COMPOSITION D'INFORMATIQUE B (XELC), FILIÈRE PC

1 Remarques générales

Dans cette épreuve, 534 copies ont été corrigées. La moyenne est de 8,98 et l'écart-type de 3,59.

Un certain nombre d'erreurs semblent indiquer un manque certain de préparation des candidats. En particulier, de nombreux candidats ignorent le symbole python pour 'différent d e' (`!=`) e t ont utilisés selon les copies `=!` ou `=!=`, voire ont biaisé avec `not ..==...` C'est pourtant un symbole courant que devrait connaître tout utilisateur même occasionnel de python.

Comme chaque année, les consignes de l'épreuve ont été négligées par certains candidats. Les opérations autorisées dans l'épreuve étaient indiquées en début de document, et en particulier `copy` n'en faisait pas partie. De même, il était précisé : "Sauf mention contraire, les fonctions à écrire ne doivent pas modifier leurs entrées.".

L'utilisation de base des boucles est globalement maîtrisée, mais les relations logiques sont parfois hasardeuses. Par exemple, les boucles `while A: while B:` ne sont pas équivalentes à la boucle `while A and B:`. De même, les candidats sont trop habitués aux boucles `for` sur la variable `i`, et quand ils doivent utiliser un autre nom de variable dans une boucle, mélangent allègrement les noms de variables. Plus technique, la question de parcours de tableau `T` avec une condition d'arrêt. Ce type de parcours est en général effectué par une boucle `while condition(T[i]):` Les candidats ayant utilisé cette approche dans l'une des questions ont souvent oublié la seconde condition d'arrêt qui est de ne pas sortir du tableau: `i<len(T)`. Cependant, cette condition doit être vérifiée avant de tester `condition(T[i])`. Ce qui donne `while i<len(T) and condition(T[i]):` et non `while condition(T[i]) and i<len(T):`

Les questions de complexité sont aussi sources d'erreurs. Beaucoup de candidats les ignorent, ce qui représente une perte de points conséquente. Certains candidats considèrent qu'une fonction est en temps constant $O(1)$ si elle ne contient pas de boucle, en ignorant la complexité des sous-fonctions. Tout aussi problématique, dans les premières questions, certains candidats ont indiqué une complexité linéaire en $O(n)$, mais sans avoir de variable n dans leur programme. Il n'est pas clair si cette erreur est due à une incompréhension du sujet ou à une mauvaise habitude comme pour les boucles `for`. A l'inverse, pour des sous-fonctions `f` et `g` de complexité C_f et C_g, la complexité de l'appel `f(g(x))` n'est pas de complexité $O(C_f * C_g)$ mais $O(C_f + C_g)$.

2 Partie I

2.1 Question 1

Une première question simple toujours traitée avec très peu d'erreurs. Les erreurs étaient essentiellement de logique (sortie de la boucle à la première égalité au lieu de la première différence) et à la question de complexité.

2.2 Question 2

Cette question était très similaire à la première, avec encore moins d'erreurs dans les copies.

2.3 Question 3

C'était encore une des questions simples du sujet. Ici, la complexité attendue ne pouvait être atteinte qu'avec l'utilisation d'un dictionnaire Python. La syntaxe était donnée dans l'introduction du sujet, ce qui n'a pas empêché de nombreuses erreurs, y compris une confusion entre liste et dictionnaire Python. Des erreurs de logiques ont aussi été retrouvées, comme chercher une lettre de `liste1` dans les caractères de `liste2`, puis recommencer dans l'autre sens.

2.4 Question 4

Cette question était la première à présenter une difficulté en terme d'algorithmique. Ici, il y a eu essentiellement deux approches proposées par les candidats :

Première méthode : celle de deux boucles `while` imbriquées. La première pour trouver la position de l'intervalle suivant, et la seconde pour remplir les deux tableaux et trouver la dernière position de l'intervalle. L'erreur la plus fréquente avec cette méthode était d'oublier le risque de sortie de tableau si les derniers caractères des deux listes étaient différents. La complexité était bien linéaire, malgré l'utilisation de deux boucles imbriquées.

