Faculté des Sciences
Aix*Marseille Université

Année universitaire 2018/2019

Site : ☒ Luminy ☒ St-Charles ☐ St-Jérôme ☐ Cht-Gombert ☒ Aix-Montperrin ☐ Aubagne-SATIS
Sujet de : ☒ 1er semestre ☐ 2ème semestre ☐ Session 2 Durée de l'épreuve : 2h
Examen de : L1 Nom du diplôme : Portail René Descartes
Code du module : SPOU01 Libellé du module : Introduction à l'informatique
Calculatrices autorisées : NON Documents autorisés : NON

Exercice 1. (Questions de cours)

1. « Un *algorithme* est la description d'une séquence finie d'instructions permettant de résoudre un problème. » Cette phrase définit-elle un algorithme ? Justifiez brièvement.

   Solution : Non, cette phrase ne définit pas le concept d'un algorithme. Les propriétés énoncées (séquence finie d'instructions) sont nécessaires mais pas suffisantes. En particulier, il faut que la description soit *non ambigüe*. On pourrait aussi ajouter que les instructions doivent être fondées sur des opérations élémentaires mais c'est moins important.

2. Quelle la complexité de l'algorithme de recherche dichotomique dans un tableau trié de taille n ? Donnez sur votre copie la réponse, sans justification, qui est l'une des suivantes :

   $$O(\log_2(n)) \qquad O(n) \qquad O(n\log_2 n) \qquad O(n^2) \qquad O(2^n)$$

   Solution : $O(\log_2(n))$

3. Décrivez à l'aide d'une phrase ce que calcule l'algorithme suivant :

```
fonction mystère(n: entier):
    resultat := 0
    Pour i de 0 à n−1 faire
        resultat := resultat + 2×i+1
    FinPour
    retourner(resultat)
```

   Solution : L'algorithme calcule la somme des n premiers nombres entiers impairs.

4. Trouvez un algorithme plus simple qui renvoie le même résultat que l'algorithme `mystère`.

   Solution : Remarquons que

   $$\sum_{i=0}^{n-1}(2i+1) = 2\sum_{i=0}^{n-1} i + \sum_{i=0}^{n-1} 1 = 2\frac{n(n-1)}{2} + n = n^2$$

   Il suffit donc d'écrire un algorithme qui renvoie n^2. Sa complexité est en $O(1)$, si l'on considère que la multiplication est une opération élémentaire.

5. Quelles sont les conditions nécessaires et suffisantes pour qu'un graphe non orienté admette un cycle eulérien, c'est-à-dire un cycle qui emprunte chaque arête du graphe une et une seule fois ?

   Solution : L'objet de la question est de donner l'énoncé du théorème d'Euler, à savoir : un graphe non orienté admet un cycle eulérien si et seulement s'il est connexe et que tous ses sommets sont de degré pair.

Exercice 2. (Codage)

1. Donnez le codage en binaire de l'entier naturel 147, en précisant brièvement votre méthode.

   Solution : On commence par retrouver les puissances de 2 successives :

n	0	1	2	3	4	5	6	7	8
2^n	1	2	4	8	16	32	64	128	256

   Dans 147, on peut donc placer $128 = 2^7$ comme plus grande puissance de 2. Il reste $147-128 = 19$: on peut donc ajouter la puissance $16 = 2^4$. Il reste alors $19-16 = 3$ qui s'écrit $2+1$. Ainsi, l'écriture en binaire de l'entier naturel 147 est 10010011.

2. Donnez l'entier naturel dont le code binaire est 1001100, en précisant brièvement votre méthode.

Solution : On utilise la table de la question précédente pour sommer les puissances de 2 correspondantes :

$$2^2 + 2^3 + 2^6 = 4 + 8 + 64 = 76$$

3. Donnez le codage en binaire de l'entier relatif -54, en précisant brièvement votre méthode.

Solution : Pour coder un entier relatif, on utilise un bit de signe. On commence donc par trouver le codage en binaire de l'entier naturel 54, comme avant. On remarque que $54 = 32 + 16 + 4 + 2 = 2^5 + 2^4 + 2^2 + 2^1$, donc son codage en binaire est 110110. En ajoutant le bit de signe qui est 1, puisque -54 est négatif, on trouve le codage 1110110.

