les candidats.

1 L'épreuve

Le sujet 2017 était un sujet de combinatoire des mots et traitait de la recherche efficace de carrés.

La première partie du sujet permettait aux candidats de se familiariser avec les définitions du sujet et de tester des notions basiques de compréhension de code Caml et d'écriture de programmes

simples. Les candidats devaient y écrire des programmes Caml simples permettant de tester l'existence d'un carré, d'en analyser la complexité dans le pire des cas (cubique en la longueur du mot) et de montrer que si l'alphabet ne contient que deux lettres, alors on dispose d'un algorithme en temps constant, puisque tout mot de longueur quatre contient un carré.

La deuxième partie du sujet visait à démontrer l'existence de mots infinis sans carré sur un alphabet de taille quatre. Il s'agissait de la partie la moins algorithmique et la plus mathématique du sujet. Il était demandé d'utiliser des propriétés de la représentation binaire des entiers pour construire un mot infini sur $\{0,1\}$ ne contenant pas de carré de période paire, puis d'obtenir un mot infini sans carré en ajoutant une information de parité, excluant ainsi également les périodes impaires.

La troisième partie du sujet avait pour but de concevoir un algorithme efficace utilisant la méthode diviser pour régner, en calculant les carrés créés par la concaténation de deux carrés. La dernière partie proposait une méthode de calcul efficace de ces nouveaux carrés, à l'aide d'une implémentation des fonctions `lms` et `lmp`.

2 Commentaires généraux

Les correcteurs estiment que, dans l'ensemble, le niveau de préparation des candidats de l'épreuve 2017 était correct : peu de copies étaient de mauvaise qualité. La première partie du sujet a ainsi été globalement bien traitée, montrant que les bases de l'informatique sont correctement assimilées (écriture d'un programme, évaluation de complexité, compréhension du sujet). Les correcteurs ont tout de même noté quelques réponses farfelues dès cette première partie, notamment dans le raisonnement sur la complexité, l'utilisation des booléens dans les tests de boucles `while`, l'usage d'exceptions après un test booléen lui-même dans une boucle `while`... pour gérer la sortie de la boucle, etc.

Les copies ont été différenciées d'une part sur la précision des réponses (erreurs de calculs, erreurs sur les bornes et indices imparfaits, erreurs dans les programmes notamment le mélange entre programmation impérative et fonctionnelle, l'oubli du déréférencement) et d'autre part sur les questions qui demandaient une compréhension fine des notions ou de la créativité (invention d'un mot infini bien construit, généralisation d'un argument précédent, traduction d'une idée mathématique en algorithme efficace, passage d'intuitions sur des exemples à une véritable preuve, etc).

La deuxième partie, plus mathématique, est la partie principale où les candidats ont eu à construire des raisonnements mathématiques non élémentaires et des raisonnements plus abstraits, la construction du mot étant indirecte en passant par l'écriture binaire. Le jury a eu le sentiment qu'une part significative des candidats travaillaient mécaniquement sans être guidés par l'intuition des objets qu'ils étudiaient.

Après les deux premières parties qui motivaient la recherche d'un algorithme efficace pour identifier les carrés d'un mot, l'objectif des deux parties suivantes du sujet était la conception d'un algorithme efficace pour tester l'existence de carrés et sa programmation en `Caml`.

La troisième partie était principalement une partie utilisant des raisonnements de combinatoire des mots. Seuls les candidats les plus rigoureux sont arrivés à bout des calculs sans erreur (dans les

raisonnements comme dans les programmes). La difficulté principale de la partie semble avoir été la question 15 – simplement ébauchée dans la plupart des copies – qui demandait de montrer le caractère nécessaire et suffisant d'une condition pour la création de nouveaux carrés (dont le centre était contenu dans le second mot). La fin de la partie utilisait une méthode diviser pour régner, et les correcteurs ont pu constater que si la très grande majorité des candidats connaît le principe de la méthode et les résultats de complexité associés, une minorité des copies a de sérieuses difficultés à l'utiliser correctement, oubliant le cas de terminaison.

