+1/1/60+

EPFL

1

Ens. : O. Lévêque, M. Stojilovic
Information, Calcul, Communication - A
Vendredi 20 décembre 2024
Durée : 180 minutes

U.N.Owen

SCIPER : **0**

Salle : INF 1

Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 12 pages, les dernières pouvant être vides. Ne pas dégrafer.

- Posez votre carte d'étudiant sur la table.
- Document autorisé pour cet examen : un formulaire constitué de deux pages A4 recto-verso, manuscrites, préparées avec stylet+tablette ou à l'ordinateur.
- L'utilisation de tout appareil électronique (calculatrice, ordinateur, smartphone/watch, tablette) est interdite pendant l'épreuve.
- L'examen est composé de deux parties:
  - une partie avec 13 questions à choix multiple valant en tout 32 points ; chaque question admet une seule réponse correcte : la réponse correcte vaut 2 points (pour les 7 premières questions sur la partie théorique) ou 3 points (pour les 6 questions suivantes sur la partie programmation) ; toute autre option (pas de réponse, réponse fausse, ou plusieurs cases cochées) vaut 0 point.
  - une partie avec 5 questions de type ouvert, valant en tout 48 points.
- Merci d'avance de soigner la présentation de vos réponses !
- Si une question est erronée, les enseignants se réservent le droit de l'annuler.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

choisir une réponse | select an answer
Antwort auswählen

ne PAS choisir une réponse | NOT select an answer
NICHT Antwort auswählen

Corriger une réponse | Correct an answer
Antwort korrigieren

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

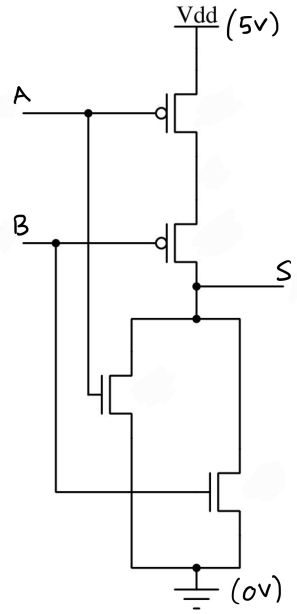
Première partie, questions à choix multiple

Pour chaque question, marquer la case correspondant à la réponse correcte sans faire de ratures. Il n'y a qu'**une seule** réponse correcte par question.

Question 1
Supposons qu'on échantillonne un signal $(X(t),\, t \in \mathbb{R})$ ayant la particularité de ne prendre que des valeurs entières comprises entre 0 et 1'000 à chaque instant t. Avec une fréquence d'échantillonnage de 100 Hz, de combien de bits a-t-on besoin pour enregistrer le signal X sur une durée d'une minute?

- [x] 60'000
- [ ] 6'000
- [ ] 600'000
- [ ] 6'000'0000

Question 2
On considère le circuit ci-dessous. Quelle affirmation est correcte?



- [ ] S = NON (A ET B)
- [ ] Certaines entrées A et B peuvent générer un court-circuit.
- [x] S = NON (A OU B)
- [ ] La sortie S vaut toujours 0.

Question 3
Quelle est l'entropie H de la séquence de 12 lettres GLOUBIBOULGA ?

- [ ] $H = \log_2(12) - \frac{5}{12}$
- [x] $H = \log_2(3) + \frac{7}{6}$
- [ ] $H = \frac{5}{6}\log_2\left(\frac{6}{5}\right) + \frac{1}{6}\log_2(6)$
- [ ] $H = \log_2(3) + \frac{7}{12}$

Question 4
Pour encoder la séquence GLOUBIBOULGA sous forme binaire, on propose les quatre dictionnaires ci-dessous. Lequel utilise le moins de bits par lettre en moyenne, tout en permettent un décodage facile lorsque la séquence de bits est lue de gauche à droite ?

- [ ] G: 111, L: 110, O: 101, U: 100, B: 011, I: 010, A: 00
- [x] G: 111, L: 10, O: 110, U: 01, B: 001, I: 0001, A: 0000
- [ ] G: 11, L: 10, O: 01, U: 000, B: 001, I: 0001, A: 0000
- [ ] G: 111, L: 110, O: 101, U: 100, B: 011, I: 010, A: 001

Question 5
Pour envoyer un message M d'une longueur de n bits à Bob, Alice utilise deux clés secrètes K_1, K_2, chacune d'une longueur de n bits, générées aléatoirement, indépendamment l'une de l'autre et du message M, et partagées au préalable avec Bob. Elle envoie ensuite les trois messages suivants:

$$M \oplus K_1, \quad M \oplus K_2, \quad K_1 \oplus K_2$$

et Eve intercepte ces trois messages. Quelle affirmation ci-dessous est correcte?

