5 Informatique

5.1 Informatique pour tous

Le sujet d'informatique commune portait cette année sur des techniques algorithmiques autour du thème de la discrétisation spatiale en facettes d'une scène de cinéma.
L'épreuve abordait, comme toujours, un large spectre des notions vues durant les deux années de préparation des candidats.

5.1.1 Remarques générales

- Les copies sont très contrastées et l'épreuve a bien joué son rôle de classement, il y a en particulier un nombre non négligeable de copies qui montrent une bonne maîtrise du langage et une compréhension relativement bonne, et parfois excellente, des problématiques abordées.
- Certaines copies sont trop proches d'un brouillon, avec beaucoup de ratures réalisées sans soin. Si l'on est capable de comprendre qu'un candidat se reprenne sur une question particulière ou une ligne de code, il ne nous semble pas acceptable de croiser des copies contenant de nombreuses ratures sales sur toute la copie. Les candidats ont eu un temps de formation conséquent pour cadrer leurs productions écrites et doivent respecter le correcteur dans la mise en forme de leurs copies.
- Nous avons constaté un phénomène gênant qui prend de l'ampleur : certaines parenthèses, pourtant indispensables, tendent à disparaître. On a souvent lu
  `for i in range m :` ou encore `m = len L` par exemple. Si on peut le laisser passer une fois, nous l'avons sanctionné lorsque c'était trop redondant dans la copie.
- La première partie du sujet définissait des opérations algébriques simples sur les vecteurs. Il pouvait être tentant de dévier vers des réflexes propres à l'utilisation du module `numpy` (par exemple la multiplication d'un scalaire par un vecteur ?), mais il fallait résister à cette tentation : on rappelle en particulier qu'une opération de type `a*L` avec a un flottant et `L` une liste n'a pas de sens en python. Les candidats sont donc invités à réfléchir attentivement aux nombreuses différences entre les listes et les tableaux numériques (`ndarrays`).
- Une remarque déjà faite les années précédentes, mais que nous pensons nécessaire de renouveler : la syntaxe `L=L+[a]` est beaucoup moins efficace pour ajouter un élément à la fin d'une liste que `L.append(a)`. Il serait souhaitable que les étudiants privilégient systématiquement la deuxième solution au cours de leur formation.

5.1.2 Commentaires spécifiques à chaque question

Q1 - Trop de candidats ne maitrisent pas la syntaxe d'une fonction d'agrégation, confondent `COUNT` avec `SUM`, ou n'utilisent pas la bonne relation.

Q2 - Mieux réussie que **Q1**, sans doute parce que l'énoncé donnait une exemple de jointure à la question suivante.

Q3 - Assez bien réussie, mais il manque souvent une information : selon l'axe x, ou la mention d'une largeur/longueur/taille.

Q4 - Une simple expression était demandée, pas une fonction. On a pu constater quelques confusions avec la syntaxe numpy pour les indices (`L[0,0,1]` ne fonctionne pas pour une liste).

Q5 - L'indexage du maillage ne correspondait pas à la numérotation des facettes sur la figure, il fallait donc être attentif et ne pas répondre `S2` trop vite.

Q6 - L'import d'une fonction d'un module est une syntaxe importante et doit être maîtrisée.

Q7 - On peut au choix se féliciter que 95% des candidats sachent reconnaître le calcul de la norme d'un vecteur, ou s'inquiéter que 5% des candidats n'y parviennent pas ?

Téléchargé gratuitement sur Doc-Solus.fr

Q8 - Cette question d'apparence simple contenait pourtant un certain nombre de pièges. En particulier, on demandait explicitement une nouvelle liste : comme les listes sont mutables, on ne pouvait pas écrire `V[i]=a*V[i]` par exemple, qui modifiait la liste en argument. On ne peut pas non plus utiliser de syntaxe « à la numpy » comme `return a*V`. Enfin, un certain nombre de candidats tombent dans le piège classique d'initialiser une nouvelle liste vide (`W=[]`) puis tentent de la remplir avec `W[i]=` ?

Q9 - La définition du barycentre, explicitement au programme de physique de toutes les filières, n'est pas toujours maitrisée.

Q10 - Il y avait deux pièges sur cette question : il fallait éviter de recoder à la main une ou plusieurs fonctions données dans le module `operations_vectorielles` (surtout la soustraction), et on ne peut pas non plus utiliser de syntaxe « à la numpy » comme `return n/norme`.

Q11 - Trop de candidats recodent à la main la fonction soustraction. On peut également souhaiter une syntaxe plus élégante ; on rappelle que la syntaxe suivante :

```
if bool :
return True
else :
return False
```

peut s'écrire plus simplement : `return bool`
Enfin, même s'il ne s'agit pas d'une épreuve de mathématiques, nous avons été surpris par le nombre de candidats confondant la norme d'une différence avec la différence des normes.

Q12 - On a pu lire un nombre incalculable de fois « la fonction renvoie `True` si tous les sommets de `L` sont proches de `S1` », ce qui témoigne d'une mauvaise compréhension de l'algorithmique de base. De même, « la fonction renvoie `True` si `S1` et `S2` sont proches » n'était pas une réponse acceptable car le deuxième paramètre de la fonction était une liste `L` qu'il fallait impérativement évoquer.