La dernière partie avait pour objectif de construire efficacement les deux fonctions cruciales sur lesquelles repose l'algorithme ; c'était donc ici le coeur de l'algorithme. À part la question 23 nécessitant un raisonnement vraiment élaboré de combinatoire des mots, le reste de la partie était plus algorithmique et contenait de nombreuses questions de programmation. La question 24, qui demandait de démontrer la correction, la terminaison et la complexité d'un programme a été généralement mal traitée.

3 Commentaires détaillés

Ci-dessous nous donnons pour chaque question le nombre de points associés, puis :
- le nombre de copies ayant reçu des points pour cette question (c'est à dire le nombre de copies où la question a été traitée et non nulle) ;
- la moyenne de toutes les copies ayant reçu des points sur la question ;
- la moyenne de toutes les copies sur la question.

Partie I. Existence d'un carré.

▷ **Question 1. [0,25 pt, 309 / 0.25 / 0.22]** Il s'agissait d'une question très simple visant à se familiariser avec les notions du sujet en exécutant à la main un programme Caml simple. Elle a été, heureusement, bien réussie dans l'ensemble. Le jury attendait une exécution parfaite : la moindre erreur sur la question faisait perdre l'ensemble des points de la question.

▷ **Question 2. [1 pt, 329 / 0.73 / 0.7]** La question demandait de définir précisément le résultat du programme Caml donné en énoncé. Là encore, étant donné la simplicité de la question, le jury attendait une exécution et une rédaction précises. Les erreurs récurrentes consistaient à indiquer que la fonction renvoyait le nombre d'indices j tels que $m_{i+j} = m_{i+j+p}$ (oublie du caractère successif de ces indices), la longueur du plus grand carré en position i, à oublier la condition de maximalité ou encore, plus rarement, à confondre suffixes et préfixes.

▷ **Question 3. [0,5 pt, 332 / 0.45 / 0.44]** La question consistait à utiliser le programme fourni dans l'énoncé pour écrire une fonction qui teste si un mot m contient un carré de période p. Le jury a accordé la moitié des points dès que la fonction proposée correspondait à un algorithme correct, et la totalité des points si de plus il n'y avait pas d'erreur de programmation *de détail* (décalage

de l'indexation, oubli de déréférencement, typage, etc...) et que la complexité de la fonction était quadratique. Dans de nombreuses copies, les boucles continuaient plus loin que nécessaire, ce qui n'a pas été sanctionné tant que le code restait correct et la complexité quadratique. Parmi les erreurs récurrentes, on a également relevé des copies renvoyant un `int`, de nombreuses erreurs de bornes (`p = ref 0` à l'initialisation ou `while !p<n/2`, etc.), des boucles descendantes initialisées à `n-2p` avec condition d'arrêt à `i=-1`, des boucles qui ne s'arrêtent pas si p est trop grand, ou encore une mauvaise utilisation des tests booléens sur la condition d'un `while`. À noter que dans un nombre significatif de copies on pouvait lire `while !b = true`, ce qui n'a pas été sanctionné dans la notation, mais ne peut pas positivement impressioner le jury. Une copie utilisait des exceptions après un test booléen pour sortir des boucles `while`, méthode assez originale...

▷ **Question 4. [0,5 pt, 340 / 0.46 / 0.45]** Le jury a appliqué le même principe de notation que pour la question précédente : moitié des points pour un code *grosso modo* correct, totalité des points s'il n'y a pas d'erreur de détail et que la complexité est linéaire. Plusieurs copies faisaient débuter leur boucle à 0 ce qui ne pouvait pas aboutir à un programme correct, la période d'un carré étant nécessairement strictement positive.

▷ **Question 5. [1 pt, 272 / 1 / 0.77]** La question demandait de donner la complexité dans le pire des cas de la fonction obtenue précédemment, et le préambule du sujet précisait que toute complexité devrait être justifiée. Les candidats ont eu la totalité des points pour une complexité cubique sans erreur et justifiée, la moitié des points si la réponse est correcte mais sans justification et 0 pour toute erreur, notamment en cas de mauvaise utilisation du $O()$, par exemple des réponses du type $O(3n^3+n^2+C)$ ou bien $O(n^2 \times min(p, n-i-p))$.

