Composition d'Informatique A, Filières MP-MPI (XULSR)

. La moyenne des 857 candidats français et internationaux de la filière MP est de 9,05/20 et l'écart-type de 3,52
. La moyenne des 320 candidats français et internationaux de la filière MPI est de 10,68/20 et l'écart-type de 3,71

1 L'épreuve

Dans ce sujet, on s'intéresse au problème des arbres équilibrés et à leur utilisation pour résoudre un problème de géométrie algorithmique.

La première partie introduit les arbres équilibrés dits AVL (du nom de leurs auteurs, Adelson–Velsky et Landis) utilisés dans le sujet et demande au candidat de démontrer des propriétés élémentaires de ces arbres.

La deuxième partie, très longue, s'intéresse aux problèmes algorithmiques liés à l'insertion et la suppression de nœuds dans l'arbre. La principale difficulté est l'étude de la fonction de rééquilibrage de l'arbre qui nécessite une analyse fine, tant du point de vue de la correction que de l'analyse de complexité.

La troisième partie s'intéresse à une application des arbres équilibrés pour résoudre un problème géométrique classique. Étant donné un ensemble fini de polygones dont les arêtes ne se croisent pas, on souhaite rapidement trouver pour un point donné dans quelle zone il se trouve.

2 Remarques générales

Il est rappelé que la présentation de la copie est importante. Trop de candidats écrivent de façon illisible ou raturent leurs réponses, alors qu'ils ont accès à des feuilles de brouillon. Même si la propreté de la copie ne joue pas dans la note, une réponse ne peut rapporter des points que si le correcteur arrive à la déchiffrer.

Lecture de l'énoncé. Le sujet proposé nécessitait d'être rigoureux dans la lecture du sujet afin de bien répondre aux questions. Par exemple, il fallait bien faire la distinction entre un arbre binaire de recherche et un AVL. Par ailleurs, de nombreuses questions contenaient plusieurs sous-questions ou demandaient plusieurs éléments. Par exemple,écrire un algorithme **et** analyser sa complexité. Un nombre bien trop important de candidats se contentent de répondre à la première partie des questions. Il n'est pas rare que la seconde partie d'une question représente entre 30%et 50% des points. Il est donc fortement recommandé aux candidats de bien relire les questions après y avoir répondu pour être sûr de n'avoir rien manqué.

Algorithmes. Trop de candidats semblent se jeter dans l'écriture d'algorithmes complexes sans réflexion préalable. Tous les codes attendus tiennent en moins de dix lignes (à l'exception de la question 22), voire moins de cinq lignes. Il est utile de rappeler que plus un code est long et moins il est probable qu'il soit correct. Pourtant, les correcteurs ont régulièrement vu des algorithmes faisant plus d'une page ! Ceux-ci ont rarement rapporté beaucoup de points.

OCaml. Les listes et les tableaux sont deux structures de données très différentes. Les structures de données utilisées dans les énoncés des questions n'ont pas été choisies pour rendre les réponses artificiellement compliquées, mais au contraire pour simplifier la vie des candidats. Malgré cela, trop de copies transforment des listes en tableaux ou vice-versa, ce qui conduit à des codes inutilement compliqués, voire faux. Par ailleurs, trop de candidats, probablement inspirés de Python, ne font pas la différence entre listes et tableaux et se permettent d'écrire `l[i]` avec `l` une liste, ou `x::t` avec `t` un tableau. De plus, ces opérations sont alors souvent considérées comme étant en temps constant, ce qui a évidemment été sanctionné.

Analyse d'algorithmes. L'analyse de code est souvent mal traitée par les candidats qui restent trop vagues. À l'exception de codes particulièrement simples, il est essentiel de faire référence dans l'analyse aux numéros de lignes du code en question. Par ailleurs, il n'est pas admissible de la part des candidats de traiter un seul cas de l'analyse puis d'expédier les autres « par un argument similaire » ! On rappelle par ailleurs que l'analyse d'une fonction récursive doit toujours se faire par une récurrence pour laquelle le candidat doit fournir un invariant. Trop de candidats négligent cet aspect important et « concluent par une récurrence » qui est souvent incomplète, voire fausse. Enfin, il est recommandé aux candidats de ne pas négliger le cas de base des inductions. Bien que celui-ci soit le plus souvent simple, trop de candidats se contentent d'affirmer sa trivialité, ce qui est peu convaincant.

