Épreuves écrites NSI
Corrigé du sujet
Centre étranger 2023 Jour 1

Exercice 2

1. a.

* La conversion de 2 en binaire sur un octet est $0000\ 0010$ car $2 = 2^1$

* La conversion de 13 en binaire sur un octet est $0000\ 1101$ car

$$13 = 8 + 4 + 1 = 2^3 + 2^2 + 2^0$$

Donc la conversion de l'adresse en binaire est :

$$10100100.1011000.\mathbf{00000010}.\mathbf{00001101}$$

b. La partie réseau est exprimée par les 24 bits de poids fort, c'est-à-dire par les 3 premiers octets. L'adresse de réseau est donc $164.178.2.0/24$.

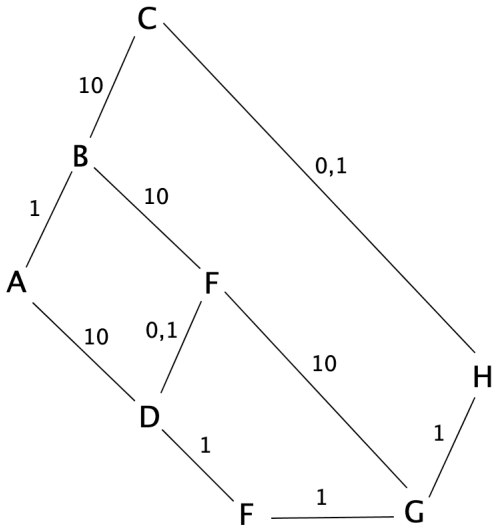
2. Les chemins optimaux (c'est-à-dire ceux qui minimisent le nombre de sauts) pour aller du routeur A au routeur G sont les suivants :

* A - B - E - G

* A - D - E - G

* A - D - F - G

3. a. On a le schéma de réseau suivant :



b. Le chemin de parcours pour un paquet partant de A pour aller en G qui respecte le protocole OSPF est celui qui minimise le coût. Il s'agit donc du chemin

A - D - F - G

avec un coût de 12.

c. Si le routeur F est en panne le chemin qui respecte le protocole OSPF sera alors

A - B - C - H - G

avec un coût de 12,1.

Exercice 3

1. On doit compléter le code de la façon suivante :

```
def ajout(f):
    couleurs =("bleu", "rouge", "jaune", vert")
    indice = randint(0, 3)
    enfiler(f, couleurs[indice])
    return f
```

2. On peut écrire le code suivant :

```
def vider(f):
    while not est_vide(f):
        defiler(f)
```

3. On doit compléter le code de la façon suivante :

```
def affich_seq (sequence):
    stock = creer_file_vide()
    ajout(sequence)
    while not est_vide(sequence) :
        c = defiler(sequence)
        enfiler(stock, c)
        time.sleep(0.5)
        affichage(c)
    while not est_vide(stock):
        enfiler(sequence, defiler(stock))
```

4. a. On doit compléter le code de la façon suivante :

```
def tour_de_jeu(sequence):
    affich_seq(sequence) # zone A
    stock = creer_file_vide()
    while not est_vide(sequence) :
        c_joueur = saisie_joueur()
        c_seq = defiler(sequence) # zone B
        if c_joueur  == c_seq :
            enfiler(stock, c_seq) # zone C
        else :
            vider(sequence) # zone D
    while not est_vide(stock) : # zone E
        enfiler(sequence, defiler(stock)) # zone F
```

b. On peut modifier le code de la façon suivante :

• 1ère instruction de la fonction (avant la ligne 2)

```
    gagne = True
```

• substituer aux lignes 7 à 9 le bloc suivant :

```
        if c_joueur  == c_seq :
            enfiler(stock, c_seq)
        else :
            vider(sequence)
            gagne = False
```

• substituer aux lignes 11 et 12 le bloc suivant :

```
    if gagne :
        while not est_vide(stock) : # zone E
                enfiler(sequence, defiler(stock))
    tour_de_jeu(sequence)
```