Banque MP inter-ENS – Concours Informatique
Session 2017

Rapport sur l'épreuve pratique d'algorithmique et programmation

• Écoles partageant cette épreuve :
ENS de Paris-Saclay, ENS de Lyon, ENS de Paris et ENS de Rennes.

• Coefficients (en % du total d'admission) :
  — Paris-Saclay: 13,2 %
  — Lyon : 12,7 % (pour les deux options)
  — Paris : 13,3 %
  — Rennes : 17,1 %

• Jury : Pierre-Évariste Dagand, Étienne Lozes, Marc Mezzarobba, Ocan Sankur.

Organisation de l'épreuve

Comme à l'accoutumée, l'épreuve se composait d'un travail sur machine d'une durée de 3h30, suivi d'une présentation orale d'environ 25 minutes.

Lors de l'épreuve pratique, les candidats doivent mettre en œuvre une chaîne complète de résolution d'un problème informatique, du choix d'algorithmes et de structures de données pour répondre à une spécification au calcul effectif de résultats numériques.

Juste avant la distribution des sujets, les candidats disposent d'une période de 10 minutes pour prendre en main l'environnement informatique proposé. Il s'agissait cette année d'un système GNU/Linux, basé sur Debian 7.11 (Wheezy) avec le bureau graphique Xfce, proche de celui des années précédentes. Les surveillants sont autorisés à aider, dans la mesure du possible, les candidats à surmonter les difficultés d'ordre purement pratique liées à l'environnement de travail. On ne saurait cependant trop recommander aux candidats de se familiariser à l'avance avec les logiciels (en particulier, l'environnement de développement et d'exécution des programmes) par exemple grâce à l'image disque ("live CD") disponible sur le site web de l'épreuve.

Une majorité de candidats programment en Python (Python 2 ou Python 3). Les candidats restants programment en Caml (Caml Light ou Ocaml) voire C++. Un candidat a remarqué la présence d'un interpréteur Ruby et choisi ce langage bien qu'il ne fasse pas partie de ceux officiellement proposés.

À l'issue de la partie pratique, les candidats rendent une « fiche réponse » qui liste les résultats numériques obtenus et une clé USB contenant une copie du code qu'ils ont écrit. Sauf cas exceptionnel, la partie pratique de l'épreuve est évaluée uniquement sur la base de la fiche réponse. La capacité à tester son code et corriger les bugs joue donc un rôle essentiel.

En plus des questions demandant des réponses numériques, les sujets comportent des questions de nature plus théorique auxquelles les candidats présentent leurs

solutions pendant l'oral, en s'aidant de leurs notes. La présentation orale vise à tester la bonne compréhension du sujet et le recul des candidats. Les examinateurs s'efforcent d'aborder toutes les questions que la candidate ou le candidat a préparées, et, suivant le temps disponible, des extensions de ces questions ou d'autres points du sujet. Remarquons qu'il est donc dans l'intérêt des candidats de présenter leurs réponses avec clarté et concision, afin de pouvoir traiter un maximum de questions orales dans le temps imparti. Si les examinateurs demandent souvent une brève description des algorithmes mis en œuvre pour répondre à telle ou telle question, il est là encore tout à fait exceptionnel qu'ils consultent le code remis sur la clé USB.

La partie orale de l'épreuve représentait 40% de la note totale. On observe dans l'ensemble, mais pas systématiquement, une bonne corrélation entre les résultats obtenus aux deux parties. Pour réaliser un bon oral, il est important de prendre le temps de réfléchir aux questions à préparer mentionnées dans le sujet, y compris si possible celles correspondant à des points non traités en pratique.

Remarques générales

Les examinateurs ont été surpris de découvrir que quelques candidats incluent la génération des objets sur lesquels on travaille, et notamment le calcul de la suite u_n définie et début de sujet, dans leurs estimations de complexité. Cela donne des réponses acceptables (lorsque l'estimation est correcte), mais trop compliquées par rapport à ce qui était attendu. Le rôle d'une analyse de complexité est de fournir une *estimation* du comportement du programme, cette estimation devra donc être raisonnable et intuitivement exploitable. Une formule alambiquée devra donc être traitée avec suspicion. Par ailleurs, nous rappelons que, dans beaucoup de sujets, il est crucial en pratique de stocker les u_n calculés plutôt que de les recalculer à partir de u_0 à chaque appel.