Analyse de complexité. Le fait qu'une fonction effectue $O(n)$ appels récursifs n'implique en rien que la complexité temporelle totale soit aussi en $O(n)$. Encore faut-il pouvoir garantir que le nombre d'opérations élémentaires effectuées à chaque appel est borné. En particulier, la concaténation de liste n'est pas une opération élémentaire ; sa complexité est linéaire en la taille de la liste de gauche ! Lorsque l'analyse de complexité n'est pas immédiate, il est fortement recommandé aux candidats d'écrire l'équation de récurrence, puis de la résoudre. En effet, le candidat qui reste vague et se trompe dans la complexité finale se retrouve avec peu d'éléments dans sa réponse susceptibles de lui rapporter des points.

3 Commentaire détaillé

Pour chaque question, sont indiqués entre crochets le pourcentage de copies ayant traité la question et le pourcentage de copies ayant obtenu la totalité des points, en fonction des filières.

Question 1 [**MP : 100%, 67% ; MPI : 100%, 65%**] Il s'agit là d'une question très facile. Cependant, un nombre non négligeable de candidats ont oublié certaines symétries et listé seulement trois arbres.

Question 2 [**MP : 100%, 55% ; MPI : 100%, 56%**] Trop de candidats ont oublié de prouver une direction de l'équivalence. Par ailleurs, beaucoup de candidats n'ont pas utilisé la définition d'arbre équilibré du sujet (séquence infixe triée) et se sont contentés de preuves vagues, ce qui a évidemment été sanctionné.

Question 3 [**MP : 100%, 61% ; MPI : 100%, 59%**] Cette question a été bien traitée dans l'ensemble, mais trop de candidats se sont contentés de lister les arbres, alors qu'une justification est explicitement demandée dans la question.

Question 4 [**MP : 82%, 21% ; MPI : 82%, 22%**] Trop de candidats ignorent l'indication et affirment informellement que « le dernier niveau d'un AVL est au moins à moitié rempli et contient donc au moins 2^{h-1} nœuds ». En plus d'être fausse, une preuve à base d'arguments trop informels ne saurait rapporter des points. Les candidats ayant suivi l'indication du sujet ont généralement proposé une équation de récurrence correcte, même si elle n'est pas toujours correctement résolue.

Question 5 [**MP : 97%, 61% ; MPI : 100%, 72%**] Il s'agit là d'une question de programmation classique plutôt bien traitée. Toutefois, un trop grand nombre de candidats ne donne pas ou ne justifie pas la complexité alors qu'elle est demandée dans le sujet.

Question 6 [**MP : 92%, 7% ; MPI : 95%, 9%**] Cette question demandait de la rigueur et a été très mal traitée dans l'ensemble. Trop de candidats ne justifient soit que la termination, soit que le « non-plantage », ou bien un mélange des deux qui les arrange ! Par ailleurs, beaucoup de copies oublient que `join_right` effectue un appel récursif à la ligne 9, ce qui nécessite donc un argument.

Question 7 [**MP : 72%, 4% ; MPI : 72%, 5%**] Cette question nécessite une analyse de cas assez fastidieuse qui a le plus souvent été mal traitée. La plupart des candidats se sont contentés de traiter le cas de base et d'invoquer « un raisonnement similaire » pour le cas inductif (ou bien n'ont traité que quelques sous-cas).

Question 8 [**MP : 65%, 1% ; MPI : 71%, 3%**] La plupart des candidats se sont contentés de démontrer l'inégalité sur la hauteur, mais n'ont pas vérifié que l'arbre est bien un AVL !

Question 9 [**MP : 53%, 10% ; MPI : 59%, 14%**] Cette question assez facile est mal traitée par beaucoup de candidats. La formule de complexité attendue étant fournie par l'énoncé, il est souhait que les candidats justifient précisément celle-ci. En aucun cas, un raisonnement vague du type « on voit bien cette quantité décroît » ne peut suffire.

Question 10 [**MP : 88%, 34% ; MPI : 91%, 42%**] Cette question assez facile est généralement bien traitée. Il convient toutefois d'être précis dans la réponse : de nombreux candidats annoncent que `split` renvoie deux arbres et un élément, mais ne précisent pas que l'ensemble des valeurs retournées est exactement celui de l'arbre en argument. En effet, sans cette précision, une fonction retournant deux arbres vides conviendrait !

