ÉPREUVE ORALE PRATIQUE D'ALGORITHMIQUE ET PROGRAMMATION
ENS : PARIS LYON CACHAN
COEFFICIENTS : PARIS 4 LYON 4 CACHAN 6
MEMBRES DE JURYS : Benoît Caillaud, Arnaud Legrand, Sylvain Pion

Comme l'année dernière, cette épreuve demandait aux candidats de mettre en œuvre la chaîne complète de résolution d'un problème informatique : analyse des spécifications, choix des structures de données, mise en forme et évaluation de l'algorithme et de sa complexité, programmation sur machine et test des programmes. En plus de la partie programmation, la présentation orale permettait d'évaluer les candidats sur leur capacité à expliquer leurs approches et solutions.

Le jury a examiné cette année 109 candidats, sur 5 sessions organisées comme les années précédente en « pipeline » : les candidats d'une session commune arrivent par groupe de 3 toutes les demi-heures, de sorte que les derniers arrivés rentrent avant que les premiers n'aient fini leur présentation orale. Une fois encore, le jury a fait un barème répartissant les points à égalité entre la partie « résultats pratiques » et la présentation des algorithmes à l'oral.

Comme l'année dernière, plusieurs langages et environnements de programmation étaient proposés :

- PC sous Windows XP, avec CAML Light, Maple, Java ou Pascal (Delphi).
- PC sous Debian/Linux gnome, avec CAML Light, Objective CAML, C, C++, Pascal, ou Java.

Il était demandé aux candidats de choisir le langage et l'environnement à l'avance. Encore une fois, CAML Light sous Windows a été très majoritairement choisi : 59,5 % des étudiants ont opté pour cet environnement même si c'est en forte baisse par rapport à l'an dernier (71%), 26 % des étudiants ont choisi CAML Light ou Ocaml sous Linux, 10 % ont choisi Pascal sous Windows, 1,8 % Pascal sous Linux et 2,7 % C/C++ sous Linux. Ni Java, ni MuPad, ni Maple n'ont été choisis cette année.

Plusieurs sujets proposaient plusieurs tailles d'instances de problèmes à résoudre. Le but était à la fois de permettre l'utilisation d'algorithmes « naïfs » peu efficaces pour résoudre la taille moyenne, et d'inciter les candidats à tester leur programme sur des petites valeurs solubles à la main (ce que beaucoup de candidats ont fait). Le jury rappelle une fois de plus l'importance pour les candidats d'apprendre à tester et corriger leur programme, en utilisant éventuellement des commandes d'affichage pour tracer l'exécution de l'algorithme.

Quelques remarques générales :

- Les candidats (surtout sous CAML) utilisent beaucoup les structures de liste chaînée (ou, plus généralement, des types de données récursifs). Si celles-ci peuvent être très utiles, elles peuvent aussi se révéler handicapantes et lourdes à gérer. Dès lors qu'ils ne sont pas inadaptés et que le nombre maximum d'éléments est connu, il vaut mieux préférer l'utilisation d'un tableau ou d'une matrice.