Nous conseillons aux candidats de réfléchir systématiquement à l'implémentation des algorithmes vus en cours en détaillant les structures de données et les opérations nécessaires. Un candidat qui n'est pas en mesure de proposer une implémentation (même au tableau) d'un algorithme du cours avec la complexité optimale n'a pas compris son cours. Ceci fut particulièrement flagrant dans le traitement de la QO1 du sujet 4 présenté ci-dessous.

Comme chaque année, nous rappelons qu'il est impossible de réussir cette épreuve sans une certaine pratique de le programmation. Il est indispensable d'être à l'aise avec le langage de programmation de son choix. Les candidats peuvent s'entraîner avec les sujets des années précédentes. Il nous semble préférable de s'entraîner avec un seul langage toute l'année et bien connaître ses subtilités, les erreurs classiques, les messages d'erreur, etc. Une des compétences nécessaires pour la réussite de l'épreuve est de savoir déboguer un programme. Plusieurs candidats ont été bloqués par diverses erreurs classiques telles que « Type Error » en Python qu'ils n'ont pas su interpréter, ou bien ont atteint la limite de la pile de récursion sans comprendre ce qu'il se passait. Une pratique régulière de programmation avec le langage choisi permet de rencontrer et savoir résoudre ce genre de problèmes classiques.

Diverses questions demandaient l'ordre de grandeur des paramètres pour lesquels un algorithme était applicable en pratique. Les candidats semblent avoir retenu que les fréquences des microprocesseurs actuels sont de l'ordre du gigahertz (comme rappelé dans un précédent rapport). Malheureusement, une immense majorité a tendance à se précipiter sur cette estimation pour juger complètement infaisable tout

calcul demandant plus de 10^9 opérations, ou à ne pas envisager qu'un algorithme dont le coût en temps et en mémoire est cubique puisse être plus contraint par la mémoire que par le temps de calcul.

Enfin, nous insistons sur le fait qu'une preuve de terminaison ne consiste pas nécessairement à faire décroître une certaine mesure positive vers 0 : comme ce fut le cas cet année dans le Sujet 4, il existe parfois une borne supérieure naturelle et autrement plus simple à vérifier.

Sujet 1 : La Poste sous l'austérité

Le sujet traitait une variante d'un problème classique en optimisation combinatoire : celui de choisir un sous-ensembles de sommets dans un graphe afin d'ouvrir un service (ici, des bureaux de poste) de sorte à minimiser le coût d'ouverture et la distance totale des clients vers le service le plus proche.

La première section proposait de construire des villes, c'est-à-dire les instances du problème, et les questions s'assuraient que la construction était faite correctement. Ces trois premières questions du TP ont été réussies par la quasi-totalité des candidats. La deuxième section s'intéressait à la recherche exhaustive de solutions. La Q4 mettait en œuvre une astuce pour énumérer simplement tous les sous-ensembles de $[1, m]$ et elle a été réussie par tous les candidats. La Q5 se servait de cette énumération pour choisir la solution optimale et elle a été réussie par la plupart des candidats. La Q6 était une question difficile qui incitait à réécrire un algorithme de recherche exhaustive par une fonction récursive, et à couper une branche de recherche si le minorant proposé était en dessous de la solution optimale vue jusque là. Un seul candidat a répondu correctement à cette question. La section 3 développait un algorithme d'approximation plus efficace mais sous-optimal. La Q7 demandait d'implémenter la première itération de l'algorithme donné dans le sujet, et la Q8 demandait le résultat final. Un quart des candidats ont bien répondu à la Q7, ont donc réussi à implémenter au moins la première itération de cet algorithme, et seulement 10 % ont réussi à l'implémenter entièrement, souvent avec des erreurs. La Q9 demandait de comparer l'algorithme exact et l'algorithme d'approximation ; un seul candidat a répondu à cette question. Enfin, la dernière section généralisait ces algorithmes dans le cas d'un graphe non-complet, mais n'a été traitée par aucun candidat.

