Numéro d'anonymat : **Groupe :**

UE LU3IN003 - Algorithmique.
Licence d'informatique.

Partiel du 10 novembre 2023. Durée : 1h30

Documents non autorisés. Seule une feuille A4 portant sur les cours est autorisée.
Tous objets connectés éteints et rangés dans vos sacs.

Exercice 1 (3 points)

Question 1 (1/3) — Si je prouve que la complexité dans le pire des cas d'un algorithme est en $O(n^2)$, est-il possible qu'il soit en $O(n)$ sur *toutes* les données ? Justifier la réponse.

Question 2 (1/3) — Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'il soit en $O(n)$ sur *certaines* données ? Justifier la réponse.

Question 3 (1/3) — Si je prouve que la complexité dans le pire des cas d'un algorithme est en $\Theta(n^2)$, est-il possible qu'elle soit en $O(n)$ sur *toutes* les données ? Justifier la réponse.

Exercice 2 (5 points)

Une grenouille se situe sur la rive d'un étang, devant une rangée de n nénuphars, indicés de 1 à n (du nénuphar le plus proche au plus éloigné de la rive). La rive correspond à l'indice 0 par convention. La grenouille se déplace systématiquement vers l'avant, en sautant de nénuphar en nénuphar pour aller de la rive au dernier nénuphar de la rangée, autrement dit le nénuphar n. Chaque saut est de longueur 1 (la grenouille atterrit sur le nénuphar $i+1$ depuis le nénuphar i) ou 2 (la grenouille atterrit sur le nénuphar $i+2$ depuis le nénuphar i). Plus précisément, la grenouille est susceptible de réaliser cinq types de sauts depuis un nénuphar i :

A : saut simple vers le nénuphar $i+1$,
B : saut périlleux vers le nénuphar $i+1$,
C : saut simple vers le nénuphar $i+2$,
D : saut périlleux vers le nénuphar $i+2$,
E : double saut périlleux vers le nénuphar $i+2$.

Un saut simple est un saut sans figure, un (double) saut périlleux un saut où la grenouille réalise une (double) roulade en l'air. Une séquence de sauts est donc caractérisée par une suite de caractères sur l'alphabet $\{A, B, C, D, E\}$. Par exemple, pour $n = 4$, si la grenouille réalise un double saut périlleux de la rive au nénuphar 2, puis un saut simple de 2 à 3, et enfin un saut périlleux de 3 à 4, cela correspond à la séquence $[E, A, B]$.

L'objet de cet exercice est de concevoir et d'analyser une fonction récursive qui énumère toutes les séquences de sauts possibles pour aller de la rive au nénuphar n. Cette fonction retourne une liste de listes, chaque liste correspondant à une séquence de sauts possibles.

Question 1 (1/5) — Compléter le pseudo-code ci-dessous de manière à ce qu'il permette d'énumérer toutes les séquence de sauts possibles pour aller de la rive au nénuphar n.

Grenouille(n)

Entrée : $n \in \mathbb{N}^*$
Sortie : liste des séquences de sauts possibles pour aller de la rive au nénuphar n
si $n=1$ **alors**
    retourner [ ]
si $n=2$ **alors**
    retourner [ ]
sinon
    $P \leftarrow$ [ ]
    $Q \leftarrow$ [ ]
    retourner $[[A]+p : p \in P]+[[B]+p : p \in P]+[[C]+q : q \in Q]+[[D]+q : q \in Q]+[[E]+q : q \in Q]$

Question 2 (1.5/5) — Quel est l'ordre de grandeur en $O(.)$ du nombre de nœuds dans l'arbre des appels récursifs de Grenouille(n) ? Justifier la réponse.

Question 3 (1.5/5) — Quel est l'ordre de grandeur en $O(.)$ de la longueur des listes P et Q ? Justifier la réponse.

Question 4 (1/5) — En déduire la complexité de Grenouille(n) en fonction de n.

Exercice 3 (6 points)

Dans cet exercice, on conçoit et on analyse un algorithme pour convertir un nombre entier naturel x représenté en base décimale (base 10) en sa représentation en base binaire (base 2). Par exemple, le nombre entier $x = 42$ (en base 10) est converti en 101010_2, où l'indice 2 sert à signifier que la représentation est en base binaire. Dans tout l'exercice, on suppose que l'on dispose d'un tableau `bin[0...9]`, où `bin[`c`]` contient la représentation binaire du digit (chiffre) c.

c	0	1	2	3	4	5	6	7	8	9
`bin[`c`]`	0_2	1_2	10_2	11_2	100_2	101_2	110_2	111_2	1000_2	1001_2

On suppose également disposer d'une fonction `mult2`(x, y) réalisant efficacement le produit de deux nombres entiers binaires x et y (algorithme de Karatsuba). En s'appuyant sur `mult2`, on va concevoir une méthode diviser pour régner pour la conversion en base 2. Dans la suite, on fait l'hypothèse simplificatrice que n est une puissance de 2 (l'approche peut toutefois se généraliser à n'importe quel n sans détériorer la complexité).

Question 1 (1/6) — Compléter la fonction `pui2bin` ci-dessous, qui permet de convertir un nombre entier décimal 10^n (un 1 suivi de n zéros) en binaire, où n est une puissance de 2.

```
fonction pui2bin(n)
    si n = 1 alors retourner 1010₂
    sinon :
        z = [                    ]
        retourner mult2(z, z)
```

Question 2 (1.5/6) — Soit $T(n)$ la complexité de `pui2bin`(n). Donner l'équation de récurrence satisfaite par $T(n)$. En déduire la complexité de `pui2bin`(n).