▷ **Question 6. [0,5 pt, 334 / 0.4 / 0.39]** La question demandait de montrer que sur l'alphabet $\{a,b\}$ tout mot suffisamment long contient un carré et d'en déduire un algorithme en temps constant. On a accordé la moitié des points pour l'argument et l'autre moitié pour l'algorithme. Parmi les erreurs typiques, on a vu des candidats traiter correctement les mots de taille au moins 3 mais oublier de traiter les mots de taille 2.

Dans cette question, on a noté une très grande variabilité de la qualité et la clarté de la rédaction avec des rédactions parfois très inefficaces (énumération de tous les mots, factorisations m = uvw pas clairement définie pour conclure que v est un carré...) là où d'autres candidats faisaient preuve d'élégance.

À noter qu'une copie a cherché à utiliser le lemme de l'étoile pour le cas d'un langage rationnel sur $\{a,b\}^\star$, tentative qui laisse le jury encore perplexe au jour de la rédaction de ce rapport.

Partie II. Construction d'un mot infini sans carré.

▷ **Question 7. [0,25 pt, 330 / 0.25 / 0.24]** Il s'agissait d'une question facile visant à familiariser les candidats avec les notions introduites dans cette partie, en construisant un mot infini $(c(i))_{i\in\mathbb{N}}$

sur $\{0,1\}$ où $c(i)$ est égal au bit qui est immédiatement à gauche du premier bit égal à 0 dans la représentation binaire de i. Cette question a été traitée sans problème, même si une partie non négligeable des candidats trahit sa lecture superficielle de l'énoncé, oubliant trop souvent de préciser les carrés demandés dans l'énoncé. Une erreur fréquente des candidats a consisté à confondre longueur d'un carré et période du carré.

▷ **Question 8. [0,5 pt, 310 / 0.48 / 0.43]** La question conduit à travailler sur la représentation binaire des entiers ; il s'agissait essentiellement de remarquer que le i^{e} chiffre des représentations binaires de n et $n+2^k$ étaient égaux pour tout $i<k$, ce qui assurait d'avoir la moitié des points de la question. Écrire correctement la représentation binaire de $2^{k-1}-1$ et en tirer la conclusion permettait d'obtenir le reste des points de la question.

▷ **Question 9. [1 pt, 270 / 0.83 / 0.65]** Cette question proposait de démontrer une relation sur les $c(i)$, utile pour la question suivante. On attendait un raisonnement sur l'écriture binaire de $i(d,k)$ et de 2^k, dont l'écriture binaire correcte apportait la moitié des points de la question, mais certains candidats ont résolu la question de manière satisfaisante par d'autres moyens.

▷ **Question 10. [0,5 pt, 273 / 0.45 / 0.36]** La question nécessitait d'appliquer, de manière assez directe, la question précédente : remarquer que $c(d)\dots c(d+2^k-1)=c(d+2^k)\dots c(d+2^{k+1}-1)$ et que $c(i(d,k))=c(i(d,k)+2^k)$ apportait la moitié des points, la conclusion correcte assurait d'avoir le reste des points de la question. L'absence de référence explicite à la question précédente a été pénalisée.

▷ **Question 11. [2 pt, 79 / 1.77 / 0.41]** La question demandait de montrer que le mot ne contenait aucun carré de période paire. Cette question plus difficile a été très peu traitée, la plupart des candidats sautant directement à la question suivante, et encore moins souvent traitée correctement. Il s'agissait d'utiliser le fait que tout nombre pair s'écrit, de manière unique, sous la forme $2^k\times(2l+1)$ pour pouvoir appliquer la question précédente. Les raisonnements ont trop souvent été incomplets ou non convaincants. Une copie a, de manière surprenante, établi le résultat en s'appuyant sur le fait que tous les entiers pairs seraient des puissances de 2...

▷ **Question 12. [1,5 pt, 137 / 1.36 / 0.54]** La dernière question visait à construire un mot infini sans carré à partir du mot construit précédemment, en doublant la taille de l'alphabet, ce qui permettait de stocker une information de parité dans les lettres et donc d'exclure les carrés de période impaire.

