l'Etudiant

BAC 2026
Correction épreuve de NSI
Mardi 16 juin 2026

Exercice 1

PARTIE A

1.

Adresse IP : 10101100.00010000.00100000.00001111
Masque : 11111111.11111111. 11110000.00000000

2. Ce réseau est défini par le masque 240. Cela signifie que les 12 derniers bits peuvent être utilisés. Donc il peut avoir 2^12 = 4096 adresses.
À ce nombre d'adresses, il faut retrancher les adresses réservées. (.0 et .255). Cela fait donc 4094 hôtes possibles.
La première IP utilisable est 172.16.32.1 et la dernière 172.16.47.254.
Pour trouver 47, il faut faire 32 + 15 (les 4 derniers bits à zéro du masque).

3. L'adresse de la passerelle est à modifier en 172.16.4**7**.254.

4. Avec un masque de réseau de 30, il reste 2 bits possibles soit 4 possibilités. Si on enlève la première adresse, celle de réseau (.8) et la dernière de broadcast (.11), il reste 192.168.0.9 et 192.168.0.10.

PARTIE B

5. L'information passe par R2 → R3 → R4 → R5. Il y a donc 3 liens réseau. Son coût est de 3.

6. Il y a une route meilleure avec un saut de moins en passant directement de R2 à R3. Les tables de routage ne sont peut-être pas à jour car le lien est nouveau ou a connu des dysfonctionnements.

7. Sur un réseau, on souhaite maximiser le débit et minimiser le coût.

8. Il faut calculer les coûts des différentes routes pour trouver le plus faible.
Le coût d'un lien ethernet est de $10^8/10^7 = 10$. (10 Mbits/s = $10x10^6=10^7$)
Le coût d'un lien fast-ethernet est de $10^8/10^8 = 1$.
Le coût d'un lien fibre optique est de $10^8/(4,10^9) = 0,025$

Une fois mis sur le graphe, on remarque que le chemin minimisant le coût entre R2 (site 4) et R5 (internet) est le chemin :
R2 → R1 → R3 → R4 → R5 qui totalise un coût de 1,075.

PARTIE C

9. Un chiffrage symétrique permet de déchiffrer si on connaît la phrase secrète. Il faudra donc préférer un chiffrement asymétrique.

10. Le protocole HTTPS est basé sur un chiffrement asymétrique. Le protocole HTTP n'est pas chiffré. Donc HTTPS est plus sécurisé car les messages ne transitent pas en clair.

Exercice 2

PARTIE A

1. Chaque solution diffère d'une autre par le changement d'état d'un jeton. Or, chaque jeton ne peut avoir que deux états. Il y a 9 jetons donc il y a 2^9 configurations possibles.

2. La suite de coups proposée ne permet pas la victoire. Après avoir retourné les deux premières lignes, on se retrouve avec les colonnes 1 et 2 noires. Il faudrait donc inverser la colonne 3 et non la 2.

3. La figure 1 est représentée par l'octet : 001 001 110 soit 78. (2+4+8+64)

4. En python :

```
def representant(liste_position):
    """renvoie un entier sous forme décimale à partir d'une liste de 0 et de 1
    In: liste_position, une liste de 0 et de 1
    Out : un entier sous écriture décimale
    """
    #on va prendre les 0 et les 1 par la fin et les ajouter en multipliant par 2 à la puissance :

    entier = 0
    puissance = 0
    for i in range(len(liste_position)-1,-1,-1): #parcourt la liste à l'envers
        entier += 2**(puissance)*liste_position[i]
        puissance +=1
    return entier
```

5. Avec une compréhension de liste :

```
def xor_etendu(l_x, l_y):
    return [xor(l_x[i],l_y[i]) for i in range(len(l_x)) ]
```

6. Il faut parcourir les valeurs du dictionnaire et appliquer pour chaque coup le xor_etendu.

```
def configurations_suivantes(config):
    config_bin = binaire(config)
    voisins = []
    for coup in coups_possibles.values():
        ... # ligne facultative
        suivant_bin = xor_etendu(config_bin, coup)
        voisins.append(representant(suivant_bin))
    return voisins
```

PARTIE B

7. Si on passe d'une configuration s1 à une configuration s2 et que donc une arête s1 → s2 existe, il suffit de refaire le même coup pour revenir à s1. Donc l'arête s2 → s1 existe aussi, il suffit de refaire le même coup.

