

# Informatique

## Présentation du sujet

Le sujet est construit autour d'un des thèmes du programmes de seconde année, le traitement des images. Il s'intéresse à la mise en œuvre de méthodes numériques visant à concevoir des photomosaïques, images composées à la manière d'une mosaïque d'une multitude de petites images appelées vignettes. Le sujet comporte 32 questions réparties sur 4 parties et fait largement appel aux connaissances algorithmiques et pratiques du programme de première année :

- la première partie traite du codage des images en termes de pixels et de codage RGB pour se terminer par l'écriture d'une fonction de conversion d'une image en niveaux de gris ;
- la deuxième partie étudie plusieurs solutions algorithmiques de redimensionnement d'images de complexités temporelles différentes. La partie se termine par une synthèse discutant des usages respectifs de ces solutions ;
- la troisième partie aborde le thème des bases des données par l'écriture de requêtes sélectionnant une image source et des vignettes ;
- la quatrième partie aboutit à la construction d'une photomosaïque. Les deux dernières questions laissent une part importante à l'initiative des candidats.

Outre la maîtrise des connaissances informatiques du programme, l'écriture syntaxiquement correcte de codes et l'analyse de leurs performances, le sujet évalue l'aptitude des candidats à porter un regard critique sur des propositions de codes. Ce sujet a très largement permis au jury d'évaluer la qualité et le niveau de compétences de chaque candidat.

## Analyse globale des résultats

Chacune des parties du sujet est progressive, ce qui a permis d'étaler les notes des candidats.

Les candidats ont pu montrer leurs compétences en informatique. Certaines compétences semblent assez bien maîtrisées, comme par exemple la programmation en Python, qui ne pose pas de problème à la majorité d'entre eux. D'autres sont plus discriminantes, comme la compréhension de la complexité, qui a permis à certains candidats de tirer leur épingle du jeu.

La partie sur les bases de données a été globalement bien réussie, voire mieux réussie que les autres parties. Pour 14 % des candidats, cette partie représente même la majorité des points. Le jury recommande aux candidats d'utiliser la liberté qu'offre SQL de sauter des lignes ou d'indenter pour rendre la structure de leur code plus facilement lisible.

Les questions d'analyse de programme et de compréhension ont permis de mesurer la capacité à prendre du recul face au problème posé. Les questions ouvertes ont permis d'évaluer la créativité des candidats et leur capacité à défricher un problème.

Le jury est globalement satisfait du niveau en informatique atteint par les candidats et encourage les futurs candidats à travailler cette matière importante dans le cursus d'un ingénieur.

## Commentaires sur les réponses apportées et conseils aux futurs candidats

### Programmation en Python

Le sujet comporte un certain nombre de questions de programmation en Python (**Q4, Q6, Q8, Q13, Q26, Q27, Q28, Q29**) où il est demandé aux candidats d'écrire une fonction satisfaisant une certaine spécification. Cette spécification contient plusieurs exigences (type d'entrée, type de sortie, ce que doit calculer la fonction, gestion des problèmes de débordements d'entiers, complexité...). Les candidats sont invités à faire attention à toutes ces exigences. Idéalement, la fonction écrite par le candidat les satisfait toutes.

Ces questions ont globalement été assez bien réussies. Certains candidats, n'arrivant pas à satisfaire toutes les exigences, ont parfois proposé des fonctions ne satisfaisant qu'une partie de ces exigences, que le jury a évaluées comme valant une partie des points de la question. Malheureusement, dans un certain nombre de copies, certaines exigences n'ont pas été satisfaites par simple négligence, par exemple en renvoyant un entier au lieu d'un `uint8`.

La plupart des candidats n'ont pas hésité à utiliser les fonctions précédemment définies, c'est une bonne chose. Quelques candidats préfèrent reprogrammer les fonctions précédentes au lieu de les utiliser. Le jury ne peut que décourager cette pratique qui est source d'erreurs et qui n'est pas une bonne manière de programmer.

Lorsque les candidats ont le choix entre plusieurs manières de programmer, ils ne doivent pas hésiter à préférer les outils qu'ils maîtrisent, fussent-ils plus simples. Utiliser une fonctionnalité avancée de Python qu'on connaît mal alors qu'une fonctionnalité plus basique peut convenir n'est pas une bonne stratégie.

Le jury a remarqué sur quelques copies des confusions entre le langage Python et le langage mathématique : les symboles comme la barre de fraction ou le symbole mathématique de la partie entière ne peuvent pas être utilisés en Python.