Indication : La représentation en base 2 d'un entier à n digits (chiffres 0 à 9) comporte $\Theta(n)$ bits, et `mult2`(x, y) est de complexité $\Theta(n^d)$ pour multiplier deux entiers binaires x et y à n bits, où $d = \log_2 3 (\simeq 1.58)$.

Question 3 (1.5/6) — L'algorithme `dec2bin2`(x) ci-dessous vise à convertir n'importe quel entier décimal x à n digits (où n est une puissance de 2) en binaire. Compléter l'algorithme.

```
fonction dec2bin2(x)
   si n = 1 alors retourner bin[x]
   sinon :
       partitionner x en x_G = x_1 ... x_{n/2} et x_D = x_{n/2+1} ... x_n

       retourner
```

Question 4 (2/6) — Soit $T(n)$ la complexité de `dec2bin2`(x) pour un entier décimal x à n bits. Donner l'équation de récurrence satisfaite par $T(n)$. En déduire la complexité de `dec2bin2`(x).

Exercice 4 (7 points)

Un graphe orienté $G(S, A)$ est dit **sans détour** s'il comporte une racine et si pour tout couple (u, v) de sommets dans S, il existe au plus un chemin élémentaire de u à v. Dans cet exercice, on cherche à concevoir un algorithme pour vérifier si un graphe est sans détour.

Question 1 (2/7) — On cherche tout d'abord à déterminer l'existence ou non d'une racine dans G.
a) Proposer une méthode faisant appel à un ou plusieurs parcours en profondeur pour déterminer si un graphe G comporte une racine. On ne demande pas de justification.
b) Quelle est la complexité de cette méthode ? Justifier brièvement la réponse.
Précision : Réponse d'autant mieux notée que la complexité de la méthode proposée sera faible.

Tourner la page SVP

On suppose désormais que G comporte une racine et on note r celle-ci. On considère un parcours en profondeur L de G depuis r. On rappelle que les arcs du graphe n'appartenant pas à la forêt $F(L)$ associée au parcours (constituée ici d'une unique arborescence) peuvent être partitionnés en : arcs avant, arrière ou transverse [1].

Question 2 (1/7) — Le graphe G est-il sans détour si l'on détecte au terme du parcours L :
a) un arc (u, v) avant ? Justifier la réponse.
b) un arc (u, v) transverse ? Justifier la réponse.

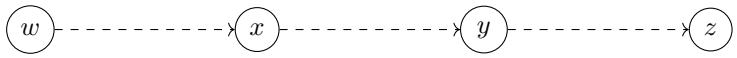
1. Un arc est ici dit *transverse* s'il n'est ni dans $F(L)$, ni avant, ni arrière.

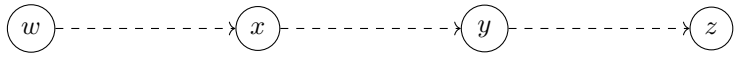
Question 3 (1.5/7) — On suppose dans cette question que G ne comporte ni arc avant ni arc transverse au terme du parcours L.

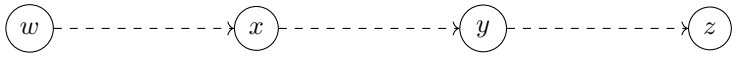
Remarquons tout d'abord que si l'on détecte un *unique* arc arrière au terme du parcours, alors il est clair que G est sans détour (G est une arborescence avec un arc supplémentaire qui crée un circuit unique).

Supposons maintenant que l'on détecte (au moins) deux arcs arrière le long d'un même chemin de l'arborescence $F(L)$, arcs arrière que l'on suppose *sans extrémité commune* dans cette question. Indiquer les trois agencements possibles des deux arcs arrière en les dessinant sur les graphes ci-dessous, où les pointillés entre deux sommets figurent une portion de chemin dans $F(L)$ (pas nécessairement un unique arc). Dans chaque cas, on indiquera si cela permet de conclure que G est *avec* détour, et si oui on indiquera le couple u, v de sommets qui en témoigne et les deux chemins élémentaires distincts de u à v.

Précision : Pour la lisibilité du dessin, on tracera des arcs arrière incurvés.

a) 

b) 

c) 

Question 4 (1/7) — En s'appuyant sur les numéros de prévisite et de postvisite des sommets dans le parcours en profondeur, il est possible de caractériser les paires a, b d'arcs arrière qui conduisent à conclure que le graphe est *avec* détour. Notons $a = (a^-, a^+)$ et $b = (b^-, b^+)$ une paire d'arcs arrière au terme du parcours. En vous aidant de la réponse à la question précédente, compléter la formule ci-dessous qui donne cette caractérisation (pas de justification demandée) :

$$\text{pre}\left(\boxed{}\right) < \text{pre}\left(\boxed{}\right) < \text{post}\left(\boxed{}\right) < \text{post}\left(\boxed{}\right)$$

$$\text{et pre}\left(\boxed{}\right) < \text{pre}\left(\boxed{}\right) < \text{post}\left(\boxed{}\right) < \text{post}\left(\boxed{}\right)$$

Question 5 (1.5/7) — Conclure en décrivant la méthode complète pour déterminer si un graphe orienté G est sans détour, et en donnant sa complexité (justifier la complexité).

Précision importante : On admettra dans la réponse que détecter l'existence de deux arcs arrières vérifiant la condition de la question 4 peut se faire sans "surcoût" de complexité.