– De la même façon, l'utilisation de la récursivité est parfois très utile et de nombreux candidats maîtrisent bien cette technique de programmation, mais tous les algorithmes ne s'y ramènent pas. De nombreux problèmes peuvent se résoudre plus facilement à l'aide de boucles, et le jury encourage les candidats à programmer aussi des algorithmes itératifs en CAML.
– La grande majorité des candidats sont souvent capables d'évaluer la complexité de leur algorithme mais assez souvent cette complexité ne correspond pas à la réalité du code qu'ils ont produit. Beaucoup de candidats (même parmi les meilleurs !) rencontrent parfois des difficultés pour résoudre des problèmes impliquant des matrices 200 par 200 et semblent persuadés que c'est la taille du problème qui est en cause et non leur code. Le temps de calcul nécessaire pour traiter chacun des sujets par le code des examinateurs variait entre moins de 2 secondes et moins de 15 secondes, selon le sujet. Il était pourtant très courant de voir des candidats expliquer qu'au bout de 10 minutes, ils n'avaient toujours pas obtenu le résultat à l'*une* des questions (même si ces candidats avaient visiblement mis en œuvre un algorithme de complexité raisonnable en apparence).
  L'enseignement du fragment fonctionnel pur du langage CAML offrant un vision très pure et abstraite des objets, certains ne font pas vraiment le lien avec les problèmes de complexité, ce qui a des conséquences désastreuses sur leur capacité à résoudre des problèmes simples. Rappelons par exemple, qu'il est inutile de recalculer deux fois la même chose et que c'est bien souvent ce qu'il se produit lorsque l'on écrit une fonction récursive sans y prendre garde. Voici un exemple très représentatif d'une mauvaise utilisation de la récursivité pour traiter les premières questions du sujets (la distribution des n premiers éléments de la suite u modulo m ici) :

```
(* Première version : complexité quadratique, ce qu'il ne faut
 * pas faire. *)

let u0 = 37
let rec u n =
  if n <= 0 then u0 else (15091 * (u (n-1))) mod 64007
let v m n = (u n) mod m

let distribution m n =
  let c = Array.make m 0 in
  begin
    for k = 0 to n-1 do
      let i = v m k in
        c.(i) <- c.(i) + 1
    done;
    c
  end;;
```

```
distribution 31 50000 ;;    (* 50 secondes ! *)
distribution 31 1000000 ;;  (* Approximativement 5h30 de calcul ! *)

(* Deuxième version : complexité linéaire, ce qu'il était nécessaire
 * de faire pour espérer résoudre les questions avec des n grands. *)

let u0 = 37
let u = ref u0
let init () = u := u0
let suivant () = u := (15091 * ( !u)) mod 64007
let v m = ( !u) mod m

let distribution m n =
  let c = Array.make m 0 in
  begin
    init () ;
    for k = 0 to n-1 do
      let i = v m in
        c.(i) <- c.(i) + 1 ;
        suivant ()
    done ;
    c
  end ;;

distribution 31 1000000 ;; (* Moins de 0,15 secondes de calcul *)
```

Les sujets étaient longs mais assez bien dimensionnés puisque pour tous les sujet (à part celui sur les surfaces aléatoires), au moins un candidat à traité l'ensemble des questions (avec parfois des erreurs cependant). Néanmoins, tous les sujets disposaient de questions de difficultés croissantes permettant de tester les capacités des candidats. Voici maintenant des commentaires pour chacun des sujets :

Dératisation à Manhattan L'objectif de ce sujet était d'évaluer la capacité des candidats à programmer des algorithmes relativement simples (trouver un minimum dans un tableau, imbriquer des boucles, …) mais dont la complexité pouvait très souvent être améliorée, voire amortie. La plupart des candidats ont d'ailleurs assez bien réagi à l'oral aux problèmes de complexité amortie. La première partie s'intéressait au cas mono-dimensionnel et proposait la mise en œuvre d'algorithmes exacts. La seconde partie s'intéressait au cas bi-dimensionnel et proposait la mise en œuvre d'heuristiques plus ou moins coûteuses. On notera que 40% des candidats n'ont pas réussi à traiter correctement la question 2 (calcul de la taille de la colonie la plus en haut à gauche) qui était pourtant extrêmement simple ! À peu

près la même proportion de candidats ont également échoué à la question 3 (calcul de la position la plus intéressante) mais, curieusement, ce n'était pas forcément les même candidats. La moitié des candidats à réussi à traiter les questions 4 et 5 (respectivement calcul des deux meilleures positions et du nombre minimal de bombes nécessaires à l'éradication de tous les rats). La moitié des candidats on traité la question 6 mais seulement 20% d'entre eux l'ont traitée correctement. La partie 2 n'a été abordée que par un peu moins de la moitié des candidats, a été terminée par un candidat et presque terminée par un autre.

