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.

Informatique

Présentation du sujet

Le sujet porte sur le thème de la recherche de plus court chemin, appliqué au cas d'un système embarqué sur le robot Spirit, où une optimisation des déplacements entre points de mesure permet d'économiser de l'énergie.

La première partie s'intéresse à l'extraction de points d'intérêt où le robot doit réaliser des mesures, soit de façon aléatoire pour des besoins de tests de l'algorithme, soit par analyse d'image, ou encore par extraction des informations dans une base de données.

La deuxième partie propose une première approche du calcul du chemin entre les points d'intérêt par l'algorithme du plus proche voisin. Une comparaison de complexité est faite entre le traitement « par force brute », c'est-à-dire en déterminant la longueur de la totalité des chemins possibles et l'algorithme du plus proche voisin, bien plus rapide. La dernière question permet néanmoins de montrer que l'algorithme ne détermine pas nécessairement le chemin le plus court.

La troisième partie aborde une seconde approche, par un algorithme génétique, mieux à même de trouver un chemin plus court, mais pour lequel il faut trouver un critère d'arrêt adapté.

Analyse globale des résultats

Le sujet est de longueur raisonnable pour le temps imparti. De nombreux candidats abordent la totalité du sujet.

À nouveau cette année, le jury se réjouit du niveau satisfaisant des copies. Le langage est bien maitrisé et permet de traduire les solutions aux questions sans difficultés. Seule une petite proportion des candidats (moins de 5%) rend une copie vide ou contenant des programmes sans aucune cohérence, ne sachant pas écrire une boucle, réaliser une simple affectation correctement ou discerner le nombre de dimensions d'un tableau. Ces copies conduisent à quelques notes très faibles et demeurent une énigme pour le jury, après trois semestres de cours et de travaux dirigés en informatique. Il est possible que ces copies soient celles de candidats n'ayant pas suivi d'enseignement d'informatique de classe préparatoire.

Les petites erreurs syntaxiques n'ont pas été retenues par le jury comme un élément discriminatoire, dans la mesure où elles ne cachent pas des erreurs de fond. Les réponses pertinentes d'un point de vue algorithmique sont valorisées.

Certaines copies proposent des programmes particulièrement élégants et concis et reflètent un vrai recul sur les différentes stratégies de programmation. Ces copies ont été valorisées.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Au regard des copies évaluées, le jury propose aux futurs candidats de prêter attention aux remarques suivantes.

L'indentation en Python délimite les blocs d'instructions et doit apparaitre clairement dans la rédaction. Toute présentation claire est bienvenue ; bien souvent, un trait vertical marquant l'alignement du bloc d'instruction est suffisant.

L'initialisation d'une variable dans une boucle ou hors de la boucle n'a pas les mêmes conséquences pour l'algorithme.

Le nombre d'itérations d'une boucle doit être bien réfléchi pour s'assurer que les indices des éléments d'une liste appelée dans la boucle sont bien définis. L'instruction `range(n)` parcourt `n` itérations indicées de `0` à `n-1`.

La concision et l'élégance des programmes sont appréciées dans l'évaluation. Les candidats qui réinvestissent les fonctions déjà codées sont valorisés par rapport à ceux qui recopient les lignes de code équivalentes. Bien souvent, une condition booléenne bien choisie permet d'éviter de longues listes de conditions aux instructions identiques.

Des noms de variables explicites aident à la compréhension du code. De trop nombreux candidats utilisent des noms de variables non significatifs (`a`, `b`...) ce qui nuit à la compréhension du programme. La clarté du programme (en particulier le choix des noms de variables) ainsi que la présence de commentaires opportuns sont prises en compte dans l'évaluation.

Lors d'un commentaire sur une complexité, une justification chiffrée minimale est attendue : un programme de complexité exponentielle pourrait être utilisable pour peu que le nombre d'opérations soit faible au regard de la puissance de calcul d'un micro-processeur actuel.