La dernière question du sujet (**Q32**) était une question de programmation ouverte, laissant les candidats exprimer leur créativité. Cette dernière question a permis de valoriser les meilleures copies.

### SQL

Le sujet comporte un certain nombre de questions demandant d'écrire des requêtes SQL (**Q19, Q20, Q21, Q22, Q23, Q25**). Les premières questions de SQL ont été réussies par une large majorité de candidats. Ces questions, très progressives, ont permis de bien différencier les candidats selon leur maîtrise du SQL.

Le jury attire l'attention des candidats sur l'importance de bien comprendre la nature des objets manipulés (valeurs, attributs, tables, ...).

### Questions de compréhension

Le sujet comporte des questions de compréhension de code (**Q10 et Q14**). Il ne s'agit ici évidemment pas de paraphraser le code, mais bien d'en extraire les idées principales pour en comprendre le fonctionnement.

### Prise de recul

Certaines questions, demandent de prendre du recul par rapport au problème (**Q5, Q10, Q14, Q16, Q18, Q24, Q31**). Il est préférable de rédiger de manière claire et synthétique, sans diluer le propos et souvent un schéma est le bienvenu.

## Complexité

Plusieurs questions portent sur la complexité (**Q9**, **Q11**, **Q15**, **Q17**, **Q30**). Le jury s'attend à une justification laconique de la complexité. Cette justification permet au jury d'évaluer la compréhension du problème de la complexité par le candidat, mais aide aussi grandement les candidats à clarifier leur pensée. En effet, la majorité des complexités qui ont été données dans les copies sans justification sont fausses.

Certaines copies ont oublié la complexité cachée due à des slicing ou à la fonction `mean`.

## Remarques particulières sur certaines questions

**Q1.** C'est une question de dénombrement, 66 % des candidats ont donné la bonne formule. Lorsqu'une question demande un nombre, le jury s'attend à ce que l'application numérique soit effectuée, 52 % des candidats ont la bonne formule et ont pensé à faire l'application numérique.

**Q2.** Pour l'essentiel, les erreurs à cette question sont dues à une mauvaise lecture de l'énoncé, notamment à une confusion entre le pixel blanc demandé et le pixel noir donné en exemple. Le jury est satisfait que l'essentiel des candidats sache faire la différence entre une instruction et une fonction.

**Q3.** Que 280 ne soit pas représentable sur 8 bits a échappé à un certain nombre de candidats.

**Q4.** Il est demandé aux candidats d'écrire une fonction avec de nombreuses spécifications (meilleure approximation, `uint8`, etc.). 88 % des candidats ont écrit une fonction répondant à au moins une spécification, mais seuls 10 % ont répondu à toutes les spécifications.

**Q13.** La principale difficulté à cette question a été l'exigence d'une complexité en  $O(N)$ . La majorité des candidats a donné une solution satisfaisant au moins une partie des exigences, seule une minorité a obtenu la complexité en  $O(N)$ .

**Q19.** Cette première question de SQL permet de tester les compétences élémentaires des candidats dans ce langage. Elle a été traitée par 96 % des candidats et a été parfaitement réussie par 82 % des candidats. Moins de 1 % des candidats ayant traité cette question ont donné une réponse totalement fausse.

**Q20** Cette question teste la capacité à écrire des jointures et à écrire une condition correcte (avec un « OR »). 30 % des candidats ont parfaitement réussi cette question, 83 % l'ont partiellement réussie. Les erreurs viennent en majorité d'une mauvaise écriture du JOIN ou d'une confusion entre AND et OR dans la condition du WHERE.

**Q25** Les étudiants devaient utiliser la structure qu'ils ont eux-mêmes créée à la **Q24**, ce qui a permis de valoriser ceux qui ont proposé, à la question précédente, une structure simple et efficace par rapport à ceux qui ont choisi une structure complexe et difficile à utiliser.

**Q27** Une des difficultés de l'énoncé était l'exigence : « en prenant garde aux dépassements de capacité ». Certains candidats ont purement et simplement ignoré cette contrainte.

**Q31 et Q32.** Le sujet termine par deux questions ouvertes, permettant d'évaluer la capacité des étudiants à réfléchir sur un problème qui n'a pas été déjà cadré. Ces questions ont été peu abordées, mais le jury est positivement surpris par la pertinence des réponses de certains candidats.

## Conclusion

Le sujet aborde de nombreux points du programme. Les résultats sont globalement satisfaisants.

Le recul vis-à-vis du problème posé, la compréhension des objets manipulés, la clarté et la concision de l'expression sont autant d'atouts qui serviront les candidats en informatique et, plus généralement, dans leur carrière d'ingénieur.