ÉPREUVE PRATIQUE D'ALGORITHMIQUE ET DE PROGRAMMATION DU CONCOURS COMMUN DES ÉCOLES NORMALES SUPÉRIEURES — RAPPORT DE JURY 2016

Écoles concernées : Cachan, Lyon, Paris, Rennes
Jury : Pierre-Évariste Dagand, Marc Mezzarobba, Michaël Rao, 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.

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. 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 des questions qu'elle ou il n'aurait pas eu le temps de traiter complètement. Pour réaliser un bon oral, il est important de prendre le temps de réfléchir aux questions à préparer mentionnées dans le sujet et de préparer suffisamment de notes au brouillon pour être capable d'exposer clairement les solutions au tableau.

Juste avant la distribution des sujets, les candidats disposent d'une période de 10 minutes pour se familiariser avec l'environnement informatique et poser des questions aux surveillants s'ils rencontrent des difficultés d'ordre pratique. L'environnement de travail était un système GNU/Linux, basé sur Ubuntu 14.04 avec le bureau graphique Xfce, proche de celui de l'année précédente. Si aucun candidat n'a rencontré de problème technique majeur pendant le travail sur machine, certains semblent avoir perdu du temps par manque de familiarité avec les logiciels. Le jury encourage les candidats à se préparer à l'épreuve en utilisant un environnement similaire, par exemple grâce aux images iso disponibles sur le site web de l'épreuve.

Le langage Python, autorisé depuis l'an dernier, est cette année le plus fréquemment choisi, légèrement devant Caml (presque toujours Caml Light, rarement OCaml) et loin devant C++ ou C. En moyenne, les candidats qui programment en Python obtiennent de meilleurs résultats à la partie sur machine que ceux qui choisissent Caml ; cependant, les toutes meilleures notes reviennent souvent aussi à des candidats utilisant C++ ou Caml.

La partie pratique de l'épreuve représentait cette année de l'ordre de 60% de la note finale, le coefficient exact variant suivant le sujet. On observe dans l'ensemble, mais pas systématiquement, une bonne corrélation entre les résultats obtenus aux deux parties.

Remarques générales

Les sujets sont longs et difficiles ; il n'y a pas lieu d'être inquiet de ne pas arriver à la fin.

Comme l'année précédente, plusieurs sujets comportaient des questions demandant d'estimer l'ordre de grandeur d'un temps de calcul. Les remarques à ce propos semblent avoir été entendues : une large majorité de candidats est capable de donner une estimation réaliste de la puissance de calcul d'un ordinateur personnel, avec cependant un recul variable sur les conséquences de cette estimation.

Les questions de dénombrement, qui revenaient elles aussi dans plusieurs sujets, posent en revanche toujours beaucoup de problèmes aux candidats.

Sujet 1 : Awalé

Ce sujet demandait de « résoudre » une version simplifiée du jeu d'Awalé. La première partie portait sur l'implémentation de l'usuel générateur pseudo-aléatoire ainsi qu'une fonction (efficace, idéalement) calculant les coefficients binomiaux sur un domaine restreint. Dans les deux cas, il était crucial de stocker en mémoire les résultats afin de ne pas constamment les re-calculer par la suite. La seconde partie visait à établir, par le truchement d'un algorithme glouton, une représentation compacte des configurations de l'Awalé. Cette représentation servait à la fois à la production aléatoire de configurations et à leur identification par les candidats. La partie suivante modélisait la dynamique de jeu : règles de capture, fin de jeu et calcul du score. Si, algorithmiquement, cette partie ne présentait pas de difficulté particulière, il s'agissait de traduire correctement les opérations spécifiées dans le sujet. Il était donc essentiel de tester ces opérations sur des cas limites afin de s'assurer de leur correction avant de les mettre en œuvre dans des algorithmes plus complexes. Enfin, la dernière partie s'attachait à déterminer si le premier joueur ou son adversaire dispose d'une stratégie gagnante grâce à une analyse rétrograde.

Tous les candidats ont traité les questions « pratiques » 1 à 4, sans erreur pour environ la moitié des candidats. Les questions 5 à 8 ont permis de départager les candidats tandis que la question 9 distingue les bons candidats (un candidat sur six). Globalement, lorsque les candidats ont répondu à une question, celle-ci était complètement juste : on apprécie donc le fait que les candidats ont vraisemblablement exploité les valeurs témoins afin de s'assurer de la correction de leurs implémentations, se censurant en cas de d'erreur. Seuls deux candidats ont réussi à traiter la question 10, laissant ainsi les deux dernières questions sans réponse.

