Informatique

Présentation du sujet

Ce sujet était constitué de deux problèmes complètement indépendants.

Le premier était issu d'un article de Karp, Miller et Rosenberg (1972). Les candidats devaient étudier un algorithme cherchant le plus grand facteur répété dans un mot. Ils devaient mettre en œuvre les qualités habituelles que nous cherchons à évaluer : comprendre un algorithme dont on présente les grandes lignes, préciser certains points, analyser les performances et programmer ledit algorithme.

Le second traitait une question très proche du cours, puisqu'il s'agissait de prouver que tout langage reconnaissable (par automate fini) est rationnel (le résultat est au programme mais pas sa preuve). Les candidats devaient prouver une maitrise raisonnable dans la manipulation des langages rationnels puis des automates.

Analyse globale des résultats

Nous avons été très agréablement surpris par la réaction des candidats face au premier problème : l'algorithme était présenté de façon beaucoup moins détaillée que par le passé et en comprendre le fonctionnement nécessitait vraiment une réflexion poussée. Des questions que nous avions peur de voir « bloquantes » ont en fait été correctement traitées par une proportion substantielle de candidats.

Dans le second problème, les copies montrent un spectre assez large de qualités et défauts, comme tous les ans, mais avec une petite amélioration sur le traitement du lemme de l'étoile.

Commentaires sur les réponses apportées, et conseils aux candidats

Identification de facteurs répétés

Signalons tout d'abord un point de vocabulaire : le terme « sous-mot » utilisé dans le problème était à prendre au sens usuel de « facteur » (lettres contiguës). Le vocabulaire n'est pas complètement fixé dans la littérature, mais « sous-mot » désigne majoritairement « mot constitué d'une suite extraite de lettres ». Quelques candidats ont signalé ce point. Sur les 1455 copies corrigées, nous n'avons pas vu le moindre contre-sens à ce sujet, mais bien entendu quelques candidats ont pu être troublés et perdre quelques minutes, nous en sommes navrés.

Nous avons choisi de faire programmer les fonctions permettant de manipuler des piles (pour faire ensuite des manipulations de piles et non de listes). Malheureusement, ces questions restent filtrantes : de trop nombreux candidats n'ont aucune idée de la façon de manipuler des objets d'un type enregistrement ; de nombreux candidats sont incapables de programmer la fonction `depile`, et beaucoup encore oublient de donner les types des fonctions programmées : en début d'épreuve, c'est réellement pénalisant (surtout pour ces questions autour de piles, où le type porte beaucoup d'informations sur ce que la fonction est censée faire).

Pour les questions I.C.2-3-4, une simple lecture de l'énoncé devrait permettre d'éviter des erreurs grossières d'analyse : les algorithmes linéaires sans structure annexe avaient peu de chance d'être justes au vu de l'ensemble des trois questions. Nous savons qu'il est parfois difficile pour un candidat de prendre du recul sur l'épreuve, mais il est de notre devoir de leur rappeler de le faire !

Pour le calcul de la classe E_{a+b} à l'aide de E_a (partie I.D), une vision d'ensemble était nécessaire pour aborder les premières questions : puisqu'il s'agit de stocker les classes d'équivalences dans un tableau de piles, il était intéressant de noter que le nombre de classes ne pouvait que diminuer de E_a à E_{a+b}. Nous avons tout de même compté juste les réponses à la question I.D.1 qui donnait pour majorant $b - a + 1$. Nous sommes favorablement impressionnés par l'attitude qu'ont eu beaucoup de candidats sur cette partie I.D : comprendre la façon dont on pouvait constituer les piles pour s'assurer à la fois de la E_a équivalence de (d, e) et $(d+b, e+b)$ n'allait pas de soi, mais a été comprise par plus de candidats que ce que nous envisagions. Ils ont été payés en retour ; particulièrement ceux ayant fait l'effort de programmer le résultat de leur analyse.

Alors même que la dernière question de ce premier problème est abordée dans une proportion importante de copies, nous sommes un peu surpris de voir que les approches dichotomiques sont finalement assez rarement évoquées, et encore plus rarement décrites précisément ! Les versions incrémentales depuis 2^{q-1} (ou décrémentale depuis $2^q - 1$) n'étaient guère satisfaisantes !

Langages et automates

Le fait que tout langage reconnaissable est rationnel est au programme, mais pas sa preuve, même si celle-ci est présentée dans beaucoup de classes. La preuve qui était proposée ici (du moins, qu'on pouvait déduire des outils mis en place), était suffisamment détaillée pour ne pas avantager outrageusement les candidats l'ayant déjà rencontrée.

Dans un premier temps, il s'agissait de manipuler des ensembles, parties, bijections : nous sommes un peu surpris des difficultés terribles qu'éprouvent plus de 10% des candidats avec ces notions qu'ils manipulent pourtant depuis le début de leur classe préparatoire, à défaut de les avoir rencontrées avant : il semble difficile de suivre avec profit un cours de mathématiques ou informatique post-bac avec de telles lacunes...

Pour le caractère irrationnel de L_P (question II.A.4), le lemme de l'étoile a été plutôt moins malmené que les années précédentes (et là, il était vraiment, sinon indispensable, très bien venu). Les énoncés « élaborés » où on choisit le facteur de longueur n (le nombre d'états) dans lequel on trouvera l'étoile est bien entendu plus puissante que la version élémentaire... mais en pratique, elle conduit beaucoup de candidats à tout mélanger et penser finalement qu'ils peuvent choisir eux-mêmes la position précise de l'étoile. Nous pensons que l'énoncé élémentaire est largement suffisant : si la preuve de cet énoncé est comprise, le candidat peut sans aucun mal l'adapter lui même si le besoin s'en fait sentir.

Dans les dernières questions, on calculait le langage associé à un automate en résolvant un système linéaire via le lemme d'Arden. Une approche classique consiste à définir comme on l'imagine des langages L_q associés aux différents états de l'automate, prouver qu'ils vérifient un système d'équations et en déduire les valeurs de ces langages (sens « unicité » d'Arden) puis du langage de l'automate. Mais ici, les X_q étaient introduits comme des (les) solutions du système d'équations. Si de nombreux candidats ont affirmé que X_{q_1} était le langage de l'automate, aucun ne l'a prouvé soigneusement avec un raisonnement du type : « les langages associés aux états vérifient telles équations ; dans la question précédente, on a trouvé une solution, mais il y en une seule solution, donc on connait finalement les langages associés aux états, puis à l'automate ». À ce sujet, les

calculs sont souvent faits correctement à la question II.B.7, mais le raisonnement mené (plus ou moins implicitement) est « si les X_q sont solutions, alors on a telles équations, donc on a telles autres équations... donc les X_q valent ceci ; on a donc trouvé une solution ». Il serait important que les candidats réfléchissent au problème que pose ce type de raisonnement sur une problématique pourtant essentielle : nécessaire/suffisant ; existence/unicité...

Conclusions

Cette année encore, une bonne connaissance du cours, une pratique raisonnable de la programmation et des capacités d'analyse acquises dans le cours d'informatique (mais aussi dans d'autres disciplines) ont permis à une proportion notable de candidats de produire de bonnes copies.

Nous encourageons les futurs candidats à produire les mêmes efforts. En complément des aspects théoriques développés en cours, un travail sérieux pendant les séances devant ordinateurs leur permettra pendant leur courte préparation d'acquérir une bonne partie des qualités qui leur fera réussir les épreuves d'informatique aux concours... mais aussi et surtout d'avoir un socle théorique et pratique en informatique qui leur sera bien utile dans leurs futures études et le métier qui suivra.