UFR IM²AG
UNIVERSITÉ Grenoble Alpes

L3-INFO Algorithmique

Quick 1 – 4 octobre 2019 – durée 1 h

Les documents et les téléphones (incluant smartphone, tablettes,... tout ce qui contient une interface réseau) interdits. Les calculettes sont autorisées.
Seuls les dictionnaires pour les personnes de langue étrangère sont autorisés.
Les questions faisant apparaître le symbole ♣ peuvent présenter une ou plusieurs bonnes réponses.
Les autres ont une unique bonne réponse.
Le barème est indicatif.

Complexité *(5 points)*

Question 1 Soit l'algorithme suivant :

Pour i = 1 à n faire
    Pour j = 1 à 2 × n faire
        Schtroumpfer()

Combien effectue-t-il de Schtroumpfages en ordre de grandeur ?

[A] $3 \times n$ [B] $2n!$ [C] n^2 [D] n^3 [E] n^n

Question 2 Soit l'algorithme suivant :

Pour i = n à 1 faire
    Pour j = 1 à i faire
        Schtroumpfer()

Combien effectue-t-il de Schtroumpfages exactement ?

[A] $n \times i$ [B] n^2 [C] $\frac{n(n-1)}{2}$ [D] $\frac{n(n+1)}{2}$ [E] $\frac{n}{2}$

On considère un tableau T indexé de 1 à n dans lequel une certaine valeur a cherchée peut apparaître zéro, une ou plusieurs fois. Plus précisément, on sait quelle est la probabilité que a soit présent dans chacune des cellules du tableau : à chaque indice i, la probabilité que $T[i] = a$ vaut un certain réel $p \in [0, 1]$.

On exécute alors l'algorithme suivant :

i = 1
Tant que $i \leqslant n$ et $T[i] \neq a$ faire
    i = i + 1

Pour les trois questions suivantes, on s'intéresse au coût en **nombre de comparaisons entre un élément de T et a.**

Question 3 Quel est le coût au pire de cet algorithme ?

[A] $n - 1$ [B] $n \times p$ [C] n [D] p [E] a

Question 4 Vers quelle valeur tend le coût en moyenne lorsque n devient grand ?

[A] $n \times p$ [B] n [C] p [D] $\frac{n}{2}$ [E] $\frac{1}{p}$

Question 5 Quel est le coût au mieux de cet algorithme ?

[A] 1 [B] n [C] p [D] 0 [E] a

Preuve d'algorithme *(7 points)*

On dispose d'une urne contenant un certain nombre de billes noires et de billes blanches, ainsi que d'une réserve de billes supplémentaires à l'extérieur de l'urne. On applique alors la procédure suivante :

(1) Tant qu'il reste au moins 2 billes dans l'urne
(2) Piocher 2 billes au hasard
(3) Si les deux billes sont de couleurs différentes
(4) Remettre une bille blanche dans l'urne
(5) Sinon
(6) Remettre une bille noire dans l'urne

On cherche à démontrer :

- D'une part, la terminaison : quelle que soit la composition initiale de l'urne, la procédure s'arrête avec une seule bille dans l'urne.
- D'autre part, la conjecture suivante : si le nombre initial de billes blanches est impair, la dernière bille est blanche.

Question 6 ♣ Parmi les propriétés suivantes, lesquelles sont des invariants de cet algorithme ?

[A] Il y a plus de billes blanches que de billes noires dans l'urne.
[B] Il y a au moins une bille blanche dans l'urne.
[C] Le nombre de billes noires a la même parité qu'en début d'algorithme.
[D] Le nombre de billes noires change de parité à chaque itération.
[E] Le nombre de billes blanches a la même parité qu'en début d'algorithme.

Question 7 ♣ Parmi les quantités suivantes, lesquelles sont des variants de cet algorithme ?

[A] la différence entre le nombre de billes noires et de billes blanches dans l'urne
[B] le nombre de billes blanches dans l'urne
[C] le nombre total de billes dans l'urne
[D] le nombre de billes noires dans l'urne
[E] le nombre de billes vertes dans l'urne

Question 8 ♣ Parmi les propriétés suivantes, lesquelles, si elles étaient vraies en sortie de boucle, permettraient de démontrer la conjecture proposée ci-dessus ?

[A] Il y a au moins une bille blanche dans l'urne.
[B] Le nombre de billes blanches a la même parité qu'en début d'algorithme.
[C] Le nombre de billes blanches et le nombre de billes noires ont la même parité.
[D] Il y a plus de billes blanches que de billes noires dans l'urne.
[E] Le nombre de billes noires a la même parité qu'en début d'algorithme.

Écriture et preuve d'algorithmes *(8 points)*

Soient deux tableaux A indexé de 1 à n, et B indexé de 1 à p contenant des éléments de même type. On s'intéresse à vérifier si A est **inclus** dans B, autrement dit si tout élément présent dans A est présent dans B.

On se place bien entendu dans le cas où $n \leqslant p$.

Dans chacun des tableaux, on suppose que les éléments sont tous distincts deux à deux (il n'y a pas de doublon). On suppose également qu'on dispose d'un ordre total sur les éléments, et que les tableaux A et B sont triés (en ordre croissant).

1. Écrire un algorithme de coût au pire $n + p$ pour résoudre le problème posé.

2. Proposer un invariant pour cet algorithme qui permettrait de démontrer sa correction. Il est recommandé d'illustrer cette réponse par un ou plusieurs schémas.