- [x] Ce système est sûr: Eve ne peut en aucun cas retrouver le message M envoyé par Alice.
- [ ] Ce système n'est pas sûr car les clés K_1 et K_2 sont des clés à usage unique, qui ne devraient donc pas être réutilisées deux fois pour un envoi.
- [ ] Ce système serait sûr *seulement* si Alice n'envoyait que deux des trois messages ci-dessus.
- [ ] Aucune des trois affirmations ci-dessus n'est correcte.

Question 6
Supposons maintenant qu'étant donné trois messages M, M_1, M_2, chacun d'une longueur de n bits, Alice envoie les trois messages suivants à Bob:

$$M \oplus M_1, \quad M \oplus M_2, \quad M \oplus M_1 \oplus M_2$$

Notez bien que Bob ne connaît aucun des messages M, M_1 ou M_2 à l'avance. Quelle affirmation ci-dessous est correcte?

- [ ] Dans tous les cas, Bob n'est pas capable de retrouver le message M envoyé par Alice.
- [ ] Bob peut retrouver le message M envoyé par Alice, même si un des trois messages est effacé.
- [x] Bob n'est capable de retrouver le message M envoyé par Alice que s'il reçoit les 3 messages.
- [ ] Aucune des trois affirmations ci-dessus n'est correcte.

Question 7
Pour décrypter une clé binaire composée de 40 bits avec une attaque par force brute, 10 serveurs travaillant en parallèle pendant 6 secondes sont nécessaires. Que faudrait-il (environ) pour décrypter une clé binaire composée de 60 bits?

- [x] 10'000 serveurs travaillant en parallèle pendant 100 minutes.
- [ ] 20 serveurs travaillant en parallèle pendant 12 secondes.
- [ ] 100 serveurs travaillant en parallèle pendant 1 minute.
- [ ] 1'000 serveurs travaillant en parallèle pendant 10 minutes.

Question 8

Qu'affiche ce programme ?

```
def addme(x, y):
    global cnt
    cnt += 1

    res.add(x + y)

    if x > 0 and y > 0:
        addme(x - 2, y)
        addme(x, y - 1)

cnt = 0
res = set()
addme(4, 2)
print(cnt, len(res))
```

☐ 7 3 ☐ 11 11 ☐ 5 3 ■ 11 6 ☐ 3 3

Question 9

Qu'affiche ce programme ?

```
def foo(a):
  b = 100
  c = a + b or a - b
  a, b, c = b, c, a
  return a, b, c

a = 10
b = a + 10
c = b + 10

x, y, z = foo(b)
print(y + z, a + b)
```

☐ 120 30 ☐ 220 110 ☐ 140 220 ☐ 40 30 ■ 140 30 ☐ 21 30

Question 10

Qu'affiche ce programme ?

```
i = 1
new_dict = dict.fromkeys(range(6), i)
elements = list(new_dict.keys())

for elem in elements:
    i += 1
    for e in elements[::i]:
        new_dict[elem] -= e

new_dict.popitem()
new_dict[new_dict.pop(2)] = i

print(len(new_dict), set(new_dict.values()))
```

- ☐ 6 {1}
- ■ 5 {1, 7, -5, -4, -2}
- ☐ 6 {1, 7, -5, -4, -2}
- ☐ 5 {1, -5, -4, -3, -2}
- ☐ 6 {1, 7, -5, -4, -3, -2}
- ☐ 6 {1, -5, -4, -3, -2}

Question 11

Quel pourrait être le résultat de ce programme ?

```
lst = []
for i in range(6):
    for j in range(i):
        lst.append(i)
s = set(lst)

s = (s & {4, 5, 6, 7}) - {6, 7}
s = (s | {8}) ^ {4, 9}

print(s)
```

- ☐ {1, 2, 3, 5, 8, 9}
- ■ {8, 5, 9}
- ☐ {5, 6, 7, 8, 9}
- ☐ {9, 4}
- ☐ {5, 8, 6, 7}

Question 12

Qu'affiche ce programme ?

```
def bar(n, r = 1):
    if n == 0:
        return r
    elif n % 2:
        return bar(n - 1, r + n)
    else:
        return bar(n - 1, r + 2 * n)

print(bar(10))
```

