Révision

Information, Calcul et Communication

Abdellah El Mrini

Rappel

Poids dans la note finale

65%

Date

Le 26.06.2026 à 09:15

Format similaire au midterm

- Une partie QCM
- Une partie Vrai/Faux
- Questions ouvertes

Conseils pour l'Examen

1 Maîtrisez les concepts fondamentaux

Revoyez tous les concepts clés, définitions et algorithmes essentiels abordés en cours.

2 Pratiquez activement

Refaites les exercices, les travaux pratiques et les questions des examens précédents pour vous familiariser avec le format.

3 Gérez votre temps

Allouez un temps réaliste à chaque section (QCM, Vrai/Faux, Questions ouvertes) et ne restez pas bloqué sur une question.

4 Lisez attentivement

Prenez le temps de bien comprendre chaque énoncé avant de répondre, surtout pour les questions détaillées.

Conseils pour l'Examen

Définir entrées & sorties

Identifier clairement les arguments, le type de retour et les **cas limites** avant toute implémentation.

Tracer à la main sur des petits examples

Simuler l'algorithme avec de **petites valeurs** pour visualiser les appels et repérer les erreurs.

Pour les algorithmes récursifs

- le **cas de base** ;
- la **taille du sous-problème** ;
- ce qui est **renvoyé** par l’appel récursif.

Syntaxe

Faire attention aux erreurs de syntaxes (par exemple indentation)

Les erreurs qui coûtent des points

Évitez ces pièges courants pour maximiser vos chances de réussite à l'examen.

Oublier le cas de base en récursivité

Oublier self dans les classes

Oublier que les indices Python commencent à 0

Modifier une liste pendant qu'on la parcourt

Confondre = et ==

Oublier de retourner une valeur avec return

EXERCICE 1

Exercice 1 : Algorithme Récursif Mystère

mystère

entrée : *deux listes L_1, L_2 de nombres entiers positifs, toutes deux de taille $n = 2^k$ avec $k \geq 0$, et toutes deux ordonnées dans l'ordre croissant*

sortie : *nombre entier positif*

Si $n = 1$

  Sortir: $\frac{L_1(1)+L_2(1)}{2}$

$m \longleftarrow \frac{n}{2}$

Si $L_1(m) > L_2(m)$

  Sortir: $\textbf{mystère}(L_1(1:m), L_2(m+1:n), m)$

Sinon

  Sortir: $\textbf{mystère}(L_1(m+1:n), L_2(1:m), m)$

1. (2 points) Quelle est la sortie de l'algorithme mystère(L1, L2, n) si L1 = (1, 3, 6, 9), L2 = (2, 4, 7, 8) et n = 4 en entrée?
2. (1 point) Est-ce que la sortie de l'algorithme est modifiée si on remplace en entrée la liste L1 par L1 = (1, 3, 6, 355)?
3. (1 point) Quelle est la complexité temporelle de l'algorithme mystère en notation Θ(·)) ? Justifier votre réponse.
4. (4 points) Écrire une version itérative (c'est-à-dire non-récursive) de l'algorithme mystère(L1, L2, n).

EXERCICE 1 — SOLUTION DÉTAILLÉE

Exercice 1 — Solution

1. Regardons l'exécution pas à pas avec L1 = (1, 3, 6, 9), L2 = (2, 4, 7, 8) et n = 4.

Premier appel : mystère((1, 3, 6, 9), (2, 4, 7, 8), 4)

- n = 4 ≠ 1
- m = 4 / 2 = 2
- Comparaison : L1(2) = 3 et L2(2) = 4
- 3 > 4 est Faux → bloc Sinon
- Appel récursif : mystère(L1(3 : 4), L2(1 : 2), 2)

Deuxième appel : mystère((6, 9), (2, 4), 2)

- Nouvelles listes : L1' = (6, 9) et L2' = (2, 4) avec n = 2
- n = 2 ≠ 1
- m = 2 / 2 = 1
- Comparaison : L1'(1) = 6 et L2'(1) = 2
- 6 > 2 est Vrai → bloc Si
- Appel récursif : mystère(L1'(1 : 1), L2'(2 : 2), 1)

Troisième appel (Condition d'arrêt) : mystère((6), (4), 1)

- n = 1 → condition Si n = 1 remplie
- Retour : (L1(1) + L2(1)) / 2 = (6 + 4) / 2 = **5**

Réponse : La sortie de l'algorithme est 5.