Concernant l'oral, les candidats ont généralement traité les questions 1 à 6, une faible minorité abordant partiellement les questions 7 à 9. La question 1, portant sur la combinatoire des configurations d'Awalé, a permis à l'imagination débordante des candidats de s'exprimer, et ce malgré les indications du sujet et les remarques des examinateurs. Nous recommandons vivement aux candidats de se forger une intuition solide de quelques problèmes de dénombrement classiques (permutations, arrangements et combinaisons ; avec et sans répétitions) puis de tenter de résoudre ce type de question par réduction vers l'un de ces cas. Sans surprise, la question 3 a similairement été maltraitée par la majorité des candidats. À notre grand soulagement, la preuve de correction de l'encodage (question 4) et les études de complexité (question 6 et 8) ont été généralement bien traitées, tant sur le plan théorique que sur le plan de la faisabilité pratique. Aucun étudiant n'a

cependant eu le temps de traiter la question 7, portant sur la correction de l'analyse rétrograde.

Le candidat ayant obtenu la meilleure note a atteint, sans erreur, l'implémentation de la phase d'initialisation de l'analyse rétrograde (question 10) et a pu partiellement traiter les dernières questions orales sur la base de ses intuitions.

Sujet 2 : Contraintes et planification

Le sujet portait sur la manipulation des systèmes d'inégalités linéaires simples et sur leurs applications. Il s'agissait d'une classe de systèmes restreints aux contraintes de différences du type $x - y \leq k$ où x et y sont des inconnues, et k une constante. Quand on se donne n inconnues, ces contraintes peuvent être rangées dans un tableau, et représentent l'ensemble des solutions qui satisfont l'ensemble des contraintes. La première section traitait la satisfiabilité de ces systèmes (c-à-d l'existence d'une solution) ainsi que le calcul d'une solution minimale pour un ordre lexicographique défini. La deuxième section était une application sur un problème d'ordonnancement dont la solution optimale s'exprimait par des contraintes linéaires sur des inconnues en utilisant les algorithmes de la section précédente. Enfin, la dernière section traitait d'un problème de planification où on cherchait un sous-ensemble d'actions non contradictoires à effectuer, étant données des contraintes logiques entre elles, mais aussi des contraintes de temps linéaires.

Les questions de 1 à 4 ont été traitées correctement par la majorité des candidats : il s'agissait de construire des systèmes de contraintes, déterminer si une valuation donnée est une solution, et décider si un système de contrainte donné est satisfiable (admet une solution). La question sur la satisfiabilité (question 4) se résolvait par l'algorithme de Floyd-Warshall, qui est au programme des classes préparatoires, et dont le pseudocode était reproduit dans le sujet. La plupart des candidats ont su le programmer. La question 5 qui demandait à calculer la forme canonique d'un système de contraintes se résolvait avec exactement le même algorithme que la question 4, mais seulement 65% des candidats l'ont réussi, contre 84% pour la question 4. La question 6 demandait à calculer une solution positive minimale pour l'ordre lexicographique, et était une application directe du théorème 3 qui était admis. Il s'agissait d'une question qui demandait un peu de réflexion avant de commencer à programmer. Tous les éléments étaient là pour déduire qu'il suffisait de calculer la plus petite valeur possible pour chaque variable, dans l'ordre, tout en recanonisant après chaque étape. Seulement 2 candidats ont cependant réussi à répondre correctement à cette question, et 2 autres ont répondu partiellement. Il a été remarqué à l'oral que la plupart des candidats n'avaient pas lu ce théorème attentivement, ou bien n'avaient pas cherché à l'interpréter. La question 7 était indépendante du reste et était relativement facile, un tiers des candidats l'ont traitée correctement. Un seul candidat a traité le reste des questions et est arrivé jusqu'à la question 11.

