Informatique

Présentation du sujet

Le sujet porte sur le thème du trafic aérien, et en particulier la gestion optimisée de l'attribution des niveaux de vol par Eurocontrol et le système anti-collision TCAS.

La première partie traite des requêtes d'extraction des données de vol utiles, à partir des bases de données stockant les caractéristiques des vols programmés et des aéroports.

La deuxième partie s'intéresse à l'attribution des niveaux de vol aux compagnies aériennes. Plusieurs algorithmes sont envisagés, permettant de détecter les conflits et de minimiser les couts de changement de niveaux de vol. Le nombre important de vols à gérer à l'échelle européenne nécessite une attention particulière au regard de la complexité des solutions envisagées.

La troisième partie porte sur le système anti-collision TCAS, embarqué dans les avions, qui traite les données de vol reçues de la part des autres avions environnant afin d'alerter le pilote en cas de danger de collision.

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 (de l'ordre de 5%) a de réelles difficultés à construire un programme, en respectant les bases de syntaxe.

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.

Les questions relatives aux bases de données, placées en début de sujet, sont relativement bien réussies cette année.

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.

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, en notant que l'instruction `range(n)` parcourt `n` itérations indicées de `0` à `n-1`.

La somme de deux listes `L1+L2` conduit à la concaténation des listes, la somme de deux tableaux (`numpy.ndarray`) `A1+A2` conduit à la somme des éléments du tableau.

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`, `c`, ...) 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 sont prises en compte dans l'évaluation.

Dans une démonstration ou dans l'écriture d'un code, une justification minimale est attendue.

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

La présentation d'une copie fait également partie des compétences attendues d'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 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 candidat ont utilisé la variable `n` sans la définir, ce qui a soulevé une ambiguïté, `n` pouvant être le nombre de vol comme le nombre de sommets.

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

Enfin, le jury invite les candidats à ne pas se décourager en milieu d'épreuve lorsqu'une question difficile n'est pas réussie. Les sujets comportent de nombreuses parties indépendantes permettant de valoriser les compétences d'un candidat.

Première partie

Les questions sur les bases de données sont abordées par pratiquement tous les candidats, et souvent de façon satisfaisante.

La fonction `count` dans la première question est souvent oubliéee, ou remplacé par autre chose (`len` par exemple) faute de savoir quoi mettre.

La question **I.D** conduit souvent à des jointures inappropriées, en particulier avec la table `aeroport`.

Deuxième partie

Cette partie débute avec des fonctions nécessitant des parcours de listes et des conditions. Les sous-parties ±QII.A et **II.B** sont bien abordées par la plupart des candidats. Le paramètre `n` n'étant pas défini comme une variable globale, il convenait de le définir dans les fonctions à partir des dimensions des tableaux. Certains candidats confondent l'affectation (`=`) et le test d'égalité (`==`). De même, le test de différence a parfois été noté `=!` au lieu de `!=`. Le jury a été indulgent cette année vis-à-vis de ces erreurs, mais souhaite attirer l'attention des candidats sur ces nuances.

Un bon nombre de candidat ne semblent pas à l'aise avec la notation « grand O » et préfèrent parler de complexité en $9n^2$ plutôt qu'en $O(n^2)$.

Les sous-parties **II.C** et **II.D** sont très souvent abordées mais sans recueillir la totalité des points. Certains candidats oublient dans la fonction `cout_du_sommet` de ne pas compter les sommets supprimés, ou au contraire ne comptent pas les sommets non encore attribués.

L'algorithme de recuit simulé (sous-partie **II.D**) montre une forte disparité entre les meilleurs candidats qui proposent un algorithme juste et concis, d'autres qui proposent un algorithme très long suite à des répétitions, et d'autres encore qui n'abordent pas la question. À noter que la syntaxe `liste1 = liste2` en python ne permet pas d'obtenir une copie de `liste2` mais fournit simplement deux noms pour accéder au même objet.

Troisième partie

La sous-partie **III.A** est généralement bien abordée, les candidats sachant plutôt bien comment les nombres sont représentés en mémoire, même si les réponses comportent souvent de petites erreurs de calcul.

Les sous-parties **III.B** et **III.C** s'intéressent au calcul et à la mise à jour de la liste d'intrus, ordonnée en fonction de la dangerosité. Ces questions sont moins abordées et les réponses rarement correctes. Beaucoup de candidats n'ont pas lu l'annexe et n'utilisent pas la fonction `time`.

La question **III.C.2** aborde le cas d'un tri dans une liste déjà triée, où seule une ligne est à replacer. Cette question nécessitait quelques explications sur la stratégie de tri adoptée.

La question **III.C.3** n'a pas été comprise par la plupart des candidats qui l'ont abordée.

La sous-partie **III.D** permet de conclure l'étude sur les performances du TCAS. Ces questions, lorsqu'elles sont abordées, sont souvent correctement traitées.

Conclusion

Le sujet aborde la majeure partie du programme d'informatique commune. Le choix d'un sujet s'appuyant sur une application concrète de l'informatique assure une cohérence avec la formation d'ingénieur. Cette approche sera reconduite sur des problématiques de simulation ou d'algorithmique courantes en informatique, à partir du programme des 3 semestres d'informatique.

Les bons résultats à cette épreuve montrent que les étudiants, soutenus par leurs professeurs, ont su montrer 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.