- ☐ 56
- ☐ 81
- ☐ 79
- ☐ 85
- ■ 86

Question 13

Qu'affiche ce programme ?

```
def matrix_manipulation(v_in, matrix_in, matrix_out):
    global n
    for i in range(n):
        for j in range(n):
            if i == 2:
                continue
            if j == 1:
                break
            matrix_out[j][0] += matrix_in[j][i] * v_in[i]

n = 3
vector = [1, 2, 3]
matrix_a = [[1, 2, 3], [1, 3, 2], [2, 1, 3]]
matrix_b = [[0] for i in range(3)]

matrix_manipulation(vector, matrix_a, matrix_b)
print(matrix_b)
```

- ☐ `[[3], [0], [0]]`
- ☐ `[[14], [0], [0]]`
- ☐ `[[1], [4], [0]]`
- ■ `[[5], [0], [0]]`
- ☐ `[[5], [7], [4]]`

Deuxième partie, questions de type ouvert

Répondre dans l'espace dédié. Votre réponse doit être soigneusement justifiée, toutes les étapes de votre raisonnement doivent figurer dans votre réponse. Laisser libres les cases à cocher : elles sont réservées aux correcteurs.

Question 14: *Cette question est notée sur 6 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ■6

Pour indiquer à Bob le temps qu'il fait chez elle, Alice utilise le code binaire suivant:

Il fait beau: 00000 Il pleut: 11111 Il neige: 10101 Il vente: 11100 Il fait froid: 00111

a) (3 points) Calculez la distance minimale de ce code binaire, ainsi que le nombre d'effacements / d'erreurs qu'il peut corriger, dans le pire des cas.

Réponse: $d = 2$; c'est la distance entre les mots de code 11111 et 10101, par exemple; toutes les autres distances sont plus grandes ou égales à 2. Donc ce code corrige au plus un effacement, et aucune erreur.

b) (2 points) Supposons qu'on sache qu'*au plus* une erreur (de type $0 \to 1$ ou $1 \to 0$) survienne lors d'une transmission. Est-il toujours possible pour Bob de *détecter* une telle erreur? Justifiez votre réponse.

Réponse: Oui. La distance minimale de code valant 2, si seulement un bit est modifié, il sera toujours possible de détecter une telle erreur.

c) (1 point) Toujours sous la même hypothèse qu'à la question b), si Bob reçoit la séquence 00100, peut-il en déduire le temps qu'il fait chez Alice? Si oui, quel est-il? Si non, expliquez pourquoi.

Réponse: Oui, car le seul mot de code qui diffère de la séquence reçue en seul bit est le mot de code 00000 (correspondant au message "il fait beau").

Note: Ici, il est donc possible de corriger l'erreur (mais on n'est pas dans le pire des cas).

Question 15: *Cette question est notée sur 10 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 ☐6 ☐7 ☐8 ☐9 ■10

Rappel: Pour $a, b \in \mathbb{R}$, $\sin(a) \cdot \sin(b) = \frac{1}{2} \cdot (\cos(a-b) - \cos(a+b))$, $\sin(-a) = -\sin(a)$, $\cos(-a) = \cos(a)$.

a) (2 points) On considère le signal Z suivant:

$$Z(t) = \sin(4\pi t) \cdot \sin(3\pi t) - \sin(5\pi t) \cdot \sin(2\pi t), \quad t \in \mathbb{R}$$

Exprimez Z comme une somme de (co-)sinusoïdes et calculez sa bande passante.

Réponse: En utilisant la formule donnée dans le rappel, on trouve:

$$Z(t) = \frac{1}{2} \cdot (\cos(\pi t) - \cos(7\pi t)) - \frac{1}{2} \cdot (\cos(3\pi t) - \cos(7\pi t)) = \frac{1}{2} \cdot (\cos(\pi t) - \cos(3\pi t))$$

donc la bande passante de Z vaut 1,5 Hz.

b) Pour reconstruire un signal X à partir de sa version échantillonnée, on utilise la formule d'interpolation suivante:

$$X_I(t) = \sum_{m \in \mathcal{Z}} X\left(\frac{m}{2}\right) \cdot \text{sinc}(2t - m), \quad t \in \mathbb{R}$$

où sinc est la fonction "sinus cardinal" vue au cours.

b1) (2 points) Est-il toujours vrai que $X_I(n) = X(n)$ pour tout $n \in \mathbb{Z}$? Justifiez votre réponse.

Réponse: Oui, car il est toujours vrai que pour un nombre entier m, $X_I(mT_e) = X(mT_e)$, et ici $T_e = \frac{1}{2}$, donc

$$X_I(n) = X_I(2nT_e) = X(2nT_e) = X(n) \quad \text{pour tout } n \in \mathbb{Z}$$

b2) (3 points) Quelle condition doit satisfaire la bande passante $f_{\max}$ du signal X pour garantir que $X_I(t) = X(t)$ pour tout $t \in \mathbb{R}$? A nouveau, justifiez votre réponse.

