L3-INFO Algorithmique

Quick 1 – 7 octobre 2022 – 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.
Le barème est indicatif.

Vous êtes vivement encouragés à illustrer vos recherches et vos réponses par des schémas.

Question 1 : Écriture et preuve d'algorithme (13 points)

On considère un tableau T de n entiers (ces entiers peuvent être négatifs). Le tableau T est indexé de 0 à $n-1$.

L'objectif est de construire un algorithme qui renvoie un indice i tel que $T[i] = i$, ou bien -1 s'il n'en existe pas.

1. (a) *(1 pts)* Écrire un algorithme (simple) qui résoud ce problème.
   (b) *(2 pts)* Calculer son coût dans le meilleur et dans le pire des cas en justifiant bien toute votre réponse.

2. On se place ensuite dans un cas particulier : le tableau T est **trié en ordre croissant** et ne contient que des **entiers distincts**.
   (a) *(2 pt)* Démontrer que si $T[j] > j$, alors pour tout indice $k \geqslant j$ on a $T[k] > k$.
   (b) *(3 pts)* En déduire un algorithme en $\mathcal{O}(\log_2 n)$ pour ce cas particulier où T est trié et sans doublon.
   (c) *(1 pts)* Démontrer que la complexité de l'algorithme que vous avez écrit est effectivement en $\mathcal{O}(\log_2 n)$.

3. On s'intéresse enfin à une variante du problème initial : pour un tableau T de n entiers (quelconques, non ordonnés), déterminer **deux** indices i et j tels que $T[i] = j$ et $T[j] = i$, ou bien poser $i = j = -1$ s'il n'en existe pas.
   (a) *(2 pts)* Écrire un algorithme, le plus efficace possible, résolvant ce dernier problème.
   (b) *(1 pt)* Déterminer sa complexité au pire.
   (c) *(1 pt)* Dans le cas où T est trié en ordre croissant, pouvez-vous améliorer la complexité de votre algorithme pour cette variante du problème ?

Question 2 : Structure de données (7 points)

On souhaite implémenter une structure de données appelée $PileDouble$ vérifiant la spécification suivante :

Opérations

Vide	:	$void \rightarrow PileDouble$
Empiler	:	$Element \times PileDouble \rightarrow PileDouble$
Enfiler	:	$Element \times PileDouble \rightarrow PileDouble$
EstVide	:	$PileDouble \rightarrow bool$
Dépiler	:	$PileDouble \rightarrow PileDouble$
Sommet	:	$PileDouble \rightarrow Element$

Préconditions

Dépiler(p)	:	p est non vide
Sommet(p)	:	p est non vide

Axiomes

EstVide(Vide())	=	vrai	
EstVide(Empiler(e, p))	=	faux	
EstVide(Enfiler(e, p))	=	faux	
Sommet(Enfiler(e, Vide()))	=	e	
Sommet(Enfiler(e, p))	=	Sommet(p)	si p est non vide
Sommet(Empiler(e, p))	=	e	
Dépiler(Enfiler(e, Vide()))	=	Vide()	
Dépiler(Enfiler(e, p))	=	Enfiler(e, Dépiler(p))	si p est non vide
Dépiler(Empiler(e, p))	=	p	

Intuitivement, cette structure se comporte comme une pile, et permet en plus d'insérer un nouvel élément en bas de la pile, à l'aide de l'opération Enfiler.

1. *(1 pt)* Comment se comporte cette structure si on interdit l'opération Empiler ?
2. *(3 pts)* Proposer une implantation du type $PileDouble$ basée sur un tableau d'éléments (et des compteurs auxiliaires si besoin). Pour cette question, il est possible de proposer une réalisation par effets de bord.

   Plus vous parviendrez à réaliser d'opérations en temps constant, plus votre solution sera valorisée.

   Un schéma préliminaire vous sera d'un grand secours et sera apprécié par le correcteur.
3. *(3 pts)* Démontrer que votre implémentation respecte les deux axiomes suivants :

Sommet(Enfiler(e, p))	=	Sommet(p)	si p est non vide
Dépiler(Empiler(e, p))	=	p	