Spécialité NSI en terminale
Protocoles de routage

1 Protocoles de routage

Quelle est la route à suivre par un paquet selon le protocole de routage utilisé ?

Deux protocoles sont présentés dans la suite : le protocole RIP basé sur le nombre de sauts entre deux machines et le protocole OSPF basé sur la rapidité des sauts. Un réseau peut se représenter par un graphe et le lien est évident avec la recherche de chemin sur un graphe. Les liaisons entre deux machines sont les arêtes du graphe, les switchs et les routeurs sont les sommets du graphe. Les algorithmes de routage sont donc basés sur des algorithmes utilisés avec les graphes.

Une table de routage est un tableau contenant en général les éléments suivants : destination réseau, masque réseau, adresse passerelle, adresse interface, métrique.

- La destination réseau est une adresse IP, celle du destinataire (une machine ou un réseau).
- Le masque indique quelle est la partie de l'adresse destination utile pour le réseau et celle utile pour l'hôte.
- L'adresse de la passerelle est une adresse IP, celle du routeur où sont envoyés les paquets.
- L'adresse interface indique l'adresse de l'interface réseau utilisée.
- La métrique indique le nombre de sauts maximum pour arriver à la destination.

Une passerelle est un routeur. Elle fait le lien entre différents réseaux ou sous-réseaux. Un paquet qui doit passer d'un réseau à un autre passe par un routeur. S'il y a plusieurs routeurs entre les deux réseaux, chaque routeur doit connaître l'adresse du routeur suivant.

Sur un ordinateur relié à une box internet dans un réseau privé, on obtient par exemple la table de routage qui suit, résultat de la commande `route print` sous Windows.

```
IPv4 Table de routage
===========================================================================
Itinéraires actifs :
Destination réseau    Masque réseau  Adr. passerelle   Adr. interface Métrique
          0.0.0.0          0.0.0.0      192.168.1.1     192.168.1.36     35
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      192.168.1.0    255.255.255.0         On-link      192.168.1.36    291
     192.168.1.36  255.255.255.255         On-link      192.168.1.36    291
    192.168.1.255  255.255.255.255         On-link      192.168.1.36    291
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link      192.168.1.36    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link      192.168.1.36    291
===========================================================================
```

Les adresses commençant par 127, 224, 255 et celles se terminant par 255 sont spécifiques au loopback, au multicast et au broadcast. (Revoir le cours de première si nécessaire). Il reste, pour ce qui nous concerne, la partie du tableau suivante :

```
IPv4 Table de routage
===========================================================================
Itinéraires actifs :
Destination réseau    Masque réseau  Adr. passerelle   Adr. interface Métrique
          0.0.0.0          0.0.0.0      192.168.1.1     192.168.1.36     35
```

```
      192.168.1.0    255.255.255.0         On-link      192.168.1.36    291
     192.168.1.36  255.255.255.255         On-link      192.168.1.36    291
===========================================================================
```

Pour effectuer un envoi, la machine examine les lignes du tableau selon l'ordre des masques, du plus grand au plus petit. Dans notre exemple, nous avons les adresses : 255.255.255.255, puis 255.255.255.0, et enfin 0.0.0.0.

Donc, si le destinataire est la machine qui a pour IP 192.168.1.36, (masque 255.255.255.255), l'interface utilisée a pour adresse 192.168.1.36. C'est la même machine.

Si le destinataire a pour IP 192.168.1.0 avec le masque 255.255.255.0, il s'agit du réseau local auquel est connectée la machine, et l'interface utilisée a pour adresse 192.168.1.36. C'est encore la même machine.

Si le destinataire a pour IP 0.0.0.0, avec le masque 0.0.0.0, il s'agit de toutes les autres adresses possibles. L'interface utilisée a pour adresse 192.168.1.36, c'est la machine locale, mais il faut sortir du réseau local et passer par la passerelle, qui appartient au réseau local, d'adresse 192.168.1.1. Autrement dit, la destination 0.0.0.0 avec le masque 0.0.0.0 propose une route par défaut, si aucune autre route n'a été trouvée.

Précision : la mention On-link pour l'adresse de la passerelle, indique que la route cherchée est sur le réseau auquel est connectée la machine émettrice.