EXERCICE 1 — SOLUTION DÉTAILLÉE

Exercice 1 — Solution

> **2. Question : Est-ce que la sortie change si L1 = (1, 3, 6, 9) devient L1 = (1, 3, 6, 355) ?**
>
> **Réponse : Non, la sortie reste 5.**

Explication :

Lors du premier appel, la condition L1(2) > L2(2) détermine quelles moitiés des listes seront conservées.

L'élément à l'indice 4 de L1 (qui passe de 9 à 355) est transmis dans le deuxième appel sous forme de liste L1' = (6, 355).

Cependant, lors du deuxième appel :

- On vérifie L1'(1) > L2'(1), soit 6 > 2 ✓ Vrai
- Cela déclenche l'appel de la **première moitié** de L1'
- Le nombre 355, situé dans la **seconde moitié**, est définitivement ignoré
- Il n'est jamais évalué dans les appels suivants

Conclusion : L'algorithme ne dépend que des éléments aux positions médiales. Les valeurs extrêmes sont éliminées par la dichotomie.

EXERCICE 1 — SOLUTION

Exercice 1 — Solution

> **3. Quelle est la complxité temporelle de l'algorithm mystère ?**

Complexité : Θ(log n)

À chaque étape, on conserve deux moitiés pertinentes. La taille du problème est **divisée par deux** — comportement similaire à une recherche dichotomique.

EXERCICE 1 — CODE

Exercice 1 — Solution

> **4. Proposer une version itérative de l'algorithm mystère.**

Principe

Quatre indices **a, b, c, d** délimitent les fenêtres courantes. À chaque itération, la fenêtre est réduite de moitié jusqu'à convergence.

```
def mystere_iteratif(L1, L2, n):
    a, b = 0, n - 1
    c, d = 0, n - 1
    while a < b:
        m = (b - a + 1) // 2
        i1, i2 = a + m - 1, c + m - 1
        if L1[i1] > L2[i2]:
            b, c = i1, i2 + 1
        else:
            a, d = i1 + 1, i2
    return (L1[a] + L2[c]) / 2
```

EXERCICE 2

Exercice 2 : Chaînes de caractères

Nous souhaitons implémenter un algorithme pour chiffrer un message en mélangeant ses lettres selon la règle suivante :

Soit $m = c_1c_2...c_n$, le message à chiffrer contenant n caractères, alors le message chiffré contient également n caractères et s'obtient en prenant la première lettre de m, puis la dernière, puis la seconde, puis l'avant-dernière, et ainsi de suite jusqu'à ce que toutes les lettres de m soient ajoutées dans le message chiffré.

Exemples :

- Si le message initial est "bonjour", alors le message chiffré est "brounoj"
- Si le message initial est "examen", alors le message chiffré est "enxeam"

⚠ On attire votre attention sur l'importance de la parité du nombre de caractères dans le mot à chiffrer.

---

a) Écrivez une fonction `encrypt(s: str) -> str` qui prend en paramètre le message en clair `s` et le chiffre selon l'algorithme décrit plus haut.

b) Écrivez une fonction `decrypt(s: str) -> str` qui prend en paramètre le message chiffré `s` et le déchiffre en sachant que la méthode de chiffrement est celle décrite plus haut.

EXERCICE 2 — SOLUTION

Exercice 2 - Solution

encrypt(s)

Deux indices **i** (début) et **j** (fin) convergent. Cas `i == j` traité séparément pour les longueurs impaires.

```
def encrypt(s: str) -> str:
    res, i, j = "", 0, len(s)-1
    while i <= j:
        if i == j:
            res += s[i]
        else:
            res += s[i] + s[j]
        i += 1; j -= 1
    return res
```

decrypt(s)

Séparer les indices **pairs** (début) et **impairs** (fin inversée), puis concaténer.

```
def decrypt(s: str) -> str:
    debut, fin = "", ""
    for i in range(len(s)):
        if i % 2 == 0:
            debut += s[i]
        else:
            fin = s[i] + fin
    return debut + fin
```

EXERCICE 3

Exercice 3 : Compression

Soit un message formé de 2^n lettres (avec $n \geq 3$), contenant :

- 2^{n-2} fois la lettre A
- 2^{n-2} fois la lettre B
- 2^{n-3} fois la lettre C
- le reste étant composé d'espaces (qu'on considère ici comme des lettres à part entière)

a) On encode ce message en une suite de 0 et de 1 en utilisant l'algorithme de Huffman. Écrire ci-dessous le dictionnaire obtenu et dessiner l'arbre correspondant.

b) Calculer le nombre moyen de bits par lettre utilisé par cet encodage.

c) En utilisant l'approximation $log_2(3) \simeq 1.6$, calculer (approximativement) l'entropie du message.

