Faculté des Sciences
Aix*Marseille Université

Année universitaire 2022/2023

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 : SPO1U65 Libellé du module : Introduction à l'informatique
Calculatrices autorisées : NON Documents autorisés : NON

Les exercices sont indépendants et peuvent donc être traités dans n'importe quel ordre. Vous pouvez passer une question vous semblant trop difficile, quitte à y revenir ensuite. Par ailleurs, toute réponse non correctement rédigée (avec explication précise et concise) ou non justifiée sera considérée comme fausse (sauf mention explicite du contraire sur le sujet).

Dans cet examen, on considère qu'un tableau de taille n admet des indices allant de 0 à $n-1$.

Exercice 1. (Exécution et analyse d'un algorithme)

On considère ici l'algorithme suivant, qui prend en entrée un tableau d'entiers A :

```
def mystère(A):
    n = len(A)
    i = 1
    j = 0
    while i < n:
        A[j] = A[i]
        j = j + 1
        i = 2 * i
    return A
```

1. Exécutez l'algorithme `mystère` sur le tableau $A = [8, 1, 6, 3, 2, 9, 7, 4, 5]$ en remplissant une table comme la suivante, qui montre les valeurs de toutes les variables au cours de l'exécution, et donnez le résultat renvoyé par l'algorithme.

A	n	i	j
$[8, 1, 6, 3, 2, 9, 7, 4, 5]$			
$\vdots$	$\vdots$	$\vdots$	$\vdots$

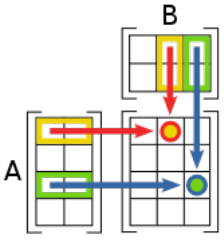
2. Expliquez pourquoi cet algorithme termine sur toute entrée.
3. Évaluez le nombre d'instructions (définies ici par les lignes de code exécutées, tests inclus) de l'algorithme en fonction de la taille n du tableau, d'abord de façon exacte et puis simplifiez l'expression avec la notation « grand O ».
4. Trouvez un tableau A tel que l'algorithme `mystère` renvoie $[9, 7, 4, 1, 4, 2, 3, 8, 1]$ quand il est exécuté sur l'entrée A.

Exercice 2. (Matrices d'entiers) Une matrice d'entiers est un tableau d'entiers à deux dimensions. Une matrice d'entiers M à m lignes et n colonnes est plus précisément un tableau « rectangulaire » contenant $m \times n$ entiers. Dans cet exercice, nous allons nous intéresser au produit ordinaire de matrices d'entiers.

Soit A une matrice de k lignes et ℓ colonnes et soit B une matrice de m lignes et n colonnes. Tout d'abord, soulignons que le produit $A \times B$ n'est défini que si le nombre de colonnes de A est égal au nombre de lignes de B, c'est-à-dire si $\ell = m$. Admettons que $\ell = m$. Leur *produit* $A \times B = C$ est une matrice de k lignes et n colonnes telle que, pour tout i et j :

$$C[i][j] = \sum_{x=0}^{m-1} A[i][x] \times B[x][j] = A[i][0] \times B[0][j] + A[i][1] \times B[1][j] + \cdots + A[i][m-1] \times B[m-1][j].$$

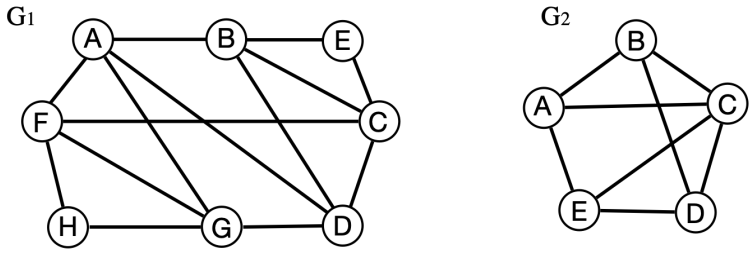
À titre d'exemple, si l'on considère deux matrices A (de taille 4×2) et B (de taille 2×3), on a :



$$C[0][1] = A[0][0] \times B[0][1] + A[0][1] \times B[1][1]$$
$$C[2][2] = A[2][0] \times B[0][2] + A[2][1] \times B[1][2]$$