Cette question a été peu traitée, sans doute parce qu'elle arrivait en fin de partie alors qu'elle ne présentait pas de grande difficulté. Les candidats qui s'y sont essayés ont d'ailleurs plutôt été couronnés de succès. On obtenait la moitié des points en produisant une construction correcte et l'autre moitié en la justifiant de manière convaincante.

Partie III. Recherche efficace des carrés.

▷ **Question 13. [0,5 pt, 290 / 0.43 / 0.36]** La première question de cette partie demandait simplement d'appliquer la définition donnée en préambule de la partie. Une réponse qui contenait deux erreurs ou plus n'avait aucun point, une réponse contenant une erreur avait la moitié des points.

▷ **Question 14. [0,25 pt, 328 / 0.23 / 0.22]** La question proposait d'étudier le cas où la concaténation de deux mots u et v crée un nouveau carré centré sur v et d'obtenir des expressions sur les indices où un suffixe de u se retrouve dans v. Si la question ne pose pas de difficulté particulière lorsque les équations sont bien posées (d'autant que le sujet proposait un schéma illustrant la situation considérée), de trop nombreux candidats ont échoué, notamment sur des erreurs d'indices sur les mots.

▷ **Question 15. [2 pt, 287 / 1.07 / 0.89]** La question consistait à établir une condition nécessaire et suffisante sur l'existence d'un nouveau carré de uv centré sur v, ne dépendant que de la taille des plus longs suffixes de u apparaissant dans v et des plus longs préfixes de v apparaissant dans v à une position donnée.

Peu de candidats ont réussi à traiter correctement l'ensemble de la question, la plupart échouant sur la réciproque alors que le raisonnement pouvait être conduit par équivalence et non par double implication. Plusieurs candidats conduisaient avec aplomb des raisonnements clairement incorrects. Cette technique qui peut s'apparenter à du *bluff*, n'est pas à recommander : plus loin dans la correction, en cas de doute sur la compréhension du candidat, le correcteur ne sera pas enclin à accorder les points.

▷ **Question 16. [1 pt, 220 / 0.46 / 0.29]** La question proposait d'établir une condition nécessaire et suffisante similaire pour les carrés centrés sur u, mais la plupart des candidats n'ont pas su adapter la condition de la question précédente.

La formulation de la question permettait deux interprétations de ce que serait une "CNS similaire", interprétations que les correcteurs ont admises même si l'une d'elles était plus naturelle et utile pour la suite du sujet.

▷ **Question 17. [0,5 pt, 184 / 0.45 / 0.24]** La question 17, simple question d'application des CNS précédentes, permettait de vérifier la bonne compréhension des candidats (ou leur non-compréhension).

Les trois dernières questions de la partie supposent donnée une implémentation efficace des fonctions lmp et lms, qui feront l'objet de la quatrième partie du sujet.

▷ **Question 18. [1 pt, 153 / 0.76 / 0.34]** La question 18 demandait simplement d'utiliser la CNS de la question 15 pour programmer une fonction `nouveau_carre_u` en temps linéaire de la somme de la taille des mots considérés.

Aucun point n'a été accordé si le programme n'avait pas la complexité demandée. Si la question 15 était mal exploitée, la moitié des points était retirée. L'absence de justification de la complexité ou des erreurs de bornes des boucles faisaient perdre des points, les autres erreurs de programmation n'ont pas été pénalisées dans la mesure où les candidats arrivaient en général à la fin du temps de l'épreuve : le jury a donc plutôt évalué le programme de cette question comme du pseudo-code.

▷ **Question 19. [1 pt, 142 / 0.81 / 0.34]** La question demandait de programmer une fonction calculant les nouveaux carrés en temps linéaire.

Si la plupart des candidats ont bien utilisé les fonctions `nouveau_carre_u` et `nouveau_carre_v`, certains ont oublié de considérer le cas où le nouveau carré n'était centré ni sur u, ni sur v. Une copie a astucieusement traité ce problème en cherchant les nouveaux carrés centrés de la concaténation de uv_0 et de $v_1 \dots v_N$: même si on détecte ainsi trop de nouveaux carrés, le résultat final est correct et la complexité inchangée, ce qui a donc été admis.

