Faculté des Sciences
Aix*Marseille Université

Année universitaire 2019/2020

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)

Pour les trois premières questions, aucune justification n'est attendue.

1. Que vaut en base 10 le nombre 01011000111100000000000000000000 encodé en virgule flottante en base 2 ?

   a) $0{,}5 \times 2^{-20}$ b) $-1{,}75 \times 2^{50}$ c) $1{,}875 \times 2^{50}$ d) $1{,}8125 \times 2^{25}$

   Solution : La réponse correcte est la c). En effet :
   - le 1er bit 0 indique un nombre positif ;
   - les 8 bits qui suivent forment l'exposant, à savoir comme vu en cours : $k = (10110001)_2 - (127)_{10} = 1 \times 2^7 + 1 \times 2^5 + 1 \times 2^4 + 1 - 127 = 128 + 32 + 16 + 1 - 127 = 177 - 127 = 50$;
   - les 13 bits restant forment la mantisse : $m = 1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} = 1 + 0{,}5 + 0{,}25 + 0{,}125 = 1{,}875$.

2. Une machine de Turing est un automate qui permet de calculer tout ce qui est calculable.

   Vrai ou Faux ?

   Solution : Vrai.

3. On considère le tri par insertion et le tri par fusion. Quelles sont leurs complexités respectives ?

   a) $\mathcal{O}(n^2)$ et $\mathcal{O}(n)$ b) $\mathcal{O}(n^2)$ et $\mathcal{O}(n \log(n))$ c) $\mathcal{O}(n^3)$ et $\mathcal{O}(n)$ d) $\mathcal{O}(n^3)$ et $\mathcal{O}(n^2)$

   Solution : La réponse correcte est la b) (cf. cours).

4. Montrez que l'algorithme de coloration de graphe de Welsh-Powell n'est pas optimal, à savoir qu'il ne garantit pas de colorier un graphe avec le nombre nécessaire et suffisant de couleurs. Pour ce faire, vous exhiberez un cycle ayant 6 sommets dont la coloration donnée par l'algorithme de Welsh-Powell utilise 3 couleurs, alors qu'un tel cycle peut être colorié avec 2 couleurs seulement.

   Solution : cf. cours

Exercice 2. (Tableaux) Considérez l'algorithme suivant :

```
1  fonction mystère(A : tableau d'entiers):
2      n := longueur(A)
3      B := tableau longueur n rempli de 0
4      Pour i de 0 à n-1 faire
5          j := 0
6          Tant que j < n faire
7              B[i] := B[i] + A[j]
8              j := j + 2
9          FinTantQue
10         B[i] := B[i] * A[i]
11     FinPour
12     retourner B
```

1. Exécutez l'algorithme `mystère` sur le tableau d'entrée `[2, 1, 3, 2]` en donnant toutes les valeurs des variables pendant l'exécution et le résultat.

Solution : Le résultat est $[10, 5, 15, 10]$, et les valeurs des variables évoluent comme dans la table suivante :

A	n	B	i	j
[2, 1, 3, 2]	4	[0, 0, 0, 0]	0	0
		[2, 0, 0, 0]		2
		[5, 0, 0, 0]		4
		[10, 0, 0, 0]	1	0
		[10, 2, 0, 0]		2
		[10, 5, 0, 0]		4
		[10, 5, 0, 0]	2	0
		[10, 5, 2, 0]		2
		[10, 5, 5, 0]		4
		[10, 5, 15, 0]	3	0
		[10, 5, 15, 2]		2
		[10, 5, 15, 5]		4
		[10, 5, 15, 10]		

2. Calculez, en justifiant, le nombre d'instructions effectuées par l'algorithme `mystère` en fonction de la longueur `n` du tableau d'entrée, pris quelconque. Vous pourrez simplifier le résultat en utilisant la notation « grand $\mathcal{O}$ ».