Avec un autre poste fonctionnant sous Linux, sur le même réseau, la commande `route -n` permet d'afficher un résultat similaire. Seules les lignes hors multicast, broadcast et loopback sont présentées.

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic  Metric  Ref  Use  Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG     303     0      0  wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U      303     0      0  wlan0
```

L'interface `wlan0` a l'adresse de la carte réseau 192.168.1.24. On l'obtient avec la commande `ifconfig`. À la deuxième ligne, Linux indique 0.0.0.0 pour la passerelle à la place de "On-link" pour Windows.

La commande `route -v` affiche le résultat :

```
Table de routage IP du noyau
Destination     Passerelle      Genmask         Indic  Metric  Ref  Use  Iface
default         lan.home        0.0.0.0         UG     303     0      0  wlan0
192.168.1.0     0.0.0.0         255.255.255.0   U      303     0      0  wlan0
```

L'adresse 0.0.0.0 est remplacée par le mot `default`. IL s'agit bien de l'adresse par défaut et la passerelle indiquée est utilisée pour envoyer des données en dehors du réseau d'adresses 192.168.1.0.

De manière simplifiée, une table de routage contient tous les réseaux auxquels est connectée la machine, une route par défaut et éventuellement des routes pour d'autres réseaux. La passerelle est le routeur à utiliser pour joindre un réseau. C'est donc le routeur lui-même s'il est dans ce réseau.

Considérons l'extrait suivant :

Destination réseau	Masque réseau	Adr. passerelle	Adr. interface	Métrique
0.0.0.0	0.0.0.0	192.168.1.1	192.168.1.36	40
192.168.1.0	255.255.255.0	On-link	192.168.1.36	296

Concrètement, comment expédier un paquet vers une adresse ?

On commence par extraire, à partir de cette adresse, l'adresse du réseau ou sous-réseau. Si c'est la même que celle de l'expéditeur, le paquet est envoyé directement à l'adresse. Dans l'exemple, c'est la deuxième ligne qui indique que pour accéder aux machines du réseau 192.168.1.0, l'accès est direct.

Nous pouvons observer avec la commande `tracert` sous Windows et `traceroute` sous Linux, les trois cas de figure.

- Depuis la machine sous Windows, on cherche l'itinéraire jusqu'à la machine elle-même.

```
C:\Users\toto>tracert 192.168.1.36

Détermination de l'itinéraire vers LAPTOP-HLS1HM9S.home [192.168.1.36]
avec un maximum de 30 sauts :

  1    <1 ms    <1 ms    <1 ms  LAPTOP-HLS1HM9S.home [192.168.1.36]

Itinéraire déterminé.
```

- Depuis la machine sous Windows, on cherche l'itinéraire jusqu'à la machine sous Linux.

```
C:\Users\serge>tracert 192.168.1.24

Détermination de l'itinéraire vers piserge.home [192.168.1.24]
avec un maximum de 30 sauts :

  1   5 ms    4 ms     4 ms  piserge.home [192.168.1.24]

Itinéraire déterminé.
```

- Depuis la machine sous Linux, on cherche l'itinéraire jusqu'à la machine elle-même.

```
pi@piSerge:~ $ traceroute 192.168.1.24
traceroute to 192.168.1.24 (192.168.1.24), 30 hops max, 60 byte packets
1 piserge.home [192.168.1.24] 0.117 ms 0.037 ms 0.037 ms
```

Si l'adresse correspond à un autre réseau, il faut chercher une correspondance entre l'adresse et celle d'un routeur capable de transmettre les paquets à destination. Il peut y avoir l'adresse de l'hôte, celle du sous-réseau, ou celle du réseau.

Si aucune adresse n'est trouvée, le paquet est envoyé à une passerelle par défaut. C'est le cas de la première ligne de l'exemple. L'adresse 192.168.1.1 est celle de la box.

Dans tous ces cas, on joint au paquet l'adresse IP de destination et l'adresse MAC du routeur.

Avec le logiciel Filius, dans les exemples présentés auparavant la case "Routage automatique" est cochée. Il suffit de la décocher pour accéder à la table de routage qui est présentée sous une forme similaire à celle obtenue par la commande `route` sous Linux ou Windows.

Note

Sous Windows, si l'invite de commande n'est pas disponible, nous pouvons la remplacer par un programme en Python.