4. On considère une bouée de mesure de houle qui permet de mesurer les variations du niveau de la mer. Les relevés de la bouée sont écrits dans un fichier texte dont chaque ligne contient exactement 8 caractères : une ligne est par exemple de la forme `+0.4256` ou `-0.0825` pour signifier la variation du niveau de la mer en mètres, auquel s'ajoute un caractère de fin de ligne. Chaque caractère est codé sur 8 bits. Déterminez le nombre de bits, puis d'octets, nécessaires pour stocker 20 minutes d'enregistrement de la bouée à la fréquence d'échantillonnage de 2 Hz (c'est-à-dire qu'il y a deux mesures par seconde). Vous choisirez les préfixes (kilo, Méga, Giga...) les plus adaptés pour donner vos réponses.

Solution : Chaque ligne nécessite $8 \times 8 = 64$ bits. 20 minutes correspondent à 1200 secondes et il y a deux mesures par seconde : au total, cela correspond à 2400 mesures, soit une taille totale de $64 \times 2400 = 153\ 600$ bits $= 153,6$ kilobits. Puisqu'un octet équivaut à 8 bits, cela représente $153,6/8 = 19,2$ kilooctets.

5. La bouée enregistre 20 minutes toutes les 30 minutes. En supposant qu'elle dispose d'une carte mémoire ne disposant que de 40 Mégaoctet d'espace mémoire, combien de jours de relevés environ peut-elle stocker ?

Solution : Chaque heure, la bouée nécessite donc de stocker environ 40 kilooctets (2 relevés de 20 minutes). Dans 40 Mégaoctets, soit 40 000 kilooctets, on peut donc stocker $40\ 000/40 = 1\ 000$ heures, soit $1\ 000/24 \approx 41$ jours de relevés.

Exercice 3. (Algorithmes sur les tableaux)

1. Exécutez l'algorithme ci-dessous sur le tableau `[1,4,5,8]` puis sur le tableau `[1,5,4,8]`, en détaillant les opérations effectuées à chaque tour de boucle.

```
fonction f(t: tableau d'entiers):
    n := longueur(t)
    Pour i de 0 à n−2 faire
        Si t[i] > t[i+1] alors
            retourner(faux)
        FinSi
    FinPour
    retourner(vrai)
```

Solution : L'algorithme renvoie `vrai` avec le tableau `[1,4,5,8]` en entrée, car aucun des tests `t[i] > t[i+1]` n'est vérifié pour `i` allant de 0 à `n−2`. Par contre, l'algorithme renvoie `faux` avec le tableau `[1,5,4,8]` en entrée, puisque le test `t[1] > t[2]` est vérifié.

2. Décrivez en une phrase ce que renvoie l'algorithme `f` pour un tableau d'entiers quelconque donné en argument.

Solution : La fonction `f` prend un tableau d'entiers en entrée et renvoie `vrai` si le tableau est trié, et `faux` sinon.

3. Quelle est la complexité de l'algorithme `f` dans le pire des cas, en fonction de la longueur n du tableau en entrée ?

Solution : Chaque itération de la boucle `Pour` exécute un nombre constant d'opérations élémentaires, disons $a \geq 1$ opérations. Dans le pire des cas, la boucle `Pour` exécute `n−1` itérations : la boucle exécute donc au plus $a \times$ (`n−1`) opérations. On exécute encore 2 opérations élémentaires (affectation de la longueur du tableau dans `n` et renvoi de la valeur de retour) en dehors de la boucle. Au total, la complexité est donc $a \times$ (`n−1`) $+ 2$, qui est en $O(n)$ puisque $a \times (n-1) + 2 \leq an + 2 - a \leq an + 1 \leq (a+1)n$ dès que $n \geq 1$.

4. Écrivez le pseudo-code d'une fonction `minimum` qui prend en entrée un tableau `t` d'entiers et deux indices `i` et `j` de cases du tableau `t` qu'on supposera vérifier $\texttt{i} \leq \texttt{j}$, et renvoie l'indice d'une case de `t` contenant l'élément minimal parmi les éléments $\{\texttt{t[i]}, \texttt{t[i+1]}, \ldots, \texttt{t[j]}\}$. Ainsi, `minimum([3,0,4,2,8,1], 2, 4)` doit renvoyer l'indice 3 qui héberge le minimum du sous-tableau `[4,2,8]`.