Solution : On exécute les lignes 2, 3 et 12 de l'algorithme seulement une fois, pour un total de 3 instructions.
On exécute le contenu de la boucle qui commence à la ligne 4 n fois ; la ligne 4 elle-même est exécutée $n + 1$ fois (on sort de la boucle quand $i = n$, donc une fois de plus). Donc on exécute n fois les lignes 5 et 10.
Comme la variable j est augmentée de 2 à chaque tour de la boucle « tant que », on n'exécute les instructions des lignes 7 et 8 approximativement $n/2$ fois pour chaque tour de la boucle « pour », et la ligne 6 une fois de plus, donc $n/2 + 1$, ce qui donne $3n/2 + 1$ exécutions des lignes 6 à 8 pour chaque tour de la boucle « pour ».
Donc, dans la boucle « pour », on exécute n fois multiplié par $3n/2 + 1 + 2 = 3n/2 + 3$ fois les lignes de 5 à 10, ce qui donne $n \times (3n/2 + 3) = 3n^2/2 + 3n$, et il faut ajouter $n + 1$ exécutions de la ligne 4, ce qui donne $3n^2/2 + 4n + 1$ instructions.
En ajoutant les lignes 2, 3 et 12 on obtient $3n^2/2 + 4n + 4$, qu'on peut simplifier en $\mathcal{O}(n^2)$.

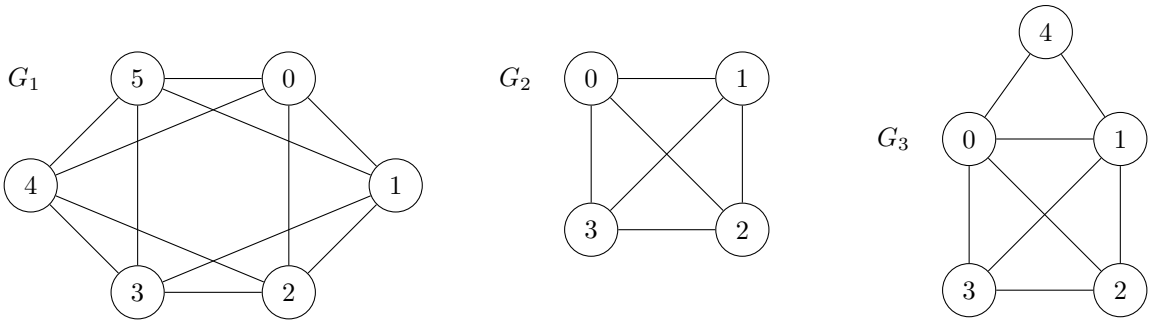
3. Écrivez un algorithme `décroissant(A)` en pseudo-code qui prend en entrée un tableau d'entiers `A` et renvoie `vrai` si le tableau est trié *en ordre décroissant*, et `faux` autrement.

Solution :
```
fonction décroissant(A : tableau d'entiers):
    n := longueur(A)
    i := 0
    Pour i := 0 à n - 2 faire        {on s'arrête avant d'arriver à n - 1}
        Si A[i] < A[i+1] alors
            retourner faux
        FinSi
    FinPour
    retourner vrai
```

Exercice 3. (Graphes) Dans un graphe non orienté, on appelle *cycle eulérien* tout cycle (un chemin qui revient à son point de départ) qui traverse chaque arête du graphe une et une seule fois. On a vu en cours la caractérisation suivante :

> « Un graphe non orienté connexe admet un cycle eulérien si et seulement si chaque sommet est de degré pair. »

1. En utilisant la caractérisation précédente, dites pour chacun des graphes suivants s'ils admettent un cycle eulérien.



Solution : Tous les sommets de G_1 sont de degré pair donc G_1 contient un cycle eulérien.
Les graphes G_2 et G_3 possèdent chacun au moins un sommet de degré impair (le sommet 2 par exemple), donc ils ne possèdent pas de cycle eulérien.

L'algorithme de Hierholzer qu'on a décrit en cours nous permet de construire un cycle eulérien (s'il en existe un). On l'a décrit de manière informelle ainsi :

- Choisir n'importe quel sommet initial v
- Suivre un chemin arbitraire d'arêtes jusqu'à retourner à v, obtenant ainsi un cycle c
- **Tant qu**'il y a des sommets u dans le cycle c avec des arêtes qu'on n'a pas encore choisies **faire**
  - Suivre un chemin à partir de u, n'utilisant que des arêtes pas encore choisies, jusqu'à retourner à u, obtenant un cycle c'
  - Prolonger le cycle c par c'

2. Pour les graphes précédents qui admettent un cycle eulérien, trouvez un tel cycle en appliquant l'algorithme de Hierholzer, en essayant de générer des *petits cycles* c' le long de l'algorithme, afin que l'algorithme utilise au moins 3 itérations de la boucle **Tant que** pour converger.

