PC

CONCOURS COMMUN INP

RAPPORT DE L'ÉPREUVE ÉCRITE D'INFORMATIQUE

1/ REMARQUES GÉNÉRALES

L'épreuve d'informatique du concours CCINP proposait de travailler sur l'astro-informatique.

La première partie traitait de l'extraction d'informations contenue dans une base de données astronomique.

La deuxième partie traitait de la classification de ces données à l'aide de l'algorithme de k-moyennes.

La troisième partie traitait de la classification à l'aide de l'algorithme des k plus proches voisins (à ne pas confondre avec les k-moyennes).

Enfin la dernière partie traitait de la reconnaissance de la structure de l'Univers par détection de filaments de galaxies à l'aide de la théorie des graphes.

Le sujet permettait de balayer un large ensemble des compétences décrites dans le programme d'informatique en CPGE, de première comme de deuxième année.

Les candidats continuent d'utiliser correctement le document réponse. Il est rappelé qu'il est utile de réfléchir sur un brouillon afin de porter le plus proprement possible la solution sur le document. La qualité de la présentation et de la rédaction est prise en compte dans la notation. On peut toutefois rappeler que les réponses peuvent commencer au bord gauche de la grille proposée : en effet certains candidats débutent leur réponse à partir de 1, 2 voir 3 cm… Pour l'indentation des programme Python, 1 cm (soit deux carreaux) semble être une valeur raisonnable.

Dans l'ensemble, les questions de SQL ont été bien traitées avec de 80 à 90% des copies qui les abordent, bien qu'on observe assez fréquemment des imprécisions sur les mots clés (`WHILE` ou `WHEN` au lieu de `WHERE`, `ORDRE BY` au lieu de `ORDER BY`) ainsi que l'ordre de référence d'un attribut dans une table (`Attribut.Table` plutôt que `Table.Attribut`). On note également beaucoup de confusion entre `HAVING` et `WHERE`. Les candidats se sont dans l'ensemble investis dans l'apprentissage du SQL et de nombreux points y ont été gagnés.

Pour la lisibilité des programmes, il est vivement conseillé de choisir des noms de variables clairs et intelligibles, de bien soigner son indentation avec l'alignement sur les carreaux ou un trait vertical (avec une position bien définie). En revanche, on évitera absolument d'utiliser un caractère censé représenter l'indentation et répété sur toute la ligne, que ce soit un trait horizontal, un point ou le caractère ␣. La lecture d'un tel code par le correcteur est très difficile et ne peut que désavantager le candidat.

Pour les questions en SQL, il pourrait être intéressant d'utiliser une ligne par mot clé pour assurer la lisibilité de la requête, par exemple

```
SELECT objID, COUNT(*) as C
     FROM SpecObj JOIN PhotoObj ON objID = bestObjId
     GROUP BY objID
     HAVING C > 1
     ORDER BY C DESC
```

L'utilisation des commentaires, bien qu'appréciée quand elle est pertinente, ne doit pas surcharger la copie et faire perdre trop de temps aux candidats. Il n'avait pas été rappelé de ne pas recopier les signatures des fonctions mais la plupart ne l'ont pas fait.

Concernant les manipulations sur les listes, on rappelle que l'opération `L+[i]` nécessite une copie complète de la liste en mémoire, ce qui est beaucoup plus coûteux que l'opération `L.append(i)`. Si la différence est parfois négligeable, elle peut devenir catastrophique dans certaines situations, dont certaines rencontrées dans ce sujet.

On notera que ce sujet présentait un certain nombre de questions de compréhension, de commentaires. Quelques candidats en ont profité pour ne traiter que très très peu de questions de programmation en Python. Attention, le barème a tendance à privilégier les candidats montrant leur maîtrise du langage au programme des CPGE.

Même s'il reste des candidats réfractaires à la matière, on note tout de même un investissement important de la majorité avec 99% de tentative de réponse à la première question de programmation (Q7) et un taux de réussite approchant les 90%. La question était volontairement simple mais permettait de vérifier l'investissement sur les deux ans et mettre en confiance pour la suite.

Le jury rappelle aux candidats que seule une pratique régulière du langage sur machine peut leur permettre de mettre en place un certain nombre d'automatismes dont l'absence est flagrante dans de nombreuses copies. Il est utile d'essayer de se réentraîner sur quelques anciens TP sur machine lors de la préparation aux écrits pour que ces automatismes reviennent juste avant les épreuves. En outre, s'obliger à écrire systématiquement le code sur feuille pendant toute l'année avant de le taper à l'ordinateur pendant les TP vous permettra aussi de soigner la présentation et la lisibilité de celui-ci.

