4 Informatique

4.1 Informatique pour tous

4.1.1 Généralités et présentation du sujet

Le sujet d'informatique commune comportait cette année trois parties indépendantes, chacune étant axée sur un type de marche différent. Dans la première partie, on menait l'étude d'une randonnée concrète via des questions sur les bases de données et des questions plus algorithmiques visant à déterminer les caractéristiques usuelles (distance parcourue, dénivelé) d'une randonnée. Dans la seconde partie, on étudiait une marche aléatoire sur un réseau, modélisant le mouvement brownien d'une petite particule en suspension dans un fluide. La dernière partie était consacrée à la génération de chemins auto-évitants dans $\mathbb{Z}^2$.
L'épreuve abordait ainsi un large éventail de notions étudiées durant les deux années de préparation des candidats.

4.1.2 Commentaires généraux

- Si certaines copies sont très faibles (voire presque vides), certaines sont excellentes et frisent parfois la perfection. La longueur et la difficulté du sujet étaient ainsi tout à fait adaptées à ce type d'épreuve, ce qui a permis de classer les candidats.
- Cette année encore, le jury souhaite souligner l'importance de la présentation des copies. Certaines sont très brouillonnes, sales, voire parfois illisibles. Un nombre trop important de ratures nuit forcément à la lecture des codes Python produits, et peut même provoquer des erreurs de syntaxe. On peut certes tolérer quelques ratures propres (correction d'un oubli, d'une erreur de syntaxe), qui ne nuisent pas à la poursuite de la lecture ni à la structure des codes proposés, mais un code trop difficile à déchiffrer (excès de ratures ou de rajouts par le biais de flèches ou d'astérisques) est forcément sanctionné.
- De la même façon, une erreur ponctuelle de syntaxe (oubli d'une parenthèse fermante) peut être tolérée. En revanche, l'absence récurrente des parenthèses (en écrivant par exemple systématiquement `for i in range n` ou `len L`) a été sanctionnée.
- L'importation des bibliothèques en Python doit être maîtrisée. Il existe plusieurs façons de procéder, qui ne doivent pas être confondues. On pourra par exemple écrire `import math`, `import math as m`, `from math import *` ou encore `from math import cos, sin`. Il est cependant demandé que la syntaxe de l'appel aux fonctions Python de ces bibliothèques soit en adéquation avec la syntaxe d'importation choisie.
- Le sujet demandait explicitement de manipuler des listes. À ce titre, il n'était pas opportun d'utiliser le module `numpy`. Les candidats doivent maîtriser la manipulation des listes, notamment :
  - la construction d'une liste élément par élément.Par exemple, l'initialisation d'une liste `L = [ ]` suivie, dans une boucle `for`, d'une affectation `L[i] = elt` provoque une erreur ;
  - l'ajout d'un élément à la fin d'une liste. Comme indiqué dans les rapports des années précédentes, la syntaxe `L.append(elt)` est à privilégier. D'une part, elle est plus efficace, mais elle est également moins source d'erreurs. L'emploi de la syntaxe `L = L + [elt]`(ou `L+ = [elt]`) a par exemple provoqué beaucoup d'oubli de crochets, quand `elt` était lui-même une liste ;

— les erreurs de syntaxe “à la `numpy`” : l'addition terme à terme de deux listes de même taille (respectivement la multiplication terme à terme d'une liste par un flottant ou un entier) ne peut pas se faire à l'aide d'une syntaxe du type `L1 + L2` (respectivement `a * L`), réservée aux tableaux `numpy`. L'accès à un élément d'une liste de listes se fait quant à lui *via* une syntaxe du type L[i][j], et pas `L[i, j]` ;

— le caractère modifiable des listes ; il s'agit d'un aspect subtil de Python, dont les candidats doivent avoir pris conscience au cours de leur formation. Par exemple, quand l'énoncé demande de créer une nouvelle liste, on ne peut pas juste modifier la liste entrée en paramètre d'une fonction ; de même, l'affectation `L1 = L2` ne permet pas de créer une copie indépendante de la liste `L2` ;

— la modification des éléments d'une liste ; la commande `for elt in L : elt = float(elt)` n'apporte par exemple aucun changement à la liste `L`.

- Le jury a remarqué cette année un nombre croissant de copies dans lesquelles les affectations des variables sont faites à l'envers ; par exemple, si `L` était une liste placée en paramètre d'une fonction, beaucoup de candidats ont écrit `L = a, b, c, d` au lieu de `a, b, c, d = L` pour affecter les valeurs des éléments de `L` dans les variables `a, b, c, d`. En outre, rappelons que le choix du nom d'une variable doit être valide et pertinent. Un nom de variable ne peut être composé que de caractères alphanumériques et du caractère _ (pas de ', -, ., qui provoquent une erreur de syntaxe, ni de lettres grecques, comme φ). Enfin, rappelons que l'affectation d'une variable se fait en Python à l'aide d'un =, que beaucoup de candidats ont utilisé à tort à la place d'un == pour tester l'égalité des valeurs de deux objets.

4.1.3 Commentaires spécifiques à chaque question

Q1 - Trop de candidats oublient le `SELECT` quand ils utilisent une fonction d'agrégation. De plus, la syntaxe `1998 < ne < 2004` ne permet pas de répondre à la question ; on préférera écrire par exemple `ne > 1998 AND ne < 2004`, ce qui n'est pas équivalent en `SQL`.

Q2 - La fonction d'agrégation AVG n'est pas toujours maîtrisée, ainsi que l'usage de `GROUP BY`.

