Informatique

Présentation du sujet

Le sujet expose quelques aspects des stratégies mises en œuvre par les robots Mars Explorer Rovers pour leur déplacement. Il est composé de trois parties. La première explique la notion de points d'intérêts, qui sont les points de la surface de la planète à visiter : pour minimiser l'énergie nécessaire au parcours, il faut également minimiser la distance. La seconde partie constate d'abord l'impossibilité pratique de comparer tous les chemins possibles puis propose une première stratégie qui consiste à se rendre à chaque étape au plus proche point non visité, pour un résultat rapide mais non optimal. La troisième partie met en œuvre un algorithme génétique, pour un meilleur résultat.

Chaque partie évalue des compétences complémentaires : la première porte sur la manipulation de tableaux et l'écriture de requêtes SQL. La seconde est d'essence algorithmique, avec de nombreuses évaluations de complexité. La troisième demande de prendre du recul sur un algorithme complexe, en utilisant à bon escient les fonctions utilitaires proposées en annexe.

Analyse globale des résultats

L'épreuve d'informatique est une épreuve abordable : les meilleurs candidats ont traité de façon très satisfaisante la totalité du sujet. Elle vise à valider un socle minimal de compétences informatiques que doit posséder le futur ingénieur, mises en œuvre dans un contexte réel. Les candidats en ont bien compris l'enjeu et les notes sont bien étalées, gage d'une évaluation de qualité.

De nombreux candidats produisent des copies remarquables, où chaque question reçoit une réponse soignée, concise et élégante. Le jury est alors enclin à pardonner les inévitables petites fautes inhérentes à l'informatique sur papier. Des points transversaux sont également attribués globalement, pour bonifier les copies où la syntaxe est rigoureuse, le code raisonnablement commenté, les noms de variables judicieusement choisis.

À contrario, certains cumulent les handicaps : souvent par manque de maitrise de la syntaxe de base, il proposent des solutions raturées, mal indentées, longues et confuses. Le jury fait en général l'effort de repérer les idées intéressantes, mais ne peut rémunérer de telles rédactions à la hauteur des copies précédentes, quand bien même la probabilité que la solution soit fonctionnelle n'est pas nulle.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Partie I

Sélectionner des points deux à deux distincts a perturbé de nombreux candidats. Comme stipulé dans le préambule de l'épreuve, une liste de fonctions utiles était donnée à la fin du sujet. Ceux qui ont su en extraire l'expression « `b in a` » ont généralement bien surmonté l'obstacle.

L'écriture de requêtes simples sur les bases de données est une compétence relativement bien acquise. Les deux dernières requêtes étaient plus difficiles et les concepts de jointure et de groupement sont moins bien maitrisés. Ce sont sur ces questions que les meilleurs candidats se sont départagés.

Partie II

Dans le calcul de la longueur d'un chemin, beaucoup de candidats oublient le point de départ. Normaliser les chemins a conduit également à de grosses erreurs, la faute la plus fréquente étant de supprimer sans précaution des éléments d'une liste qu'on est en train de parcourir.

L'algorithme du plus proche voisin était la question algorithmique la plus difficile de ce problème. Toute solution raisonnable nécessitait la gestion d'une liste des points déjà visités. Les autres approchent n'étant en général pas fonctionnelles, elles ont coûté aux candidats et aux correcteurs beaucoup d'énergie pour peu de points.

Les complexités sont trop souvent données sans justification ; La complexité linéaire cachée dans l'expression « `x in list` » a échappé à beaucoup.

Partie III

Cette partie ne comportait pas de difficulté majeure, elle à permis aux candidats suffisamment entrainés de montrer leur aptitudes à traduire l'algorithme proposé en langage Python. Ceux qui on soigneusement lu l'annexe ont su proposer les solutions les plus élégantes, les plus concises et sont allé au bout du problème. Il n'est pas raisonnable d'écrire un algorithme de tri de listes, parfois plusieurs fois et souvent de manière erronée, quand la fonction `list.sort` est donnée.

Conclusion

L'impression globale de sérieux de la préparation perçue les années passées perdure pour cette troisième session. Pour poursuivre dans cette voie, le jury invite les étudiants et leurs formateurs à insister encore une fois sur la clarté et la lisibilité du code.