Devoir surveillé n°7 – Correction

Exercice 1 : (10pt) *Cet exercice porte sur la notion de listes, la récursivité et la programmation dynamique.*

Pour extraire de l'eau dans des zones de terrain instable, on souhaite forer un conduit dans le sol pour réaliser un puits tout en préservant l'intégrité du terrain. Pour représenter cette situation, on va considérer qu'en forant à partir d'une position en surface, on s'enfonce dans le sol en allant à gauche ou à droite à chaque niveau, jusqu'à atteindre le niveau de la nappe phréatique.

Le sol pourra donc être représenté par une pyramide d'entiers où chaque entier est le score de confiance qu'on a dans le forage de la zone correspondante. Une telle pyramide est présentée sur la figure 1, à gauche, les flèches indiquant les différents déplacements possibles d'une zone à une autre au cours du forage.

Un conduit doit partir du sommet de la pyramide et descendre jusqu'au niveau le plus bas, où se situe l'eau, en suivant des déplacements élémentaires, c'est-à-dire en choisissant à chaque niveau de descendre sur la gauche ou sur la droite. Le score de confiance d'un conduit est la somme des nombres rencontrés le long de ce conduit. Le conduit gris représenté à droite sur la figure 1 a pour score de confiance $4+2+5+1+3=15$.

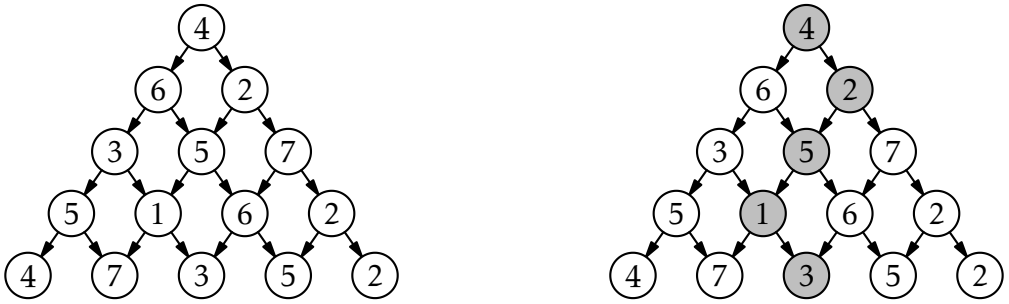
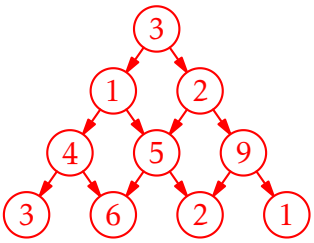


Figure 1

On va utiliser un ordinateur pour chercher à résoudre ce problème. Pour cela, on représente chaque niveau par la liste des nombres de ce niveau et une pyramide par une liste de niveaux.

La pyramide ci-dessus est donc représentée par la liste de listes
`ex1 = [[4],[6,2],[3,5,7],[5,1,6,2],[4,7,3,5,2]]`.

1) Dessiner la pyramide représentée par la liste de listes
`ex2 = [[3],[1,2],[4,5,9],[3,6,2,1]]`.



2) Déterminer un conduit de score de confiance maximal dans la pyramide `ex2` et donner son score.
Solution : On peut faire $3+2+5+6=16$ ou $3+2+9+2=16$.

On souhaite déterminer le score de confiance maximal pouvant être atteint pour une pyramide quelconque. Une première idée consiste à énumérer tous les conduits et à calculer leur score pour déterminer les meilleurs.

3) Énumérer les conduits dans la pyramide de trois niveaux représentée sur la figure 2.

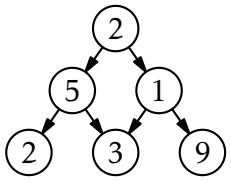


Figure 2

Solution : $2+5+2=9$, $2+5+3=10$, $2+1+3=6$, $2+1+9=12$.

Afin de compter le nombre de conduits pour une pyramide de `n` niveaux, on remarque qu'un conduit est uniquement représenté par une séquence de $n-1$ déplacements `gauche` ou `droite`.

4) En considérant le fait que pour passer d'un niveau à celui du dessous, on a toujours le choix entre deux possibilités, gauche ou droite, déterminer le nombre de conduits dans une pyramide de `n` niveaux.
Solution : Il y a $n-1$ changements de niveaux, donc 2^{n-1} conduits différents.