Q3 - Beaucoup de candidats ont voulu utiliser une jointure entre les tables `Rando` et `Participant` en utilisant une condition du type `ON rid = pid`, montrant ainsi un manque de compréhension de cette notion. Un produit cartésien (suivi d'une clause `WHERE`) ou l'utilisation d'une sous-requête permettaient de répondre à la question.

Q4 - Cette question était plus délicate. Une auto-jointure permettait par exemple de répondre à la question. Certains candidats ont également cherché à utiliser une sous-requête ; attention dans ce cas à la confusion entre `HAVING` et `WHERE`. De manière générale, beaucoup de candidats utilisent une syntaxe erronée de la forme `attribut.table` en lieu et place de `table.attribut`.

Q5 - Les fonctionnements de `readline`(), `readlines`() et `split`() n'ont pas été assez compris. La conversion d'une chaîne de caractères en flottant a également posé de nombreux problèmes, voire a été souvent complètement occultée. Si des progrès ont été notés concernant la gestion des fichiers texte par les candidats, des efforts sont néanmoins à poursuivre sur ce point.

Q6 - Lors d'une recherche de maximum dans une liste, l'initialisation du maximum courant à 0 est erronée.

Q7 - Le principe est souvent maîtrisé, mais la syntaxe a trop souvent été entachée d'une erreur de gestion des indices dans la boucle `for`. Les candidats sont invités à vérifier systématiquement les éventuels dépassements d'indices lors d'un parcours d'une liste. Beaucoup de candidats ont également commis une erreur de signe concernant le dénivelé négatif ; le canevas fourni en fin d'énoncé permettait pourtant de lever un quelconque doute à ce sujet.

Q8 - Cette question était assez difficile. Il fallait dans un premier temps bien comprendre le type d'un point de passage (beaucoup de candidats l'ont interprété à tort comme une liste de deux flottants, contredisant l'énoncé). La conversion (en radians pour les angles, en mètres pour les longueurs) était ensuite attendue. La prise en compte de l'altitude moyenne de l'arc a porté à confusion, ainsi que l'utilisation du théorème de Pythagore. Certains candidats ont illustré leur code d'un dessin, les aidant vraisemblablement à mieux comprendre la question.

Q9 - Question plutôt réussie, excepté l'erreur fréquente sur le type d'un point de passage.

Q10 - Beaucoup trop de candidats ont cherché à utiliser une syntaxe "à la `numpy`" dans cette question, ce qui n'est pas possible avec des listes.

Q11 - Outre la difficulté de projeter l'équation différentielle sur les deux axes, de nombreux candidats ont eu des difficultés à coder correctement la force $\vec{f_B}$. Les fonctions `uniform` ou `gauss` génèrent des valeurs différentes à chaque utilisation, ce qui a rendu faux le code de beaucoup de candidats.

Q12 - Le principe de la méthode d'Euler n'est pas toujours maîtrisé. L'enchaînement des questions incitait également à utiliser les fonctions précédentes (`vma` et `derive`).

Q13 - Le principe a globalement été compris. Les erreurs ont souvent porté sur les voisins possibles de `p` (les voisins en diagonale n'étaient pas à considérer) ainsi que sur le test de leur appartenance à atteints. Les conditionnelles utilisant `elif` ne permettaient pas de répondre correctement. La construction de la liste voulue à partir d'une liste vide [ ] et de `append` successifs est un incontournable à maîtriser.

Q14 - De nombreux candidats ont utilisé un point en trop dans leur exemple de CAE, ou oublié la moitié des cas dans le dénombrement des CAE les plus courts.

Q15 - Beaucoup de candidats ont confondu la liste vide [ ] avec la valeur spéciale `None`. Trop de copies ont également recalculé deux ou trois fois la même valeur de $\texttt{positions}_\texttt{p}\texttt{ossibles}(\texttt{p}, \texttt{atteints})$ au lieu de la stocker.

Q16 - Comme souvent, la détermination rigoureuse de la complexité (dans le cas le pire) d'une fonction a beaucoup posé problème. Trop de candidats pensent qu'une telle complexité est toujours en $O(n^k)$, où k est le nombre de boucles for présentes dans la fonction. On attendait ici que l'on détermine et somme les complexités de chaque fonction appelée dans la boucle.

Q17 - L'interprétation rigoureuse de la valeur de l'ordonnée représentée dans ce graphique a souvent posé problème. Il ne s'agissait par exemple pas d'un pourcentage.

Q18 - Les tris, ainsi que leur complexité respective, ne sont pas toujours maîtrisés. Rappelons que le tri rapide a une complexité dans le cas le pire en $O(n^2)$, alors que celle du tri-fusion est en $O(n \log n)$.

Q19 - Le principe de trier la liste `chemin` a été plutôt bien compris, et la question assez réussie.

Q20 - L'obtention des formules pour les coordonnées de l'image d'un point par une rotation a souvent été entachée d'erreur.

Q21 à 23 - Figurant en fin d'énoncé, ces questions ont permis de valoriser la prise de recul et la maîtrise des listes des meilleurs candidats.

4.2 Informatique option MP

4.2.1 Généralités

Le sujet s'intéresse à l'analyse et à la programmation du jeu du solitaire. Il est composé de deux parties : une première partie avec un jeu sur le tablier européen en un minimum de coups et une autre partie où le jeu se déroule sur un tablier unidimensionnel.
Le sujet est composé de 30 questions. Elles permettent de reconnaitre des motifs, de définir des coups simples puis composés, d'identifier des parties minimales, de reconnaître des motifs sur un tablier