Épreuves écrites NSI
Corrigé du sujet
Métropole 1 2023 Jour 1

Exercice 2

1. Même si une liaison est coupée, un paquet de données pourra être transmis depuis tout site vers tout autre site en utilisant les autres liaisons.

2. En suivant les tables de routage, on voit que le paquet suivra le chemin suivant :

R2 - R3 - R4 - R5

3. On aura la table de routage suivante pour le routeur R1 :

Routeur R1 (RIP)		
Destination	Suivant	Nombre de sauts
SiteA	Local	0
SiteB	R2	1
SiteC	R2	2
SiteD	R3	2

4. Selon le protocole RIP, le fait d'emprunter la liaison 2 a le même poids que d'emprunter une autre liaison. Ceci ne tient donc pas compte du fait que le débit est faible sur cette liaison. Ainsi selon ce protocole, pour aller du siteA au siteC on suivra le chemin R1 - R2 - R3 et donc on utilisera la liaison 2 malgré son faible débit.

5. a. La liaison dont le débit est le plus faible est celui pour lequel le coût est le plus élevé. Il s'agit donc de la liaison 2.

b. Les chemins possibles pour aller de SiteA à SiteC sont :
* R1 - R2 - R5 ; le coût est de $1\,000\,000 + 100\,000 = 1\,100\,000$
* R1 - R2 - R3 - R4 - R5 ; le coût est de $100\,000 + 5 + 5 + 10 = 100\,020$
* R1 - R3 - R2 - R5 ; le coût est de $50\,000 + 5 + 1\,000\,000 = 1\,050\,005$
* R1 - R3 - R4 - R5 ; le coût est de $50\,000 + 5 + 10 = 50\,015$

c. On aura la table de routage suivante pour le routeur R1 :

Routeur R1 (OSPF)		
Destination	Suivant	Coût total
SiteA	Local	0
SiteB	R3	$50\,005$
SiteC	R3	$50\,015$
SiteD	R3	$50\,005$

Exercice 3

Partie 1

1. `nom`, `tab_voisines`, `tab_couleurs_disponibles` et `couleur_attribue` sont des attributs de la classe `Region`.

2. Le paramètre `nom_region` est de type `str` : c'est une chaîne de caractère.

3.

```
ge = Region("Grand Est")
```

4.

```
def renvoie_premiere_couleur_disponible(self) :
    """..."""
    return self.tab_couleurs_disponibles[0]
```

5.

```
def renvoie_nb_voisines(self) :
    """..."""
    return len(self.tab_voisines)
```

6.

```
def est_coloriee(self) :
    """..."""
    return self.couleur_attribuee != None
```

7.

```
def retire_couleur(self):
    """..."""
    if couleur in self.tab_couleurs_disponibles :
        self.tab_couleurs_disponibles.remove(couleur)
```

8.

```
def est_voisine(self, region)
    """..."""
    return region in self.tab_voisines
```

Partie 2

9.

```
def renvoie_tab_regions_non_coloriees(self):
    """..."""
    res = []
    for reg in self.tab_regions :
        if not reg.est_coloriee() :
            res.append(reg)
    return res
```

ou bien :

```
def renvoie_tab_regions_non_coloriees(self):
    """..."""
    return [reg for reg in self.tab_regions \
           if not reg.est_coloriee()]
```

10. a. La méthode renvoie `None` si et seulement si l'instance sur laquelle elle est appelée ne possède pas de régions non coloriées, c'est-à-dire si l'appel de méthode `self.tab_regions_non_coloriees()` renvoie un tableau vide.

b. La région renvoyée est non coloriée et comporte un maximum de régions voisines parmi les régions non coloriées.

11.

```
def colorie(self) :
    reg = self.renvoie_max() :
    while reg != None :
        coul = reg.renvoie_premiere_couleur_disponible()
        reg.couleur_attribuee = coul
        for reg_vois in reg.tab_voisines() :
             reg_vois.retire_couleur(coul)
        reg = self.renvoie_max()
```