4 Informatique

4.1 Informatique commune aux filières MP, PC et PSI

4.1.1 Généralités et présentation du sujet

Le sujet d'informatique commune traitait cette année de la communication numérique. Il était découpé en 2 parties : une première partie portait sur la compression d'un message et une seconde partie portait sur le décodage d'un message à l'aide de l'algorithme de Viterbi.

Les 25 questions du sujet balayaient une partie conséquente du programme d'informatique des deux années de CPGE : calculs de complexité, algorithme glouton, utilisation des dictionnaires, programmation dynamique... Certaines questions étaient d'un niveau élémentaire (requête SQL simple, recherche du maximum d'une liste ...), quand d'autres (essentiellement en fin de sujet) exigeaient une maîtrise et une compréhension plus fines. Un nombre non négligeable de questions portait sur la compréhension des problématiques liées à la communication numérique. L'épreuve abordait donc un large éventail de notions étudiées durant les deux années de préparation tout en évaluant la capacité des candidats à relier ces notions aux problèmes concrets rencontrés lors de la mise en œuvre d'une communication numérique. Elle a ainsi permis d'évaluer et de classer l'ensemble des candidats.

Une analyse détaillée des questions est présentée dans l'annexe R.

4.1.2 Commentaires généraux

- Si certaines copies sont très faibles (voire presque vides), d'autres 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 bien classer les candidats.
- La présentation des codes Python est primordiale : les candidats doivent prêter attention au choix des noms de variables, à l'insertion de commentaires pertinents dans le corps de leurs programmes à l'aide du symbole # (ou en amont si ces commentaires sont trop longs). Il est rarement utile d'écrire un paragraphe de plusieurs lignes pour présenter l'idée générale d'un code Python ; une ou deux phrases suffisent largement.
- Le sujet requiert l'utilisation de différents types de variables (chaines de caractères, dictionnaires, listes...). Le jury déplore le nombre important de confusions liées à leur utilisation. En particulier, un nombre important de candidats utilise *append*, réservé aux listes, avec des chaines de caractères ou des dictionnaires.
- Plusieurs calculs de complexité sont demandés dans le sujet.La notion de complexité semble ne pas être acquise pour une partie des candidats qui ne maîtrise pas l'utilisation de $\mathcal{O}$. L'absence de justification a été sanctionnée. Le jury note aussi l'absence d'esprit critique de certains candidats qui proposent des complexités exponentielles pour un programme ne comportant qu'une boucle *for*.
- Beaucoup d'erreurs de calcul ont été repérées.

4.1.3 Conseils aux futurs candidats

Nous conseillons aux futurs candidats une lecture attentive du rapport du jury ainsi que du programme officiel d'informatique commune : celui-ci peut aider à vérifier la maîtrise des points exigibles aux concours. De plus, certains candidats, de manière quasi systématique, ne traitent pas les questions

nécessitant d'écrire un programme dans le langage Python. Cela montre un manque d'entraînement à écrire des codes, mêmes simples. Un investissement un peu plus important des candidats en informatique commune produirait certainement une nette amélioration.

4.2 Informatique option MP

4.2.1 Généralités

Le sujet s'intéresse à l'analyse et à la programmation d'une méthode de calcul d'une couverture minimum d'un ensemble ordonné par des chaînes disjointes. Il est composé de trois parties : une première partie s'intéresse à la mise en œuvre d'un tri topologique, une seconde partie à la manipulation de la notion de chaînes et d'antichaînes et une troisième partie traite da la notion de couverture.

25 questions composent le sujet.

Dans la première partie, il s'agit, dans un premier temps, de mettre en œuvre une vérification algorithmique élémentaire des propriétés d'une relation d'ordre. Nous avons constaté que de nombreux candidats éprouvaient des difficultés sur ces premières questions de programmation élémentaire. Il s'agit, dans un second temps, de programmer et d'analyser un algorithme de tri topologique d'un ensemble ordonné dont le principe est expliqué en amont. Peu de candidats ont compris le principe et n'ont sans doute pas cherché à vérifier l'interprétation qu'ils en faisaient sur un exemple simple : ils auraient constaté leur erreur.

Dans la seconde partie, le candidat doit s'approprier les codes fournis afin d'en vérifier la validité, de les corriger ou de saisir l'usage qui est fait d'une structure de données abstraites qu'on lui demande de clarifier. Ces questions sont globalement bien traitées.

La troisième partie traite de la construction d'un graphe biparti associé à un ensemble ordonné. Puis de son utilisation pour construite une couverture minimum par des chaînes disjointes : questions souvent abordées, mais rarement bien faites.

Une analyse détaillée des questions est présentée dans l'annexe S.

4.2.2 Analyse de forme

Les programmes présentés par les candidats respectent les règles d'indentation avec des retours à la ligne facilitant la lecture du code. Plusieurs copies restent malgré tout mal présentées, voire même illisibles, pour le correcteur, sans indentation, avec de grosses ratures, des renvois avec des flèches en bas de page, etc. Ces copies sont très difficiles à interpréter.

Plusieurs compositions utilisent des fonctions auxiliaires, ce qui est une bonne chose pour décomposer un programme. Il est recommandé d'utiliser des noms significatifs à l'image de ce que font les concepteurs des sujets. Un commentaire additionnel permet de mieux évaluer la compréhension du sujet ou de la question par les candidats.