Les 2 premières question d'oral portaient sur la satisfiabilité d'un système de contraintes et ont été globalement bien réussies. La question d'oral 3 demandait comment modifier un système de contrainte pour se restreindre aux solutions positives. 63% des candidats ont bien répondu à cette question, mais elle a déstabilisé le reste des candidats. Ceux-là n'avaient pas l'air d'avoir saisi qu'un système de contraintes représentait un ensemble de points dans l'espace, et qu'on pouvait se restreindre aux solutions positives en ajoutant quelques contraintes de positivité

supplémentaires. La question d'oral 4 demandait à justifier l'algorithme de la question 6 qui calcule une solution positive minimale. Un seul candidat a eu la bonne réponse immédiatement. Plusieurs candidats ont trouvé la solution quand ils ont été invités à réfléchir sur le théorème 3. Les questions d'oral 5 et 6 ont été abordées par une partie des candidats avec une réussite mitigée. Peu d'entre eux avaient regardé la section 2, et certains ont pu expliquer comment lire le temps de terminaison de la solution optimale au problème d'ordonnancement immédiatement dans le système de contraintes. D'autres candidats ont proposé diverses façons de chercher le temps de terminaison optimal de manière itérative, ce qui donnait une complexité peu avantageuse. Peu de candidats ont remarqués que le problème pouvait être représenté par un graphe acyclique, et qu'on pouvait calculer la solution optimale en temps linéaire dans la taille de ce graphe (donc en $O(mn)$), qui était la bonne réponse à la question d'oral 6. La question d'oral 7 était indépendante du reste, et demandait à formuler une récurrence pour exprimer la plus longue sous-suite croissante d'une suite. Une forte indication était déjà donnée dans le sujet. Ceux qui ont eu le temps d'aborder cette question, c'est-à-dire un tiers des candidats, ont plus ou moins réussi à trouver la solution. Un seul candidat a eu le temps d'aborder les questions d'oral 8 et 9 qui cherchaient à justifier les algorithmes de la section 3.

Conseils. On rappelle aux futurs candidats qu'il est indispensable de faire plusieurs sujets des années précédentes avant de se rendre à l'épreuve. En effet, chaque année, de nombreux candidats sont confrontés aux même problèmes. Voici quelques grands classiques :

- La plupart des sujets commencent par la construction d'une suite récurrente u_n. Par définition, le calcul de u_n se fait en temps $O(n)$. Plutôt que créer un tableau de taille suffisante et le remplir en temps linéaire, beaucoup de candidats calculent ces valeurs-là par récursion sans les stocker dans un tableau. Ainsi calculer u_i pour $1 \leq i \leq n$ prend un temps $O(n^2)$ plutôt qu'un temps linéaire. Les algorithmes deviennent alors trop coûteux.
- Dans le calcul par récursion de u_n en Caml, plusieurs candidats se sont heurtés à la profondeur limitée de la récursion, et obtenait une exception "Stack Overflow". En effet, cette profondeur est bornée sauf pour les fonctions sous la forme dite *récursives terminales*. Le problème apparaît dans la plupart des sujets depuis 2002, donc les candidats pouvaient s'en rendre compte en s'entraînant avec quelques sujets précédents.
- Pour reprendre une remarque du rapport du jury de l'épreuve de 2007, plusieurs candidats ne savaient pas comment créer un tableau à deux dimensions (une matrice) en Caml. Ils ont remarqué pendant l'épreuve que `Array.make n (Array.make m 0)` crée un tableau dont tous les éléments pointent vers un même tableau, et qu'en modifiant une ligne, on modifie donc toutes les lignes.

On remarque qu'il n'est pas possible de réussir cette épreuve en se satisfaisant des TP de programmation qui accompagnent les cours d'informatique dans les classes préparatoires. Comme c'est le cas, par exemple, pour les épreuves de math, un investissement personnel des candidats et un entraînement régulier sur les sujets des années précédentes semblent nécessaires.

Sujet 3 : Découpage en lignes d'un paragraphe de texte

Ce sujet portait sur le découpage en lignes d'un texte, avec la possibilité de couper un mot en fin de ligne par un tiret. La première partie demandait d'implémenter un

modèle (un peu compliqué) de production aléatoire de texte ; elle ne comportait pas de difficulté algorithmique particulière. La partie suivante pouvait être résolue en programmant, outre les règles de césure décrites dans le sujet, un algorithme glouton de découpage en lignes et un algorithme dynamique pour compter les découpages possibles. La partie 3 s'intéressait à une version simplifiée de l'algorithme de mise en forme de paragraphes de Knuth et Plass, également à base de programmation dynamique mais plus complexe. Enfin, la dernière partie demandait comment mettre à jour efficacement le découpage glouton d'un long texte après insertion d'un court fragment, ce qui est faisable par un algorithme relativement simple mais délicat à programmer correctement. Trois questions intermédiaires portaient sur les problèmes de dépassement et d'erreurs d'arrondi dans le calcul « numérique » sur ordinateur, avec notamment des rudiments d'arithmétique décimale à virgule fixe.