Solution : On applique l'algorithme de Hierholzer au graphe G_1, unique graphe qui contient un cycle eulérien.
On part du sommet 0 et on trouve (par exemple) le cycle $0, 1, 2, 0$. À partir du sommet 1, on ajoute le cycle $1, 3, 5, 1$ pour obtenir $0, 1, 3, 5, 1, 2, 0$. À partir du sommet 5, on ajoute le cycle $5, 0, 4, 5$ pour obtenir $0, 1, 3, 5, 0, 4, 5, 1, 2, 0$. À partir du sommet 2, on ajoute alors le cycle $2, 4, 3, 2$ pour obtenir le cycle eulérien $0, 1, 3, 5, 0, 4, 5, 1, 2, 4, 3, 2, 0$.

3. La notion de cycle eulérien peut être étendue : un *chemin eulérien* est un chemin (pas nécessairement un cycle) qui traverse chaque arête du graphe une et une seule fois. Parmi les graphes précédents, quels sont ceux pour lesquels vous pouvez trouver un chemin eulérien ? *Dans le cas où vous n'en trouvez pas, on ne demande pas de preuve qu'il n'en existe pas.*

Solution : Le graphe G_1 possède un chemin eulérien, puisqu'il possède un cycle eulérien.
Le graphe G_2 ne possède pas de chemin eulérien.
Le graphe G_3 possède un chemin eulérien : $2, 1, 4, 0, 1, 3, 0, 2, 3$.

4. Considérons un graphe non orienté quelconque. Si le graphe possède un chemin eulérien allant du sommet u_0 au sommet $u_1 \neq u_0$, quelle propriété a le degré des sommets u_0 et u_1 ? Quelle propriété a le degré de tous les autres sommets du graphe ?

Solution : Supposons que le graphe possède un chemin eulérien allant du sommet u_0 au sommet u_1. Alors, le long de ce chemin, on sort une fois du sommet u_0, puis ensuite, à chaque fois qu'on y rentre, on en ressort juste après. Puisque le chemin est eulérien, le degré de u_0 est donc impair. Le même raisonnement s'applique pour u_1. Pour les autres sommets, on y rentre autant qu'on en ressort, donc leur sommet est pair.

5. Dans le cas où il existe un chemin eulérien allant d'un sommet u_0 à un sommet $u_1 \neq u_0$, transformez l'algorithme de Hierholzer pour produire un tel chemin. On ne demande pas de prouver la correction de l'algorithme, mais **vous l'illustrerez sur un exemple pertinent**.

Solution : Notons u_0 et u_1 les deux seuls sommets de degrés impairs (dont on connait l'existence par la question précédente).

- Suivre un chemin arbitraire d'arêtes allant de u_0 à u_1 (par exemple, à l'aide d'un parcours de graphe)
- **Tant qu**'il y a des sommets u dans le chemin avec des arêtes qu'on n'a pas encore choisies **faire**
  - Suivre un chemin à partir de u, n'utilisant que des arêtes pas encore choisies, jusqu'à retourner à u, obtenant un cycle c'
  - Prolonger le chemin courant avec le cycle c'

Sur l'exemple du graphe G_3. On commence par le chemin composé d'une unique arête allant du sommet 2 au sommet 3. Puis, partant de 2, on trouve le cycle $2, 1, 0, 2$. On étend le chemin courant pour obtenir le chemin $2, 1, 0, 2, 3$. En partant de 1, on trouve le cycle $1, 4, 0, 3, 1$. On étend le chemin courant pour obtenir le chemin $2, 1, 4, 0, 3, 1, 0, 2, 3$. C'est un chemin eulérien du graphe G_3.

