EPREUVE ORALE D'INFORMATIQUE FONDAMENTALE
ENS : PARIS – LYON

***Coefficient* : 4**

MEMBRES DE JURYS : G. VILLARD, P. ZIMMERMANN

Ce document fait le point sur l'oral d'informatique fondamentale du concours commun d'entrée à l'ENS Cachan, l'ENS Lyon et l'ENS Paris. En 2002, les oraux ont eu lieu du 1er au 6 juillet, avec 81 candidats interrogés sur 14 sujets différents.

Remarques générales

Les points suivants sont apparus au cours de l'épreuve :

- **écrire un algorithme**: de nombreux candidats montrent des connaissances approfondies en algorithmique -l'un d'eux a cité l'algorithme de Knuth-Morris-Pratt, un autre a cité deux algorithmes de tri en O(*n* log *n)* dans le pire cas -, mais sont réticents à expliciter les algorithmes, et ont beaucoup de difficultés quand il s'agit de passer d'une formule mathématique à un véritable programme en pseudo-code. Certains ne semblent pas trouver dans quel ordre enchaîner les instructions, d'autres quelles constructions (boucle *pour* ou *tant que)* utiliser. Chaque exercice commençait ainsi par un « filtre » consistant à écrire un algorithme simple. Un grand nombre de candidats se sont empêtrés dans ce « filtre », notamment le changement de racine d'un arbre, la division de polynômes, ou simplement la division de deux rationnels avec arrrondi au plus proche ou l'évaluation d'un polynôme en un point. Un candidat a ainsi affirmé en partant «je pensais que ce serait plus théorique», car il n'a pas été plus loin que le « filtre » !
- **Caml versus pseudo-code** : plusieurs candidats essaient d'exprimer leurs algorithmes en langage Caml, et en cherchant à tout prix à respecter la syntaxe et la sémantique de Caml, rencontrent des difficultés qui nuisent à la lisibilité de l'algorithme, et en fin de compte les pénalisent. Par exemple le mécanisme de *pattern-matching* de Caml est mal adapté quand on doit dépiler les éléments d'une liste par la fin. Le style Caml est également malaisé pour remplir un tableau défini de façon récursive. Un symptôme est le syndrome du rec dans let rec, que les candidats s'empressent d'ajouter quand ils se rendent compte que leur programme est récursif. Il est préférable pour cette épreuve d'écrire les algorithmes demandés en pseudo-code, quitte à inventer des constructions ou fonctions manquantes.

- expliquer : peu de candidats expliquent ce qu'ils font (ou essaient de faire) . Il est recommandé de détailler l'idée de l'algorithme avant de l'écrire, et tout en l'écrivant d'expliciter ce que fait chaque fonction ou procédure, et ce que représente chaque variable. Cela aide la compréhension de l'examinateur, mais aussi du candidat! On préfère en général un candidat qui s'engage sur une fausse piste, plutôt qu'un candidat qui ne dit rien du tout.

- exemples : très peu de candidats également essaient de « voir ce qui se passe » sur un exemple pour mieux comprendre le problème posé. Ceux qui le font déduisent en général très vite une solution algorithmique au vu de l'exemple. Ce problème fut flagrant sur le sujet « Réduction des réseaux », qui demandait de comprendre ce que faisait un algorithme donné : ceux qui se sont lancés dans des calculs sans regarder ce que faisait l'algorithme ont perdu un temps précieux.

