Informatique

Présentation du sujet

Le sujet s'intéresse au jeu de go, il commence par demander une implémentation en langage C permettant de jouer au go (partie II), puis s'intéresse à différentes stratégies pour gagner :

— une utilisation de la méthode des k plus proches voisins à partir d'une base de données de partie de go (partie III-A en langage C) ;

— une autre utilisation de la base de données en se concentrant sur le contexte local, et en optimisant le temps de calcul grâce au hachage de Zobrist (partie III-B en langage C) ;

— une exploration en parallèle de différentes possibilités de parties de go avec une méthode Monte carlo (partie IV en langage OCaml).

Enfin, la partie V est plus théorique et traite d'une réduction de 3-SAT pour démontrer qu'une version généralisée de la stratégie gagnante au go est NP-dur.

Le sujet couvre une partie assez large du programme. Il demande de programmer en langage C et en langage OCaml, et de maitriser certains aspects techniques de programmation comme l'allocation dynamique ou les threads. Il teste aussi différentes compétences : écriture de code, lecture et amélioration de code déjà écrit, création d'un jeu de test, rédaction de preuves, etc.

Analyse globale des résultats

Le sujet était long, comportait 46 questions. Les candidats ont traité entre 5 et 46 questions, pour une moyenne de 22 questions par candidat.

Les candidats ont bien réussi les questions évaluant les compétences de base et ont été départagés par les questions plus complexes. La dernière partie (questions 37 à 46), plus abstraite, a permis aux meilleurs candidats de se distinguer.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Les candidats maitrisent globalement bien les langages C et OCaml. La syntaxe n'est pas un problème pour une large majorité d'entre eux. Un conseil qui peut être donné aux candidats est de bien vérifier la spécification des fonctions qu'ils programment. Leur fonction doit la satisfaire entièrement. Sur certaines copies, seule une partie de la spécification était respectée. En particulier, les exigences de complexité sont importantes ; typiquement, les réponses aux questions 8 et 14 ne respectant pas cette contrainte passent à côté de l'enjeu de la question.

Le sujet demande de nombreuses preuves. La plupart des candidats savent écrire des preuves, et le jury constate avec satisfaction que certaines erreurs mentionnées dans le rapport de l'année précédente sont, cette année, absentes des copies.

Dans l'ensemble, les candidats ont montré qu'ils savent raisonner sur un problème nouveau, qu'ils savent s'adapter. Le jury se félicite des bonnes compétences des candidats. Il reste cependant un point négatif : les réponses aux questions demandant moins de réflexion, mais demandant plutôt de mobiliser directement les connaissances du cours, sont peu traitées par les candidats. Ainsi les questions 12 (trouver la bonne

structure de données vue en cours) ou 14 (appliquer directement une méthode du cours) ont été peu traitées et peu réussies.

Quelques remarques sur certaines questions

Q3. Il faut bien gérer le tas. Notamment, seule la grille du goban (le champ m) est allouée sur le tas, la structure représentant le goban n'est pas allouée sur le tas. Il est demandé de renvoyer une structure et non un pointeur vers une structure.

Q5. Cette question laisse une certaine latitude au candidat, qui a le choix de la méthode utilisée pour identifier les pions appartenant au groupe. Il est attendu que les candidats précisent bien quelle méthode ils choisissent. Les réponses les plus classiques attendues sont un parcours en largueur ou un parcours en profondeur, mais les solutions plus originales ont bien évidemment été acceptées.

Cette question a été plutôt bien réussie malgré de petites erreurs dans de nombreuses copies (souvent l'oubli de la pierre initiale).

Q8. Cette question demande d'écrire une fonction en temps linéaire en le nombre de pierres à retirer. Tout l'enjeu ici est qu'on ne peut pas lire en entier le goban.

Q10. Il s'agit d'une analyse d'ordre de grandeur. Des calculs simples sont attendus à partir des données fournies.

Q12. Il est demandé de citer une structure de données vue en cours et adaptée au problème. Cette question porte ainsi sur la maitrise du cours. Les résultats à cette question ont été un peu décevants, seuls 67 % des candidats ont abordé cette question, et seule une minorité a obtenu tous les points. Trouver une structure de données adaptée est une question classique auquel un ingénieur doit savoir répondre. En outre, il est préférable de privilégier une structure simple et efficace plutôt qu'une structure complexe et moins performante.

Q13. Il s'agit ici de montrer qu'un algorithme comparatif doit « au moins » réaliser une certaine comparaison. La question a pu dérouter certains candidats plus habitués à montrer des « au plus ».

Q14. Dans cette question, il suffit de dérouler la méthode « diviser pour régner » pour résoudre le problème. Seuls 35 % des candidats ont traité cette question d'application directe du cours, c'est assez peu. Le jury attire l'attention sur le fait que la question demande de « proposer un algorithme », ce qui signifie que les candidats ne sont pas tenus de proposer une implémentation en langage C, et qu'il peuvent décrire cet algorithme en langage naturel dès lors que la description est suffisamment précise.

De surcroit, la contrainte sur le nombre de comparaisons est importante. Si on s'autorise $O(n)$ comparaisons, la question ne pose plus aucune difficulté et la méthode « diviser pour régner » exigée par l'énoncé n'a plus d'intérêt.

Q24. à **Q27**. Ces questions ont principalement pour objet de tester les compétences en programmation en langage OCaml.

Q25. La gestion des exceptions a été la principale difficulté pour les candidats.

Q37. C'est une question de cours. Les définitions demandées sont assez bien connues des candidats qui ont traité la question. Il y a eu parfois des imprécisions sur ces définitions, mais le jury a pu constater sur les questions suivantes que ces imprécisions ne provenaient en général pas d'une mécompréhension des concepts.

Quant à l'exemple demandé, il est ici sous-entendu qu'on le souhaite significativement différent de SAT et 3SAT. Cependant, tous les problèmes NP-complets, mêmes proches de SAT et 3SAT ont été acceptés.

Conclusion

Les candidats écrivent des programmes de qualité, malgré la difficulté qu'il peut y avoir à écrire du code sur du papier ; et le jury est satisfait du niveau global des copies.

Les candidats ont des connaissances théoriques et savent raisonner et écrire des preuves. Ils savent aborder des questions plus abstraites, comme la NP-complétude. Les meilleurs candidats se distinguent par leur maitrise poussée de l'abstraction.

Cette seconde session du concours montre les compétences et le bon niveau des étudiants suivant la filière MP2I/MPI.