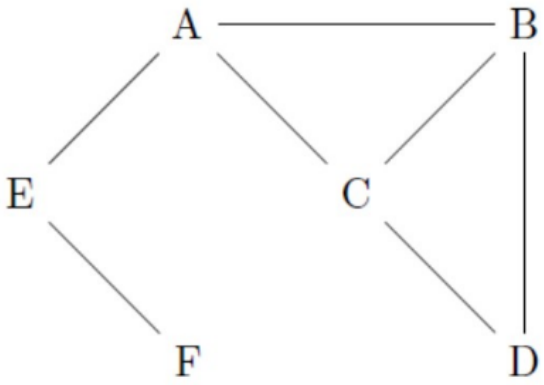
Travaux pratiques - Protocoles de routage

Implémentation du protocole RIP en python

Pour simuler un réseau appliquant le protocole RIP en python, nous allons d'abord utiliser la programmation objet pour créer des objets routeurs. Puis nous implémenterons l'algorithme du protocole qui permettra aux routeurs d'échanger les informations sur leurs routes.

Exercice 1 Création des routeurs et du réseau

1. Créer une classe `Routeur`. Chaque instance doit être initialisée par les attributs suivants :
   - `nom` : chaîne de caractères correspondant au nom du routeur donné lors de l'instanciation.
   - `voisins` : liste vide qui contiendra les voisins immédiats de chaque routeur.
   - `table` : dictionnaire dont les clés sont les noms des routeurs destination. La valeur de chaque clé est un dictionnaire de deux clés. Une clé `passerelle` de valeur le nom du routeur passerelle et une clé `metrique` de valeur la métrique de la route correspondante.
     L'attribut est initialisé avec {`nom:{'passerelle':nom, 'metrique':0}`}.
2. Dans le programme principal, instancier les six routeurs du réseau ci-contre. Leurs noms seront tout simplement les caractères 'A', 'B', 'C', 'D', 'E', 'F'.



3. Une remière étape consiste à initialiser la table de routage d'un routeur avec ses plus proches voisins uniquement. Pour chaque voisin, une route sera ajoutée dans la table de routage avec une métrique de 1.
   (a) Donner, par écrit, le contenu des tables de routage des routeurs A, B, C D, E et F à l'issue de cette initialisation.
   (b) Dans la classe `Routeur`, ajouter une méthode `initialisation`. Dans un premier temps, cette méthode ne fera rien (on dit que la méthode est "bouchonnée").
   (c) **Test Driven Design** :
   Ecrire un programme de test qui, pour chaque routeur, compare le résultat de la fonction `initialisation` avec les résulats attendus (ceux de la question 3.a).
   - Tant que la méthode `initialisation` est incomplète, le test devra retourner la valeur `False`.
   - Si tout se passe bien le test retournera la valeur `True` lorsque la fonction sera complétée.

   (d) Completer la méthode `initialisation` afin qu'elle change l'attribut `voisins` par la liste d'objets, et qu'elle ajoute dans `table` les routes vers les routeurs voisins avec une métrique de 1.
   Vous pouvez lancer le programme de test en cours de dévelopement(le test retourne `False` tant que la fonction n'est pas complète).

4. Si test de la méthode `initialisation` sont bons (tous les tests retpirnent `True`, alors, dans le programme principal, initialiser les six routeurs du réseau avec leurs voisins respectifs.
5. Regrouper l'ensemble des routeurs dans une liste d'objet `Routeur` nommée reseau.
6. Tester l'ensemble en affichant les noms des six routeurs à l'aide d'une boucle. On affichera de plus les noms de leurs voisins respectifs avec une seconde boucle imbriquée dans la première.
7. Créer une méthode `affiche_table` qui affiche la table de routage du routeur de la manière suivante.
   Tester la méthode sur un routeur puis sur l'ensemble du réseau.

```
Table de A
  pour A donne a A , metrique = 0
  pour B donne a B , metrique = 1
  pour C donne a C , metrique = 1
  pour E donne a E , metrique = 1
```

Exercice 2 Échange des données entre les routeurs

Dans cette partie nous allons nous focaliser sur l'implémentation de la méthode `mise_a_jour_table` qui permettra aux routeurs d'échanger les informations sur les routes.

Chaque routeur devra :

- récupérer les informations sur les routes connues par ses voisins immédiats
- s'il découvre une route vers une destination inconnue, l'inscrire dans sa table
- s'il découvre une route connue plus courte, la remplacer dans sa table

1. Ajouter la méthode `donne_table` qui retourne une copie de la table du routeur.

```
def donne_table(self):
        t = copy.deepcopy(self.table)
        return t
```

   On utilise pour cela la méthode deepcopy du module `copy`.
   Tester cette méthode et expliquer pourquoi il st nécessaire de retourner une copie de la table et non la table elle-même.
2. Donner, par écrit, le contenu des tables de routage des routeurs A, B, C D, E et F àprès la mise à jour des tables en fonction des tables de routage des voisins.
3. Dans la classe `Routeur`, ajouter une méthode `mise_a_jour_table`. Dans un premier temps, cette méthode ne fera rien (on dit que la méthode est "bouchonnée").
4. **Test Driven Design** :
   Ecrire un programme de test qui, pour chaque routeur, compare le résultat de la fonction `mise_a_jour_` avec les résultats attendus (ceux de la question 3.a).
   - Tant que la méthode `mise_a_jour_table` est incomplète, le test devra retourner la valeur `False`.
   - Si tout se passe bien le test retournera la valeur `True` lorsque la fonction sera complétée.

   Pour construire la méthode `mise_a_jour_table` il va d'abord falloir récupérer une copie de la table de chaque routeur et incrémenter ses métriques de 1.

5. Définir la méthode et une fonction `incremente_metrique(table)` à l'intérieur. La fonction retournera la table modifiée. On réalise ce qu'on appelle une fermeture. Tester la fonction sur la propre table du routeur.
6. Compléter la méthode à l'aide du boucle incrémentant les copies des tables de chacun des voisins du routeur. La tester.

```
def mise_a_jour_table(self):
       modif = False
       def incremente_metrique(t):
            for cle, valeur in tmp.items():
               valeur["metrique"]= # ACOMPLETER
       for r in self.voisins:
           tmp = #A COMPLETER
           incremente_metrique(tmp)
           print(r.nom)
           ...
           # A COMPLETER
           ...
       return modif
```

7. Finir la méthode avec une seconde boucle imbriquée appliquant le reste du protocole. La méthode devra retourner True si elle a ajouté ou modifié une route, False sinon.
8. Vérifier le bon fonctionnement de la méthode sur un des routeurs du réseau.
9. L'appliquer ensuite aux six routeurs. Est-ce suffisant d'appliquer une seule fois la méthode `mise_a_jour_table` au six routeurs ?
10. Combien de fois doit-on appliquer la méthode aux six routeurs ? Comment en être sûr ?