le barycentre pour en extraire ses trois coordonnées, ce que nous avons sanctionné puisqu'une solution simple et plus économe existait.

Q22 - Assez bien réussie ; on attendait des candidats qu'ils réinvestissent les fonctions définies précédemment.

Q23 - L'opérateur + n'est pas adapté pour les listes (il les concatène). La projection sur l'axe vertical a parfois été oubliée.

Q24 - On a pu croiser certaines structures en boucles imbriquées :

```
for i in range(len(L1) :
for j in range(len(L2)) :
...
```

qui témoignent d'une mauvaise compréhension de l'algorithme de fusion de deux listes.
De manière générale, cet algorithme classique de cours devrait être mieux maitrisé par les candidats. Parmi ceux qui connaissaient l'algorithme, beaucoup ont oublié de comparer les aires des éléments (facettes) et pas simplement les éléments eux-mêmes. Enfin, la version récursive de la fusion de deux listes nous semble à éviter absolument, car très peu efficace : nous encourageons les candidats à bien maitriser (et présenter) la version itérative.

Q25 - Le cas de base a souvent été oublié ou mal traité.

Q26 - Le principe (tri puis extraction) a plutôt été maitrisé, mais la condition sur le nombre de facettes a posé de nombreux problèmes. On pouvait très facilement vérifier soi-même la cohérence de l'expression sur de petites valeurs (`len(maillageG)=2` et `len(maillageG)=3`) pour l'inclusion de la médiane.

Q27 - Question assez simple si on avait bien compris le sujet et le principe de la méthode d'Euler. Nous avons fait preuve de tolérance concernant le type de variable utilisé (scalaire ou vecteur) en tenant compte de l'ambiguïté de l'énoncé.

5.2 Informatique option MP

5.2.1 Généralités

Le sujet s'inscrit dans le domaine de la compression de données. Il traite particulièrement le cas du comptage efficace d'occurrences de symboles (codes Unicode) dans un texte, puis comment coder, à l'aide d'un encodeur efficace, un texte dans une suite de bits.
Le sujet comporte 33 questions. Même s'il traite d'un unique problème de bout en bout, le sujet est découpé en deux parties indépendantes. Les candidats ont bien pris en compte cette séparation et ont bien su gérer le passage de l'une à l'autre en vue de répondre au maximum de questions.
En général, les candidats ont bien compris le problème posé. Ils répondent aussi bien aux questions de programmation et de complexité qu'aux questions portant sur des démonstrations et/ou des justifications rigoureuses.
Enfin, les correcteurs ont noté, avec satisfaction, qu'il y avait une diminution importante de compositions traitant exclusivement les questions portant sur l'écriture de programmes ou les questions portant sur des démonstrations.

5.2.2 Analyse Globale

Les candidats ont une bonne compréhension des structures complexes mises en œuvre dans l'énoncé du problème. Ils font un usage de la récursivité souvent clair et efficace. Le niveau des réponses et des copies est satisfaisant voire bon. Pour les questions programmation, les correcteurs ont pris en

compte l'exactitude algorithmique : bonnes valeurs initiales et bonnes valeurs héréditaires pour les programmes récursifs, bonnes bornes et bons indices dans les boucles, mises à jour des bonnes variables avec les bonnes valeurs.
Les petites erreurs de syntaxe qui seraient relevées immédiatement par un compilateur ne sont pas sanctionnées pour les questions les plus difficiles. Une partie non négligeable de candidats écrivent des fonctions OCaml imprécises. Il est important de distinguer la programmation fonctionnelle et la programmation impérative avec effets de bord. On trouve beaucoup de confusions entre quantités calculées renvoyées et arguments modifiés. A titre d'illustration, on peut mentionner :

1. les concepts de références à des variables en OCaml avec des confusions entre l'usage de `x`, `!x`, `:=`, `ref` et `let`
2. l'opérateur d'ajout `::` sur les listes qui est souvent vu comme un opérateur à effet de bord. Les manipulations en présence d'effets de bords restent moyennement maîtrisées par plusieurs élèves.

Un nombre important de programmes comporte des fonctions auxiliaires internes. Si certaines sont indispensables, beaucoup se révèlent inutiles et nuisent à la lisibilité des programmes produits. Il n'est pas pertinent d'implanter des fonctions auxiliaires récursives terminales, elles ne sont plus au programme et les compilateurs récents savent les gérer. Le recours à des noms de fonctions et de variables difficiles à interpréter (comme `aux`, `aux1`, `aux2`, et les accumulateurs `acc`) rend difficile la compréhension des programmes rédigés. Concernant les calculs de complexité, une mauvaise maitrise de ces calculs est observée. Il arrive de trouver des expressions de la forme $O(t+1)$. De plus, des calculs de complexité qui ne correspondent pas au programme rédigé sont souvent observés. Le mélange de programmes rédigés en Python et en OCaml dans une même copie a été constaté à plusieurs reprises. Les candidats doivent être vigilants sur le langage utilisé. Le sujet rappelle explicitement que ce langage est OCaml. Enfin, les correcteurs recommandent une lecture plus attentive de l'énoncé afin d'éviter des réponses ou des programmes inutiles et des réponses qui ne correspondent pas aux questions posées.

5.2.3 Analyse de Forme

De manière générale, les candidats présentent des programmes avec des retours à la ligne et des indentations qui facilitent la lecture. Il reste malgré tout plusieurs copies avec une présentation de codes médiocres, sans retour à la ligne, sans indentation, et avec plusieurs ratures. Ces copies sont extrêmement difficiles à décrypter.
De plus, les correcteurs ont constaté, sur plusieurs copies, l'utilisation d'encre trop claire qui réduit la lisibilité de ces copies.
Enfin, plusieurs copies renferment encore des expressions de la forme « C'est évident », « il est clair que ... », « Trivial ». Ces expressions ne sont pas acceptées lorsque la question exige une justification et/ou bien une démonstration qui doit être convaincante et rigoureusement établies.

5.2.4 Analyse par question

Q1 - Question bien traitée par la majorité des candidats. Il faut penser à être rigoureux dans la rédaction de la récurrence.

Q2 à 4 - Questions bien traitées par la majorité des candidats.

Q5 - On attendait une caractéristique précise du texte pour l'estimation de la complexité et non un majorant trop grossier.

Q6 - Il est demandé une caractéristique du texte et pas simplement un encadrement.

Q7 - Question bien traitée par la majorité des candidats.

Q8 - Beaucoup d'acrobaties ont été observées pour le calcul de l'indice de la case à incrémenter alors qu'une opération modulo suffisait.

Q9 - Question bien traitée par la majorité des candidats. Le point clé était de comprendre comment retrouver la valence d'un texte à l'aide de sa table modulaire.

Q10 et 11 - Ayant constaté qu'un grand nombre de candidats a eu de grandes difficultés à répondre à ces questions (qui n'étaient sans doute pas assez détaillées), les correcteurs ont décidé de ne pas sanctionner l'ensemble des candidats sur ces deux questions.

Q12 - Question bien traitée par une majorité des candidats l'ayant abordée.

Q13 - Au vu de l'emplacement de la question, on attendait une valeur pour optimiser en ordre de grandeur la complexité en mémoire.

Q14 - Une quantité importante de raisonnements imprécis, voir qualitatifs et faux, sont produits. Si la majorité des candidats ont repéré qu'il s'agissait de propriétés sur les opérations entre ensembles finis, une bonne proportion de candidats ont affirmé sans justifier que I était à valeurs dans l'ensemble des lettres du mot w, sans le dépasser (ce qui était précisément la question posée).

Q15 et 16 - Des démonstrations inachevées ont été observées pour ces questions. La notion de correction évoquée dans la question 16 (qui ne comportait pas de structure répétitive de par la question 15) était une invitation à vérifier l'absence de dépassement d'indice.

Q17 - Un bon nombre de candidats confondent la syntaxe de tableau avec celle de quadruplet en OCaml. Il était pertinent d'utiliser une structure conditionnelle avec la fonction `est_present`, et de n'oublier la mise à jour d'aucune des composantes.

Q18 à 20 - Questions bien traitées par la majorité des candidats ayant abordé ces questions.

Q21 - Plusieurs confusions entre méthode « réaliste » et méthode « optimale » ont été relevées.

Q22 - L'utilisation d'une fonction auxiliaire permettant de trouver le code d'une lettre simplifie l'écriture de la fonction demandée (et son calcul de complexité). Il est dommage que cette utilisation soit absente dans une quantité importante de copies.

Q23 - Question bien traitée par la majorité des candidats ayant abordé cette question.

Q24 et 25 - Plusieurs erreurs sur la portée de l'opérateur somme généralisée (?) ont été constatées, ainsi que sur le fait que la profondeur d'un nœud dans un arbre fils diffère de celle dans l'arbre père.

Q26 - Elle est peu abordée. Il faut savoir faire le lien avec les notions de programmation dynamique et/ou de mémoïzation (programme de première année).

Le reste du sujet a été abordé par une petite minorité de candidats, à l'exception de la **Q28**(grappillage).