La première question d'oral était un simple exercice de dénombrement et a été réussie par la plupart des candidats. La QO2 demandait la complexité de l'algorithme énumératif de la Q5. Les candidats avaient souvent compris le nombre d'itérations de cet algorithme, mais certains d'entre eux oubliaient le coût de calcul du coût des solutions énumérées. On rappelle que l'analyse de complexité doit être faite rigoureusement en comptant le coût d'accès aux structures de données et celui des appels récursif. La QO3 demandait de démontrer une simple inégalité sur le minorant proposé dans le sujet. Un tiers des candidats ont traité cette question. Certains candidats semblaient avoir été découragé par l'énoncé avant même d'avoir tenté de le démontrer. Celui-ci était pourtant simple une fois qu'on pensait à l'interpréter en termes de solutions. La QO4 était également une simple question de dénombrement et a été réussie par la plupart des candidats. La QO5 demandait à démontrer le lemme donné dans le sujet, qui permettait d'exécuter une itération de l'algorithme d'approximation en temps polynomial. La majorité des candidats ont réussi à démontrer ce lemme et ont compris qu'il fallait choisir les maisons les plus proches

de l'emplacement choisi. Cependant, plutôt que de trier la liste des maisons selon leurs distances à l'emplacement, beaucoup de candidats ont choisi de sélectionner incrémentalement la maison la plus proche à l'emplacement, ce qui revient à trier en temps quadratique. La QO6 considérait un exemple qui permettait d'établir que l'algorithme étudié n'admet pas de rapport d'approximation. Moins d'un tiers des candidats ont réussi cette question. La QO7 était une question ouverte qui demandait aux candidats de proposer des améliorations de la qualité de l'algorithme. Quelques idées simples étaient de basculer vers l'algorithme exact quand la taille de l'instance devient suffisamment petite, tenter d'améliorer localement la solution calculée en essayant d'ajouter ou d'enlever un petit nombre d'emplacements, ou bien modifier l'algorithme pour qu'un emplacement choisi puisse être choisi à nouveau avec un coût d'ouverture nul. 3 candidats ont traité cette question. Enfin, un seul candidat est arrivé jusqu'à la QO8 qui demandait de décrire l'algorithme de Dijkstra et sa complexité appliqué ici pour pouvoir généraliser ces algorithmes aux graphes généraux.

Sujet 2 : Marches à petits pas dans le quart de plan

Ce sujet s'intéressait aux marches sur $\mathbb{N}^2$ contraintes à n'utiliser qu'un sous-ensemble des huit pas vers les plus proches voisins, considérées d'un point de vue principalement combinatoire.

La première partie du sujet était un exercice de programmation, où il s'agissait pour l'essentiel d'implémenter correctement la spécification fournie. Les trois premières questions faisaient tirer aléatoirement des pas et des ensembles de pas, sur la base d'un générateur congruentiel linéaire. Les deux suivantes demandaient de simuler une marche aléatoire, et de collecter des statistiques sur les distances à l'origine atteintes par ces marches. Environ 85% des candidats ont abordé ces cinq questions, avec cependant des erreurs dès la question 2.

La première difficulté sérieuse arrivait en début de partie 2, où l'on demandait d'explorer exhaustivement les marches d'une longueur donnée. Seuls 60% des candidats y sont parvenus en pratique, et seuls 20% l'ont fait de manière vraiment efficace. Lors de l'oral, cependant, une large majorité voyait plus ou moins que l'on pouvait parcourir récursivement les marches suivant leur préfixes communs. L'analyse de la complexité de ce parcours a été diversement traitée.

Les questions 7 et 8 demandaient ensuite de dénombrer plus efficacement les marches en fonction de leur longueur. Cela pouvait se faire par programmation dynamique, en exprimant le nombre de marches de longueur n aboutissant en (i, j) en fonction du nombre de marches de longueur $n-1$ aboutissant à un des sommets voisins. Ces questions ont été abordées par moins de 20% des candidats, et seuls les meilleurs ont fait mieux qu'adapter leur code d'énumération exhaustive. Une bonne moitié, cependant, avait compris l'idée de l'algorithme efficace ou l'a comprise avec un peu d'aide lors de l'oral.