De manière non exhaustive, voici quelques erreurs relevées au gré des copies :

- `range 2` ou `range([0 :10])`
- `a = b` au lieu de `b = a`
- `TRUE, 'True'`
- `If, While, Else`, etc avec une majuscule
- Non parenthésage dans les divisions `somme / j+1` est différent de `somme / (j+1)`
- Condition d'égalité avec un seul symbole `=` ou affectation avec `==`
- Structure : `valeur if condition` avec un `else` sans instruction ou des instructions inutiles `max = max`…
- Utilisation de noms de variable non valides ou de fractions mathématiques au lieu de la barre oblique.

Pour les questions d'algorithmique simple, il n'est pas attendu que le candidat utilise les fonctions `max, sum` ou autre, surtout dans les premières questions qui visent à tester les compétences de bases en Python.

2/ ANALYSE DÉTAILLÉE DES QUESTIONS

Q1 : La majorité des candidats trouvent qu'il faut 8 bits pour représenter une donnée. En revanche, 8 bits = 1 octet semble être souvent inconnu. À noter qu'une réponse sur cinq était complètement fausse. Il a été proposé jusqu'à 8^{256} bits pour coder un unique pixel ou $2^{3049472}$ octets pour stocker une image. Quelques ordres de grandeur sur les tailles des images usuelles auraient pu guider ces copies vers les erreurs de raisonnement y ayant mené.

Q2 : Assez bien traitée, essentiellement des erreurs de calcul et la confusion bit/octet. Par ailleurs, il était demandé de « déterminer », pas de « donner », on attendait donc au moins de voir le calcul posé pour expliquer la démarche.

Q3 : Première question de SQL, traitée par 95% des candidats. Certains candidats oublient ou ne connaissent pas le mot clé `COUNT`.

Q4 : Question assez difficile, seulement un tiers des candidats arrivent à mettre tous les éléments en place sans erreur : sélection des bons attributs, jointure, `GROUP BY` (oublié la plupart du temps), confusion entre `HAVING` et `WHERE`, `ORDER BY` avec un nombre important de `ORDER BY DESC` sans mention d'attribut par rapport auquel faire le classement et un ordre global des instructions assez peu respecté.

Q5 : Question mieux traitée malgré l'oubli de certains critères parmi ceux demandés. Il fallait bien penser à distinguer les deux attributs `z`, seul doublon entre les tables (soit en spécifiant la table d'appartenance, soit en utilisant une requête imbriquée avec aliasing comme cela a été croisé dans quelques copies). À noter que les questions de SQL n'ont pas de raison d'être par ordre strictement croissant de difficulté, il était donc déconseillé de sauter Q5 sous prétexte que Q4 semblait déjà difficile.

Q6 : Énormément de confusion sur la définition de l'algorithme des k-moyennes. Sur les 75% de copies qui ont tenté une réponse, plus de la moitié n'a eu aucun point, très souvent du fait d'une confusion avec l'algorithme des k plus proches voisins qui était exploré plus tard en Q15. Le fait qu'il s'agisse d'un algorithme de classification a été très peu mis en évidence.

Q7 : Première question de programmation, traitée par 99% des copies, parfaitement réussie ou presque par 90% d'entre elles. Il s'agissait pour le candidat de montrer qu'il savait manipuler une boucle `for`, utiliser un accumulateur et faire un dernier calcul avant de renvoyer le résultat voulu. On retrouve toutefois des erreurs sur les bornes du « range » ; des oublis de cumul pour le calcul de la somme. On rappelle en outre qu'il ne faut surtout pas écraser les arguments donnés à la fonction en les « initialisant » par un `S1 = []` en début de fonction (croisé un nombre non négligeable de fois) et que les noms de variables ne peuvent pas comporter de parenthèses, on ne peut donc pas utiliser `d(S1, S2) =` …

Q8 : Question traitée par 98% des copies et la plupart du temps réussie. Attention à bien utiliser une condition composée pour éviter de dupliquer un indice pour lequel à la fois `S1[i]` et `S2[i]` seraient à `None` par utilisation de deux `if` successifs plutôt que la combinaison `if`/`elif`.