- **calculs** : les examinateurs ont été surpris de constater qu'une bonne moitié des candidats n'était pas à l'aise dans des calculs mathématiques simples, comme $1+2+\cdots+n$ qui fait $\frac{n(n-1)}{2}$ selon au moins trois candidats. Un autre ne savait plus la somme d'une suite géométrique $1+x+x^2+\cdots$ Un troisième semblait bloqué par le calcul de la dérivée de $\prod f_i(x)^{e_i}$. Un quatrième prétend que $\sum_{i=0}^{k-1}\alpha_i$ donne α_0 pour $k=0$. De nombreux confondent encore poteaux et intervalles, et prétendent qu'une boucle qui va de d à $n+d-1$ s'effectue $n-1$ fois. Un candidat connaissait des formules très compliquées sur les coefficients binomiaux, mais pas la formule simple $\sum_i C_k^i = 2^k$! Au contraire, certains se plongent dans des calculs infinis (ou presque), si bien qu'il fallut leur dire « réfléchissez, ce n'est pas un oral de math ! »
- **récursivité** : certains candidats veulent à tout prix recourir à la récursivité, là où un algorithme itératif ferait très bien l'affaire, et en plus ont du mal à mettre au point leur algorithme récursif. Une déformation due à Caml ? L'un d'entre eux voulait même faire un algorithme récursif pour le rendu de monnaie en pièces de 1, 2, et 5 !
- **preuve par récurrence** : consommer avec modération. Un candidat a proposé de calculer la valeur de $f(x) \bmod (x-x_i)$ par récurrence ! Un autre a posé la division pour ce calcul (f est un polynôme) !
- **invariants** : certains font plusieurs appels à une même fonction avec les mêmes arguments, au lieu de sauvegarder les résultats intermédiaires, par exemple une inversion de matrice en $O(n^3)$ à chaque pas d'une boucle !
- **rigueur** : beaucoup de candidats ont une bonne intuition, mais ne cherchent pas à vérifier rigoureusement, ce qui leur joue des tours : au moins trois candidats ont prétendu que l'algorithme glouton pour le rendu de monnaie restait optimal pour des pièces ayant un rapport d'au moins deux entre elles.
- **complexité** : la plupart des candidats maîtrisent les calculs de complexité en $O(\cdot)$ des algorithmes. Cependant l'un d'entre eux a demandé si le coût indiqué de $O(nm)$ pour la multiplication de deux entiers correspondait à des entiers de n et m bits, ou de valeur n et m !
- **perles** : voici quelques morceaux choisis entendus en quelques 60 heures d'oral : « on peut trier n nombres en $O(\log n)$ grâce aux arbres tournois », « on peut calculer le pgcd de deux polynômes de degré n en $2n$ opérations », « on peut diviser deux polynômes de degré n en $O(n)$ », « on peut multiplier deux polynômes en temps constant », « la division d'un polynôme de degré m par un autre de degré n coûte $O(n^{m-n+1})$ », « peut-être qu'on peut faire arrêter l'algorithme une fois que le degré arrête de diminuer », « il faut déjà détailler la chose en langage naturel »...

Exemples de sujets proposés

Voici quelques exemples de sujets qui ont été proposés : occurrences de motifs (proposer et analyser différents algorithmes d'occurrences de motifs) ; maximum, minimum, médian (proposer et analyser différents algorithmes de recherche de l'élément maximum, minimum ou médian d'une liste) ; arbres et matrices (liens entre arbres, graphes, et leur représentation matricielle) ; produit court (analyse d'algorithmes pour le produit tronqué de séries formelles) ; décomposition sans carré (analyser différents algorithmes pour la décomposition sans facteur carré de polynômes) ; rendu de monnaie (proposer et analyser différents algorithmes pour le rendu de monnaie) ; calculs x-adiques (calcul de proche en proche de solutions d'équations en séries formelles) ; réduction des réseaux (analyse de l'algorithme de réduction de Gauss en dimension 2) ; évaluation, interpolation (calcul de polynôme par évaluation et interpolation) ; calcul récursif de pgcd ; plus long sous-mot commun (proposer et analyser des algorithmes de recherche du plus long sous-mot commun de deux mots donnés) ; translation (proposer et analyser des algorithmes de calcul de $P(x+1)$ où $P(x)$ est un polynôme) ; forme échelonnée (mise sous forme échelonnée de matrices) ; points dominants (détermination des points dominants d'un ensemble de points en dimension 2).