Réponse: Vu que $T_e = \frac{1}{2}$ sec, $f_e = 2$ Hz, et donc la bande passante $f_{\max}$ du signal X doit satisfaire la condition $f_{\max} < \frac{f_e}{2} = 1$ Hz.

c) (3 points) Le signal Z de la question a) ne satisfait malheureusement pas la condition trouvée à la question b2). Expliquez en détail ce qu'il faut faire avec le signal Z pour éviter l'effet stroboscopique lors de sa reconstruction, tout en conservant au mieux celui-ci. Que vaut alors le signal reconstruit?

Réponse: Pour éviter l'effet stroboscopique, il faut, avant d'échantillonner le signal, supprimer la fréquence $f = 1{,}5$ Hz présente dans le signal X avec un filtre passe-bas idéal de fréquence de coupure f_c comprise entre 0,5 Hz et 1,5 Hz (strictement), de manière à conserver l'autre fréquence $f = 0{,}5$ Hz présente dans le signal Z. Le signal reconstruit est alors le même que le signal filtré et vaut

$$\widehat{Z}(t) = \frac{1}{2} \cdot \cos(\pi t)$$

Question 16: *Cette question est notée sur 10 points.*

0 1 2 3 4 5 6 7 8 9 10

a) (3 points) Calculez les trois entropies suivantes:

$H(\text{HONO})$ $H(\text{LULU})$ $H(\text{HONOLULU})$

Réponse: $H(\text{HONO}) = 1,5$ $H(\text{LULU}) = 1$ $H(\text{HONOLULU}) = 2,25$

b) (3 points) Soient X et Y deux séquences composées chacune de n lettres, d'entropies respectives $H(X)$ et $H(Y)$ et telles qu'aucune lettre de la séquence X ne se retrouve dans la séquence Y (comme dans l'exemple ci-dessus avec $X = \text{HONO}$ et $Y = \text{LULU}$).

Soit aussi XY la séquence composée de la juxtaposition des deux séquences X et Y (HONOLULU dans l'exemple ci-dessus). Dans le cas général, exprimez l'entropie $H(XY)$ de la séquence XY en fonction des entropies $H(X)$ et $H(Y)$.

Note: Si vous êtes bloqués sur cette question, un conseil: passez à la suivante (question c) et revenez après.

Réponse: (voir question c) ci-dessous pour une réponse alternative à cette question)

Soient $p_1, \ldots, p_k$ les probabilités d'apparition des k lettres différentes qui forment la séquence X et $q_1, \ldots, q_m$ les probabilités d'apparition des m lettres différentes qui forment la séquence Y. Vu que les deux séquences sont de même longueur (n), les probabilités d'apparition de toutes ces lettres dans la séquence XY sont donc données par

$$\frac{p_1}{2}, \ldots, \frac{p_k}{2}, \frac{q_1}{2}, \ldots, \frac{q_m}{2}$$

et donc l'entropie vaut

$$\begin{aligned} H(XY) &= \sum_{i=1}^{k} \frac{p_i}{2} \log_2\left(\frac{2}{p_i}\right) + \sum_{j=1}^{m} \frac{q_j}{2} \log_2\left(\frac{2}{q_j}\right) \\ &= \frac{1}{2}\left(\log_2(2) + \sum_{i=1}^{k} p_i \log_2\left(\frac{1}{p_i}\right)\right) + \frac{1}{2}\left(\log_2(2) + \sum_{j=1}^{k} q_j \log_2\left(\frac{1}{q_j}\right)\right) \\ &= \frac{1}{2}\left(1 + H(X) + 1 + H(Y)\right) = \frac{H(X) + H(Y)}{2} + 1 \end{aligned}$$

comme on peut le vérifier dans l'exemple HONOLULU de la question a) : $2,25 = \frac{1,5+1}{2} + 1$

c) (4 points) Supposons maintenant que les séquences X et Y de la question b) soient encodées sous forme binaire, chacune avec un dictionnaire sans préfixe.

Proposez une méthode systématique pour obtenir un dictionnaire sans préfixe pour encoder la séquence XY. Illustrez votre méthode avec les séquences HONO, LULU et HONOLULU de la question a).

Note: Cette question peut aussi vous donner un indice sur la formule à obtenir pour la question b).

Réponse: En général, pour obtenir un code sans préfixe pour la séquence XY, on rajoute un 0 au début de chaque mot de code des lettres de la séquence X, et un 1 au début de chaque mot de code des lettres de la séquence Y.