```
import subprocess

com1 = ['ipconfig']
com2 = ['arp', '-a'] # traduction adresse IP adresse physique
com3 = ['route', 'print'] # table de routage
com4 = ['tracert', '192.168.1.1']
com5 = ['ping', '192.168.1.1']

# exemple avec com1
with subprocess.Popen(com1, stdout=subprocess.PIPE) as proc:
    s = proc.stdout.read()
```

```
ch = s.decode("cp850")
print(ch)
```

Résumons : une machine A envoie un paquet à une machine B par l'intermédiaire d'un routeur.

La trame part de la machine A vers le routeur. Elle contient l'adresse IP de la machine A, celle de la machine B, l'adresse MAC de la machine 1 et celle de l'interface 1 du routeur. Le paquet repart du routeur et la trame contient alors l'adresse IP de la machine A, celle de la machine B, l'adresse MAC de l'interface 2 du routeur et celle de la machine 2. Ce procédé utilise les tables de routage de chaque appareil. Le routeur doit être configuré pour contenir les adresses des postes ou des autres routeurs vers lesquels envoyer les paquets.

Avec uniquement un système d'adresses, chaque routeur utilise sa table de routage et le routage se fait de proche en proche. Cependant, pour atteindre une machine lointaine, la notion de coût est importante.

Il y a différentes manières de mesurer ce coût. Le passage d'un routeur à un autre peut compter pour un saut, et on compte alors le nombre de sauts nécessaires pour atteindre une machine. Sur de grands réseaux il peut être plus rapide de passer par deux routeurs distincts que par un seul. Cela est fonction de la vitesse de transmission, du débit des liaisons, c'est-à-dire le nombre de bits qui peuvent être transférés d'un routeur à un autre par unité de temps. De manière similaire, le kilométrage effectué sur une autoroute peut être plus important que celui effectué par des petites routes mais la vitesse y est plus rapide. Un débit se mesure en bits par seconde (bps). On peut ensuite évaluer le temps de parcours entre deux routeurs pour une quantité de bits donnée. Par exemple, pour une quantité de 10^8 bits, on divise cette quantité par le débit et on obtient le temps en seconde. Ensuite on ajoute les coûts calculés entre chaque élément afin d'obtenir le coût total d'une route.

L'état du réseau peut changer rarement ou souvent. Dans le cas d'un petit réseau, on peut définir les routes de manière statique. Les tables de routage sont remplies une fois et ne sont pas modifiées. Dans le cas où l'état du réseau change souvent, il est nécessaire de déterminer régulièrement les routes les moins coûteuses (en nombre de sauts ou en débit) et des algorithmes performants sont utilisées pour cela.

Les protocoles RIP et OSPF sont dynamiques et sont utilisés pour des réseaux assez grands. Le premier utilise le nombre de sauts (des distances), le second l'état des liens (la rapidité).

2 Le protocole RIP

Nous commençons par le RIP, Routing Information Protocol en anglais, ce qui signifie en français protocole d'information de routage. Il existe en deux versions : RIPv1 et RIPv2.

Pour choisir la meilleure route, on applique l'algorithme de Bellman et Ford, utilisé pour déterminer un chemin le plus court dans un graphe.

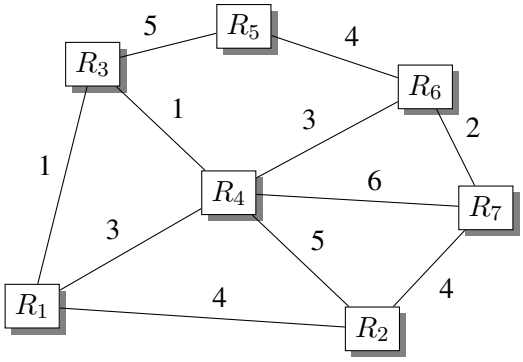
Nous considérons un programme applicable à un graphe et l'utilisons avec un réseau exemple.

On représente un graphe à l'aide d'une liste ou d'un dictionnaire. Dans le cas du dictionnaire, les clés sont les sommets du graphe, la valeur associée à une clé est la liste des couples (voisin, distance).