4.3 Informatique 1 filière MPI

4.3.1 Remarques générales

Le sujet s'intéresse à l'extraction d'un sous-graphe le plus dense d'un graphe non orienté quelconque. Il est composé d'une unique partie comprenant au total 33 questions.

Q Informatique commune MP, PC et PSI

Q1 - Question mal comprise par la majorité des candidats : l'ambiguïté du décodage n'est que rarement détectée.

Q2 - Question traitée et réussie par la majorité des candidats. Plusieurs erreurs de syntaxe Python sont récurrentes : utilisation de = au lieu de == lors des tests d'égalité ; confusion entre la variable `c` et la chaine de caractères `'c'`.

Q3 - Oubli fréquent des guillemets pour les chaines de caractères. Certains candidats perdent du temps en proposant des explications très longues du fonctionnement de la fonction ; une explication claire et concise est à privilégier.

Q4 - Beaucoup de candidats proposent des complexités dépendant uniquement de n et non de k et n. La notion de complexité semble ne pas être acquise pour une partie des candidats : manque de simplification des $\mathcal{O}$, calculs d'applications numériques. L'absence de justification a été sanctionnée.

Q5 - Question globalement réussie malgré l'oubli fréquent des guillemets pour les chaines de caractères.

Q6 - *Idem* que la question 4.

Q7 - Mauvaise manipulation des dictionnaires, en particulier lors de l'ajout d'un couple (clé,valeur) et du test de la présence d'une clé. Certains candidats font le choix d'utiliser `keys()`, mais beaucoup d'entre eux oublient les parenthèses. Certains candidats utilisent `if/if` quand il est nécessaire d'utiliser `if/elif` ou `if/else`. Les fonctions proposées ne respectent pas toujours la complexité demandée.

Q8 - Question bien traitée dans l'ensemble malgré une syntaxe approximative lors de l'utilisation de `DISTINCT`. De rares candidats ne connaissent pas la syntaxe `SELECT ... FROM ...` ; certains proposent des requêtes se terminant par `WHERE` sans condition. Plusieurs candidats, à cause d'une mauvaise lecture du sujet, pensent que table fait partie du nom de chaque table.

Q9 - Requête assez compliquée. Plusieurs candidats proposent des sous-requêtes alors que le sujet demande explicitement UNE requête. L'utilisation des jointures est assez peu maitrisée. Erreurs de syntaxe SQL récurrentes : confusion entre `WHERE` et `HAVING` ; confusion entre `SUM` et `COUNT` ; utilisation de attribut.table au lieu de `table.attribut` ; mauvaise maîtrise de `GROUP BY`.

Q10 - Question bien traitée, à condition de ne pas faire d'erreurs de calcul.

Q11 - La méconnaissance du dépaquetage d'un t-uple et de la syntaxe d'assignation de plusieurs variables de manière simultanée est la source de nombreuses erreurs : l'instruction `t = codeCar(c, g, d)` suivie par `g = t[0]` et `d = t[1]` est correcte alors que l'instruction `g = codeCar(c, g, d)[0]` suivi de `d = codeCar(c, g, d)[1]` ne l'est pas.

Q12 - *Idem* que la question 10.

Q13 - Mauvaise justification en général.

Q14 - La condition d'arrêt de la boucle while est souvent incorrecte à cause d'un manque de compréhension du sujet. Trop de candidats utilisent append pour ajouter un élément à une chaine de caractères. Certains candidats font des appels multiples et coûteux de la fonction `decodeCar`. Il est préférable de stocker le résultat de l'appel d'une fonction dans une variable plutôt que d'appeler une fonction de manière redondante.

Q15 - Beaucoup d'erreurs dans le dénombrement des arcs. Certains candidats proposent des résultats aberrants ; il est recommandé aux candidats d'utiliser leur esprit critique pour vérifier la cohérence de leur réponse.

Q16 - Structure du graphe généralement correcte mais beaucoup d'erreurs de calculs sur les probabilités.

Q17 - Manque de justification concernant l'impossibilité d'utiliser une exploration exhaustive.

Q18 - Programme assez classique malgré le fait qu'il soit en fin de sujet. Question globalement bien traitée. Un point d'attention doit être donné à l'initialisation qui n'est pas toujours correcte. Certains

candidats proposent de multiples parcours inutiles de la liste.

Q19 - Question peu traitée, mais globalement réussie par les candidats y accordant le temps suffisant.

Q20 - Question peu traitée. Certains candidats semblent considérer que tous les algorithmes gloutons ont la même complexité.

Q21 - Question globalement réussie par les candidats la traitant.

Q22 - L'algorithme de Dijkstra est souvent mentionné, mais peu de candidats pensent à utiliser l'opposé du logarithme. Même si cela n'est pas sanctionné, il est regrettable qu'un aussi grand nombre de candidats ne connaisse pas l'orthographe, même approximative, de Dijkstra.

Q23 - Question très peu traitée, mais correctement lorsque c'est le cas.

Q24 - Question très peu traitée et abordée souvent de manière superficielle ; surement par manque de temps.

Q25 - Question très peu traitée. Manque de justification.

↑RETOUR