MPCI

Programmation et Algorithmes

Devoir Surveillé 01

Vendredi 25 février 2022

On rappelle qu'aucun document, ni équipement électrique ou électronique n'est autorisé.

Tout algorithme devra être prouvé et on devra en donner sa complexité.
Pas de blabla inutile, on ne tente pas d'enfumer le correcteur.

1 Questions de cours

1.1 Code

Des trois parties suivante d'un programme, quelle est la plus importante ? Et pourquoi ?

- le programme principal
- le code des fonctions
- les tests

1.2 Complexités

1.2.1 Définitions

Rappelez les définitions de :

1. la **complexité maximale** d'un algorithme
2. la **complexité minimale** d'un algorithme
3. la **complexité en moyenne** d'un algorithme
4. la **complexité d'un problème** algorithmique

1.2.2 Complexité du tri

Donnez et justifiez[1] la complexité du **problème du tri** d'un tableau d'entiers.

[1] vous pourrez utiliser le fait que $\mathcal{O}(\ln(n!)) = \mathcal{O}(n\ln(n))$ et donner les complexités des algorithmes de tri du cours sans les démontrer.

2 Tri de crêpes

Définition tirée de Wikipédia :

> Un cuisinier fait des crêpes et les pose en pile à côté de la poêle lorsqu'elles sont cuites. Toutes les crêpes sont de taille différente. On dispose donc d'une pile de crêpes, chacune de taille différente et il s'agit d'ordonner les crêpes dans la pile par diamètre croissant, avec la crêpe de plus petit diamètre en haut de la pile.
>
> Un seul type d'opération est autorisé pour manipuler la pile : le **retournement**. Il consiste à insérer une spatule à un endroit de la pile et retourner d'un coup sur la pile toutes les crêpes qui se trouvent au-dessus de la spatule.

2.1 Problème

Formalisez le problème de façon informatique. En particulier, il vous faudra :

- montrer que l'entrée du problème peut être un tableau de taille n contenant tous les entiers de 0 à $n-1$
- montrer que le problème peut être vue comme un tri
- définir ce que signifie l'opération de **retournement** avec votre modélisation

2.2 Algorithme de retournement

Proposez un algorithme permettant d'effectuer l'opération de retournement.

2.3 Résolution du problème

2.3.1

Montrez que l'on peut toujours :

1. placer un élément quelconque de la pile en haut de celle-ci en 1 opération de retournement
2. placer le premier élément de la pile à une position quelconque de celle-ci en 1 opération de retournement

2.3.2

En supposant que l'on doive ordonner n crêpes. Déduire de la question précédente que le problème du tri de crêpes admet toujours une solution et donnez un algorithme qui le résout avec une complexité de $\mathcal{O}(n)$ retournements.

2.3.3

Le résultat de la question précédente est-il compatible avec la complexité du problème du tri d'un tableau d'entiers ?

2.4 Bornes

2.4.1 Borne maximum

Utilisez l'algorithme de la question précédente pour donner une borne maximale (pas forcément atteinte) du nombre de retournements nécessaires pour trier une pile de crêpes.

2.4.2 Borne minimum

On note D_i la taille (diamètre) de la crêpe d'indice i dans le tableau T. Une **adjacence** est formée de crêpes qui se suivent dans la pile (d'indices i et $i+1$) et telle qu'il n'existe pas de crêpe de taille intermédiaire entre les deux (pour tout $j \notin \{i, i+1\}$, on a soit $D_j < \min(D_i, D_{i+1})$ soit $D_j > \max(D_i, D_{i+1})$).

Montrez :

1. qu'il existe au plus $n-1$ adjacences pour une pile de crêpes de taille n
2. qu'une pile de n crêpes est triée si et seulement si la crêpe la plus grande est en bas de la pile et qu'il y a $n-1$ adjacences
3. qu'un retournement augmente le nombre d'adjacences d'une pile de crêpes d'au plus 1
4. que pour $n \geq 4$ il existe des piles de crêpes ayant 0 adjacence et telle que la crêpe la plus grande ne soit pas en bas de la pile (on pourra séparer les crêpes par tailles paires et impaires)

En déduire que la borne minimum du problème du tri de crêpes est d'au moins n retournements lorsque $n \geq 4$.

3 Echange d'indices

Dans toute cette partie, T sera un tableau de taille n contenant tous les entiers allant de 0 à $n-1$.

On considère un algorithme qui échange les éléments d'indice 0 et d'indice $T[0]$ de T tant que $T[0] \neq 0$.

Par exemple soit $T = [1, 2, 0]$:

1. à l'indice 0 il y a $T[0] = 1$ et à l'indice $T[0] = 1$ il y a $T[T[0]] = 2$.
2. après échange, $T = [2, 1, 0]$. Comme $T[0] \neq 0$, il faut continuer.
3. à l'indice 0 il y a maintenant $T[0] = 2$ et à l'indice $T[0] = 2$ il y a $T[T[0]] = 0$.
4. après échange, $T = [0, 1, 2]$. Comme $T[0] = 0$, on s'arrête.

Il a fallut 2 échanges pour que $T[0] = 0$

3.1

1. Ecrivez un pseudo-code de cet algorithme
2. testez le sur le tableau $[2, 1, 6, 7, 0, 4, 5, 3]$

3.2

On associe à T la liste $L(T)$ de taille n définie telle que (pour $0 \leq i < n$) :

$$L(T)[i] = \begin{cases} 1 & \text{si T[i] = i} \\ 0 & \text{sinon.} \end{cases}$$

1. Donnez la valeur de $L(T)$ pour chaque itération de l'exécution de l'algorithme avec le tableau $[2, 1, 6, 7, 0, 4, 5, 3]$ comme entrée
2. Montrez qu'à chaque itération de l'algorithme, le nombre de 1 de $L(T)$ augmente strictement.

3.3

Déduire de la question précédente que l'algorithme s'arrête forcément au bout d'au plus $\mathcal{O}(n)$ itérations et donc que sa complexité est de $\mathcal{O}(n)$.

3.4

En déduire un algorithme (que vous expliciterez) de complexité $\mathcal{O}(n)$ qui trie le tableau T.

3.5

Le résultat de la question précédente est-il compatible avec la complexité du problème du tri d'un tableau d'entiers ?

4 Inversion de préfixes

Dans toute cette partie, T sera un tableau de taille n contenant tous les entiers allant de 0 à $n-1$.

On considère un algorithme[2] qui **retourne** tous les éléments de T entre les indices 0 et $T[0]$ inclus (cette opération de **retournement** est identique à celui du tri de crêpes de la question 2), tant que $T[0] \neq 0$.

4.1

1. Ecrivez un pseudo-code de cet algorithme
2. testez le sur le tableau $[2, 1, 6, 7, 0, 4, 5, 3]$

4.2

On note $N(T)$ le nombre défini tel que :

$$N(T) = \sum_{i=0}^{n-1} L(T)[i] \cdot 2^i$$

Montrez qu'à chaque itération de l'algorithme, $N(T)$ augmente strictement.

4.3

Déduisez de la question précédente que l'algorithme s'arrête forcément et donnez une borne maximale de sa complexité en nombre de retournements.

4.4

En utilisant le fait qu'il faut en moyenne k essais pour obtenir un évènement de probabilité $\frac{1}{k}$ (c'est une loi géométrique), démontrez que la complexité en moyenne de l'algorithme est de $\mathcal{O}(n)$ retournements.

4.5

Que pouvez vous conclure de ces complexités ?

[2]algorithme tiré d'un cours de M. Habib