Tous les candidats ont abordé au moins les questions « pratiques » 1 à 4, deux candidats sur trois les traitant complètement sans erreur. Les questions 5 à 7 ont été les plus discriminantes. Si plus de la moitié des candidats a résolu au moins une partie de la question 6, moins d'un candidat sur quatre l'a traitée complètement. Un peu plus d'un sur huit a donné une réponse correcte pour la première des instances de la question 7 (qui pouvait être traitée à la main), mais un seul est parvenu à mettre en œuvre un algorithme suffisamment rapide pour résoudre le reste de la question. Un quart des candidats a répondu correctement à une partie des questions introductives de la partie 3 (questions 7 et 8). Les deux dernières questions, nettement plus difficiles, n'ont jamais été traitées correctement.

En ce qui concerne les questions à développer pendant l'oral, pratiquement tous les candidats ont abordé, spontanément ou avec l'aide de l'examinateur, les questions des deux premières parties (1 à 5) et souvent la question 6, certains les suivantes. La question 2 a été presque universellement bien traitée. Toutes les autres ont posé des difficultés à au moins un quart des candidats. Les questions 4 et 5, qui demandaient d'expliquer et analyser les algorithmes dont la mise en œuvre faisait la différence dans la partie programmation, ont été diversement traitées. À la question 4, de nombreux candidats présentent de façon à peu près convaincante une solution en temps linéaire à nombre de colonnes fixé, mais peu voient que l'on peut obtenir une complexité linéaire en n indépendamment de c. Beaucoup n'ont qu'une solution exponentielle à la question 5. Par ailleurs, certains candidats ne comprennent pas ce qui était demandé dans la question 1 ou semblent avoir été désarçonnés par la formulation un peu ouverte de la question 3.

Le candidat qui a obtenu la meilleure note avait traité presque parfaitement les trois premières parties, y compris l'idée de l'algorithme de justification mais à l'exclusion de sa mise en œuvre (question de programmation 10).

Sujet 4 : Le castor affairé

Le sujet demandait de simuler l'exécution de machines de Turing, puis de programmer divers heuristiques afin de détecter qu'une exécution ne s'arrête pas, dans le but de trouver des machines s'arrêtant après un nombre maximum de pas de calculs. Il s'agit du problème du « busy beaver », ou « castor affairé ».

La première partie (questions 1 à 4) concernait la simulation d'une exécution. Elle ne comportait pas de difficulté particulière, et environ deux tiers des candidats ont parfaitement réussi ces questions. On peut noter que quelques candidats ont eu des difficultés pour créer correctement des tableaux de dimension 2 en Caml.

La partie suivant (questions 5 à 8) demandait d'implémenter des tests, partiellement indépendants, de détection de non-terminaison. Le premier test (questions 5 et 6) cherchait les états qui ne peuvent plus accéder à un état final dans le graphe des transitions. Il s'agissait d'un algorithme simple sur les graphes orientés. Environ la moitié des candidats ont réussi la question 5, et un tiers la question 6. Le deuxième test (question 7) cherchait à détecter les cycles dans les suites de configurations, et le troisième (question 8) cherchait à détecter les transitions inaccessibles par une recherche inverse. Seuls 3 candidats ont réussi la question 7, et personne n'a abordé la question 8.

La dernière partie (questions 9 et 10), plus libre et non guidée par la fiche réponse type, demandait de trouver des « castors affairés », c'est à dire des machines qui s'arrêtent après un nombre maximum de pas d'exécutions, ou avec le nombre maximum de « 1 » écrits sur la bande. Cette partie pouvait être traitée sans avoir programmé tous les tests de détection de non-terminaison. Deux candidats ont abordé cette partie.

Comme souvent, une partie importante des candidats a éprouvé des difficultés pour de simples dénombrements (question orale 1, où il était demandé de calculer le nombre de machines de Turing à n états). Les questions orales 2 et 3 n'ont pas posé de difficultés particulières. La question orale 4, où il était demandé d'expliquer comment simuler une machine de Turing à partir d'une autre, et la question orale 5 qui abordait des questions sur les graphes, ont naturellement été moins bien traitées. À partir de la question orale 6, les questions orales ont été traitées sporadiquement.