La dernière partie se fondait sur le fait que le nombre de marches de longueur n satisfait dans de nombreux cas une récurrence linéaire. Il s'agissait de remarquer que l'on peut alors « deviner » cette récurrence à partir des premiers termes de la suite en résolvant des systèmes linéaires (méthode dont des extensions ont conduit à de nombreux progrès récents en combinatoire des marches), puis de s'en servir pour dénombrer les marches en un nombre linéaire d'opérations arithmétiques. Cette partie n'a été traitée par personne lors de l'épreuve pratique, mais deux

candidats ont su démontrer la correction de l'algorithme d'Euclide étendu demandée en préliminaire.

Côté théorique, les petites questions de combinatoire présentes en début de sujet ont été dans l'ensemble bien traitées. Beaucoup de candidats se précipitent sur le lemme de l'étoile pour prouver qu'un langage n'est pas rationnel quand un raisonnement ad hoc un peu plus simple aurait fait l'affaire, et ont du mal à répondre quand on leur demande de détailler leur preuve. La question demandant de commenter les différences entre les statistiques de distance à l'origine collectées en simulant une marche aléatoire et en faisant une moyenne sur les marches de longueur donnée s'est avérée trop ouverte. L'idée principale que l'auteur du sujet avait en tête était de faire observer que simuler une marche aléatoire de longueur n en tirant à chaque étape un pas parmi ceux autorisés n'est pas équivalent à tirer uniformément parmi les marches de longueur n. Quelques candidats ont proposé des réponses plausibles et pertinentes.

Sujet 3 : Réseaux de comparateurs

L'épreuve portait sur des circuits composés de comparateurs. Les deux premières parties s'intéressaient à leur représentation sous forme de tableau et sur le calcul de divers paramètres (temps de calcul, profondeur, etc.) ainsi que la mise en oeuvre d'un réseau calculant le min. La deuxième partie introduisait le principe du 0/1 permettant de tester si un réseau de comparateurs était un réseau de tri. Les deux dernières parties étudiaient deux constructions efficaces de réseaux de tri.

Les questions de programmation pouvaient se traiter en une quinzaine de lignes, ce qui offrait la possibilité aux candidats de consacrer un temps non négligeable sur le raisonnement algorithmique et sur les questions d'oral. Certains candidats ont ainsi pu traiter des questions d'oral des parties 3 et 4 sans avoir complètement traité toutes les questions d'oral et d'écrit précédentes.

La première question de programmation non triviale était la question Q4: on demandait de calculer le temps de calcul d'une sortie d'un réseau de tri. Cette question a été traitée par 30% des candidats. Une approche récursive naïve suffisait pour traiter quelques instances, mais la dernière instance ne pouvait être traitée que par programmation dynamique avec un algorithme en temps linéaire. Seulement la moitié des candidats ayant traité cette question l'ont fait par programmation dynamique. À la question Q5, on demandait de calculer la profondeur d'un réseau de comparateurs. Un bon tiers de candidats ont été confus à l'oral sur la définition de la profondeur du réseau, d'autres n'ont probablement pas su l'implémenter. Par ailleurs, pour traiter toutes les instances de cette question, il fallait remarquer que la profondeur du réseau correspondait au temps de calcul maximal d'une sortie du réseau, qui pouvait se calculer en temps linéaire par programmation dynamique. Au final, 10% des candidats ont traité cette question, mais aucun n'a traité toutes les instances. La question Q6 n'a été traitée par aucun candidat: il s'agissait de trier les comparateurs par rapport à une clé faisant intervenir le temps de calcul, ce qui pouvait se faire en quelques lignes si l'on utilisait une fonction de tri de la librairie standard du langage utilisé. La question Q7 pouvait se traiter sans avoir traité la question Q6 en générant un réseau déjà trié, voire sans générer le réseau, par calcul du nombre de bits à 1 consécutifs en partant du bit de poids fort pour l'entier $c_N(\ell)$ maximal. Parmi les 10% de candidats ayant traité cette question, la moitié ont suivi cette deuxième approche. Les questions Q8 et Q9 ont été traitées