Question 11 [**MP : 85%, 50% ; MPI : 86%, 59%**] Cette question assez facile est généralement bien traitée, mais encore faut-il être rigoureux et bien suivre les étapes de l'algorithme.

Question 12 [**MP : 66%, 24% ; MPI : 72%, 30%**] Cette question est généralement bien traitée, mais trop de candidats n'explicitent pas la récurrence et l'hypothèse d'induction, se contentant d'un « on conclue par récurrence » qui ne rapporte évidemment pas tous les points.

Question 13 [**MP : 59%, 0% ; MPI : 66%, 2%**] Cette question difficile nécessite une analyse précise pour montrer le résultat annoncé car il faut utiliser le « téléscopage » des complexités lors de la descente. De nombreux candidats n'ont pas remarqué ce fait et ont démontré une borne de complexité en $O(\log^2 n(t))$, ce qui rapporte quand même des points. Il faut aussi bien penser à analyser la complexité de `split` et pas seulement `join` !

Question 14 [**MP : 66%, 3% ; MPI : 73%, 9%**] Beaucoup de candidats ont mal traité la complexité en espace. En particulier, on rappelle que si un programme effectue une opération A suivie d'une opération B, la complexité temporelle est la somme des complexités temporelles de A et B, mais la complexité spatiale est le maximum. La dernière partie de la question nécessite de se rappeler la notion de partage des objets en OCaml.

Question 15 [**MP : 70%, 35% ; MPI : 75%, 42%**] Cette question assez facile est généralement bien traitée, mais encore faut-il bien suivre l'énoncé et renvoyer un AVL (donc en rééquilibrant avec `join`) et ne pas oublier de renvoyer le dernier élément !

Question 16 [**MP : 56%, 14% ; MPI : 61%, 24%**] Cette question assez facile est généralement assez bien traitée, mais trop de candidats oublient de traiter le cas de l'arbre vide ou bien oublient de rééquilibrer.

Question 17 [**MP : 57%, 31% ; MPI : 63%, 32%**] Cette question assez facile est généralement bien traitée.

Question 18 [**MP : 55%, 22% ; MPI : 69%, 33%**] Cette question est surprenamment mal traitée par trop de candidats qui ne lisent pas l'énoncé en détail. Le sujet précise bien « On considère une subdivision du plan définie par un ensemble fini de polygones. Les arêtes ne se croisent pas ». Les candidats qui ont ignoré l'une de ces contraintes n'ont évidemment pas reçu de points.

Question 19 [**MP : 64%, 7% ; MPI : 73%, 9%**] Un nombre surprenant de candidats ont proposé des formules complètement fausses consistant le plus souvent à comparer uniquement l'ordonnée du point de départ de chaque segment. Pourtant, un simple dessin permet de vérifier rapidement que cela ne marche pas. Pour ceux qui ont correctement abordé le problème, il y avait une subtilité dans la comparaison de deux lignes dont le point de départ ou d'arrivée est identique. Le plus simple était donc de comparer l'ordonnée sur les deux arêtes au niveau de l'abscisse médiane.

Question 20 [**MP : 30%, 2% ; MPI : 46%, 7%**] Trop de candidats se contentent de réponse vague alors que l'énoncé demande bien d'être précis quant aux structures de données et aux algorithmes utilisés.

Question 21 [**MP : 43%, 16% ; MPI : 58%, 28%**] Il s'agit ici d'une recherche dichotomique classique, relativement bien traitée. On note toutefois que trop peu de candidats arrivent à écrire la recherche sans problème de gestion des indices (comparaison stricte ou non stricte, comment bien trouver l'élément du milieu). Les correcteurs souhaitent rappeler aux candidats qu'il est pourtant facile d'éviter ces problèmes en faisant un essai sur des tableaux de taille 2, 3 et 4 par exemple.

Question 22 [**MP : 24%, 3% ; MPI : 39%, 7%**] Peu de candidats ont bien traité cette question, la plupart écrivant des codes beaucoup trop compliqués et donc incorrects. Il est à noter que le sujet comportait une erreur qui a été relevée par quelques candidats : l'approche proposée par le sujet ne gère pas correctement le cas des polygones qui « se replient sur eux-mêmes », comme une spirale (voir ci-dessous). En effet, l'approche suggère implicitement que si le point se trouve entre deux segments, alors il est à l'intérieur, ce qui n'est pas vrai.