Solution :
```
fonction minimum(t: tableau d'entiers, i: entier, j: entier):
    indice_minimum := i
    Pour k de i+1 à j faire
        Si t[k] < t[indice_minimum] alors
            indice_minimum := k
        FinSi
    FinPour
    retourner(indice_minimum)
```

5. Cette fonction nous permet d'imaginer une fonction de tri différente de celles étudiées en cours et en TD. Il s'agit d'un tri par *sélection*. Comme le tri par insertion, il consiste à trier le tableau de gauche à droite. À un moment de l'algorithme le tableau `t` sera trié jusqu'à l'indice `i`. On cherche alors l'élément minimal de la fin du tableau (les éléments à partir de l'indice `i+1`) et on vient l'échanger avec l'élément d'indice `i+1`. Par exemple, si on est arrivé au tableau `[1,4,5,9,8,6,7]` avec $\texttt{i} = 2$, on cherche l'indice de l'élément minimal dans la fin du tableau (c'est l'indice 5) et on échange les éléments d'indice 3 et 5 pour obtenir le tableau `[1,4,5,6,8,9,7]` qui est trié jusqu'à l'indice 3. On continue comme cela jusqu'à ce que tout le tableau soit trié.

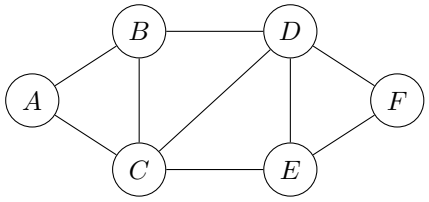
   Écrivez une fonction `tri_selection` qui prend en entrée un tableau d'entiers et trie ce tableau par ordre croissant à l'aide d'un tri par sélection. *Indication : vous pouvez utiliser la fonction `minimum` de la question précédente.*

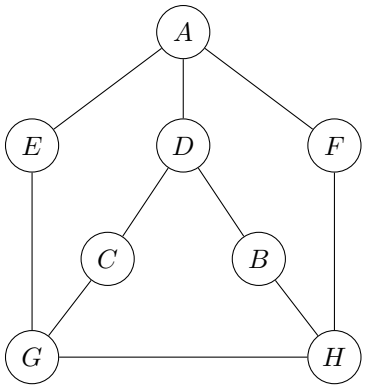
Solution : Puisqu'on implémente un tri *en place*, il n'est pas utile de renvoyer le tableau trié à la fin : on a directement modifié le tableau `t`. On peut s'arrêter dès lors que l'indice `i` est égal à `n-2`, puisqu'alors tous les éléments, sauf le dernier, sont triés, mais on sait aussi que ce dernier élément est plus grand que tous les autres : le tableau entier est donc trié.
```
fonction tri_selection(t: tableau d'entiers):
    n := longueur(t)
    Pour i de 0 à n−2 faire
        # le tableau est désormais trié jusqu'à l'indice i-1 inclus
        indice_minimum := minimum(t, i, n−1)
        # on échange les éléments i et indice_minimum
        tampon := t[indice_minimum]
        t[indice_minimum] := t[i]
        t[i] := tampon
    FinPour
```

Exercice 4. (Coloration de graphes) Colorer un graphe (non orienté) c'est associer à chaque sommet du graphe une couleur de sorte que deux sommets reliés par une arête n'ont pas la même couleur. Dans cet exercice, on propose un algorithme de coloration inédit, où les couleurs sont les entiers naturels $(0, 1, 2, \ldots)$.

L'algorithme traite les sommets dans l'ordre alphabétique de leurs noms (les sommets ont pour nom $A, B, C, \ldots$). Pour chaque sommet u, on liste l'ensemble E des couleurs utilisées par au moins un des sommets avec qui il est relié par une arête. On colorie alors le sommet u par *le plus petit entier positif ou nul qui n'est pas dans l'ensemble* E : si $E = \{0, 1, 3, 5\}$ par exemple, on colorie le sommet avec la couleur 2.