Ainsi par exemple, si pour encoder la séquence HONO, on utilise le dictionnaire $H : 00$, $N : 01$, $O : 1$ et pour la séquence LULU, on utilise le dictionnaire $L : 0$, $U : 1$, alors la méthode décrite ci-dessus donnera le dictionnaire suivant:

$$H : 000, \quad N : 001, \quad O : 01, \quad L : 10, \quad U : 11$$

qui est bien un dictionnaire sans préfixe.

Retour à la question b): Comme les dictionnaires utilisés ci-dessus sont optimaux et que leurs longueurs moyennes correspondent aux entropies des différentes séquences, on trouve que

$$L(C_{XY}) = \frac{1}{2}\left((1 + L(C_X)) + (1 + L(C_Y)\right) = \frac{L(C_X) + L(C_Y)}{2} + 1$$

ce qui nous redonne bien la réponse de la question b).

Question 17: *Cette question est notée sur 11 points.*

0 1 2 3 4 5 6 7 8 9 10 11

Écrivez une fonction Python `aggregate_score()` qui prend comme argument une liste de chaînes de caractères. Chaque chaîne de la liste contient le nom d'un étudiant suivi d'un espace et d'un score représentant les points obtenus. Votre fonction doit agréger les notes correspondant à un même étudiant et renvoyer un dictionnaire contenant la note agrégée de chaque étudiant.

Données d'entrée : Une liste de chaînes de caractères, chaque chaîne étant formatée comme « nom score », où nom est le nom de l'étudiant et score est un nombre entier non négatif. Vous pouvez supposer que les noms sont composés d'un seul mot sans espace et que chaque nom commence par une majuscule. Voici un exemple de liste de notes : `["Tom 5", "Lea 10", "Tom 7", "Lea 15"]`. Étant donné que les noms commencent par une majuscule, la situation dans laquelle le dictionnaire contient, par exemple, les noms `"Tom"` et `"tom"`, ne peut pas se produire.

Résultat : Un dictionnaire dans lequel chaque clé est un nom unique de la liste d'entrée et la valeur correspondante est la somme de tous les scores associés à ce nom. Par exemple, la sortie de l'entrée donnée sera `{'Tom' : 12, 'Lea' : 25}`.

Exemple d'utilisation :

```
scores = ["Tom 5", "Lea 10", "Tom 7", "Lea 15"]
score_dict = aggregate_score(scores)
print(score_dict)
# Output: {'Tom': 12, 'Lea': 25}
```

Réponse:

Question 18: *Cette question est notée sur 11 points.*

0 1 2 3 4 5 6 7 8 9 10 11

On vous donne deux listes, `list1` et `list2`, toutes deux de longueur n, et une fenêtre de taille w. Votre tâche consiste à écrire une fonction Python **récursive** `window_average_recursive()` pour calculer une nouvelle liste dont chaque élément représente la moyenne des valeurs des deux listes d'entrée à l'intérieur d'une fenêtre coulissante de taille w.
Étapes de la résolution :

(a) Pour chaque position i de la fenêtre coulissante ($0 \leq i \leq n - w$) :

- Considerez les sous-sections :

$$\text{window1} = \text{list1}[i], \text{list1}[i+1], \ldots, \text{list1}[i+w-1]$$

et

$$\text{window2} = \text{list2}[i], \text{list2}[i+1], \ldots, \text{list2}[i+w-1].$$

- Calculer les moyennes par élément :

$$\text{element\_average}[j] = \frac{\text{window1}[j] + \text{window2}[j]}{2}, \quad j = 0, 1, \ldots, w-1.$$

- Calculer la moyenne globale de ces valeurs à l'intérieur de la fenêtre :

$$\text{window\_average} = \frac{1}{w} \sum_{j=0}^{w-1} \text{element\_average}[j].$$

- Ajouter window_average à la liste des résultats.

(b) Répéter ce processus pour toutes les positions possibles de la fenêtre coulissante jusqu'à ce que $i = n-w$.

Résultat : La liste résultante aura une longueur de $n-w+1$. Le programme doit renvoyer une liste contenant toutes les moyennes de la fenêtre.
Exemple d'utilisation :

```
list1 = [1, 2, 3, 4, 5]
list2 = [0, 2, 1, 3, 8]
w = 3
result = window_average_recursive(list1, list2, w)
print(result)
# Output: [1.5, 2.5, 4.0]
```

Notez que, si l'implémentation de votre code est entièrement correcte mais ne se présente pas sous la forme d'une fonction récursive, le nombre maximum de points que vous pouvez obtenir pour cette question est 5.

Réponse: