Métropole – 2025 – sujet1 - Correction

Exercice 3 (8 points)

Partie A – Adressage IP

1. Adresses IP valides

Adresses IP valides : `192.168.20.2` et `192.168.20.157`.

`192.168.20.261` est invalide (octet > 255) et `192.168.24.10` n'appartient pas au réseau `192.168.20.0/24`.

2. Adresse de diffusion : `192.168.20.255`

3. Nombre de machines

Il y a donc 256 valeurs possibles. Mais il faut enlever l'adresse du réseau (0), l'adresse de broadcast (255), l'adresse du routeur et les trois autres. Il reste donc **250** possibilités.

4. Longueur du masque sous-réseau

8 (2^3) adresses nécessitent 3 bits. Le masque peut donc utiliser 32 − 3 = **29 bits.**

Partie B – Protocole RIP

5.

Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
192.168.30.0	172.16.4.1	172.16.4.2	1
172.16.1.0	172.16.3.1	172.6.3.2	1

6.

Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
192.168.10.0	172.16.4.1	172.16.4.2	2

7.

Réseau destination	Interface de sortie	Prochain routeur	Nombre de sauts
autre	172.16.3.1	172.16.3.2	

Partie C – Protocole OSPF

8. Fast Ethernet (100 Mbit/s) : coût 10 ; Fibre optique (1 Gbit/s) : coût 1.
9. La route de coût minimal est 1-2-3-4 avec un coût de 21

Partie D – Arbres binaires de recherche et Python

10. ip_bin(`'192.168.20.12'`) = `11000000.10101000.00010100.00001100`

11. La dernière instruction return de `precede` est exécutée lorsque les deux adresses sont identiques.
12.
```
def precede ( a, b ) :
    for i in range ( len ( a ) ) :
        if a[i] < b[i] :
            return True
        if a[i] > b[i] :
            return False
    return False
```

13.
- Attribut possible de la classe Abr : `adresse_ip`.
- Méthode possible : `est_vide`

14. `return self.adresse_ip == ''`

15. L'accès aux éléments d'un arbre binaire de recherche est rapide (si il est équilibré), il se fait en O(log (n)). Dans un tableau, la recherche est linéaire en O(n).

16. **Fonction `modifie`**

```
def modifie(self, adresse_ip, interface, passerelle, cout):
    if self.est_vide():
        self.gauche = Abr('','','',0)
        self.droite = Abr('','','',0)
    self.adresse_ip = adresse_ip
    self.interface = interface
    self.passerelle = passerelle
    self.cout = cout
```

17. **Ligne 35**

```
    elif precede(ip_bin(adresse_ip), ip_bin(self.adresse_ip)):
```