1. Le produit matriciel ordinaire n'est pas commutatif, c'est-à-dire que pour deux matrices A et B quelconques, $A \times B$ n'est pas nécessairement égal à $B \times A$. Donnez deux matrices qui illustrent la non-commutativité du produit matriciel ordinaire.
2. Proposez un algorithme de prototype
   `def mult_mat(A, B)`
   qui prend en paramètres d'entrée deux matrices d'entiers A et B et qui renvoie la matrice d'entiers $C = A \times B$, quand $A \times B$ est possible.
3. Expliquez pourquoi cet algorithme termine sur toute entrée.
4. Évaluez le nombre d'instructions (définies ici par les lignes de code exécutées, tests inclus) dans le pire cas de l'algorithme en fonction des tailles des matrices A et B d'abord de façon exacte, puis en simplifiant l'expression avec la notation « grand O ».

Exercice 3. (Graphes) On s'interesse à des propriétés des graphes.



1. Est-ce que le graphes G_1 est planaire ? Et G_2 ? Si c'est le cas, redessinez le ou les graphes planaires sans intersections d'arêtes.
2. Lesquels parmi les graphes G_1 et G_2 admettent un cycle eulérien ?
3. Pour chacun des graphes précédents qui admet un cycle eulérien, trouvez-en au moins un en appliquant l'algorithme de Hierholzer, dont la description suit, et énumérez les sommets traversés le long du cycle.

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

4. Rappelons ici l'algorithme de Welsh-Powell pour la coloration de graphes :

   — Trier le sommets du graphe par ordre de degré (nombre de voisins) décroissant *en gardant en ordre alphabétique les sommets ayant le même degré*
   — *couleur* = 1
   — **Tant qu'**il y a encore des sommets non coloriés **faire**
      — Parcourir la liste triée des sommets et colorier en *couleur* les sommets non coloriés qui ne sont pas connectés à d'autres sommets de la même couleur
      — *couleur* = *couleur* + 1

   Appliquez l'algorithme de Welsh-Powell aux graphes G_1 et G_2 et dessinez les graphes obtenus, en indiquant à côté de chaque sommet le numéro de la couleur correspondante.
5. Combien de couleurs sont nécessaires et suffisants pour colorier G_1 et G_2 ?

Faculté des Sciences
Aix*Marseille Université

Année universitaire 2022/2023

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 : SPO1U65 Libellé du module : Introduction à l'informatique
Calculatrices autorisées : NON Documents autorisés : NON

Les exercices sont indépendants et peuvent donc être traités dans n'importe quel ordre. Vous pouvez passer une question vous semblant trop difficile, quitte à y revenir ensuite. Par ailleurs, toute réponse non correctement rédigée (avec explication précise et concise) ou non justifiée sera considérée comme fausse (sauf mention explicite du contraire sur le sujet).

Dans cet examen, on considère qu'un tableau de taille n admet des indices allant de 0 à $n-1$.

Exercice 1. (Exécution et analyse d'un algorithme)

On considère ici l'algorithme suivant, qui prend en entrée un tableau d'entiers A :

```
1  def mystère(A):
2      n = len(A)
3      i = 1
4      j = 0
5      while i < n:
6          A[j] = A[i]
7          j = j + 1
8          i = 2 * i
9      return A
```

1. Exécutez l'algorithme `mystère` sur le tableau $A = [8, 1, 6, 3, 2, 9, 7, 4, 5]$ en remplissant une table comme la suivante, qui montre les valeurs de toutes les variables au cours de l'exécution, et donnez le résultat renvoyé par l'algorithme.

