MP

CONCOURS COMMUN INP

RAPPORT DE L'ÉPREUVE ÉCRITE D'INFORMATIQUE

1/ PRÉSENTATION DU SUJET

Le sujet est composé de trois parties indépendantes proposant chacune des questions de difficulté croissante.

La première partie traite du problème de coloration des sommets d'un graphe. Après une introduction des définitions et propriétés mises en jeu, elle propose une implémentation guidée d'un algorithme de coloration et une application. Cette partie mobilise notamment les connaissances et les compétences de l'informatique tronc commun avec une programmation imposée en Python.

Dans les deux autres parties, le langage de programmation utilisé est OCaml. La deuxième partie traite de la résolution du problème de satisfiabilité d'une formule booléenne et évoque le lien entre la logique propositionnelle et la déduction naturelle. Elle mobilise les connaissances et compétences acquises dans ces domaines en première et en deuxième année. Au niveau de la programmation, elle évalue l'aisance quant à la manipulation de tableaux et de listes en OCaml.

La troisième partie s'intéresse à la reconnaissance de motifs dans un texte via une méthode naïve de recherche textuelle puis à l'aide d'un automate. Après une introduction de définitions propres à ce domaine, elle met en œuvre la manipulation de chaînes de caractères et d'automates.

2/ REMARQUES GÉNÉRALES

Le sujet a permis de classer les candidats avec une moyenne de **10,26** et un écart type de **4,07**. Sa longueur et le niveau de difficulté semblent adaptés aux connaissances et compétences des étudiants qui ont suivi l'option informatique. Quelques candidats ont abordé toutes les questions du sujet. On peut mentionner la qualité de certaines copies et au contraire, un petit nombre de copies très lacunaires.
Les correcteurs sont globalement satisfaits de la présentation des copies et rappellent qu'une présentation du code, respectant les bonnes pratiques de programmation (indentation, retours à la ligne, noms de variables significatifs), permet une meilleure lisibilité du code. Une partie du barème prend en compte la présentation de la copie.

Le sujet a abordé largement des notions de première et deuxième années de l'informatique tronc commun et de l'option informatique. Les différentes compétences développées durant les deux années de classes préparatoires, cruciales dans l'exercice du métier d'ingénieur, sont évaluées, à savoir :

- analyser et modéliser,
- imaginer et concevoir une solution,
- décrire et spécifier,

- mettre en œuvre une solution,
- justifier et critiquer une solution,
- communiquer à l'écrit.

De façon générale, les erreurs relèvent des points suivants.

- Analyse et raisonnement : manque de clarté et de rigueur lors des justifications, difficulté à modéliser et concevoir une solution dans un contexte nouveau.

- Algorithmique :
  - Conception erronée de l'algorithme proposé qui ne répond pas ou répond partiellement à la problématique
  - Méconnaissance de certaines techniques et algorithmes au programme

- Programmation :
  - Erreurs de syntaxe : accès à une valeur et parcours d'un dictionnaire en Python, mauvaise maîtrise des indices d'un tableau, des booléens en OCaml, confusion entre les langages de programmation Python et OCaml
  - Sémantique du langage OCaml : usage inadapté de la programmation fonctionnelle ou impérative

La partie suivante décrit plus précisément les observations relevées lors de la correction des copies des candidats.

3/ REMARQUES SPÉCIFIQUES

PARTIE I

Cette partie est constituée de 11 questions.
Les 3 premières questions, relativement bien traitées, attestent de la bonne compréhension des définitions introduites.

Q1. Bien traitée par les candidats.

Q2-Q3. Ces questions ont souvent été le lieu de défauts de raisonnement. Certains candidats ont utilisé un raisonnement par récurrence avec une hérédité non démontrée. Les correcteurs attendent des arguments clairs et précis.

Les questions **Q4** à **Q9** consistent en la programmation de fonctions auxiliaires permettant de coder l'algorithme de coloration en question **Q10**. Hormis quelques erreurs de syntaxe, ces questions ont été relativement bien traitées. Un petit nombre de candidats ne maîtrisent pas du tout le langage Python. Dans les boucles *for*, on note un usage quasi systématique de l'itération sur les indices alors qu'une itération sur les éléments d'une liste peut être plus appropriée. Nous rappelons qu'une partie de l'épreuve est toujours consacrée à l'informatique tronc commun.

Q4. Généralement bien traitée sauf par un petit nombre de candidats qui confondent la représentation par liste d'adjacence avec la représentation par matrice d'adjacence.

Q5. Bien traitée par les candidats qui maîtrisent l'usage des dictionnaires.

Q6. Toutes les utilisations cohérentes de la fonction *tri* ont été acceptées (tri en place ou non).

Q7. Il est nécessaire de vérifier qu'on dispose bien d'une 2-coloration, selon la définition donnée en introduction ; en particulier, les seules couleurs possibles sont *0* ou *1*.

Q8. Cette question atteste de la bonne compréhension de l'algorithme proposé.

Q9. Bien réussie dans l'ensemble. Avant de lire la couleur d'un sommet dans le dictionnaire *dc*, il faut vérifier que la clé est présente.

Q10. Mise en œuvre de la solution qui est généralement bien réussie par les candidats qui ont traité cette question. Compte tenu de l'énoncé de l'algorithme, les 2 types de retour *liste* et *dictionnaire* ont été acceptés dans les implémentations proposées.

Q11. Un certain nombre de candidats a traité partiellement cette question qui met en œuvre la capacité à modéliser et résoudre un problème concret avec des structures informatiques.

PARTIE II

Cette partie est composée de 11 questions.

Q12. Le constructeur *Var* est souvent omis dans le codage de la clause.

Q13. Il est approprié de réinvestir le résultat obtenu dans la question précédente. Certains candidats n'ont pas implémenté la formule avec une liste comme précisé.

Q14. Les correcteurs relèvent que la syntaxe des booléens en OCaml n'est pas toujours maîtrisée. Une bonne utilisation des opérateurs sur les booléens est plus efficace que des structures conditionnelles « emboitées ».

Q15. Le traitement du cas de la liste vide est problématique dans de nombreuses copies.

Q16. Généralement bien traitée.

Q17. Question ouverte difficile pour un bon nombre de candidats. Il est opportun que les candidats commentent leur programme pour indiquer leur choix de conception. Les algorithmes proposés ne traitent pas tous les cas, notamment celui où le tableau ne contient que des booléens valant *true*. Au niveau de la programmation, la sortie prématurée d'une boucle *for* est souvent mal codée en OCaml. Un bon usage des *exceptions* est adéquat. Nous rappelons qu'il n'existe pas d'élévateur à la puissance pour les entiers en OCaml.

Q18. Attention à prendre en compte toutes les valuations.

Q19. Question souvent mal traitée avec une complexité polynomiale trouvée par certains candidats. Lorsqu'une fonction fait appel à d'autres fonctions implémentées, il est nécessaire de prendre en compte la complexité de ces fonctions pour obtenir un ordre de grandeur correct de la complexité temporelle attendue.

Q20. Question rarement traitée. Peu de candidats ont pu proposer une stratégie de retour sur trace pour résoudre le problème.

Q21-Q22. Ces deux dernières questions mettent en jeu la déduction naturelle. Pour ces questions, tout pseudo-code clair, toute preuve correcte résolvant le problème ont été acceptés.

PARTIE III

La dernière partie est constituée de 14 questions réparties en deux sous-parties.
L'introduction rappelle les notions de *mot, facteur, préfixe,* au programme de deuxième année et définit les notions de *bord* et de *périodes* d'un mot.
Les questions **Q23** à **Q30** attestent de la bonne compréhension des notions définies au début de la partie.

Q23. Manque de justification et de clarté constaté.

Q24. Généralement bien comprise mais le code proposé reste souvent inintelligible à cause des erreurs de syntaxe. Nous rappelons que l'opérateur de la concaténation ^ n'est pas un constructeur et qu'on ne peut pas procéder par filtrage.

Q25. Généralement bien réussie.

Q26. Justification souvent incomplète. Il faut justifier que la valeur trouvée est bien un minimum.

Q27. Mêmes observations que pour la question **Q24**. Dans la recherche de la période, on relève dans certaines copies des erreurs liées à la manipulation de la longueur des mots.

Q28-Q29-Q30. Questions peu traitées qui ont permis de distinguer les meilleurs candidats. Les différentes étapes du raisonnement par récurrence ne sont pas toujours maîtrisées.

La deuxième sous-partie propose une identification de motifs dans un texte à l'aide d'un automate.

Q31. Relativement bien traitée. Certains candidats omettent les apostrophes pour la désignation de caractères et ne savent pas comment définir la fonction *transition* et l'intégrer dans la variable *automate*.

Les questions suivantes ont été rarement abordées.

Q32-Q33. Questions rarement traitées. La notion d'automate émondé et l'algorithme d'élimination des états ne sont généralement pas connus. L'état initial et l'état final sont souvent omis.

Q34. Question traitée dans les meilleures copies.

Q35. Les correcteurs notent une faible utilisation de la fonction *List.mem* pour tester l'appartenance de l'état courant à la liste des états finaux. En revanche, l'usage du mot clé *in* est une erreur.

Q36. Les correcteurs attendent des arguments clairs pour justifier la correction de l'algorithme.

4/ CONCLUSION

L'épreuve écrite de l'option informatique a pour but d'évaluer les connaissances et les compétences acquises en informatique durant les deux années de classes préparatoires. Nous encourageons les candidats à travailler de concert toutes les notions au programme et la programmation dans les 2 langages Python et OCaml.