par un ou deux candidats: pour la première, il s'agissait d'énumérer un ensemble de vecteurs à l'aide de deux boucles imbriquées, pour la deuxième il fallait coder la construction par récurrence du fusionneur de Batcher. Pour la question Q8, il est un peu surprenant qu'une bonne moitié des candidats aient témoigné d'une bonne compréhension de ce qu'il fallait faire (si l'on se fie à l'oral), mais que très peu aient réussi à traduire leur idée en un code somme toute assez simple.

Les questions d'oral ont permis à certains candidats ayant peu programmé de démontrer une bonne compréhension du sujet.

La question QO1 demandait de détailler un algorithme exécutant un réseau de tri et d'en donner sa complexité. Cette question a été bien traitée dans l'ensemble, même si l'évaluation de la complexité de l'algorithme, dans quelques cas, ne tenait pas compte des opérations d'allocation mémoire et de recopie de tableaux. À la question QO2, il fallait exécuter un réseau de comparateurs et déterminer si il s'agissait d'un réseau de tri. La grande majorité des candidats ont répondu correctement à cette question, avec parfois un peu d'aide pour arriver à justifier leur réponse. La question QO3 demandait de définir un réseau de comparateurs qui calcule le minimum de ses entrées en temps parallèle logarithmique. Environ 30% des candidats ont répondu correctement à cette question, et environ 10% des candidats ont su argumenter sur l'optimalité de leur solution. La question QO4 s'intéressait à la preuve du principe du 0/1. Environ 30% des candidats ont su démontrer l'indication suggérée, et la moitié d'entre eux a su l'exploiter pour terminer la preuve. À la question QO5, il fallait donner un ensemble d'entrées permettant de tester par énumération si un réseau donné calcule la deuxième plus petite de ses entrées. Le principe du 0/1 a mis sur la bonne piste 50% des candidats, et le tiers de ces candidats ont justifié de façon satisfaisante la validité de leur réponse. La question QO6 demandait de justifier le principe du réseau de Batcher. Cette question difficile a été traitée correctement par quelques rares candidats qui ont compris comment exploiter l'indication qui était donnée. La question QO7 demandait d'étudier la complexité d'un fusionneur de Batcher. Bien que la question soit relativement abordables, peu de candidats ont essayé de la traiter. Les questions QO8 et QO9 étaient des questions de fin de sujet: seule une infime partie des candidats les a abordées, et rarement avec succès.

Sujet 4 : Automates, bisimilarité, équivalence

L'épreuve portait sur la manipulation des automates finis en implémentant des algorithmes permettant de tester le vide d'un langage rationnel, calculer l'intersection de deux langages, vérifier l'équivalence de deux automates d'abord par bisimulation, ensuite par équivalence de langage.

La première question non-triviale du TP (Q3) demandait de calculer le nombre d'états accessibles d'un automate fini. La plupart des candidats ont réussi à implémenter un parcours de graphe (mais pas toujours avec la complexité optimale). Ensuite, une série de questions demandait si un automate fini donné accepte un mot qui est aussi accepté par une expression rationnelle simple donnée (Q4), un autre automate fini (Q5), par k autres automates finis donnés. À notre grand étonnement, seuls quelques candidats ont su reconnaître qu'il suffisait de calculer l'intersection de langages rationnels pour répondre à ces questions. Ainsi, peut-être parce que le sujet ne mentionnait pas le mot intersection explicitement, et peut-être aussi parce que la clôture par intersection des langages rationnels avaient été admis comme un théorème mathématique sans contenu calculatoire, les candidats n'ont

pas réussi à faire le lien avec ce résultat du programme. Par ailleurs, nombreux étaient les candidats qui n'ont pas su démontrer la clôture par intersection des langages rationnels à l'oral. Il est désolant de devoir départager les admissibles aux ENS par des questions de cours. Beaucoup de candidats sont restés bloqués à ces questions alors que la section suivante était indépendante et le sujet les incitaient fortement à passer à la suite en cas de blocage.

