MPI

CONCOURS COMMUN INP

RAPPORT DE L'ÉPREUVE ÉCRITE D'INFORMATIQUE

1/ CONSIGNES GÉNÉRALES

Le sujet proposé comportait trois parties indépendantes. Le sujet était classant. La moyenne est de 10,12 et l'écart type de 3,62.

Le sujet est jugé par les correcteurs de bonne qualité, quelques 'typos' et imprécisions toutefois :

- en question 16, spécification de compare incomplète. Le barème a tenu compte de ces imprécisions ; les candidats qui ont traité cette question n'ont pas paru déstabilisés, d'autant qu'un certain nombre semblait connaître List.sort (dont la spécification était par ailleurs incorrecte) ;
- 'typo' dans les données D en deuxième ligne, deuxième colonne. Quelques candidats en ont fait la remarque mais cette typo n'était pas handicapante pour répondre aux questions du sujet ;
- dans l'algorithme 4, imprécision : il ne faut pas ajouter tous les arcs présents dans un circuit de G dans A mais uniquement ceux qui vont de V vers U. Cette imprécision a posé un problème à certains candidats pour répondre à la question 28 ; quoique la plupart aient eu assez de recul pour constater que suivre à la lettre l'algorithme produisait un résultat incohérent.

Il aborde des questions de programmation en OCaml et en C, ce qui permet aux candidats de montrer leurs compétences dans ces 2 langages de programmation. L'immense majorité des candidats a su aller chercher les points en allant traiter, par exemple, les questions d'application d'algorithme ou le code de la partie III. Les questions de code sont plutôt plébiscitées. Les questions de cours ou proche du cours ont souvent permis de bien classer les candidats.

2/ REMARQUES GÉNÉRALES

Erreurs les plus fréquentes des candidats

- Les arguments permettant de prouver la classe de complexité d'un problème ou la réduction polynomiale d'un problème à un autre sont incomplets.
- Confusion dérivation / arbre de dérivation.
- Le caractère strict de la décroissance du variant de boucle est souvent omis.
- Des raisonnements par disjonction de cas sont également incomplets.

Questions relatives au langage OCaml

- Non-respect du type de retour de la fonction.
- Oubli du point dans les opérations sur les flottants.

- Écriture de fonctions itératives au lieu de fonctions récursives lorsque c'est explicitement demandé.
- Mélange parfois obscur de traits impératifs avec de la récursivité. L'exemple suivant, issu de la **Q13**, illustre le propos :

```
let trouve_boite bl o =
  let a = ref 0 in
  let rec aux lst compteur = match lst with
    |[] -> !compteur
    |p::q -> if (1 -. p.charge -. o.charge) >= 0 then !compteur
             else compteur := !compteur +1;
                  aux q compteur
  in aux bl a
```

Questions relatives au langage C

- Notations relatives à l'accès d'un champ d'une structure via un pointeur (le point est utilisé au lieu de la flèche).
- Mauvaise gestion des pointeurs (allocation et libération de la mémoire).

Le nom des objets manipulés dans un programme a une importance : en particulier, *aux* n'est pas un nom de fonction pertinent pour les appels récursifs (remarque relevant de la partie II).

Les correcteurs font remarquer que les raisonnements qui affirment que les points compliqués à traiter sont « triviaux » et ne sont pas de ce fait abordés correctement ne constituent en aucun cas une preuve. À l'inverse, il est inutile de passer une page de rédaction pour montrer des points évidents ou faciles à traiter (par exemple la **Q5**).

Quelques copies étaient difficiles à lire à cause de l'écriture du candidat.

3/ REMARQUES SPÉCIFIQUES

PARTIE I

Q1. Question beaucoup traitée et classante. Quelques candidats n'effectuent pas de dérivations à gauche, d'autres proposent des arbres.

Q2. Q3. Q4. Questions beaucoup traitées, de manière correcte dans l'ensemble. Dans la **Q2**, quelques candidats proposent un arbre de dérivation qui n'en est pas un. Nous rappelons qu'un nœud interne dans un tel arbre a autant de fils que le nombre de lettres dans le membre droit de la règle qu'on souhaite utiliser.

Q5. Question régulièrement esquivée. De nombreux candidats ont produit une explication qui s'arrêtait souvent à la première difficulté, voire prétendaient que le raisonnement tenu pour montrer - typiquement - que les mots commençant par a étaient générés par G pouvait être recopié pour montrer que les mots commençant par b ou c l'étaient aussi.

PARTIE II

Dans les parties code de cet exercice, beaucoup de fonctions auxiliaires récursives internes ont été utilisées, plutôt que d'écrire directement une fonction récursive (ce qui était toujours possible pour toutes les questions du sujet).

Q6. Question beaucoup traitée et classante. Peu de candidats ont précisé que le certificat est de taille polynomiale en la taille de l'instance. Quelques candidats semblent penser que NP signifie ”non polynomial”.

Q7. Globalement, les candidats connaissent les étapes pour montrer une réduction mais certains n'ont pas trouvé la transformation, ou n'ont pas précisé qu'elle est calculable en temps polynomial.