Deuxième méthode : l'utilisation d'une boucle `for` pour parcourir les listes. Ici, il fallait à chaque case un critère pour savoir si la position courante correspond au début d'un intervalle, ou si l'intervalle est déjà en cours en cas d'inégalité. Là encore, le cas de deux caractères différents en dernière position des deux tableaux était à prendre en compte. Cela impliquait en effet soit un critère particulier dans la boucle pour mémoriser le dernier intervalle, soit d'ajouter un cas particulier après la boucle `for`. La question de complexité était plus simple à justifier avec cette seconde méthode.

2.5 Question 5

Comme dans la question précédente, il y avait ici une petite difficulté, car il fallait deux boucles imbriquées, une sur les positions du texte, et l'autre sur les intervalles du différentiel. La complexité restait cependant en $O(len(texte))$, puisque la taille du différentiel est nécessairement plus petite que la longueur du texte. Ici en particulier, il ne fallait pas modifier `texte1`. Certains candidats ont simplifié la question en recopiant d'abord le texte avant d'appliquer le différentiel sur la copie. Attention à la copie de liste dans ce cas : l'instruction `l'=l` crée un pointeur vers la même liste, pas une nouvelle liste. Par la suite, l'instruction l'[0]=... modifie l. Certains candidats ont oublié soit le début soit la fin du texte, selon la méthode employée.

2.6 Question 6

Cette question était simple à rédiger une fois le problème compris. Elle n'a pas posé de problème majeur aux candidats.

2.7 Question 7

Cette question nécessitait d'utiliser les fonctions des questions précédentes. Elle a été globalement bien réussie sauf quelques erreurs d'inattention. Certains ont cependant proposé une complexité en $O(len(texte))$ pour la fonction `annule` alors que ce n'est pas un paramètre de la fonction. Il y a également chez de nombreux candidats une incompréhension liée à l'opération `liste.pop` qui ne supprime pas seulement le dernier élément de la liste, mais le renvoie également. Donc utiliser `liste(-1)` juste avant `liste.pop` n'a pas de sens.

3 Partie II

3.1 Question 8

Encore une question très simple qui n'a pas posé de grandes difficultés. La complexité était ici triviale, mais n'a pas été donnée par de nombreux candidats.

3.2 Question 9

Il fallait avoir une bonne compréhension du sujet pour répondre à cette question. C'était la première réelle difficulté. À partir de cette question les objets utilisés se complexifient, peu de candidats ont su montrer qu'ils parvenaient à comprendre et à se représenter correctement les structures utilisées, ce qui a amené des incompréhensions sur les objectifs et les moyens de les réaliser.

3.3 Question 10

Il s'agissait ici simplement d'appliquer la réponse précédente. La notation de cette question ne dépendait pas de la correction de la réponse précédente. Que la formule de récurrence proposée soit juste ou fausse ne changeait pas les points donnés à cette question. Peu de candidats ont remarqué que les dimensions de la matrice étaient `len(liste1)+1` et `len(liste2)+1` et non `len(liste1)` et `len(liste2)`. Là encore, la complexité était une question évidente.

3.4 Question 11

Moins d'un tiers des candidats ont traité cette question, généralement sans beaucoup de succès. Il fallait ici partir de la dernière case de la matrice `M[len(liste1)][len(liste2)]` et non du début.

3.5 Question 12

Peu de réponses correctes ici. Le majorité des réponses ne donnaient pas la complexité voulue.

3.6 Question 13

Toujours moins de réponses dans cette fin d'énoncé. La difficulté était ici élevée puisque les modifications de chaque différentiel modifient les positions de départ de l'autre différentiel. Les réponses proposées se contentaient en général de concaténer les deux listes.

4 Partie III

4.1 Question 14

La question était ici très similaire à la question 9, mais peu de candidats l'ont remarqué. Peu de réponses des candidats, et encore moins de points accordés.

4.2 Question 15

Il fallait ici expliquer le lien entre la résolution de notre problème et un calcul de plus court chemin. Les quelques candidats ayant traité la question ont pu grappiller quelques points malgré des explications alambiquées.

4.3 Question 16

Quelques réponses justes sur cette question de complexité. Aucun candidat n'y a vu d'intérêt par rapport à la partie II, sauf erreur dans l'une ou l'autre des complexités.

4.4 Question 17

Aucun point accordé ici pour très peu de tentatives.