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 portait cette année sur l'utilisation de différentes méthodes de résolution du problème du sac à dos : algorithme glouton, programmation dynamique, algorithme par séparation et évaluation et optimisation par colonie de fourmis.

Certaines de ces méthodes (algorithme glouton et programmation dynamique) font partie du programme d'informatique de CPGE et les questions les traitant ont donc permis d'évaluer la capacité des candidats à adapter des algorithmes connus afin de les appliquer au problème spécifique du sac à dos.

Les autres méthodes étaient inconnues pour la majorité des candidats et leur étude a mis en évidence la capacité de certains candidats à comprendre et s'approprier un algorithme nouveau. Plusieurs questions portaient sur la compréhension des différentes méthodes et sur les différences qui existent entre elles.

Le sujet était composé de 24 questions balayant une partie conséquente du programme d'informatique des deux années de CPGE. Certaines étaient d'un niveau élémentaire (recherche de l'indice du maximum d'une liste, multiplication de tous les éléments d'une liste par une constante) quand d'autres exigeaient une maîtrise et une compréhension plus fine.

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 au problème du sac à dos.

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

4.1.2 Commentaires généraux

Dans l'ensemble, les codes Python sont correctement indentés et commentés et les candidats écrivent correctement (en majuscule ou minuscule) les mots clés des différents langages. Il est rappelé aux candidats que si des commentaires sont indispensables lors de l'écriture de codes complexes, il est rarement utile d'écrire un paragraphe de plusieurs lignes pour présenter l'idée générale d'un code Python.

Une attention particulière est attendue lors de la définition et l'utilisation des variables : une variable doit être définie et être accessible si elle utilisée ensuite ; il est déconseillé d'appeler deux variables avec le même nom même si l'un des noms est en majuscule et l'autre en minuscule (par exemple, `p` et `P`) ; un nom déjà utilisé (pour une autre variable ou une fonction) ne doit pas être réutilisé.

Il est demandé de faire preuve de rigueur dans l'utilisation de la syntaxe Python. Exemples de manque de rigueur récurrents : oubli de parenthèses, oubli de l'opérateur `*`, utilisation $\leq$ au lieu de `<=`, utilisation des notations mathématiques de l'énoncé au lieu des variables informatiques correspondant.

Plusieurs calculs de complexité sont demandés dans le sujet. Il est demandé de faire preuve d'esprit critique dans les réponses qu'ils proposent : par exemple, certains candidats proposent des complexités plus importantes pour le tri par insertion dans le meilleur cas que dans le pire cas quand d'autres proposent des complexités fantaisistes. De plus, il est attendu que la réponse soit sous forme de complexité asymptotique : $\mathcal{O}(n^2)$ et non $\frac{(n+3)(n+10)}{2}$.

4.1.3 Conseils aux futurs candidats

Nous conseillons aux futurs candidats une lecture attentive du rapport du jury afin d'éviter les erreurs récurrentes dans les copies.

Nous conseillons de se familiariser avec le programme officiel d'informatique commune afin de vérifier la maîtrise des points exigibles aux concours.

4.2 Informatique option MP

4.2.1 Généralités

Le sujet de cette année porte sur les langages rationnels et propose de construire un automate reconnaissant un langage L donné à partir des réponses fournies par un oracle qui sait répondre aux questions

- un mot w appartient-il au langage L ?
- un automate $\mathcal{A}$ reconnaît-il le langage L et, si la réponse est non, donner un contre-exemple, c'est-à-dire un mot appartenant à L mais non reconnu par $\mathcal{A}$ ou un mot reconnu par $\mathcal{A}$ mais n'appartenant pas à L.

Partie 1. Cette partie permet de s'approprier les premières notions du sujet avec quelques questions de cours et plusieurs fonctions simples à écrire en `OCaml`, certaines donnent un cadre pour guider l'écriture du code.

Partie 2. Le sujet introduit la notion de séparabilité de deux mots par un mot et fait travailler sur cette notion et la relation d'équivalence associée.

Partie 3. Le sujet utilise la séparation de mots dans un arbre de décision et étudie des propriétés de ceux-ci.

Partie 4. Cette partie donne la définition d'un automate à partir de certains arbres par itération de constructions. La dernière question montre qu'on a construit un automate minimal.

On remarque que l'oracle peut être défini à partir d'un automate reconnaissant un langage L : la construction permet donc de minimiser un automate.

12 des 28 questions demandent d'écrire des fonctions dans le langage `OCaml`.

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

4.2.2 Commentaires généraux

- Les correcteurs ont été surpris par le nombre de candidats, de l'ordre du tiers des copies, qui écrivent

  ```
  if condition then true else false
  ```

  Bien entendu cette construction donne le bon résultat mais elle laisse entendre un manque de compréhension des booléens.