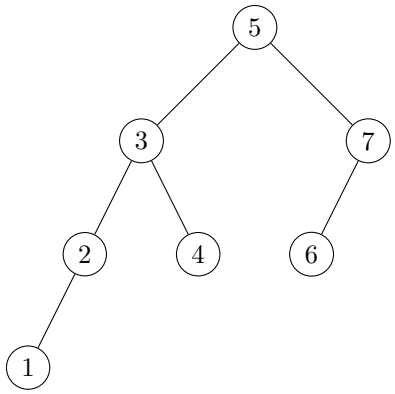
Une autre solution consiste à ajouter une fausse arête entre le sommet u_0 et le sommet u_1 (quitte à considérer des graphes avec des multi-arêtes). Le graphe ne contient plus alors que des sommets de degré pair. On peut alors appliquer l'algorithme de Hierholzer pour trouver un cycle eulérien. On retire l'arête fictive entre u_0 et u_1 ce qui fournit alors un chemin eulérien.

Exercice 4. (Arbres) La propriété fondamentale des arbres binaires de recherche est que, pour chaque nœud x de l'arbre :

- tous les nœuds dans le sous-arbre gauche ont une valeur inférieure à la valeur de x ;
- tous les nœuds dans le sous-arbre droit ont une valeur supérieure à la valeur de x.

1. Construisez un arbre binaire de recherche en insérant une par une les valeurs 5, 7, 6, 3, 4, 2, 1 dans l'ordre indiqué.

Solution :



2. Donnez la liste des valeurs des nœuds traversés si on effectue un parcours *infixe* de l'arbre binaire de recherche obtenu.

Solution : On obtient les valeurs en ordre croissant, donc 1, 2, 3, 4, 5, 6, 7.

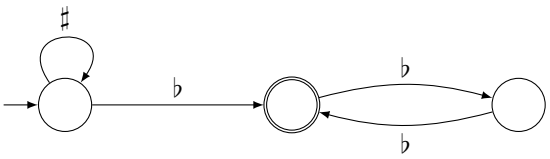
3. Donnez un algorithme (de façon informelle en français ou en pseudo-code) pour construire un arbre binaire de recherche bien équilibré (c'est-à-dire, tel que les hauteurs des sous-arbres gauche et droit de chaque nœud sont proches) à partir d'un tableau *trié* d'entiers A.

Solution :

```
procédure construire_abr(A)
    n  := longueur(A)
    Si n > 0 alors
        m = n div 2                  {division entière par 2}
        insérer A[m] dans l'arbre
        B := A[0...m-1]              {éléments de A avant la position m}
        C := A[m+1...n-1]            {éléments de A après la position m}
        construire_abr(B)
        construire_abr(C)
    FinSi
```

Exercice 5. (Automates)

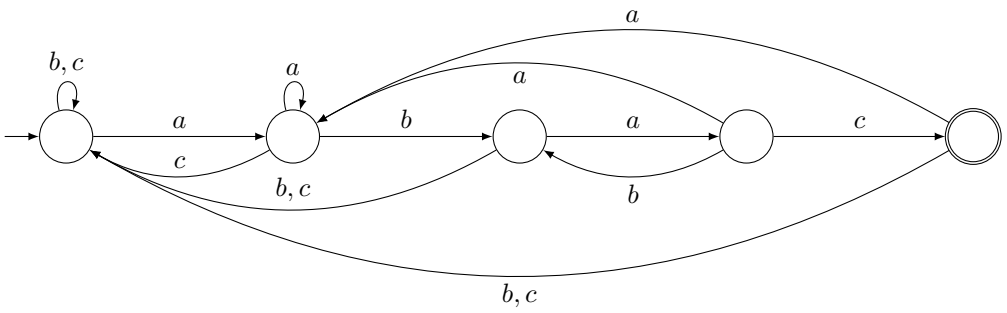
1. Donnez l'alphabet ainsi que l'ensemble de toutes les séquences (ou mots) acceptées par l'automate ci-dessous.



Solution : L'automate a pour alphabet $\{\sharp, \flat\}$. Il reconnait l'ensemble des séquences contenant d'abord un nombre quelconque de $\sharp$, suivi d'un nombre impair de $\flat$.

2. Donnez un automate sur l'alphabet $\{a, b, c\}$ qui reconnaît l'ensemble des séquences (ou mots) qui se terminent par *abac*.