A	n	i	j
$[8, 1, 6, 3, 2, 9, 7, 4, 5]$			
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Solution : Après cette exécution :

A	n	i	j
$[8, 1, 6, 3, 2, 9, 7, 4, 5]$	9		
$[1, 1, 6, 3, 2, 9, 7, 4, 5]$		1	0
$[1, 6, 6, 3, 2, 9, 7, 4, 5]$		2	1
$[1, 6, 2, 3, 2, 9, 7, 4, 5]$		4	2
$[1, 6, 2, 5, 2, 9, 7, 4, 5]$		8	3
		16	4

l'algorithme renvoie le résultat $[1, 6, 2, 5, 2, 9, 7, 4, 5]$.

2. Expliquez pourquoi cet algorithme termine sur toute entrée.

Solution : La seule façon qui empêcherait la terminaison de l'algorithme est si la boucle `Tant que` était exécutée à l'infini ; montrons que ce n'est pas le cas.
La variable i a, au début, la valeur 1. Si $n \leq 1$ (c'est-à-dire, si le tableau est vide ou de longueur 1), la boucle `while` n'est jamais exécutée et l'algorithme termine immédiatement.
Sinon, la valeur de i est doublée à chaque tour de la boucle `Tant que`, sa valeur augmente strictement. Donc, tôt ou tard elle atteindra ou dépassera la valeur de n (un entier non-négatif), ce qui rendra fausse la condition de la boucle et causera l'exécution de l'instruction `return A`, terminant ainsi l'algorithme.

3. Évaluez le nombre d'instructions (définies ici par les lignes de code exécutées, tests inclus) de l'algorithme en fonction de la taille n du tableau, d'abord de façon exacte et puis simplifiez l'expression avec la notation « grand O ».

Solution : Comptons le nombre d'exécutions de chaque ligne du code :

- Il y a 4 instructions hors boucle aux lignes 2, 3, 4 et 9.
- Comme i est doublée à chaque tour de boucle jusqu'à atteindre ou dépasser la valeur n, le nombre total d'itérations est $\lceil \log_2 n \rceil$. Chaque itération exécute les lignes 6, 7 et 8 une fois, ce qui donne un total de $3\lceil \log_2 n \rceil$ instructions.
- La condition de la boucle à la ligne 5 est exécutée une fois de plus (ce qui correspond au dernier test, quand la condition devient fausse), donc $\lceil \log_2 n \rceil + 1$ fois.

Au total, on a donc $4 + 3\lceil \log_2 n \rceil + \lceil \log_2 n \rceil + 1 = 4\lceil \log_2 n \rceil + 5$ instructions, ce qui est $O(\log_2 n)$.

4. Trouvez un tableau A tel que l'algorithme `mystère` renvoie $[9, 7, 4, 1, 4, 2, 3, 8, 1]$ quand il est exécuté sur l'entrée A.

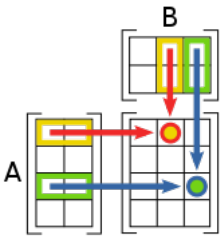
Solution : N'importe quel tableau de la forme $A = [x, 9, 7, y, 4, 2, 3, 8, 1]$ donne le résultat demandé, pour n'importe quels entiers x et y.

Exercice 2. (Matrices d'entiers) Une matrice d'entiers est un tableau d'entiers à deux dimensions. Une matrice d'entiers M à m lignes et n colonnes est plus précisément un tableau « rectangulaire » contenant $m \times n$ entiers. Dans cet exercice, nous allons nous intéresser au produit ordinaire de matrices d'entiers.

Soit A une matrice de k lignes et ℓ colonnes et soit B une matrice de m lignes et n colonnes. Tout d'abord, soulignons que le produit $A \times B$ n'est défini que si le nombre de colonnes de A est égal au nombre de lignes de B, c'est-à-dire si $\ell = m$. Admettons que $\ell = m$. Leur *produit* $A \times B = C$ est une matrice de k lignes et n colonnes telle que, pour tout i et j :

$$C[i][j] = \sum_{x=0}^{m-1} A[i][x] \times B[x][j] = A[i][0] \times B[0][j] + A[i][1] \times B[1][j] + \cdots + A[i][m-1] \times B[m-1][j].$$

À titre d'exemple, si l'on considère deux matrices A (de taille 4×2) et B (de taille 2×3), on a :



$$C[0][1] = A[0][0] \times B[0][1] + A[0][1] \times B[1][1]$$
$$C[2][2] = A[2][0] \times B[0][2] + A[2][1] \times B[1][2]$$

1. Le produit matriciel ordinaire n'est pas commutatif, c'est-à-dire que pour deux matrices A et B quelconques, $A \times B$ n'est pas nécessairement égal à $B \times A$. Donnez deux matrices qui illustrent la non-commutativité du produit matriciel ordinaire.