Q8. Question modérément traitée, globalement de façon correcte. La justification de l'utilisation de boites individuelles n'est pas toujours donnée.

Q9. Généralement bien traitée quand elle est abordée. La justification du « +1 » est parfois omise. Un nombre non négligeable de candidats ne mentionne pas le cas particulier de la dernière boîte mais concluent correctement quand même.

Q10. Question très peu traitée.

Q11. Question beaucoup traitée. Erreur de syntaxe dans de nombreuses copies : mauvaise écriture du type enregistrement ou oubli du point indiquant un flottant. Quelques candidats confondent la syntaxe OCaml pour les enregistrements et la syntaxe C pour l'initialisation de structures.

Q12. Question beaucoup traitée. Un nombre non négligeable de candidats a traité cette question en considérant que les boîtes avaient des champs mutables et a donc écrit une fonction de signature objet -> boite -> unit plutôt que celle imposée par l'énoncé.

Q13. Généralement bien traitée. Dans certains cas, une fonction itérative a été proposée.

Q14. Question beaucoup traitée. Cette question a été l'occasion de distinguer les candidats comprenant la récursivité de ceux qui ne la comprennent pas : ces derniers ayant traduit "Si i est plus grand que la taille de la liste f, f d est ajouté à la fin” par des concaténations de f d en queue avec @ ou des utilisations erronées de List.rev.

Q15. Question modérément traitée. L'utilisation de la fonction *transforme* a quelques fois posé problème.

Q16. Question modérément traitée, généralement de façon correcte. L'énoncé ne précisant pas l'ordre dans lequel la fonction compare ordonne les objets, les candidats qui ont directement utilisé compare comme fonction de comparaison dans List.sort n'ont pas été pénalisés. De nombreux candidats ayant traité cette question ont en fait écrit leur propre fonction de comparaison, généralement de manière correcte.

PARTIE III

Cette partie a été moins traitée et beaucoup plus sujette au ”grappillage”.
Dans cette partie, les questions 18 à 27 mobilisaient formellement les compétences « justifier une solution, communiquer à l'écrit ». Elles ont pénalisé bon nombre de candidats.

Q17. Question beaucoup traitée, quasi systématiquement de manière correcte. Dans certaines copies, un élément de V refait une proposition à un élément de U.

Q18. Question beaucoup traitée, très souvent de manière erronée ou incomplète. Le variant utilisé par un nombre conséquent de candidats n'est pas toujours correct (pas de décroissance stricte).

Q19. Généralement bien traitée mais la formulation est parfois confuse.

Q20. Q21. Généralement bien traitées.

Q22. Généralement bien traitée quand elle l'est.

Q23. à Q25. Questions rarement traitées dans leur globalité. Plusieurs types de raisonnements sont proposés et souvent incomplets. Les candidats ne prennent pas toujours en compte que l'acceptation d'un élément *v* de *V* par un élément *u* de *U* peut être temporaire lors de l'exécution. De rares copies se sont distinguées par leur argumentation.

Q26. Les candidats ont compris la raison de l'existence du circuit mais la rédaction manque souvent de clarté. L'efficacité et la clarté avec laquelle les candidats abordent cette question est assez corrélée à la qualité globale de la copie.

Q27. à Q31. Bien traitées quand elles le sont. Sur la **Q28**, la principale erreur vient de candidats qui ajoutaient l'intégralité du circuit trouvé au couplage, ce qui est incohérent puisque cela ne produit - justement - pas un couplage.

Certains candidats n'ont pas traité les questions théoriques précédentes mais ont répondu aux questions suivantes relatives au codage en C.

Q32. Généralement bien traitée – De rares erreurs dans les indices du tableau *rang* lors du remplissage.

Q33. Certains candidats n'ont pas pris en compte le fait que le rang commence à 0. Environ la moitié des candidats qui ont traité cette question ont proposé d'initialiser USuiv[i] avec l'une ou l'autre des cases de Lu ou de Rang.

Q34. La phase d'initialisation est souvent omise. Malgré la question précédente et même chez les candidats l'ayant traitée correctement, aucun candidat ne pense à initialiser correctement les tableaux.

Q35. Question modérément traitée et modérément réussie. Certains candidats ne maîtrisent pas la syntaxe des pointeurs, ni leur manipulation. Beaucoup de candidats font les ajouts en fin de liste. Le sujet n'indiquait pas de contrainte de complexité, mais les correcteurs s'attendaient plutôt à des ajouts en tête. Bien entendu, rien n'a été pénalisé.

Q36. Question modérément traitée. La plupart des candidats ont compris qu'on attendait d'eux un parcours de liste mais très peu parviennent à l'implémenter tout à fait correctement. Le cas particulier du pointeur NULL et la libération de la mémoire sont souvent omis.

Q37. Q38. Rarement traitées mais généralement bien réussies. La conséquence du changement de numérotation des sommets a eu diverses interprétations pour l'utilisation de *Lu* et *Lv*.