1. Exécutez cet algorithme sur les deux graphes ci-dessous.





Solution : Pour le graphe de gauche, voici les couleurs qu'on associe à chaque étape, avec l'ensemble des couleurs déjà associées à l'un au moins des voisins :

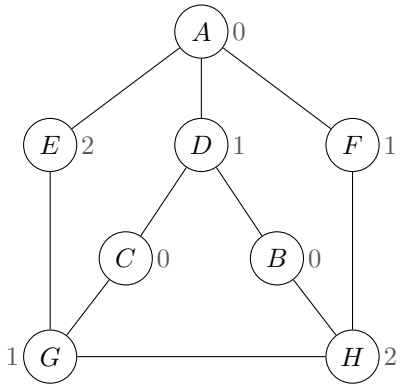
sommet traité	ensemble des couleurs de ses voisins	couleur attribuée
A	$\emptyset$	0
B	$\{0\}$	1
C	$\{0,1\}$	2
D	$\{1,2\}$	0
E	$\{0,2\}$	1
F	$\{0,1\}$	2

Pour le graphe de droite, cela donne :

sommet traité	ensemble des couleurs de ses voisins	couleur attribuée
A	$\emptyset$	0
B	$\emptyset$	0
C	$\emptyset$	0
D	$\{0\}$	1
E	$\{0\}$	1
F	$\{0\}$	1
G	$\{0,1\}$	2
H	$\{0,1,2\}$	3

2. Pour chacun des deux graphes ci-dessus, dites (en justifiant) si l'algorithme a renvoyé une coloration optimale, c'est-à-dire une coloration utilisant un minimum de couleurs possible.

Solution : On a colorié le graphe de gauche avec 3 couleurs, et on ne peut pas mieux faire, puisque le graphe contient un triangle (B, C, D par exemple) dont chaque sommet doit nécessairement être associé à une couleur distincte.
On a colorié le graphe de droite avec 4 couleurs. Cependant, il existe une meilleure coloration utilisant trois couleurs :

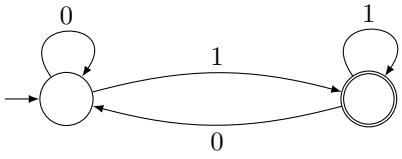


3. Donnez une borne supérieure (la plus précise possible) sur le nombre de couleurs utilisées au maximum par l'algorithme de coloration étudié dans cet algorithme.

Solution : L'algorithme de coloration étudié associe nécessairement à chaque sommet une couleur qui est inférieure à son degré (le nombre de ses voisins) plus 1. Ainsi, la coloration renvoyée utilise au maximum un nombre de couleurs qui est le degré maximal des sommets du graphe plus 1.

Exercice 5. (Automates)

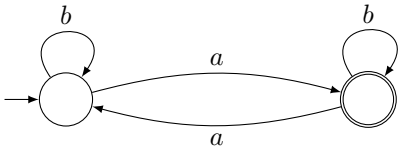
1. Donnez l'alphabet ainsi que l'ensemble des séquences acceptées (ou mots) par l'automate ci-dessous.



Solution : L'automate a pour alphabet $\{0, 1\}$. Il reconnait l'ensemble des séquences qui se terminent par un 1, c'est-à-dire l'ensemble des codes binaires d'entiers impairs.

2. Dessinez un automate qui reconnaît les séquences sur l'alphabet $\{a, b\}$ contenant un nombre impair de a.

Solution :



À présent, considérons l'alphabet $\{0, 1, \ldots, 9\}$ constitué des 10 chiffres de 0 à 9. L'objectif est de modéliser un digicode « simple », c'est-à-dire un digicode qui accepte n'importe quel mot en entrée dont le suffixe est le code attendu. Plus précisément, la porte associée au digicode s'ouvre au moment précis où le code est reconnu et le mécanisme électromagnétique maintient la porte fermée tant que le code précis n'est pas entré par l'utilisateur ou dès lors qu'un nouveau chiffre est entré après avoir saisi le code. À titre d'exemple, si le code attendu est `157`, les codes `0123157`, `86157`... permettent à l'utilisateur de pousser la porte. Tout autre code ne le permet pas.

3. Construisez un automate fini qui modélise le fonctionnement d'un digicode de ce type dont le code est `1794`.

Solution :