Solution : *Indication de correction :* Un des deux arguments ci-dessous suffit.

Admettons que les deux matrices A et B sont telles que $A \times B = C$. Cela signifie que $A \times B$ est correctement défini, autrement dit que le nombre de colonnes de A et que le nombre de lignes de B sont égaux. Or, rien n'induit dans ce cas que le nombre de colonnes de B soit égal au nombre de lignes de A et donc que le produit $B \times A$ soit défini. Par conséquent, le produit matriciel ordinaire n'est pas commutatif.
À titre d'exemple, considérons

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \text{ et } B = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}.$$

Le produit $A \times B$ est défini, ce qui n'est pas le cas de $B \times A$.

Considérons maintenant A et B telles que $A \times B$ et $B \times A$ sont correctement défini et montrons que le produit matriciel n'est pas commutatif, c'est-à-dire que $A \times B \neq B \times A$. Pour ce faire, prenons simplement les deux matrices carrées d'ordre (*i.e.* de taille de côté) 2 suivantes :

$$A = \begin{pmatrix} 2 & 3 \\ 3 & 1 \end{pmatrix} \text{ et } B = \begin{pmatrix} 5 & 2 \\ 3 & 4 \end{pmatrix}$$

Le produit $A \times B$ vaut :

$$A \times B = \begin{pmatrix} 2 \times 5 + 3 \times 3 & 2 \times 2 + 3 \times 4 \\ 3 \times 5 + 1 \times 3 & 3 \times 2 + 1 \times 4 \end{pmatrix} = \begin{pmatrix} 19 & 16 \\ 18 & 10 \end{pmatrix},$$

alors que le produit $B \times A$ vaut :

$$B \times A = \begin{pmatrix} 5 \times 2 + 2 \times 3 & 5 \times 3 + 2 \times 1 \\ 3 \times 2 + 4 \times 3 & 3 \times 3 + 4 \times 1 \end{pmatrix} = \begin{pmatrix} 16 & 17 \\ 18 & 13 \end{pmatrix},$$

ce qui montre que $A \times B \neq B \times A$.

2. Proposez un algorithme de prototype
   `def mult_mat(A, B)`
   qui prend en paramètres d'entrée deux matrices d'entiers A et B et qui renvoie la matrice d'entiers $C = A \times B$, quand $A \times B$ est possible.

Solution : L'algorithme est le suivant :

```
1  Fonction mult_mat(d A[0..k-1][0..l-1], B[0..m-1][0..n-1]: matrices): matrice
2    i, j, p: entiers;
3    C[0..k-1][0..n-1]: matrice;
4  Début
5    # Test de la définissabilité du produit A x B.
6    Si (k != m) alors
7      écrire("Erreur : Le produit A x B n'est pas défini.\n");
8      Sortir;
9    Finsi
10   Pour i de 0 à k-1 faire       # Parcours des lignes de A
11     Pour j de 0 à n-1 faire     # Parcours des colonnes de B
12       C[i][j] := 0;
13       Pour p de 0 à m-1 faire   # Parcours des colonnes de A et des lignes de B
14         C[i][j] := C[i][j] + A[i][p] * B[p][j];
15       Finfaire
16     Finfaire
17   Finfaire
18   Retour C;
19 Fin
```