5) Justifier que la solution qui consiste à tester tous les conduits possibles pour calculer le score de confiance maximal d'une pyramide n'est pas raisonnable.
Solution : Le coût de l'algorithme est exponentiel. Cela veut dire que le temps de calcul augmente de façon exponentiel par rapport à la hauteur de la pyramide. Ajouter un étage multiplie le temps de calcul par 2. Il devient donc rapidement très lent.

On dira dans la suite qu'un conduit est maximal si son score de confiance est maximal. Afin de pouvoir calculer efficacement le score maximal, on peut analyser la structure des conduits maximaux.

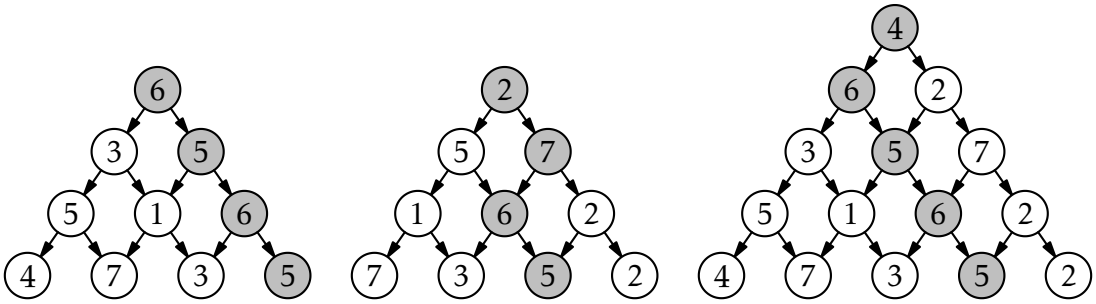


Figure 3

- **Première observation :** si on a des conduits maximaux `cm1` et `cm2` (représentés en gris dans la figure 3) pour les deux pyramides obtenues en enlevant le sommet de `ex1`, on obtient un conduit maximal en ajoutant le sommet 4 devant le conduit de plus grand score parmi `cm1` et `cm2`. Ici le score de `cm1` est $6+5+6+5=22$ et le score de `cm2` est $2+7+6+5=20$ donc le conduit maximal dans `ex1` est celui obtenu à partir de `cm1` et dessiné à droite dans la figure 3.
- **Deuxième observation :** si la pyramide n'a qu'un seul niveau, il n'y a que le sommet, dans ce cas, il n'y a pas de choix à faire, le seul conduit possible est celui qui contient le sommet et le nombre de ce sommet est le score maximal que l'on peut obtenir.

Avec ces deux observations, on peut calculer le score maximal possible pour un conduit dans une pyramide `p` par récurrence. Posons `score_max(i,j,p)` le score maximal possible depuis le nombre d'indice `j` du niveau `i`, c'est-à-dire dans la petite pyramide issue de ce nombre. On a alors les relations suivantes :

- `score_max(len(p)-1,j,p) = p[len(p)-1][j]` ;
- `score_max(i,j,p) = p[i][j] + max(score_max(i+1,j,p),score_max(i+1,j+1,p))`.

Le score maximal possible pour `p` toute entière sera alors `score_max(0,0,p)`.

6) Écrire la fonction récursive `score_max` qui implémente les règles précédentes.

```
def score_max(i, j, p):
    if i == len(p) - 1:
        return p[i][j]
    else:
        return p[i][j] + max(score_max(i+1,j,p), score_max(i+1,j+1,p))
```

Si on suit à la lettre la définition de `score_max`, on obtient une résolution dont le coût est prohibitif à cause de la redondance des calculs. Par exemple `score_max(3,1,p)` va être calculé pour chaque appel à `score_max(2,0,p)` et `score_max(2,1,p)`. Pour éviter cette redondance, on décide de mettre en place une approche par programmation dynamique. Pour cela, on va construire une pyramide `s` dont le nombre à l'indice `j` du niveau `i` correspond à `score_max(i,j,p)`, c'est-à-dire au score maximal pour un conduit à partir du nombre correspondant dans `p`.

7) Écrire une fonction `pyramide_nulle` qui prend en paramètre un entier `n` et construit une pyramide remplie de 0 à `n` niveaux.

```
def pyramide_nulle(n):
    res = []
    for i in range(n):
        res.append([0]*(i+1))
    return res
```

8) Recopier et compléter la fonction `prog_dyn` ci-dessous qui prend en paramètre une pyramide `p`, et qui renvoie le score maximal pour un conduit dans `p`. Pour cela, on construit une pyramide `s` remplie de 0 de la même taille et la remplit avec des valeurs correspondant à celles de `score_max` en commençant par le dernier niveau et en appliquant petit à petit les relations données ci-dessus.

```
def prog_dyn(p):
    n = len(p)
    s = pyramide_nulle(n)
    # remplissage du dernier niveau
    for j in range(n):
        s[n-1][j] = p[n-1][j]
    # remplissage des autres niveaux
    for i in range(n-2, -1, -1):
        for j in range(i+1):
            s[i][j] = p[i][j] + max(s[i+1][j], s[i+1][j+1])
    # renvoie du score maximal
    return s[0][0]
```

9) Montrer que le coût d'exécution de cette fonction est quadratique en `n` pour une pyramide à `n` niveaux.

