Composition d'Informatique B 2h, Filière MP (XLSR)

Statistiques

La note moyenne des candidats français admissibles est 10,40/20, et l'écart-type est 3,30.

Question	1	2	3	4	5	6	7	8	9
Moyenne*	0.87	0.93	0.83	0.64	0.65	0.75	0.69	0.91	0.65
Pourcentage de copies ayant traité la question	100%	100%	99.5%	98.5%	98.5%	94.6%	95%	96.5%	88.1%

Question	10	11	12	13	14	15	16	17
Moyenne*	0.57	0.38	0.43	0.26	0.33	0.32	0.15	0.55
Pourcentage de copies ayant traité la question	74.8%	27.2%	52%	19.8%	25.7%	18.3%	15.3%	2.5%

* La moyenne est rapportée sur 1 point, et calculée parmi les copies qui ont traité la question.

Commentaires

En grande majorité, les candidats maîtrisent la syntaxe du langage Python.

Il ne s'agit pas simplement d'écrire du code fonctionnel, il faut aussi songer à sa lisibilité. Notamment, les noms de variables doivent aider le lecteur à deviner quel est leur rôle.

On voit trop souvent des cas de base inutiles, par exemple un cas particulier si le texte est vide. Cela alourdit le code et n'a pas d'intérêt puisque le fonctionnement normal de l'algorithme traite correctement ce cas.

Il faut veiller aux dépassements de bornes de listes. Attention, dans

```
while texte1[i] == texte2[i] and i < len(texte1): ...
```

il y a dépassement puisque la comparaison texte1[i] == texte2[i] est exécutée avant d'avoir vérifié la borne.

Toutes les questions de ce sujet se résolvaient naturellement avec des fonctions écrites en style impératif : boucles for, quelques boucles while. Les candidats qui ont tenté d'utiliser un style récursif pour certaines questions ont majoritairement échoué.

Il faut utiliser les fonctions début(tr), avant(tr) définies dans le sujet pour accéder aux champs tr['début'], tr['avant'], tr['après']. Et surtout pas des indices tr[1], tr[2]...

Question 1 :

- Des candidats relativement nombreux oublient de donner la complexité. Il est inutile (mais non pénalisé) de comparer la longueur des textes puisque le sujet fait l'hypothèse qu'ils ont la même longueur.

Question 2 :

- Question très bien réussie. Mais ici aussi certains candidats oublient de donner la complexité.

Question 3 :

- Certains candidats construisent séparément le dictionnaire des caractères contenus dans texte1 et celui des caractères contenus dans texte2. C'est inutile.

Question 4 :

- Certains candidats oublient d'enregistrer la dernière tranche lorsqu'elle termine le texte.

Question 5 :

- La fonction ne doit pas modifier le texte en place, mais le recopier. Attention, l'instruction texte2 = texte1 ne recopie pas le texte1.
- Il y a essentiellement deux méthodes : soit on recopie tout texte1 puis on modifie les tranches, soit on construit progressivement le texte modifié en ajoutant les morceaux de texte, alternativement conservés et modifiés. Avec la seconde méthode, il ne faut pas oublier de recopier la fin du texte, non modifiée, après la dernière tranche.
- Si l'on utilise un compteur pour la tranche courante, attention de vérifier que celui-ci reste bien dans les bornes de la liste diff. De même diff[0] n'est pas correct si diff est vide.
- Certains candidats donnent une complexité en O(len(texte1)*len(diff)). Il faut remarquer que, même s'il y a deux boucles imbriquées, le nombre total de passages dans la boucle la plus interne est bien O(len(texte1)).

Question 6 :

- Ici aussi il ne faut pas modifier diff en place, mais le recopier. Il est important d'utiliser les primitives fournies dans le sujet pour accéder à la structure tranche.

Question 7 :

- Certains candidats confondent le dernier élément de l'historique avec la version précédente du texte.
- Moins grave, beaucoup de candidats oublient de retourner la nouvelle valeur courante du texte.
- Les parenthèses pour l'appel à pop sont régulièrement oubliées.

Question 8 :

- Question très bien réussie.

Question 9 :

- La solution proposée par plusieurs copies "quand texte1[i] != texte2[i], M[i + 1][j + 1] = M[i + 1][j] + M[i][j + 1] - M[i][j]" est fausse. Contre exemple :
  - texte1 = ['e', ' ', 'c', 'h', 'a', 't']
  - text2 = ['h', 'i', 'e']
  - cette approche donne levenshtein(c1, c2, variant=1)[4][3] == 3
  - la bonne réponse est levenshtein(c1, c2, variant=1)[4][3] == 5

Question 10 :

- L'erreur la plus fréquente est de travailler sur une matrice de taille n × m au lieu de (n+1) × (m+1).
- Quelques candidats tentent une version récursive qui n'est pas du tout adaptée.

Question 11 :

- Il faut parcourir la matrice en partant de (n+1, m+1) et remonter. Le parcours en partant de (0, 0) ne permet pas de retrouver le chemin optimal.

Question 13 :

- Bien lire la question.

Question 14 :

- Question très mal comprise.
- Il est inutile d'utiliser M pour calculer les pondérations des arcs : le coût pour une édition est simplement 1, alors que la différence entre deux valeurs adjacentes (verticalement ou horizontalement) de M n'est pas toujours 1.

Question 15 :

- On attend une caractérisation soigneuse des clefs de dist_final : ensemble des sommets visités, c'est-à-dire tous les sommets à plus proche ou à même distance de l'entrée que la sortie.

Question 16 :

- Le log venu de la complexité des appels à ajoute() et extraire_min() est souvent oublié.