8. Le jeu possède 4096 possibilités. Il aurait donc fallu faire une matrice de 4096x4096. = 16 millions de configurations.
Or, on sait qu'il y a 8 coups possibles. Donc 8 voisins. Donc au mieux dans un dictionnaire des voisins : 4096*8 = 32 768 configurations.
C'est bien plus optimal.

Exercice 3

PARTIE A

1. On parle de la racine.

2. Il ne s'agit pas d'un arbre binaire, un nœud peut avoir plus de 2 fils.

3. Liste des atttibuts :
self.contenu : chaîne de caractère
self.sorte : chaîne de caractère
self.arguments : liste
self.nb_likes : entier
self.nb_dislikes : entier

4.

```
def soutenir(self, argument):
"""
Ajoute l'Affirmation `argument` comme argument pour de
l'Affirmation `self`.
Précondition : l'Affirmation ne doit pas avoir déjà été
utilisée, ni comme sujet, ni comme pour, ni comme
contre, ce qu'on vérifie grâce à son attribut `sorte`.
"""
    assert argument.sorte == ""  #chaine vide si non utilisée
    self.arguments.append(argument)
    argument.sorte = "pour"
```

5.

```
  def nb_contre(self):
      nb = 0
      for argument in self.arguments :
          if argument.sorte = "contre"
              nb+=1
      return nb
```

6. La fonction *mystere* est récursive : elle va parcourir tout l'arbre sous la racine.
Elle trouve le maximum de contre argument et à la fin renverra le nombre maximal de contre.

7.

```
def evaluation(self):
    total = self.nb_likes - self.nb_dislikes #on gère les likes et disllikes de l'Affirmation'
    for arg in self.arguments:
        if arg.sorte == "pour":
            total = total + arg.evaluation()
        if arg.sorte == "contre":
            total = total - arg.evaluation()
    return total
```

PARTIE B

8. Un SGBD permet de :
a) sécuriser les accès à la base de données ;
~~b) assurer l'alimentation électrique des serveurs ;~~
c) assurer la persistance des données même en cas de panne matérielle ;
d) gérer les accès en parallèle de plusieurs utilisateurs ;
~~e) assurer des connexions en HTTPS au serveur.~~

9. L'attribut email aurait aussi pu servir de clé primaire.

10.

```
SELECT COUNT(*)
 FROM affirmation
WHERE nb_likes >= 50;
```

11.

```
SELECT affirmation.contenu, affirmation.nb_likes
FROM affirmation
JOIN utilisateur ON affirmation.auteur = utilisateur.pseudo
WHERE utilisateur.prenom = 'Pierre' AND utilisateur.nom = 'Durand';
```

12.
Cette requête permet de récupérer le contenu d'une affirmation et le contenu de sa réponse si la réponse a reçu au moins le double de l'affirmation de départ et que c'est un contre-argument.

13.

```
INSERT INTO reaction
VALUES (108, « i<3descartes », « like ») ;
UPDATE affirmation SET nb_likes = nb_likes + 1 WHERE id_aff = 108;
```

14.
Le SGBD va empêcher la suppression de l'utilisateur car son pseudo est une clé étrangère de la table réaction. Il faut donc d'abord vider toutes ses réactions.

15.

```
UPDATE affirmation
SET auteur = NULL
WHERE auteur = « i<3rgpd » ;

DELETE FROM reaction WHERE utilisateur = 'i<3rgpd';

DELETE FROM utilisateur WHERE pseudo = 'i<3rgpd';
```