Q13 - Il fallait donner explicitement ce qui était renvoyé ; attention au type (beaucoup d'oublis du point pour les flottants). Nous avons bien entendu fait preuve de tolérance compte tenu de l'erreur d'indentation de l'énoncé, et accepté les réponses tenant compte ou pas de cette erreur. Nous demandions seulement la cohérence avec l'analyse de complexité à la question suivante.

Q14 - Que de réponses affirmées sans aucune justification ! Il est capital de justifier proprement une analyse de complexité. Ici, il est attendu une description explicite du meilleur et du pire des cas avant la justification. Cette distinction a souvent été oubliée pour `mystere2`. De plus, trop de candidats justifient leurs complexités par le simpliste (et faux) « il y a une boucle `for` donc c'est de complexité linéaire » ? Il fallait remarquer que la complexité de `mystere3` ne pouvait dépendre que de `m` (taille de l'entrée) et pas de `n`, caractéristique d'une variable intermédiaire. Cette étude a très rarement été bien réalisée, erreur d'indentation ou pas. Enfin, on a encore retrouvé dans les copies quelques complexités en $O(n!)$, $O(3n)$ ou encore $O(1/n)$ sans que cela paraisse gêner les candidats qui les proposent.

Q15 - L'application numérique est régulièrement fausse (même avec une certaine tolérance), alors que le calcul posé est correct.

Q16 - La gestion des chaînes de caractères n'est pas maitrisée : il s'agit d'un objet non mutable (on ne peut pas écrire `ch[-1]="\n"` par exemple), pas de append, conversion en chaîne de caractères à l'aide de str non connue (on a lu un nombre incalculable de fois `"mat_h[i][j]"` pour cette conversion). Les candidats sont encouragés à s'entraîner sur la maitrise de ce type de variable.

Q17 - L'ouverture et l'écriture dans un fichier n'est pas maîtrisée.

Q18 et 19 - Très peu de candidats ont compris la question (ou lu le paragraphe précédent).

Q20 - La valeur absolue a souvent été oubliée. On ne demandait pas une fonction, mais un script/code.

Q21 - La condition d'immersion n'a pas toujours été comprise (on attendait notamment le calcul de la hauteur des vagues au barycentre de la face). Beaucoup de candidats ont également calculé trois fois

Téléchargé gratuitement sur Doc-Solus.fr

le barycentre pour en extraire ses trois coordonnées, ce que nous avons sanctionné puisqu'une solution simple et plus économe existait.

Q22 - Assez bien réussie ; on attendait des candidats qu'ils réinvestissent les fonctions définies précédemment.

Q23 - L'opérateur + n'est pas adapté pour les listes (il les concatène). La projection sur l'axe vertical a parfois été oubliée.

Q24 - On a pu croiser certaines structures en boucles imbriquées :

```
for i in range(len(L1) :
for j in range(len(L2)) :
...
```

qui témoignent d'une mauvaise compréhension de l'algorithme de fusion de deux listes.
De manière générale, cet algorithme classique de cours devrait être mieux maitrisé par les candidats. Parmi ceux qui connaissaient l'algorithme, beaucoup ont oublié de comparer les aires des éléments (facettes) et pas simplement les éléments eux-mêmes. Enfin, la version récursive de la fusion de deux listes nous semble à éviter absolument, car très peu efficace : nous encourageons les candidats à bien maitriser (et présenter) la version itérative.

Q25 - Le cas de base a souvent été oublié ou mal traité.

Q26 - Le principe (tri puis extraction) a plutôt été maitrisé, mais la condition sur le nombre de facettes a posé de nombreux problèmes. On pouvait très facilement vérifier soi-même la cohérence de l'expression sur de petites valeurs (`len(maillageG)=2` et `len(maillageG)=3`) pour l'inclusion de la médiane.

Q27 - Question assez simple si on avait bien compris le sujet et le principe de la méthode d'Euler. Nous avons fait preuve de tolérance concernant le type de variable utilisé (scalaire ou vecteur) en tenant compte de l'ambiguïté de l'énoncé.

5.2 Informatique option MP

5.2.1 Généralités

Le sujet s'inscrit dans le domaine de la compression de données. Il traite particulièrement le cas du comptage efficace d'occurrences de symboles (codes Unicode) dans un texte, puis comment coder, à l'aide d'un encodeur efficace, un texte dans une suite de bits.
Le sujet comporte 33 questions. Même s'il traite d'un unique problème de bout en bout, le sujet est découpé en deux parties indépendantes. Les candidats ont bien pris en compte cette séparation et ont bien su gérer le passage de l'une à l'autre en vue de répondre au maximum de questions.
En général, les candidats ont bien compris le problème posé. Ils répondent aussi bien aux questions de programmation et de complexité qu'aux questions portant sur des démonstrations et/ou des justifications rigoureuses.
Enfin, les correcteurs ont noté, avec satisfaction, qu'il y avait une diminution importante de compositions traitant exclusivement les questions portant sur l'écriture de programmes ou les questions portant sur des démonstrations.

5.2.2 Analyse Globale

Les candidats ont une bonne compréhension des structures complexes mises en œuvre dans l'énoncé du problème. Ils font un usage de la récursivité souvent clair et efficace. Le niveau des réponses et des copies est satisfaisant voire bon. Pour les questions programmation, les correcteurs ont pris en

Téléchargé gratuitement sur Doc-Solus.fr