3. Expliquez pourquoi cet algorithme termine sur toute entrée.

Solution : Cet algorithme est construit en deux parties : la première sert à vérifier que le produit des matrices A et B est correctement défini (et donc possible) ; la seconde réalise le produit dans le cas où la vérification préalable n'a pas échoué.
Admettons que la phase de vérification de définissabilité échoue, dans ce cas, un message est écrit à l'écran et l'algorithme force l'arrêt du programme avec `Sortir`. Par conséquent, l'arrêt est garanti.
Admettons maintenant qu'elle réussit. Dans ce cas, l'algorithme entre dans une séquence de trois boucles imbriquées, toutes définies au moyen de `Pour`, ce qui garantit qu'elles sont correctement bornées et qu'on en sortira. Enfin, une dernière instruction est exécutée qui consiste à retourner C.
En conséquence, l'algorithme `mult_mat` s'arrête bel et bien.

4. Évaluez le nombre d'instructions (définies ici par les lignes de code exécutées, tests inclus) dans le pire cas de l'algorithme en fonction des tailles des matrices A et B d'abord de façon exacte, puis en simplifiant l'expression avec la notation « grand O ».

Solution : Tout d'abord, soulignons que le pire cas est induit par le fait que le produit $A \times B$ est correctement défini, c'est-à-dire celui où les trois boucles imbriquées sont exécutées.

Commençons par dénombrer les instructions en dehors de toute boucle. Nous avons : *(i)* l'expression booléenne `k != m` évaluée une fois par la structure conditionnelle `Si` ; *(ii)* l'initialisation implicite de i à 0 juste après la structure conditionnelle ; et *(iii)* le retour du produit calculé à la fonction appelante. Soit au total 3 instructions.

À l'intérieur des boucles, nous avons :

(a) Dans la première boucle sur `i`, itérée k fois :
- le test vérifiant que i est dans l'intervalle $[0, k[$;
- l'initialisation implicite de j à 0 ;
- l'incrémentation implicite de i.

Soit au total $3k$ instructions.

(b) Dans la deuxième boucle sur `j`, itérée $k \times n$ fois :
- le test vérifiant que j est dans l'intervalle $[0, n[$;
- l'initialisation de $C_{i,j}$ à 0 ;
- l'initialisation implicite de p à 0 ;
- l'incrémentation implicite de j.

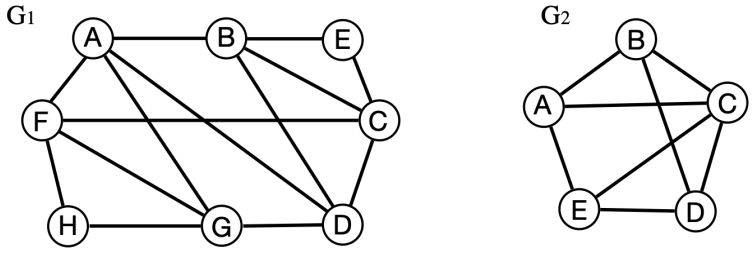
Soit au total $4kn$ instructions.

(c) Dans la troisième boucle sur `p`, itérée $k \times n \times m$ fois :
- le test vérifiant que p est dans l'intervalle $[0, m[$;
- l'incrémentation implicite de p ;
- un calcul amenant au terme de l'itération à obtenir la valeur du coefficient $C_{i,j}$.

Soit au total $3knm$ instructions.

Dans le pire cas, on a donc $3 + 3k + 4kn + 3knm$ instructions exécutées par la fonction `mult_mat`. Or, 3, $3k$ et $4kn$ sont des quantités asymptotiquement négligeables par rapport à $3knm$. Par conséquent, nous pouvons conclure que la complexité de `mult_mat` est en $O(knm)$.

Exercice 3. (Graphes) On s'interesse à des propriétés des graphes.



1. Est-ce que le graphes G_1 est planaire ? Et G_2 ? Si c'est le cas, redessinez le ou les graphes planaires sans intersections d'arêtes.
2. Lesquels parmi les graphes G_1 et G_2 admettent un cycle eulérien ?
3. Pour chacun des graphes précédents qui admet un cycle eulérien, trouvez-en au moins un en appliquant l'algorithme de Hierholzer, dont la description suit, et énumérez les sommets traversés le long du cycle.