L'ordre des questions importe. Prendre soin de rédiger les réponses aux questions en respectant leur ordre dans le sujet.

La qualité d'expression (l'orthographe notamment) et la qualité visuelle de présentation relèvent des compétences de communication indispensables à un candidat à une école d'ingénieur. Le correcteur n'attribue les points qu'aux éléments de réponse qu'il parvient à lire et à comprendre. Les copies obscures et difficiles à comprendre sont pénalisées.

Les variables utilisées dans une fonction doivent être définies dans cette fonction ou être explicitement définies comme variables globales (soit par le sujet, soit par le candidat). Beaucoup de candidats ont utilisé la variable `n` sans la définir, ce qui a soulevé une ambiguïté, `n` pouvant être le nombre de points d'intérêt incluant la position initiale du robot ou pas.

Les candidats sont invités à bien lire l'annexe contenant certaines fonctions utiles pour traiter le sujet.

Première partie

Les points d'intérêt choisis au hasard devaient être distincts. Certains candidats ne vérifient pas cette contrainte dans leur programme. Le calcul des distances devait prendre en compte (dans la dernière colonne) les distances à la position courante du robot. De nombreuses erreurs sur les indices ont été relevées, que ce soit sur le nombre d'itérations ou les indices des éléments du tableau. Certains candidats confondent encore l'affectation (`=`) et le test d'égalité (`==`).

Les explications sur la fonction `F1` se limitent parfois à paraphraser les lignes de code. Il était attendu une description de la tache globale réalisée et une interprétation du résultat renvoyé.

Les deux premières questions de bases de données sont très souvent abordées et bien réussies. C'est moins le cas pour les trois questions suivantes. Les questions 3 et 5 faisaient intervenir des jointures. La question 4 nécessitait de prendre des initiatives en définissant le nombre de bit de codage des entiers pour en déduire la surface maximale d'exploration enregistrable.

Deuxième partie

Cette partie débute avec deux fonctions relativement simples et utiles pour la suite. La longueur du chemin est souvent bien calculée, lorsque la position courante du robot est prise en compte. En revanche, la normalisation fait souvent l'objet de programmes décousus. L'erreur la plus fréquente étant d'engager une boucle sur la longueur du chemin tout en supprimant des éléments de ce même chemin, ce qui ne manque pas de conduire à un débordement et à des décalages d'indices.

La plupart des candidats trouvent le nombre de chemins possibles mais un minimum de justification était attendu. Pour conclure, il fallait non seulement réaliser une application numérique pour déterminer l'ordre de grandeur du nombre d'itérations à prévoir, mais aussi pour comparer ce nombre aux puissances de calcul des processeurs actuels.

L'algorithme du plus proche voisin était probablement le plus complexe à élaborer, sachant qu'il convenait de s'assurer de ne pas repasser sur un point déjà visité. Les réponses ont souvent conduit à des programmes sans queue ni tête, difficilement évaluables par le correcteur en l'absence de commentaires. Le calcul de complexité devait prendre en compte la complexité d'une instruction telle que `if k not in L`.

Beaucoup de candidats ont trouvé un exemple où l'algorithme ne fournit pas le plus court chemin. Un schéma était souvent plus clair qu'un long discours à cette question.

Troisième partie

Les fonctions `créer_population` et `réduire` distinguent les candidats qui savent utiliser judicieusement les fonctions mises à disposition et ceux qui tentent de se débrouiller sans. Les questions suivantes n'ont pas posé beaucoup de difficultés aux candidats attentifs aux indices. Une grande partie des candidats est à l'aise avec la manipulation des listes pour effectuer des concaténations et du slicing.

La construction de la fonction principale `algo_génétique`, réutilisant bon nombre des fonctions précédentes, est bien réussie. L'argumentation sur le critère d'arrêt est souvent pertinente lorsque la question est abordée.