Q9 : Question assez bien traitée. Deux erreurs typiques : l'inversion True/False dans le renvoi (attention à ne pas lire l'énoncé trop vite, le nom `est_absent` aurait dû être assez explicite) et, plus embêtant, le fait de renvoyer directement une réponse dès le premier tour de boucle. On rappelle qu'une instruction conditionnelle n'a pas forcément d'instruction `else` associée : quand on ne fait rien «sinon», on n'a pas besoin de le marquer. Ici, l'usage de `return not(element in L)` a été sanctionné car on attendait du candidat qu'il sache correctement itérer sur une boucle en prenant garde à la différence entre élément et position de l'élément.

Q10 : Moins de 10% des candidats ont su trouver un invariant. À la vue des réponses proposées, savent-ils vraiment ce que c'est ?

Q11 : La réponse attendue pour une question de complexité est généralement sous forme d'un grand O. Le calcul (s'il est bien expliqué) permet d'obtenir les points de justifications, mais il ne faut pas oublier de conclure. Une justification succincte convient mais encore faut-il la donner. Ici la complexité de la fonction `a_rejeter` a trop souvent été oubliée pour `distance2`.

Q12 : Question traitée par 90% des candidats et réussie presque parfaitement par 90% de ceux qui l'ont traitée. On retrouve néanmoins des confusions pour l'extraction du spectre : si on boucle sur les indices, il faut mettre `spectres[i]` ; si on boucle sur les spectres directement, il faut utiliser le nom de la variable de boucle.

Q13 et Q14 : Ces deux questions étaient plus difficiles et demandait de se souvenir du cours sur les k-moyennes. Traitées par à peine plus de 50% de copies et parmi celles-ci seulement la moitié ont tenté des choses intéressantes.

Q13: Parmi celles qui ont tenté quelque chose, beaucoup de copies n'ont pas compris la fonction `initialise` (qui permet d'obtenir des centroïdes aléatoires, nécessaires pour une première application de `produit_partition`)

et s'en sont servi pour remplacer les `spectres` dans la suite, ce qui n'a guère de sens (le nombre de spectres étant très important, contrairement au nombre `k` de centroïdes).

Q14 : Beaucoup de copies ne comprennent pas le second terme renvoyé par `principal_et_reste` et tentent de le reconstruire à partir du premier. La condition d'arrêt n'est pas toujours bien gérée, et pas mal de copies renvoient les spectres non-classés restants à la fin alors qu'il était explicitement demandé de les rejeter. Plusieurs copies se sont essayées à une version récursive mais avec généralement une erreur sur la construction du résultat renvoyé.

Q15 : Comme sa jumelle Q6, cette question a été assez mal traitée car la définition de l'algorithme des k plus proche voisins n'est pas maitrisée. Il y a eu un nombre bien trop important de copies pour lesquelles l'algorithme des k plus proches voisins consiste uniquement à trouver les k plus proches voisins d'un objet, et c'est tout. Ici aussi le fait qu'il s'agisse d'un algorithme de classification a été très peu mis en évidence. La notion de vote majoritaire suffisait en général pour distinguer les candidats qui ont passé un peu de temps à apprendre leur cours de deuxième année.

Q16 : Question assez mal traitée également, du fait de la mauvaise connaissance de l'algorithme étudié. Un petit schéma permettait d'expliquer la notion rapidement et avec élégance.

Q17 : Si cette question a été la plus réussie du sujet, quelques copies ont tout de même confondu `k` et `R`.

Q18 : Beaucoup de paraphrases, ce que l'énoncé demandait explicitement d'éviter ; le jury attendait de reconnaître les étapes de l'algorithme. Les meilleures copies sont souvent celles qui ont respecté à la lettre la consigne d'« expliquer en une ligne » plutôt que de faire un long paragraphe. En particulier, faire apparaitre la notion de vote majoritaire pour les lignes 19 à 22 était largement suffisant.

Q19 : La notion de signature semble toujours peu claire pour beaucoup de copies. Concernant le spectre, il était attendu de préciser que c'était une liste de flottants et pas juste une liste. Une réponse utilisant la nomenclature du sujet pour préciser les spécifications des signatures des fonctions était tout à fait valable (et efficace).