EXERCICE 3 — SOLUTION A)

Exercice 3 - Solution

Étape 1 : Exprimer tous les effectifs en fonction de 2^{n-3}

Pour simplifier la construction de l'arbre, on normalise :

- **Lettre C** : $2^{n-3} = 1 \cdot 2^{n-3}$
- **Lettre A** : $2^{n-2} = 2 \cdot 2^{n-3}$
- **Lettre B** : $2^{n-2} = 2 \cdot 2^{n-3}$
- **Espace (E)** : Total = $2^n = 8 \cdot 2^{n-3}$
  - Reste : $8 - (1 + 2 + 2) = \mathbf{3 \cdot 2^{n-3}}$

Résumé :

Symbole	C	A	B	Espace
Effectif	$1 \cdot 2^{n-3}$	$2 \cdot 2^{n-3}$	$2 \cdot 2^{n-3}$	$3 \cdot 2^{n-3}$

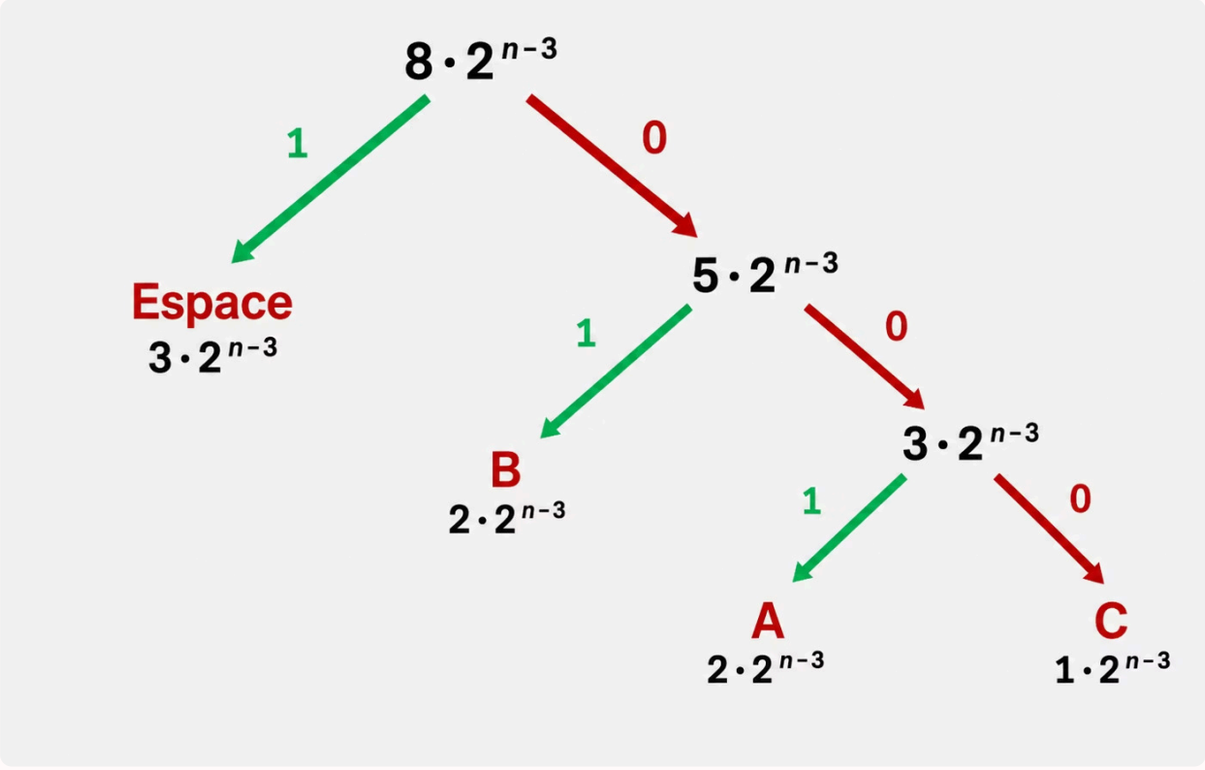
EXERCICE 3 — SOLUTION A)

Construction de l'Arbre de Huffman

Étape 2 : Fusionner les nœuds de plus faible fréquence

Ordre des fusions :