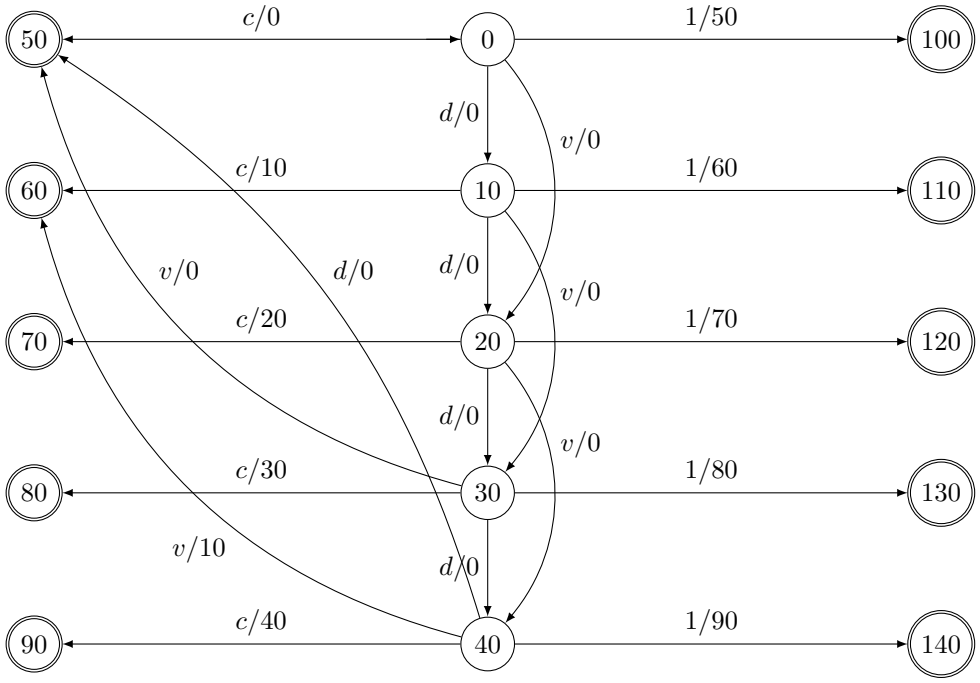
Solution :



3. L'objectif ici est de modéliser par un automate le fonctionnement d'une machine permettant d'acheter des tickets de transports en commun dont la valeur est 50 centimes d'euro. Cette machine accepte les pièces de 10, 20 et 50 centimes, et 1 euro. Par ailleurs, elle rend la monnaie. Pour éviter toute confusion :
   — l'alphabet d'entrée (on appelle ici entrée les pièces qui sont données par l'utilisateur) est l'ensemble $\Sigma = \{d, v, c, 1\}$, où d correspond à une pièce de 10 centimes, v à une pièce de 20 centimes, c à une pièce de 50 centimes, et 1 à une pièce d'1 euro.
   — l'alphabet de sortie (on appelle ici sortie les pièces qui sont rendues par la machine dans le cas où l'utilsateur a donné plus d'argent que nécessaire) est $\Sigma' = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90\}$.
   — le fonctionnement de la machine est tel que dès que l'utilisateur a fourni une pièce qui fait que le montant total donné est au moins égal à 50 centimes d'euro, la demande de ticket est validée et le rendu de monnaie est opéré si nécessaire.
   — le rendu de monnaie est modélisé directement sur les transitions de l'automate. Concrètement, chaque transition de l'automate est étiquetée par une paire a/b, avec $a \in \Sigma$ et $b \in \Sigma'$, ce qui s'interprète comme le fait que l'utilisateur a mis une pièce de valeur a et que la machine doit rendre le montant b en monnaie (on considère ici que le choix des pièces pour le rendu de monnaie est géré par un autre automate qui n'a pas à être modélisé).

   Proposez un automate validant cette spécification.

Solution : L'énoncé ne précise pas si on doit permettre à l'automate de modéliser une unique transaction ou s'il doit permettre d'enchainer les transactions. On fournit donc deux corrections envisageables.
Voici d'abord l'automate si on ne veut représenter qu'une unique transaction : les états acceptant représentent donc la fin de la transaction.



Voici ensuite l'automate qui permet d'enchainer les transactions : l'unique état acceptant est alors l'état initial qui signifie qu'on peut, si on le souhaite, commencer une nouvelle transaction.