Algorithmique III (I41) - Licence d'Informatique

Contrôle terminal (Session 2 - Juin 2025)

En 2026, les notes d'algorithmique au contrôle terminal sont remplacées par quatre smileys ☹, 😐, ☺ et 😄 quand la copie est vraiment drôle, et sont encodés par les entiers naturels 1, 2, 3 et 0, *respectivement*. On note n le nombre d'étudiant · e · s, E la liste de leurs noms et T la liste des codes des notes correspondantes ($T[i]$ est la note de l'étudiant · e $E[i]$ pour tout $i \in [\![1, n]\!]$).

1» [1.0] Zack a eu pour résultat 😐, Liza ☺, Phil 😐, Eva ☹ et Liz 😐. Quelles valeurs contiennent les listes E et T pour ces 5 étudiant · e · s si E a été triée dans l'ordre lexicographique et T en conséquence afin que $T[i]$ soit la note de $E[i]$ pour tout $i \in [\![1, n]\!]$?

Réponse. La liste des étudiants une fois triée dans l'ordre lexicographique est la suivante :

$$E = [\text{Eva, Liz, Liza, Phil, Zack}].$$

Et la liste des notes correspondantes est

$$T = [1, 2, 3, 2, 2].$$

2» [1.0] Définissez une relation d'ordre total $\preccurlyeq$ sur l'ensemble $\mathscr{S} :=$ {☹, 😐, ☺, 😄} pour que la fonction d'encodage $e : \mathscr{S} \to \{0, 1, 2, 3\}$ définie en introduction soit strictement croissante si l'on munit $\{0, 1, 2, 3\}$ de l'ordre naturel $\leqslant$. Justifiez.

Réponse. On se donne une relation d'ordre $\preccurlyeq$ sur $\mathscr{S}$ (donc réflexive, antisymétrique et transitive) et on suppose qu'elle satisfait les inégalités suivantes : 😄 $\preccurlyeq$ ☹ $\preccurlyeq$ 😐 $\preccurlyeq$ ☺. La transitivité de $\preccurlyeq$ assure que les éléments de l'ensemble $\mathscr{S}$ sont deux-à-deux comparables, il s'agit donc d'une relation d'ordre total. Il nous reste à montrer que l'application $e : \mathscr{S} \to \{0, 1, 2, 3\}$ est croissante, autrement dit que

$$\forall (x, y) \in \mathscr{S} \quad x \preccurlyeq y \Rightarrow e(x) \leqslant e(y). \tag{1}$$

Le graphe de la fonction e représenté ci-dessous permet de vérifier l'implication (1) pour les 6 couples $(x, y) \in \mathscr{S}^2$ tels que $x \preccurlyeq y$.

$$\begin{array}{rccccccc} & \text{😄} & \preccurlyeq & \text{☹} & \preccurlyeq & \text{😐} & \preccurlyeq & \text{☺} \\ e: & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ & 0 & \leqslant & 1 & \leqslant & 2 & \leqslant & 3 \end{array}$$

3» [2.0] Soit $n \in \mathbb{N}$. Définissez l'ensemble des listes sur $\mathscr{S}$ de longueur n en logique des prédicats. En déduire le nombre de listes de notes de longueur n distinctes possibles. Justifiez.

4» [2.0] Écrivez un algorithme `TrierNotes(@T)` qui trie la liste des notes T des n étudiant · e · s dans l'ordre croissant. Ce tri ne devra faire *aucune* comparaison entre notes de la liste T.

5» [1.0] Quelle est la complexité de votre algorithme ? Justifiez.

6» [1.0] Peut-on adapter l'algorithme `TrierNotes` de manière à trier simultanément la liste E afin que $T[i]$ soit la note de l'étudiant · e $E[i]$ pour tout $i \in [\![1, n]\!]$, sans changer sa complexité ? Justifiez.

Après l'examen, les copies ont été corrigées et la liste T triée par l'algorithme `TrierNotes`. Les étudiant · e · s les moins bien noté · e · s décident de se plaindre parce qu'ils estiment que l'examen était trop dur. La promotion est alors partitionnée en deux groupes : le groupe des frondeurs aux positions 1 à q, et les autres aux positions $q + 1$ à n avec $q < n$. La césure q a été réalisée de manière à ce qu'il y ait le plus de frondeurs possible, mais sans que la somme de leurs notes n'excède celle des notes des autres étudiant · e · s.

7» [1.0] Quelle est la valeur de q pour la liste des notes de la question **??** ? Justifiez.

8» [1.0] Donnez une définition formelle de l'entier q à l'aide d'une expression en logique des prédicats.

9» [3.0] Écrivez un algorithme `Frondeurs(T)` qui calcule et renvoie la valeur q, en supposant que la liste des notes T a été triée au préalable en appelant `TrierNotes(@T)`.

10» [1.0] Faites une preuve d'arrêt de votre algorithme.

11» [1.0] Justifiez que votre algorithme renvoie bien la valeur q.

12» [1.0]Calculez la complexité de votre algorithme dans le meilleur des cas et dans le pire des cas. Justifiez.