Q Informatique commune MP, PC et PSI

Q1 - La syntaxe SQL n'est pas entièrement maitrisée pour une majorité de candidats. Erreurs fréquentes : confusion dans l'ordre des instructions, utilisation de `AND` (au lieu d'une virgule) entre les différents attributs sélectionnés avec `SELECT`, utilisation d'une virgule (à la place de `AND`) entre les conditions spécifiées après `WHERE`, pas de précision de l'attribut utilisé pour le classement lors de l'utilisation de `ORDER BY`.
Il est conseillé aux candidats de faire un effort de présentation pour les requêtes SQL : mettre en majuscule les éléments de langage et en minuscule les éléments spécifiques aux tables ; aller à la ligne régulièrement.

Q2 - *Idem Q1.*

Q3 - La manipulation des booléens n'est pas optimale : sans être fausse l'utilisation du code `if a == b : return True ... else : return False` au lieu de `return a == b` montre le manque de familiarisation avec cet objet.
Certains codes manquent d'efficacité dans l'écriture des tests conditionnels. Il n'est pas nécessaire d'introduire `else` dans le bloc d'instructions suivant :

```
if condition :
    p=p +1
else :
    p=p
```

Q4 - *Idem Q3.*

Q5 - Question bien traitée.

Q6 - Le nombre de feuilles est correctement déterminé mais un grand nombre de candidats n'a pas remarqué que la complexité dépend du nombre de feuilles ET de la complexité des fonctions `profit` et `contrainte`.

Q7 - Il est conseillé aux candidats d'être concis dans leur justification pour ce type de questions : plus d'une page de justification n'est pas nécessaire et cela leur fait perdre un temps considérable.

Q8 - Pour les questions contenant un code à trous, il n'est pas nécessaire de recopier l'intégralité du code.
Certains candidats n'ont pas compris la structure des données à utiliser.

Q9 - La complexité du tri par insertion n'est pas maitrisée par un nombre important de candidats et plusieurs candidats le confondent avec le tri à bulles. Les tris sont au programme et une complexité linéaire ou quadratique doit pouvoir être reconnue par les candidats.

Q10 - *Idem Q8.*

Q11 - Les justifications proposées par les candidats manquent de précision : pour montrer que l'algorithme glouton ne fournit pas une solution optimale, il est nécessaire de comparer les profits des solutions gloutonnes et optimales.
Une solution est optimale ou ne l'est pas ; elle n'est pas "moins/plus optimale" qu'une autre solution.
Certains candidats font la confusion entre optimalité et complexité.
Le jury attend que les candidats fassent preuve d'esprit critique : le sujet pousse à montrer que l'algorithme glouton ne propose pas une solution optimale, les réponses trouvées aux questions 7 et 11 doivent donc être différentes pour aller dans ce sens.

Q12 - Beaucoup d'erreurs de calculs.

Q13 - Le résultat attendu dépend de n et de b car rien ne permet d'affirmer que $b = \mathcal{O}(n)$.

Q14 - Peu de candidats ont indiqué que la complexité de la partie ajoutée est en $\mathcal{O}(n)$.

Q15 - Trop de candidats manipulent les dictionnaires en utilisant la syntaxe propre aux listes sans réaliser que le résultat ne correspond pas à leurs attentes : par exemple `a[0]` renvoie l'éventuelle valeur

associée à la clé `0` et non le "premier" élément du dictionnaire `a`. Par ailleurs, il n'est pas possible de parler du "premier" élément d'un dictionnaire car un dictionnaire n'est pas ordonné.

Q16 - La manipulation des booléens n'est pas optimale : il est préféré l'utilisation de `return bool1 and bool2` à

```
if bool1 == False  :
    return False
elif bool2 == False  :
    return False
else  :
    return True.
```

La notion de mutabilité des listes n'est pas maitrisée par plusieurs candidats. Il est rappelé que l'instruction `S=Sk` implique que `S` et `Sk` désignent la même liste. Par conséquent, une modification de la liste `Sk` modifie aussi la liste `S`.

Q17 - Plusieurs candidats n'ont pas compris que la fonction demandée `KPpse` était une fonction récursive inspirée de la fonction `KPforceBrute`. Par conséquent, au lieu de faire appel à la fonction `KPpse`, ils font appel à la fonction `KPforceBrute`.

Q18 - Plusieurs candidats ont proposé le code `for i in T  : i=i*rho` qui ne modifie pas les éléments de la liste `T`.

Q19 - Question abordable (recherche de l'indice du maximum d'une liste) mais plusieurs candidats n'ont pas utilisé les bonnes variables.

Q20 - Question bien traitée par les candidats étant arrivé à cette question.

Q21 - Question peu traitée.

Q22 - Question peu traitée.

Q23 - Question peu traitée.

Q24 - Question traitée par plusieurs candidats n'ayant pas traité les questions précédentes. Les analyses proposées manquent parfois de précision mais dans l'ensemble, les candidats ont su relever les points essentiels : optimalité et rapidité des solutions proposées par les différents algorithmes.

RETOUR