La question Q7 demandait d'implémenter un algorithme donné dans le sujet, calculant les paires d'états bisimilaires. Aucune astuce n'était nécessaire pour réussir cette question, à part penser à inverser les arêtes de l'automate pour pouvoir calculer les prédécesseurs. Un quart des candidats l'ont réussi. Les questions Q8 et Q9 construisaient la version perturbée d'un automate et la comparait par bisimulation à l'automate original. Il s'agissait de calculer l'union disjointe de deux automates et appliquer Q7. Un seul candidat est arrivé jusqu'à la Q9. La dernière section qui portait sur l'équivalence de langage et n'a été traitée par aucun candidat.

La première question d'oral consistait à décrire un algorithme pour calculer le nombre d'états accessibles et a été réussie par la majorité des candidats. La plupart des candidats savaient en effet décrire les grandes lignes d'un algorithme et connaissaient la complexité optimale d'un tel parcours mais certains ont décrit un algorithme dont la complexité était plus élevée car ils n'utilisaient pas les bonnes structures de donnés. Une erreur typique était de stocker les sommets déjà visités dans une liste, et de parcourir cette liste à chaque appel récursif pour éviter de revisiter un sommet. Cela montre que ce résultat du cours n'avait pas été bien compris par ces candidats. La QO2 demandait de construire un automate fini qui reconnaissait l'expression rationnelle donnée, et a été bien réussie. Les candidats savaient donc construire un automate non-déterministe qui reconnaît un langage donné. Pour la QO3, plusieurs candidats ont voulu inventer un algorithme compliqué pour parcourir deux automates en même temps pour chercher un mot commun. Certains ont pensé à énumérer les mots reconnus par un des automates, et les tester dans l'autre. Aucune de ces tentatives n'a abouti en un algorithme. Seulement quelques uns ont pensé à calculer l'intersection de deux automates. Il ne semble pas acquis qu'un des intérêts de l'équivalence des langages rationnels et des automates finis est que toutes les opérations sur ces derniers sont effectives.

Les QO4 et QO5 de la section 2 ont été globalement réussies. Une partie de la QO5 et la QO6 ont été sélectives puisqu'elles demandaient de démontrer des propriétés non triviales sur la bisimulation et l'algorithme donné. La QO7 demandait de décrire un algorithme pour déterminer si un langage rationnel est inclus dans un autre, quand ceux-ci sont décrits par des automates finis. Il s'agissait de complémenter un des automates et l'intersecter avec l'autre automate. Cette question a été traitée par un seul candidat.

Sujet 5 : Très grands nombres

Le sujet traitait d'un système de numération trichotomique permettant de représenter de façon compacte certains très grands nombres ainsi que l'implémentation d'opérations arithmétiques efficace (en temps mais aussi en espace) sur ces objets.

À l'écrit, la première partie du sujet consistait, comme à l'accoutumée, à mettre en œuvre un générateur pseudo-aléatoire de très grands nombres et une fonction de signature (Q1 à Q5) à partir de spécifications fournies. Ces questions devaient également amener les candidats à choisir un type de donnée adapté aux nombres

trichotomiques, dont la nature fondamentalement récursive était mise en avant dans le sujet. Les quatres premières questions ont été traitées par environ 85% des candidats. La Q5 n'a été traitée que par environ 70% de candidats : cette question a été particulièrement sélective pour les candidats ayant choisi une mauvaise représentation machine des très grands nombres (comme, par exemple, utiliser les entiers à précision arbitrairement de Python).

Les Q6 à Q9 consistaient ensuite à implémenter les opérations arithmétiques élémentaires (décrémentation, incrémentation, addition, multiplication et comparaison d'entiers). Malgré le fait que la spécification de l'opération de décrémentation était fournie, seuls 50% des candidats l'ont traitée avec succès. Les Q7 et Q8, pour lesquelles il fallait déterminer puis implémenter les algorithmes d'incrémentation et de comparaison d'entiers, ont été abordées par environ 30% des candidats.