Conclusion

Le sujet aborde une large partie du programme d'informatique commune. Le choix d'un sujet s'appuyant sur un thème courant en informatique assure une cohérence avec la formation d'ingénieur. Cette approche sera reconduite sur des problématiques de simulation ou d'algorithmique en informatique, à partir du programme des trois semestres d'informatique.

Les bons résultats à cette épreuve montrent que les étudiants, soutenus par leurs professeurs, ont acquis des compétences affirmées en informatique. Le jury encourage les futurs candidats à travailler l'informatique en alliant réflexion sur feuille de papier et mise en œuvre des algorithmes sur ordinateur.

Informatique

Présentation du sujet

Cette année, ce sujet d'informatique propose d'étudier des aspects informatiques d'un robot effectuant une mission sur Mars.

Une première partie nous amène à créer une exploration pour le robot et gérer des points d'intérêt.

La deuxième partie présente un algorithme permettant de planifier une exploration.

Enfin, une troisième partie utilise un algorithme génétique pour proposer une autre solution au problème de l'exploration.

Analyse globale des résultats

Le sujet est de longueur satisfaisante et a permis un étalement des notes convenable avec une moyenne brute voisine de 10 et une répartition des notes adaptée à une épreuve de concours. Certains candidats ont traité quasi-correctement l'intégralité du sujet. À contrario, on peut noter quelques copies presque vides.

Le langage Python est désormais globalement plutôt bien maitrisé par les candidats et le code écrit est plutôt bien présenté même si vraiment peu de commentaires sont présents et si quelques problèmes techniques subsistent.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Les parties sont indépendantes et ont été plus ou moins toutes abordées.

Partie I

I.A.1) Le caractère distinct des points choisis est souvent mal traité.

I.A.2) Quelques difficultés à gérer la position initiale du robot, surtout au niveau des indices. On a aussi pu voir des calculs de distance farfelus.

I.B.1) La lecture du programme ligne par ligne ne permet pas au candidat d'avoir une vision globale du programme, souvent le bilan qui en résulte est inexact.

I.B.2) Parfois, la taille de la photo a posé problème avec une mauvaise utilisation de fonction.

I.C – L'écriture en SQL est dans l'ensemble plutôt réussie pour les requêtes simple (**I.C.1)** et **I.C.2)**) mais bien plus confuse pour les autres.

Partie II

II.A.1) Oubli récurrent de la position initiale du robot et gestion erronée des indices.

II.A.2) Pas de problèmes particuliers, l'énoncé décrit la marche à suivre.

II.B.1) Le nombre de chemins possibles n'est souvent pas justifié.

II.B.2) Bien peu de candidats justifient convenablement (référence au nombre d'opérations par seconde pour un processeur) le fait que l'algorithme n'est pas utilisable dans ce cas.

II.C.1) Question difficile rarement bien traitée même si souvent abordée. Des difficultés fréquentes lors de l'initialisation.

II.C.2) La notion de complexité est souvent mal traitée par les candidats.

Partie III

La partie III a été un peu moins souvent abordée que les deux précédentes, mais elle a été plutôt bien réussie par les candidats qui sont parvenus jusque-là.

III.A – Oubli fréquent de la position initiale du robot dans l'initialisation

III.B – Une majorité de candidats a pensé à trier la liste mais la suppression des éléments a été moins bien traitée (suppression des termes les plus petits, mauvaise utilisation de `del`)

III.C.1) On a souvent vu un oubli d'avoir 2 points distincts.

III.C.2) Plutôt bien traitée quand elle est abordée.

III.D.1-2) Pas de problème particulier.

III.E.1) L'algorithme génétique : ceux qui l'ont traité ont bien réussi en général.

III.E.2) Souvent bien traitée quand abordée.

III.E.3) Question plus difficile avec parfois des réponses tout à fait pertinentes.

Remarques générales

Les noms des variables doivent-être choisis de manière judicieuse et pas seulement par ordre alphabétique.

Certaines variables utilisées ne sont pas définies.

La gestion des indices dans les listes n'est pas totalement maitrisée. Il n'est pas rare de lire `L(len(L))` dans des copies.