```
def bellman_ford(G, s):
    n = len(G)
    d = {k: float('inf') for k in G}
    d[s] = 0
    for i in range(n-1): # on répète n-1 fois
        for k in G: # puis deux boucles for pour explorer toutes les arêtes
            for j in range(len(G[k])):
                v, c = G[k][j]
```

```
                d[v] = min (d[v], d[k] + c)
    return d
```

Considérons par exemple une partie d'un réseau. Cette partie est constituée de sept routeurs. Certains peuvent être connectés à d'autres réseaux.



Ce réseau est défini par le dictionnaire `reseau`.

```
reseau = {"R1" : [("R2", 4), ("R3", 1), ("R4", 3)],
          "R2" : [("R1", 4), ("R4", 5), ("R7", 4)],
          "R3" : [("R1", 1), ("R5", 5), ("R4", 1)],
          "R4" : [("R1", 3), ("R2", 5), ("R3", 1), ("R6", 3), ("R7", 6)],
          "R5" : [("R3", 5), ("R6", 4)],
          "R6" : [("R4", 3), ("R5", 4), ("R7", 2)],
          "R7" : [("R2", 4), ("R4", 6), ("R6", 2)]}
```

On exécute la fonction `bellman_ford` en donnant pour paramètres le graphe `reseau` et le sommet `"R1"`.

