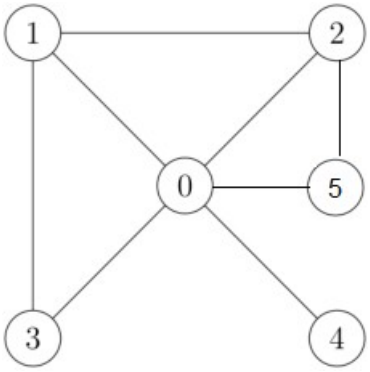
Proposition de correction

Exercice 1

Q1

```
voisins = [[1, 2, 3, 4],
    [0, 2, 3],
    [0, 1],
    [0, 1],
    [0]]
```

Q2



Q3

```
voisins = [[1, 2, 3, 4, 5],
    [0, 2, 3],
    [0, 1],
    [0, 1, 5],
    [0],
    [0, 2]]
```

Q4

la fonction s'appelle elle-même

Q5

la fonction simule un déplacement aléatoire d'un sommet à un autre dans une liste de voisins

- i : Le sommet de départ du graphe
- n : Le nombre de déplacements à effectuer

Q6

```
def simule(voisins : list, i : int, n_tests : int, n_pas : int) -> list:
    """
    simule le déplacement d'un virus
    @param i -- sommet de départ dans la liste voisins
    @param n_tests -- nombre de tests à effectuer
    @param n_pas -- nombre d'étapes, démarrant au sommet i
    @return une liste contenant en position j
            le nombre de fois que le virus a terminé son parcours au sommet j / n_tests
    """
    results = [0] * len(voisins)

    for _ in range(n_tests):
        j = marche_alea(voisins, i, n_pas)
        results[j] += 1

    return [total / n_tests for total in results]
```

Q8

L'ordinateur 0 a la fréquence la plus élevée avec 32.8% de chances d'être le point final du virus après 1000 pas

Q9

```
fonction temps_propagation(voisins : tableau d'entiers, s : entier) : entier
début
    sommet : entier
    pas : entier := 0
    visités : tableau d'entiers := [s]
    tant que ( taille(voisins) <> taille(visités) ) faire
            sommet := marche_alea(voisins, s, 1)
            si ( sommet n'est pas dans visités ) alors
                    visités := visités + [sommet]
            pas := pas + 1
    renvoyer pas
fin
```