- Choisir n'importe quel sommet initial v
- Suivre un chemin arbitraire d'arêtes jusqu'à retourner à v, obtenant ainsi un cycle partiel c
- **Tant qu'**il y a des sommets u dans le cycle c avec des arêtes encore inexplorées **faire**

— Suivre un chemin de sommets à partir de u jusqu'à retourner à u, obtenant un cycle c'
— Prolonger le cycle c par c'

4. Rappelons ici l'algorithme de Welsh-Powell pour la coloration de graphes :

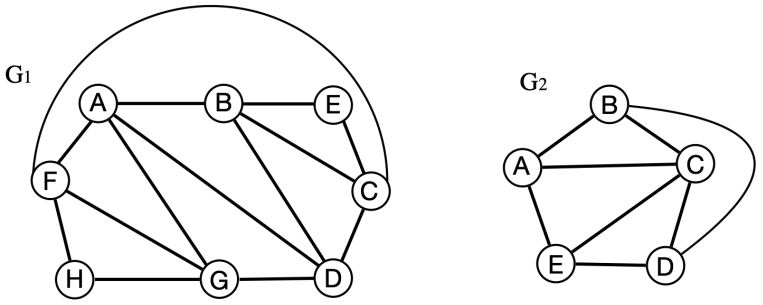
— Trier le sommets du graphe par ordre de degré (nombre de voisins) décroissant *en gardant en ordre alphabétique les sommets ayant le même degré*
— *couleur* = 1
— **Tant qu'**il y a encore des sommets non coloriés **faire**
— Parcourir la liste triée des sommets et colorier en *couleur* les sommets non coloriés qui ne sont pas connectés à d'autres sommets de la même couleur
— *couleur* = *couleur* + 1

Appliquez l'algorithme de Welsh-Powell aux graphes G_1 et G_2 et dessinez les graphes obtenus, en indiquant à côté de chaque sommet le numéro de la couleur correspondante.

5. Combien de couleurs sont nécessaires et suffisants pour colorier G_1 et G_2 ?

Solution :

— Le graphe G_1 est planaire. G_2 est aussi planaire. On peut les redessinez comme montré dans la figure suivante



— Le graphe G_1 admet un cycle euleérien pour le théorème d'Euleur et le fait que tous ses sommets ont degré pair. G_2 n'admet pas un cycle euleérien car certains de ses sommets ont dégré impair.
— Le graphe G_1 a, par exemple, le cycle eulérien obtenu par l'union des cycles $A \to B \to E \to C \to D \to G \to H \to F \to A$ et $C \to B \to D \to A \to G \to F \to C$, c'est-à-dire le cycle $A \to B \to E \to C \to B \to D \to A \to G \to F \to C \to D \to G \to H \to F \to A$.
— — L'ordre des sommets de G_1 (d'abord par degré et puis alphabéthique, en cas d'égalité) est A, B, C, D, F, G, E, H.
Dans la première itération, on colorie en couleur 0 les sommets A puis C et enfin H.
Dans la deuxième itération, on colorie en couleur 1 le sommet B puis F.
Dans la troisième itération, on colorie en couleur 2 le sommet D puis E. Dans la dernière itération on colorie en couleur 3 le sommet G.
— L'ordre des sommets de G_2 (d'abord par degré et puis alphabéthique, en cas d'égalité) est C, A, B, D, E.
Dans la première itération, on colorie en couleur 0 les sommet C.
Dans la deuxième itération, on colorie en couleur 1 le sommet A puis D.
Dans la troisième itération, on colorie en couleur 2 le sommet B puis E.
— Pour G_1 et G_2, 3 couleurs sont necessaires et suffisantes. Necessaires parce qu'il y a un triangle dans chacun de ces graphes. Pour G_1, 3 couleurs sont suffisantes, parce que on peut colorier en 0 les sommets A, C, H, en 1 les sommets B et G et on 2 les sommets D, F et E.
Pour G_2, l'algorithme de Welsh-Powell donne une coloration optimale.