▷ **Question 20. [1,5 pt, 183 / 1.21 / 0.65]** La dernière question de la partie consistait à utiliser une méthode *diviser pour régner* avec la fonction `nouveau_carre` pour obtenir une complexité en $O(n \times log(n))$ où n est la longueur du mot. Trop souvent, les algorithmes *diviser pour régner* proposés par les candidats ne terminaient pas par manque d'un cas de base.

Partie IV. Implémentation efficace des fonctions `lms` et `lmp`.

▷ **Question 21. [0,5 pt, 146 / 0.49 / 0.21]** Il s'agit d'une question d'application directe de la définition de l'énoncé. Chaque erreur retirait la moitié de points de la question. On déplore à nouveau un nombre significatif d'erreurs d'inattention ou indiquant un manque de vérification des réponses par les candidats.

▷ **Question 22. [0,25 pt, 146 / 0.13 / 0.06]** La question ne présentait pas de véritable difficulté, mais il a été fréquent qu'elle ne soit que partiellement traitée : oubli de la condition de maximalité, ou caractérisation vague du mot $v_f \dots v_g$. Dans le cas $g < i$, on a ainsi une caractérisation de $v_f \dots v_g$ comme préfixe maximal de v débutant à la position f.

▷ **Question 23. [2 pt, 42 / 1.17 / 0.14]** Dans cette question, on se place dans le cas où $g > i$ et on exprime $pref_i$ en fonction des valeurs précédentes de $pref_j$. Cette question a été très peu abordée et rarement traitée correctement. Par dessus tout, les rédactions lisibles étaient rarissimes et elles ont bénéficié d'un bonus. Il s'agissait d'une des questions les plus difficiles du sujet.

▷ **Question 24. [1,5 pt, 83 / 0.8 / 0.19]** La question visait à montrer qu'un programme réalisait effectivement le calcul du tableau $pref$, que son calcul terminait et d'en analyser la complexité.

Au sujet de la correction du programme, les copies ont assez rarement explicité les invariants de boucle nécessaires à une justification rigoureuse, ignorant notamment totalement d'intégrer les valeurs de f et g dans la correction. Au sujet de la terminaison, certaines copies semblaient avoir compris l'idée mais n'ont pas réussi à l'expliquer clairement. La justification de la complexité nécessitait de remarquer que la valeur de g ne faisait qu'augmenter et que, malgré les boucles imbriquées, on avait une complexité linéaire.

▷ **Question 25. [0,5 pt, 33 / 0.41 / 0.04]** La question demande d'implémenter en temps linéaire en la somme des longueurs des mots la fonction `calcul_lmp` qui, appliquée à deux mots u et v, renvoie un tableau d'entiers de taille $|v|$ tel que l'entier stocké à la position i soit égal à $lmp(u,v,i)$ pour $0 \leq i < |v|$. Pour cela, il suffit d'utiliser la fonction `calcul_pref` de la question précédente sur le mot $u\#v$ où le caractère # n'apparaît pas dans les mots u, v et d'extraire du résultat les $|v|$ derniers éléments.

Dans cette question, on n'attendait pas de justification particulière, simplement une mention de la complexité. Les erreurs typiques consistaient à utiliser les mots uv ou $v\#u$ au lieu de $u\#v$ et à oublier de faire appel à la fonction `calcul_pref`.

▷ **Question 26. [0,5 pt, 28 / 0.42 / 0.03]** La question 26 a été évaluée selon les mêmes critères que la question précédente. Elle demandait de programmer la fonction `calcul_suff` qui renvoie le tableau des suffixes. Pour cela, il suffisait d'appliquer `calcul_pref` au miroir du mot donné en entrée et de retourner le miroir du résultat. C'est en général ce dernier point qui a été oublié par les candidats (pris par le temps ?) qui renvoyaient le résultat de `calcul_pref` sans le renverser.

▷ **Question 27. [0,5 pt, 20 / 0.39 / 0.02]** La question 27 était très similaire à la question 25, notée selon les mêmes critères, ne présentait pas de surprise, pas plus que les solutions proposées dans les copies qui ont atteint ce point.