```
>>> bellman_ford(reseau, "R1")
{'R1': 0, 'R2': 4, 'R3': 1, 'R4': 2, 'R5': 6, 'R6': 5, 'R7': 7}
```

Le résultat nous donne les longueurs des plus courts chemins entre le sommet R1 et chacun des autres sommets.

Le dictionnaire des distances se remplit étape par étape.

- étape 1 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}
- étape 2 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}
- étape 3 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}
- étape 4 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}
- étape 5 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}
- étape 6 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}

Attention, il peut sembler que les étapes 2 à 6 sont inutiles, que les longueurs sont obtenues dès la fin de l'étape 1. Mais c'est à cause de l'ordre d'examen des sommets et des arêtes qui est particulier dans cet exemple.

Si nous cherchons les longueurs des plus courts chemins entre "R7" et chaque sommet du graphe, avec `bellman_ford(reseau, "R7")`, il en va autrement et nous obtenons le résultat qui suit :

- étape 1 : {'R1' : inf, 'R2' : 4, 'R3' : inf, 'R4' : 6, 'R5' : inf, 'R6' : 2, 'R7' : 0}
- étape 2 : {'R1' : 8, 'R2' : 4, 'R3' : 7, 'R4' : 5, 'R5' : 6, 'R6' : 2, 'R7' : 0}
- étape 3 : {'R1' : 8, 'R2' : 4, 'R3' : 6, 'R4' : 5, 'R5' : 6, 'R6' : 2, 'R7' : 0}
- étape 4 : {'R1' : 7, 'R2' : 4, 'R3' : 6, 'R4' : 5, 'R5' : 6, 'R6' : 2, 'R7' : 0}
- étape 5 : {'R1' : 7, 'R2' : 4, 'R3' : 6, 'R4' : 5, 'R5' : 6, 'R6' : 2, 'R7' : 0}
- étape 6 : {'R1' : 7, 'R2' : 4, 'R3' : 6, 'R4' : 5, 'R5' : 6, 'R6' : 2, 'R7' : 0}

Le principe du protocole RIP : c'est un protocole de routage IP (Internet Protocol) où chaque routeur communique aux routeurs voisins la distance qui les sépare d'un réseau IP. Cette distance est mesurée en nombre de sauts (*hops* en anglais) et on l'appelle la « métrique ». Aucun routeur n'a une vision globale du réseau. Les routes sont connues de proche en proche.

Un routeur envoie donc, à l'aide des adresses de ses interfaces réseau, une demande aux routeurs voisins. Ceux-ci lui répondent en envoyant une partie ou toute leur table de routage, suivant la demande. Lorsque une réponse arrive, le routeur l'analyse.

— S'il obtient une nouvelle route, il incrémente la métrique de 1 et si celle-ci est inférieure à 15, il ajoute la route à sa table.
— Si la route existe déjà, il y a deux cas.
  — Si la métrique est inférieure, il supprime l'ancienne route et la remplace par la nouvelle.
  — Si la métrique est supérieure, il change la métrique dans sa table si l'information vient du même routeur.

Un routeur garde donc en mémoire, pour chaque réseau IP connu, l'adresse du routeur voisin dont la métrique est la plus petite.

Quelques règles :

- Le nombre maximal de sauts autorisés pour un paquet est 15. Ceci permet d'éviter qu'un paquet ne se déplace en boucle. Si le paquet a effectué 15 sauts, il est supprimé. De plus seul le nombre de sauts est pris en compte. Un paquet ira de R1 à R2 même si la liaison est nettement plus mauvaise que de passer de R1 à R4 puis de R4 à R2.
- La table de routage est diffusée toutes les 30 secondes. Si une route connaît un problème, une métrique de 16 lui est affectée temporairement. Ceci signifie que la route ne peut être choisie, (16 signifie infinie). L'information est diffusée et la route est supprimée de la table après 120 secondes.
- Si un routeur ne diffuse rien pendant 3 minutes sur une route, on considère qu'il est en panne. La route est alors supprimée et est plus tard remplacée si un autre routeur a communiqué une information.

Un problème peut survenir si deux routeurs s'envoient des informations contradictoires : R_1 dit à R_2 que R_3 est en panne et R_2 envoie à R_1 une métrique valide sur R_3. Il y a un risque de bouclage infini. Une sécurité est de ne jamais retransmettre sur une interface une information qui a été reçue sur cette interface et de diffuser sans attendre une information de panne.

Le protocole RIPv2 propose des améliorations.

- Pour moins encombrer le réseau, la diffusion des métriques est en multicast plutôt qu'en broadcast.
- Pour améliorer la sécurité, l'authentification est accompagnée d'un mot de passe crypté.
- Les masques des sous-réseaux sont diffusés à la place des masques par défaut des réseaux.

3 Le protocole OSPF

Le protocole OSPF, pour Open Shortest Path First, a été développé pour remplacer le protocole RIP. La version 1 date de 1989, la version 2 de 1998 et la version 3 adaptée pour IPv6 de 1999. Ce protocole est plus complexe et utilise des routeurs plus puissants mais dont la configuration est moins simple. Il permet de gérer de grands domaines, de découper les domaines en aires ou de mieux agréger des routes.

L'algorithme prend en compte les débits pour mesurer la rapidité d'une route, son coût. Chaque routeur utilise l'algorithme de Dijkstra, Shortest Path First, pour déterminer la meilleure route vers chacun des réseaux connus.

L'algorithme de Dijkstra est plus efficace que celui de Bellman et Ford dans les cas qui nous concernent. Cependant les coût dépendent de la manière d'implémenter ces algorithmes. On peut obtenir un coût de l'ordre du produit $s \times a$ où s est le nombre de sommets et a le nombre d'arêtes pour Bellman et Ford, et de l'ordre de $(a + s) \log s$ pour Dijkstra.

Remarque 1 : de manière générale, en théorie des graphes, on obtient les mêmes coûts tant que les poids des arêtes ne sont pas négatifs. Dans ce dernier cas, l'algorithme de Dijkstra ne fonctionne pas. En revanche, celui de Bellman et Ford fonctionne correctement et permet même de détecter la présence d'un cycle de poids négatif.

Remarque 2 : avec l'algorithme de Dijkstra les sommets sont visités au plus une fois, alors qu'avec celui de Bellman et Ford ils sont visités plus d'une fois.

Pour écrire un programme avec l'algorithme de Dijkstra, on a besoin de définir une fonction qui détermine la valeur minimale d'un dictionnaire et renvoie la clé correspondante. On utilise la valeur infinie `inf` du module `math`.

```
from math import inf

def minimum(dico):
    mini = inf
    s = -1
    for k in dico:
        if dico[k] < mini:
            mini = dico[k]
            s = k
    return s

def dijkstra(G, s):
    D = {} # tableau des distances minimales
    d = {k:inf for k in G} # tableau des distances initiales
    d[s] = 0 # s sommet de départ
    while len(d) > 0: # c'est fini quand d est vide
        k = minimum(d) # sommet de distance minimale pour démarrer une étape
        for j in range(len(G[k])): # on regarde les voisins de k
                v, c = G[k][j] # v un voisin de k, c le cout
                if v not in D:
                    d[v] = min(d[v], d[k] + c)
        D[k] = d[k] # on copie le sommet et la distance dans D
        del d[k] # on supprime le sommet de d
    return D
```

Supposons que le débit entre un routeur R1 et un routeur R2 soit de 2 Mbps. S'il existe une route de R1 à R2 empruntant un routeur R3 tel que le débit entre R1 et R3 ainsi que celui entre R3 et R2 soit de 10 Mbps, alors le protocole OSPF permettra de choisir cette route. La table de routage de R1 indiquera d'utiliser la passerelle R3.

On reprend le réseau exemple, déjà utilisé avec l'algorithme de Bellman et Ford, pour tester le programme.