La Q9 proposait d'implémenter l'addition. On suggérait pour cela d'utiliser un opérateur de normalisation dont la spécification était donnée. Cela semble avoir incité les candidats à traiter cette question et la suivante (multiplication) en produisant une implémentation naïve suivi d'un appel à l'opérateur de normalisation : la complexité temporelle était alors catastrophique. Ainsi 10% des candidats ont abordés la Q9 et produit des résultats corrects.

La seconde partie abordait la question de l'efficacité mémoire et temporelle de la représentation trichotomique. Une analyse empirique (Q11) des nombres produits par le générateur pseudo-aléatoire suggérait fortement de représenter les grands nombres par des graphes plutôt que par des arbres tandis qu'un argument de complexité suggérait de mémoïser les calculs sur ces graphes. À cette fin, il était indiqué que le nombre de nœuds des graphes considérés était raisonnablement faible, ce qui permettait de mettre en œuvre le partage et la mémoïsation sans avoir à connaître les tables de hachage (hors programme). Afin d'exercer cette représentation améliorée, les Q12 à Q14 consistaient à implémenter des opérations arithmétiques fonctionnant sur de très (très) grands nombres, culminant avec l'arithmétique signée. De cette seconde partie, seule la Q11 a été partiellement traitée par un candidat et ce pour un nombre relativement petit.

À l'oral, la majorité des candidats a eu l'opportunité de traiter les QO1 à QO5, avec un succès variable. Environ 20% des candidats ont pu aborder la QO6 tandis que les questions suivantes n'ont généralement pas été traitées. La QO1 portait sur la complexité spatiale de la représentation binaire des nombres : le jury fut soulagé de constater l'aisance avec laquelle quasiment tous les candidats ont résolu cette question. La QO2 demandait simplement de vérifier l'unicité d'une décomposition et exploitait le théorème de la division euclidienne. Au vue de la simplicité de la preuve, le jury a été particulièrement exigeant quant à la clarté de la démonstration : seulement la moitié des candidats se sont vus attribuer la note maximale sur cette question, ce que l'on pourra regretter. La QO3 s'intéressait, essentiellement, à l'étude de la profondeur dans le pire cas de la représentation trichotomique d'un nombre et demandait de produire une suite de nombres réalisant cette borne. Cette question a permis d'identifier les candidats ayant bâti une intuition solide des objets informatiques manipulés, moins de la moitié d'entre eux ayant obtenu la note maximale sur cette question. La QO4 visait à vérifier que les candidats avaient une intuition de la taille des nombres manipulés dans le sujet ainsi que des contraintes spatiales posées par de tels nombres. L'application numérique a rarement posé soucis. Cependant, certains candidats semblaient surpris que l'on

puisse représenter un nombre sur plus de 64 bits, ce qui les a empêché d'aborder sereinement cette question. Les QO5 à QO7 demandaient aux candidats de décrire l'algorithme implémentant les opérations arithmétiques (incrémentation, addition et multiplication). Si la QO5 a été traitée par 75% des candidats, moins de 10% a fait une présentation et une analyse de complexité correctes. En effet, l'algorithme met en œuvre la fonction de comparaison de grands entiers dont la complexité n'est *a priori* pas $O(1)$, comme c'est généralement le cas pour les entiers machines. La QO6 a été abordée par peu de candidats (20%) mais généralement réussie. Un seul candidat a traité la QO7 (parfaitement, de surcroît). La QO8 proposait, à partir de résultats empiriques, d'améliorer la représentation trichotomique introduite au début du sujet. Ce question devait orienter les candidats vers une représentation basée sur les graphes plutôt que des arbres afin de partager les sous-arbres redondants. Moins de 10% des candidats ont pu aborder cette question, de façon majoritairement satisfaisante. Les QO9 et QO10 ont été traitées par une poignée de candidats qui ont majoritairement su faire l'analyse de complexité en QO9 et proposer des améliorations, notamment la mémoïsation des calculs sur la structure de graphe.