Exercice 2

Partie A

Q1

255.255.0.0

Q2

172.16.0.0/16

Q3

172.16.255.255/16

Q4

n = 2^(32-16) – 2 = 2^16 – 2 = 65534

Partie B

Q5

L1 → A → H → D → L2

Q6

L1 → A → B → C → D → L2

Q7

routeur H

Routeur	Réseau destinataire	Passerelle	Interface
H	L2	53.10.10.10	53.10.10.9

Q8

- 100 Mbit/s : c = 10
- 1 Gbit/s : c = 1
- 10 Gbit/s : c = 0,1

Q9

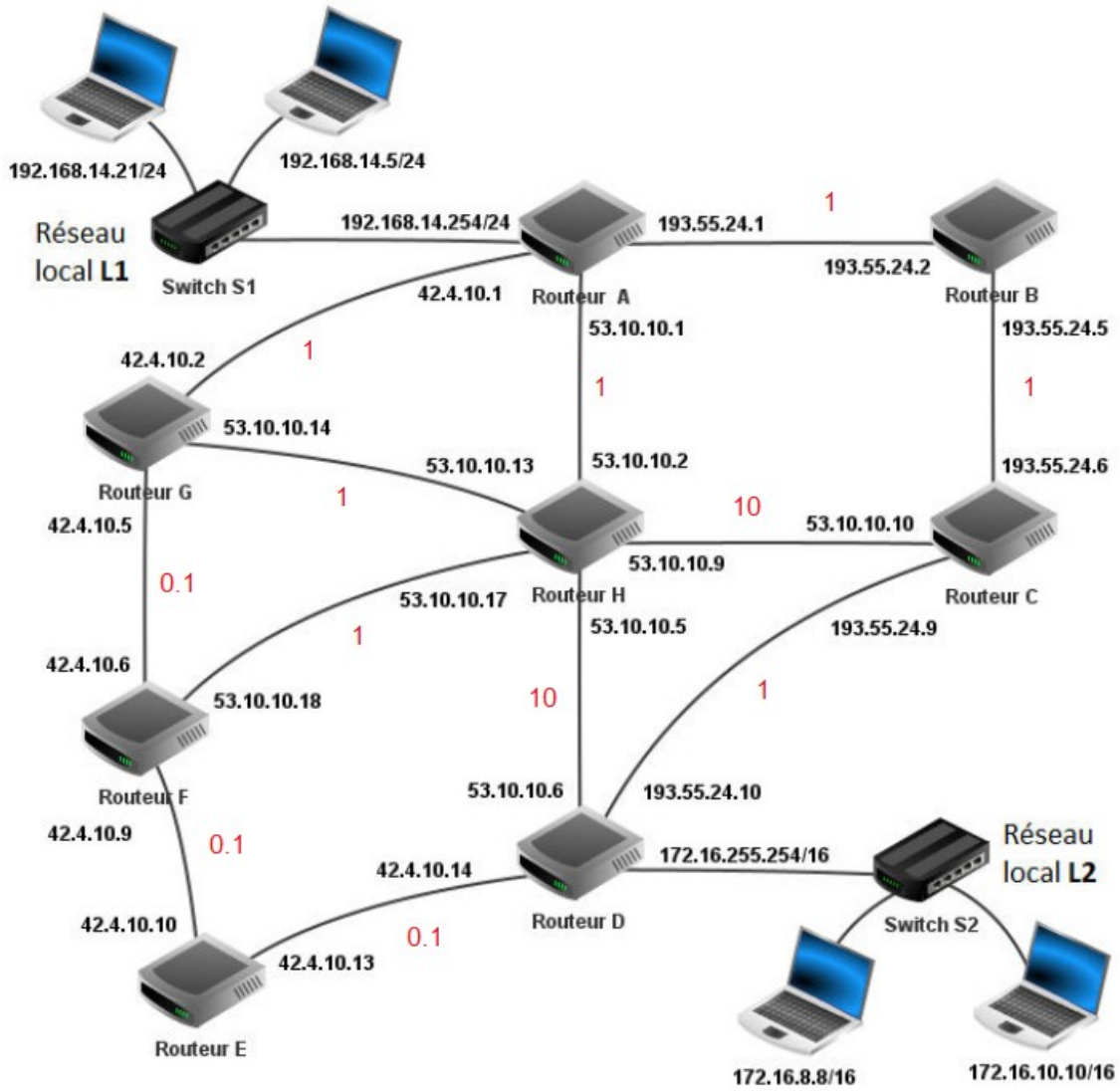
L1 → A → G → F → E → D → L2

c = 1 + 0,1 + 0,1 + 0,1 = 1,3

Q10

L1 → A → H → F → E → D → L2

c = 1 + 1 + 0,1 + 0,1 = 2,2



Exercice 3

Partie A

Q1

- meilleure intégrité des données
- performances optimisées pour les requêtes complexes

Q2

l'unicité

Q3

mettre en relation les tables Client et Emplacement

Q4

Emplacement(id_emplacement : INTEGER, nom : TEXT, localisation : TEXT, tarif_journalier : REAL)

Q5

1 myrtille A4

4 mandarine B1

6 melon A2

Q6

SELECT nom, prenom

FROM Client

WHERE ville = ‘Strasbourg’

ORDER BY nom, prenom

Q7

INSERT INTO Client

VALUES(42, ‘CODD’, ‘Edgar’, ‘28 rue des Capucines’, ‘Lyon’, ‘France’, ‘0555555555’)

Q8

SELECT Client.nom, Client.prenom, Reservation.nombre_personne, Reservation.date_arrivee, Reservation.date_depart, Emplacement.tarif_journalier

FROM Client, Reservation, Emplacement

WHERE Reservation.id_reservation = 18

AND Emplacement.id_emplacement = Reservation.id_emplacement

AND Client.id_client = Reservation.id_client

Partie B

Q9

fait référence à l'instance actuelle de la classe

Q10

client01 = Client(‘CODD’, ‘Edgar’, ‘28 rue des Capucines’, ‘Lyon’, ‘France’, ‘0555555555’)

Q11

```
def montant_a_regler(triplet):
    """ renvoie le montant en euros
    à régler pour cette réservation """
    client, reservation, emplacement = triplet
    return emplacement.tarif_journalier *  reservation.nb_jours() + \
                reservation.nombre_personne * 2.20 *  reservation.nb_jours()
```

Q12

comparaison entre chaînes de caractères et entiers

Q13

```
if not(len(annee) == 4) or not(2018 <= int(annee) <= 2024):
```

Q14

```
def facture_est_valide(chaine : str) -> bool :
    """renvoie vrai si chaine est une chaîne de
    caractères conforme au modèle de facture"""
    partie = separe(chaine)
    if not(len(partie) == 3):
            return False
    annee, mois, numero = partie[0], partie[1], partie[2]
    if not(que_des_chiffres(annee)):
            return False
    if not(len(annee) == 4) or not(2018 <= int(annee) <= 2024):
            return False
    if mois not in calendrier :
            return False
    if not(len(numero) == 3) and not(que_des_chiffres(numero)):
            return False

    return True
```