```
reseau = {"R1" : [("R2", 4), ("R3", 1), ("R4", 3)],
          "R2" : [("R1", 4), ("R4", 5), ("R7", 4)],
          "R3" : [("R1", 1), ("R5", 5), ("R4", 1)],
          "R4" : [("R1", 3), ("R2", 5), ("R3", 1), ("R6", 3), ("R7", 6)],
          "R5" : [("R3", 5), ("R6", 4)],
          "R6" : [("R4", 3), ("R5", 4), ("R7", 2)],
          "R7" : [("R2", 4), ("R4", 6), ("R6", 2)]}
```

On exécute alors la fonction `dijkstra` en donnant pour paramètres le graphe `reseau` et le sommet `"R1"`.

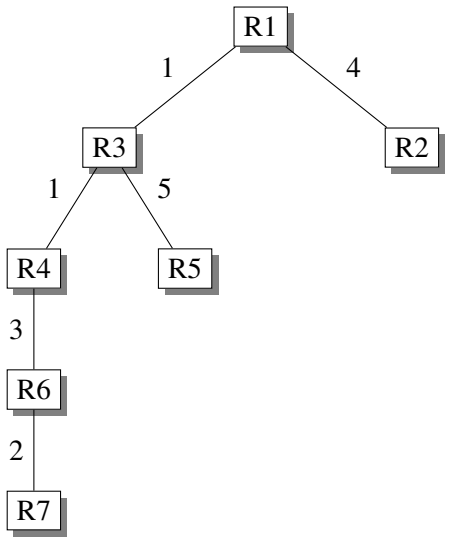
```
>>> dijkstra(reseau, "R1")
{'R1': 0, 'R2': 4, 'R3': 1, 'R4': 2, 'R5': 6, 'R6': 5, 'R7': 7}
>>>
```

La réponse obtenue avec Dijkstra est identique à celle obtenue avec Bellman et Ford.

Les étapes par contre sont différentes. On obtient une nouvelle valeur à chaque étape :

- étape 1 : {'R1' : 0, 'R2' : inf, 'R3' : inf, 'R4' : inf, 'R5' : inf, 'R6' : inf, 'R7' : inf}
- étape 2 : {'R1' : 0, 'R2' : inf, 'R3' : 1, 'R4' : inf, 'R5' : inf, 'R6' : inf, 'R7' : inf}
- étape 3 : {'R1' : 0, 'R2' : inf, 'R3' : 1, 'R4' : 2, 'R5' : inf, 'R6' : inf, 'R7' : inf}
- étape 4 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : inf, 'R6' : inf, 'R7' : inf}
- étape 5 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : inf, 'R6' : 5, 'R7' : inf}
- étape 6 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : inf}
- étape 7 : {'R1' : 0, 'R2' : 4, 'R3' : 1, 'R4' : 2, 'R5' : 6, 'R6' : 5, 'R7' : 7}

Dans les deux cas, cela revient à construire un arbre dont la racine est `R1`.



4 Le protocole BGP

Le Border Gateway Protocol est un protocole utilisé sur Internet. C'est un protocole basé sur l'échange d'informations de routage (routes, accessibilité) entre des systèmes autonomes (comme le réseau d'un FAI) qui utilisent eux, en interne, le protocole OSPF par exemple. Pour le choix d'une route, différentes règles sont définies utilisant des préférences locales, des discriminateurs, la connaissance de chemins complets pour atteindre une destination, etc. La métrique classique (nombre de sauts, débit), n'intervient pas ici. Ce protocole est utilisé entre les opérateurs (des services cloud ou télécoms) et les fournisseurs d'accès à Internet.

Avec le protocole BGP, des informations (des routes) sont échangées entre systèmes concurrents (fournisseurs) qui ont conclu des accords préalablement. L'objectif n'est donc pas forcément un système de transmission optimal. Diverses restrictions peuvent être mises en place suivant le type d'accord commercial.