1. **Fusion 1** : C $(1 \cdot 2^{n-3})$ + A $(2 \cdot 2^{n-3}) = 3 \cdot 2^{n-3}$
2. **Fusion 2** : A,C $(3 \cdot 2^{n-3})$ + B $(2 \cdot 2^{n-3}) = 5 \cdot 2^{n-3}$
3. **Fusion 3** : Espace $(3 \cdot 2^{n-3})$ + B,A,C $(5 \cdot 2^{n-3}) = 8 \cdot 2^{n-3}$ (racine)



EXERCICE 3 — SOLUTION A)

Codes Binaires Obtenus

En parcourant l'arbre de la racine à chaque feuille :

Symbole	Nombre de questions	Code
Espace	1	**1**
B	2	**01**
A	3	**001**
C	3	**000**

EXERCICE 3 — SOLUTION B)

Calcul de la Longueur Moyenne

Calcul du nombre moyen de bits par lettre

Tableau récapitulatif :

Symbole	Effectif	# bits	Contribution
Espace	$3 \cdot 2^{n-3}$	1	$3 \cdot 2^{n-3}$
B	$2 \cdot 2^{n-3}$	2	$4 \cdot 2^{n-3}$
A	$2 \cdot 2^{n-3}$	3	$6 \cdot 2^{n-3}$
C	$1 \cdot 2^{n-3}$	3	$3 \cdot 2^{n-3}$

Total des bits :

$$Total_{bits} = 3 + 4 + 6 + 3 = 16 \text{ (en unités de } 2^{n-3})$$

$$Total_{bits} = 16 \cdot 2^{n-3} = 2^4 \cdot 2^{n-3} = 2^{n+1}$$

Longueur moyenne :

$$L_{moyenne} = \frac{2^{n+1}}{2^n} = \mathbf{2} \text{ bits/lettre}$$

EXERCICE 3 — SOLUTION C)

Calcul de l'Entropie

Calcul de l'entropie du message

Espace

$P(\text{Espace}) = \frac{3}{8}$

B

$P(B) = \frac{2}{8} = \frac{1}{4}$

A

$P(A) = \frac{2}{8} = \frac{1}{4}$

C

$P(C) = \frac{1}{8}$

Formule de l'entropie :

$$H = -\sum_{i} P(i) \log_2(P(i))$$

Calcul détaillé :

$$H = -\left[\frac{3}{8}\log_2\left(\frac{3}{8}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{8}\log_2\left(\frac{1}{8}\right)\right]$$

Avec $\log_2(3) \approx 1.6$:

$$H = -\left[\frac{3}{8}(1.6 - 3) + \frac{1}{4}(-2) + \frac{1}{4}(-2) + \frac{1}{8}(-3)\right]$$

$$H = -\left[-0.525 - 1.375\right] = \mathbf{1.9}\ \text{bits/lettre}$$

EXERCICE 4

Exercice 4 : Chiffrement RSA

Contexte :

Vous souhaitez protéger une information que vous envoyez à un ami en utilisant RSA.

Clés disponibles :

Personne	Clé Publique	Clé Privée
Votre ami	(e, n) = (17, 77)	d = 41
Vous	(e, n) = (11, 65)	d = 23

Objectif :

Expliquez précisément chaque étape de la communication en partant du message en clair m jusqu'au moment où votre ami récupère le message en clair.

EXERCICE 4 — SOLUTION

Solution RSA — Chiffrement et Déchiffrement

01

Étape 1 : Chiffrement du message

Vous chiffrez le message m avec la **clé publique de votre ami** $(e, n) = (17, 77)$:

$$c = m^{17} \bmod 77$$

Le message chiffré c est calculé en élevant le message en clair m à la puissance $e = 17$, puis en prenant le reste de la division par $n = 77$.

02

Étape 2 : Transmission du message chiffré

Vous envoyez le message chiffré c à votre ami via un canal de communication (potentiellement non sécurisé).

Sécurité : Même si quelqu'un intercepte c, il ne peut pas retrouver m sans connaître la clé privée $d = 41$ de votre ami.

03

Étape 3 : Déchiffrement du message

Votre ami reçoit le message chiffré c et le déchiffre avec sa **clé privée** $d = 41$:

$$m = c^{41} \bmod 77$$

Il élève le message chiffré c à la puissance $d = 41$, puis prend le reste de la division par $n = 77$.

Résultat : Votre ami retrouve le message en clair m.

Propriété fondamentale de RSA :

$$(m^e)^d \equiv m \pmod{n}$$

C'est cette propriété mathématique qui garantit que le déchiffrement retrouve le message original.