Réchauffement climatique Ce sujet proposait la mise en œuvre de listes à enjambements pour représenter des ensembles d'intervalles. Il n'était absolument pas nécessaire d'utiliser de vraies listes ou des pointeurs. Un simple tableau bidimensionnel suffisait amplement et la majorité des candidats ont d'ailleurs utilisé cette structure et émulé les pointeurs à l'aide d'indices. Un certain nombre de candidats semblent avoir été déstabilisés et n'avaient finalement pas compris le principe des listes à enjambement, même après 3h30 de réflexion. Les questions 2 à 4 ont été correctement traitées par seulement la moitié des candidats. La question 5 revenait simplement à rechercher le rang d'un élément dans un tableau et seulement 3 candidats ont répondu correctement à cette question ! Notons que le candidat qui a traité le sujet en entier avait commencé par développer une méthode *ad-hoc* pour répondre à la dernière question afin de vérifier les résultats qu'il aurait obtenu en utilisant les listes à enjambements. La première méthode lui a finalement permis d'obtenir les réponses pour les plus grandes instances des questions…

Automates pour rechercher Ce sujet était plutôt facile et pourtant les candidats ont été très décevants. Les statistiques indiquent que pour cette journée, le niveau moyen des candidats était inférieur à celui des autres journées. Les algorithmes à mettre en œuvre, à l'exception des questions 6 et 7, étaient de complexité au pire cubique sur un paramètre de petite taille et linéaire dans la taille du texte sur lequel s'effectuait les recherches. Notons que le code pour les questions 4 et 5 était facile une fois celui de la question 3 écrit. L'ensemble du code CAML (non particulièrement «raccourci») pour les questions 1 à 5 (générateur aléatoire, description de l'automate du sujet, simulateur d'automate, et constructeur d'automate) faisait moins de 200 lignes.

La guerre des étoiles Ce sujet avait pour cœur, la construction de couplages maximaux dans des graphes bipartis. Aucune connaissance particulière n'était cependant nécessaire puisque l'algorithme (glouton) était détaillé dans le sujet. On notera que seulement 65% des candidats ont réussi à répondre correctement à la question 2 (la construction de l'instance sur laquelle ils allaient travailler par la suite). Seulement 30% ont réussi à traiter correctement la question 3 (donc à résoudre des équations linéaires). Pour le calcul des instants critiques, les programmes d'un grand nombre de candidats itéraient sur les instants et utilisaient des propriétés de la suite u_n pour savoir quand s'arrêter au lieu de simplement résoudre

les quelques équations linéaires nécessaires. Seuls 2 candidats ont réussi à mettre en œuvre la construction d'une chaîne améliorante et seul un candidat a traité l'intégralité du sujet.

Surfaces aléatoires Ce sujet ne nécessitait que des parcours dans des tableaux et n'a pas déstabilisé les candidats même si, encore une fois, assez peu ont abouti. Seul 40% des candidats ont réussi à construire correctement la surface et à calculer des propriétés élémentaires comme la hauteur la plus basse, la plus élevée, etc. Beaucoup de candidats ont mis en œuvre pour la question 4 un algorithme de complexité cubique même si un algorithme simple de complexité quadratique existait mais cela n'a pas posé de difficulté majeure pour répondre à cette question. Pour les questions 5 et 6, beaucoup de candidats ont choisi d'énumérer les chemins, ce qui n'était pas raisonnable vu que le nombre de chemins croit exponentiellement avec la longueur.

Un petit bêtisier

- Un candidat a proposé d'utiliser une méthode dichotomique pour résoudre des équations linéaires.
- Un candidat a ré-implémenté la fonction modulo pour essayer d'améliorer les performances de ses algorithmes. Sans succès…
- Des complexités algorithmiques en «*tau* de n».
- Un candidat n'a utilisé absolument aucune structure de donnée. Il ne stockait rien et recalculait tout à chaque fois.
- Dans le sujet la guerre des étoiles, un des candidats n'avait pas réalisé qu'une planète pouvait être reliée à plusieurs planètes.
- Pour accélérer le calcul de la suite u_n un candidat a fait preuve d'originalité et a tenté d'utiliser une propriété particulière de la suite pour éviter le calcul des termes intermédiaires. Il n'y est pas parvenu en raison des problèmes de dépassement de capacité des entiers avec CAML. C'était de toutes façons totalement inutile puisque l'ensemble des termes était utilisés par la suite mais le jury a néanmoins apprécié l'effort.
- Un des candidats a découvert pendant l'épreuve à son désavantage que `Array.make n (Array.make m 0)` ne construit pas une matrice $n \times m$ mais un tableau de taille n dont chacun des éléments est le même tableau de taille m.

On peut trouver les 5 sujets proposés cette année sur le site web de l'épreuve :

`http://www.ens-lyon.fr/LIP/ConcoursInfo`