Q20 : Beaucoup de « défauts » signalés portaient sur la complexité non-optimale du code, mais la notion faisait référence à « ne pas respecter la spécification ». Ce sont donc de vrais défauts algorithmiques qui étaient attendus : le fait que si la distance d'un ou plusieurs voisins était égale à celle du seuil (peu probable sur le type de données utilisé mais sait-on jamais), le nombre de voisins ne serait pas de `k` (voire pourrait être nul en cas d'égalité stricte des k+1 premières distances), et que si plusieurs types apparaissaient autant, c'était le premier dans l'ordre des types qui était renvoyé. Quelques copies ont repéré l'erreur `type_spectraux` au lieu de `types_spectraux` à la ligne 14, qui, bien qu'elle n'ait pas été placée là intentionnellement, a bien été comptée comme un défaut majeur quand elle était signalée.

Q21 : Question très rarement traitée : 30% de tentatives dont seule une sur trois aboutissait. Cela demandait un peu de recul sur l'algorithme étudié en cours.

Q22 : La définition d'une matrice de confusion semble inconnue pour la plupart des candidats tentant de répondre à cette question.

Q23-24 : Question assez bien traitées pour les candidats ayant pris la peine de bien lire l'algorithme. Beaucoup de copies ont cherché à obtenir un chemin plutôt qu'un arbre, aboutissant à un résultat faux.

Q25 : Question plutôt bien traitée. Les erreurs relevées correspondaient principalement à l'invention de variables (`x, y, z`) non définies dans les arguments. L'utilisation d'une boucle n'était pas forcément nécessaire vu que le nombre de coordonnées était connu et fixé.

Q26 : Attention à l'initialisation de la matrice d'adjacence. Beaucoup de candidats initialisent avec `[[]*N]` (ce qui revient à une liste contenant une liste vide) ou des variations comme `[[0]*N]*N` qui reproduit N fois la même ligne de zéros et donnera aux final une matrice de rang maximal 1. D'autres encore initialisent une liste vide puis tentent d'accéder à l'élément `[i][j]`. Certaines copies ont pensé à exploiter la symétrie, mais ne pas le faire n'était pas pénalisé.

Q27 : Question plutôt bien traitée par 50% des copies. Plusieurs soucis ont néanmoins été relevés : `d['depart']` utilisé au lieu de `d[depart]`, utilisation de `G[i]` ou `G[i][j]` comme clés, traitement du `depart` avant la boucle, aboutissant à son écrasement, ou parfois à des boucles commençant à `1` ou terminant à `len(G)-1` alors que le départ pouvait être quelconque. Il est curieux également de voir de nombreuses copies chercher à tester si le `i` correspond à `depart` à chaque fois plutôt que de tout remplir avec l'infini et de corriger juste le départ ensuite, même si ce point n'était pas pénalisant. Certaines copies n'ont pas compris l'usage d'une note en bas de page et ont rempli avec l'infini au cube. Cela n'a pas été pénalisé, étant considéré comme un défaut du sujet (et l'infini, même au cube, donne tout de même l'infini).

Q28 : Généralement bien traitée par les 70% de copies qui ont lu jusque-là. Comme à la question précédente, une certaine incompréhension dans l'usage des clés d'un dictionnaire en confondant chaîne de caractère et variable, soit ici `dist['gal']` au lieu de `dist[gal]`.

Q29 : Question assez bien traitée par les candidats connaissant la notion de variant. Attention néanmoins à bien définir ce qu'est un variant et à justifier correctement sa décroissance stricte.

Q30 : Question abordée dans seulement la moitié des copies avec souvent des erreurs dans les combinaisons de complexités même si celles des fonctions de base sont souvent bien évaluées.

Q31 : Question traitée par seulement un tiers de copies. La racine a souvent été oubliée malgré la dernière phrase de la question dans l'énoncé. Les solutions les plus élégantes, qui itéraient sur les valeurs du dictionnaire, n'ont naturellement pas eu ce problème en plus d'avoir facilement une complexité linéaire.

Q32 : Question souvent traitée mais pas de façon suffisamment précise en s'appuyant sur ce que fait réellement le code. Si l'aspect retrait des galaxies sans successeurs a souvent été mentionné, celui de la répétition itérative de ce retrait l'a très rarement été. Certaines copies ont néanmoins été jusqu'à faire des schémas montrant le résultat de `nb_etapes` sur un arbre inventé, ce qui fut du plus bel effet.

Q33 : Question (assez logiquement) la moins traitée du sujet, avec souvent le cas échéant un calcul de la valeur moyenne de distance sur toute la matrice `G` plutôt que sur l'`arbre` restant.