Correction 26-NSIJ1AN1

Exercice 1

Partie A

1.
```
Adresse IP : 10101100.00010000.00100000.00001111
Masque     : 11111111.11111111.11110000.00000000
```

2. D'après le masque donné, il y a 12 bits réservées à la partie machine dans les adresses IP de ce réseau. On peut donc représenter $2^{\{12\}} = 4096$ adresses différentes, mais comme l'adresse de réseau et l'adresse de diffusion ne sont pas attribuables, il ne reste que $4096 - 2 = 4094$ adresses disponibles attribuables. La première adresse attribuable sera donc $10101100.00010000.00100000.00000001$, soit $172.16.32.1$, et la dernière adresse attribuable sera $10101100.00010000.00101111.11111110$, soit $172.16.47.254$.

3. Le paramètre mal configuré est la passerelle (gateway). Comme vu à la question précédente, elle devrait être à $172.16.47.254$.

4. Vu la notation CIDR du masque réseau, il n'y a que $2^{32-30} - 2 = 2$ adresses disponibles pour les hôtes. Le dernier octet, à 8 en décimal, s'écrit 00001000 en binaire, et les adresses attribuables auront pour dernier octet, en binaire, les valeurs 00001001 et 00001010, soit les adresses : $192.168.0.9$ et $192.168.0.10$.

Partie B

5. Le chemin pris est $R2 \to R3 \to R4 \to R5$, de coût 3.

6. Comme le réseau utilise le protocole RIP et minimise le nombre de sauts, le chemin pris devrait être $R2 \to R4 \to R5$: celui proposé est trop long. Le dysfonctionnement possible doit se trouver au niveau de la liaison entre les routeurs $R2$ et $R4$, qui ne doit pas être correctement configurée.

7. Dans un réseau informatique, on cherche à minimiser le $cout$ et maximiser le $debit$.

8. Le chemin suivit serait $R2 \to R1 \to R3 \to R4 \to R5$, pour un coût total de $\frac{10^8}{100.10^6} + \frac{10^8}{4.10^9} + \frac{10^8}{4.10^9} + \frac{10^8}{4.10^9} = 1 + \frac{3}{40} = \frac{43}{40} \approx 1,075$.

Partie C

9. Si Alice et Bob ne se sont pas mis d'accord en dehors du réseau sur une clé secrète, alors il faut utiliser un chiffrement asymétrique.

   Une autre possibilité est d'utiliser un protocole asymétrique pour échanger une clé secrète, puis utiliser un protocole symétrique pour chiffrer les messages (en général plus efficace).

10. Le protocole HTTPS a une couche de sécurité supplémentaire par rapport au protocole HTTP, qui est le chiffrement des données échangées entre le client et le serveur.

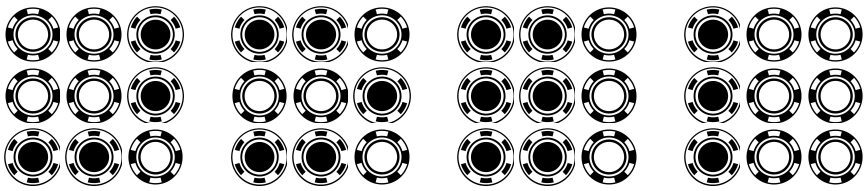
    D'autres protocoles plus adaptés à des échanges entre deux personnes existent.

Exercice 2

Partie A

1. Chaque jeton possède 2 faces, et il y a 9 jetons, donc le nombre de configurations possibles est 2^9 (on peut assimiler un jeton à un bit, et une configuration à une suite de 9 bits).

2. Non, car on aura les configurations successives suivantes (il fallait terminer par la colonne 3 et non la 2 pour gagner) :



3. La liste associée à la figure 1 est `[0, 0, 1, 0, 0, 1, 1, 1, 0]`, soit, en décimal : $0+0+64+0+0+8+4+2+0=78$.

4.
```python
def representant(lst):
    decimal = 0
    for b in lst:
        decimal = 2*decimal + b
    return decimal
# ou
# def representant(lst):
#     n = len(lst)
#     decimal = 0
#     for i in range(n):
#         decimal += lst[i] * (2 ** (n - i - 1))
#     return decimal
```

5.
```python
def xor_etendu(l_x, l_y):
    res = []
    for i in range(len(l_x)):
        res.append(xor(l_x[i], l_y[i]))
    return res
```

6.
```python
def configurations_suivantes(config):
    config_bin = binaire(config)
    voisins = []
    for coup in coups_possibles.values():
        voisin = xor_etendu(config_bin, coup)
        voisins.append(representant(voisin))
    return voisins
```

Partie B

7. Soit x et y deux configurations telles que $x \to y$. Alors, par définition, il existe un coup c qui permet de passer de x à y. On remarque alors que le coup c permet de passer de y à x, donc $y \to x$.