Le nombre d'itérations d'une boucle doit être bien réfléchi, en notant que l'instruction `range(n)` permet de réaliser `n` itérations indicées de `0` à `n-1`.

Il faut lire attentivement le sujet pour utiliser correctement les fonctions décrites et fournies par l'énoncé.

les requêtes SQL, dès qu'elles sont un peu évoluées ne sont pas bien traitées : tout candidat devrait pouvoir utiliser convenablement une jointure et une fonction d'agrégation.

L'instruction `del` est très souvent utilisée pour un résultat régulièrement faux (surtout dans une boucle).

Les calculs de complexités ne sont clairement pas assimilés par beaucoup de candidats. On a par exemple appris que $O(n^2) + O(n^2) = O(n^4)$.

Une conséquence du point précédent est que beaucoup de fonctions sont écrites sans se soucier de leur efficacité propre, ce qui peut être gênant pour la maitrise réelle de l'outil informatique.

Dans l'écriture de fonctions, il est naturel d'attendre quelques explications, mêmes succinctes.

Conclusion

Le sujet proposé cette année est de longueur raisonnable et a permis une bonne répartition des notes. Globalement, le niveau des candidats est convenable même si de nettes différences entre candidats peuvent être remarquées. Le jury rappelle qu'un investissement constant sur toutes les années de préparation est essentiel et peut permettre facilement de se démarquer. Il serait aussi souhaitable de se poser la question de l'efficacité dans l'écriture de programmes.

Informatique

Présentation du sujet

Cette épreuve traite les problématiques liées à l'exploration martienne par des robots ; en particulier la recherche de points d'intérêts dans une image satellite, leur stockage dans une base de données et la résolution du problème du voyageur de commerce pour déterminer l'ordre de leur visite.

Le sujet aborde les bases de données et les requêtes SQL, des algorithmes simples de parcours de listes et de matrices, des algorithmes plus avancés pour la résolution d'un problème de graphe.

Analyse globale des résultats

La structure de base d'une requête SQL est maitrisé par la majorité des candidats. Si la syntaxe du langage Python et les éléments fondamentaux (indentation, boucles, tests...) de l'algorithmique semblent acquis pour la majorité des candidats, un grand nombre est bloqué lorsqu'il est nécessaire d'écrire une fonction utilisant les résultats accumulés sur plusieurs questions.

Commentaires sur les réponses apportées et conseils aux futurs candidats

Partie I

Il est important de bien comprendre les structures de données utilisées. Un schéma peut souvent aider.

Le jury attend des étudiants qu'ils connaissent certaines notions mathématiques de base comme le calcul d'une distance entre deux points.

Pour description d'une fonction et son résultat, le jury attend une interprétation globale, pas uniquement une analyse ligne à ligne.

La maitrise des jointures ainsi que des opérations telles que `COUNT` ou `GROUP BY` est indispensables pour traiter toutes les questions concernant les bases de données.

Partie II

Les candidats doivent être vigilent à ne pas faire des boucles sur une liste dont on modifie les éléments.

Une minorité de candidats a proposé une solution correcte pour la question **II.C.1**. Cette question était l'une des plus difficiles. Pour traiter une question de ce type, il est souvent pertinent avant d'écrire le code python de 1) prendre un petit exemple 2) dérouleur l'algorithme imaginé dessus et 3) représenter les structures de données intermédiaire.

Partie III

Cette partie peut sembler complexe mais pour un grand nombre de questions elle ne consiste qu'a comprendre l'algorithme génétique décrit et ré-utiliser des fonctions issues des questions précédentes.

Conclusion

Les candidats doivent travailler pour être capable de traiter des requêtes SQL plus complexes, contenant des jointures. Ils doivent être capable d'écrire des fonctions faisant intervenir l'utilisation d'autres fonctions écrites au préalable ou des fonctions issues de bibliothèques. Le jury encourage les candidats à tester les fonctions qu'ils créent sur des données simples.