8. D'après la question 1, le graphe possède 2^9 noeuds. Vu la figure 3, chaque noeud possède 8 arrêtes.
   - Avec une représentation par dictionnaire, il y a donc besoin de stoker 2^9 clés et 8 valeurs dans chaque liste associée à chaque clé, soit $2^9 * 8 = 2^{12}$ nombres dans les listes, et donc, en tout, $2^9 + 2^{12}$ nombres.
   - Avec une représentation par matrice d'adjacence, il y a besoin de stoker $2^9 * 2^9 = 2^{18}$ nombres.

   Il semble donc bien plus efficace de représenter le graphe par dictionnaire.

9. ```python
def parcours_profondeur(graphe, configuration, vus):
    vus.append(configuration)
    for s in graphe[configuration]:
        if s not in vus:
            parcours_profondeur(graphe, s, vus)
```

10. La configuration gagnante correspond à `[1, 1, 1, 1, 1, 1, 1, 1, 1]` (*on pouvait calculer à la main* $255 + 256 = 2^9 - 1 = 511$), donc on peut écrire (*on peut remplacer le assert par un print, par exemple*):

```python
vus = []
parcours_profondeur(graphe, representant([1, 1, 1, 1, 1, 1, 1, 1, 1]), vus)
assert 4 not in vus
# ou
# vus = []
# parcours_profondeur(graphe, 4, vus)
# assert representant([1, 1, 1, 1, 1, 1, 1, 1, 1]) not in vus
```

11. Pour chercher à minimiser le nombre de coups pour passer d'une configuration à la configuration gagnante, il faut utiliser une parcours en largeur. En effet, un parcours en profondeur ne va pas chercher le chemin le plus court, alors qu'un parcours en largeur explore toutes les configurations à une distance donnée avant de passer à la distance suivante, garantissant ainsi de trouver le chemin le plus court vers la configuration gagnante.

Exercice 3

Partie A

1. C'est la racine.
2. Non, ce n'est pas un arbre binaire, car un noeud peut avoir 3 enfants et plus. Par exemple, la racine dans la figure 1 possède 4 enfants.
3. Les différents attributs sont :
   - contenu (type `str`),
   - sorte (type `str`),
   - arguments (type `list`),
   - nb_likes (type `int`),
   - nb_dislikes (type `int`).
4. ```python
def soutenir(self, argument):
    """
    Ajoute l'Affirmation `argument`comme argument pour de l'Affirmation `self`.

    Précondition : l'Affirmation ne doit pas avoir déjà été utilisé, ni comme sujet, ni comme pour, ni comme contre, ce qu'on vérifie grâce à son attribut `sorte`.
    """
    assert argument.sorte == "", "L'argument a déjà été utilisé dans un arbre."
    self.arguments.append(argument)
    argument.sorte = "pour"
```
5. ```python
def nb_contre(self):
    n = 0
    for argument in self.arguments:
        if argument.sorte == "contre":
            n += 1
    return n
```
```

6. La méthode `mystère` retourne le nombre d'arguments "contre" maximal contre une seule Affirmation dans l'arbre d'arguments de l'Affirmation `self`.

7.
```
def evaluation(self):
    total = self.nb_likes - self.nb_dislikes
    for arg in self.arguments:
        if arg.sorte == "pour":
            total = total + arg.evaluation()
        if arg.sorte == "contre": # ou else:
            total = total - arg.evaluation()
    return total
```

Partie B

8. Les propositions a), c) et d) font partie des rôles d'un SGBD.

9. L'attribut `email` aurait également pu servir de clé primaire, en supposant qu'un compte mail n'est utilisé que par une personne.

10.
```
SELECT COUNT(*) FROM affirmation
WHERE nb_likes >= 50;;
```

11.
```
SELECT contenu, nb_likes FROM affirmation
JOIN utilisateur ON auteur = pseudo
WHERE prenom = 'Pierre' AND nom = 'Durand' and sorte = 'sujet';;
```

12. (*coquille : rep.aff_repondue à la place de rep.affirmation_repondue et aff.id_aff à la place de aff.id_affirmation*) La requête renvoie le contenu d'une affirmation et de sa réponse ("contre") lorsque la réponse a au moins 2 fois plus de like que l'affirmation.

13.
```
INSERT INTO reaction
VALUES (108, 'i<3descartes', 'like');;

UPDATE affirmation
SET nb_likes = nb_likes + 1
WHERE id_aff = 108;;
```

14. Si on supprime l'utilisateur avant ses réactions, s'il en a fait, le SGBD va renvoyer une erreur, car une valeur de la clé étrangère `utilisateur` de la table `reaction` ne correspondra plus à une valeur de la clé primaire `pseudo` de la table `utilisateur` : c'est la contrainte d'intégrité référentielle. Il faut donc supprimer les réactions de l'utilisateur avant de supprimer l'utilisateur lui-même.

15.
```
UPDATE affirmation
SET auteur = NULL
WHERE auteur = 'i<3rgpd';;

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

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

Sous licence CC BY-NC-SA 4.0

Antoine DEQUAY