Solution : On doit parcourir toutes les valeurs de `p`. Il y en a 1 sur le 1e étage, 2 sur le 2e, jusqu'à `n` sur le `n`-ème étage. On doit donc regarder $1+2+3+\cdots+n=\dfrac{(n+1)n}{2}$ valeurs, ce qui est quadratique en `n`.

10) Expliquer comment adapter la fonction `score_max` pour éviter la redondance des calculs afin d'obtenir également un coût quadratique, tout en gardant une approche récursive.

Solution : Il faut utiliser la mémoïsation. Par exemple en rajoutant un dictionnaire en paramètre pour mémoriser les appels déjà faits et ne pas les refaire.

Exercice 2 : (10pt) *Cet exercice porte sur les réseaux, les protocoles de routage et les graphes.*

Partie A

Le réseau informatique d'une société est constitué d'un ensemble de routeurs interconnectés à l'aide de fibres optiques.

La figure ci-dessous représente le schéma de ce réseau. Il est composé de deux réseaux locaux L1 et L2. Le réseau local L1 est relié au routeur R1 et le réseau local L2 est relié au routeur R9.

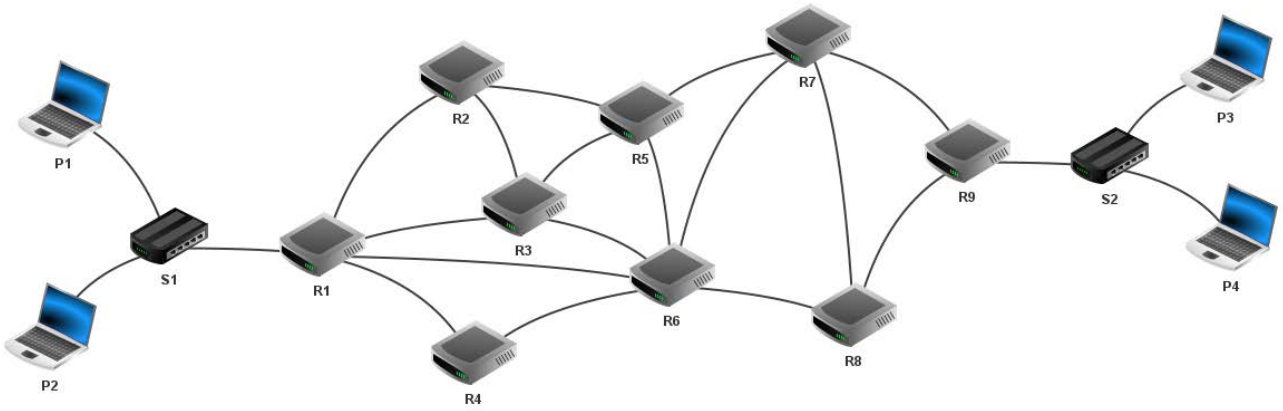


Figure 1. Réseau

Dans cette partie, les adresses IP sont composées de 4 octets, soit 32 bits. Elles sont notées `X1.X2.X3.X4`, où `X1`, `X2`, `X3` et `X4` sont les valeurs des 4 octets, converties en notation décimale. La notation `X1.X2.X3.X4/n` signifie que les `n` premiers bits de poids forts de l'adresse IP représentent la partie « réseau », les bits suivants représentent la partie « hôte ».

Toutes les adresses des machines connectées à un réseau local ont la même partie réseau.

Le tableau suivant indique les adresses IPv4 des machines constituant le réseau de la société.

NOM	TYPE	ADRESSE IPV4
R1	Routeur	Interface 1 : `192.168.1.1/24` Interface 2 : `192.168.2.1/24` Interface 3 : `192.168.3.1/24` Interface 4 : `192.168.4.1/24` Interface 5 : `192.168.5.1/24`
R2	Routeur	Interface 1 : `192.168.2.2/24` Interface 2 : `192.168.7.1/24` Interface 3 : `192.168.8.1/24`
R3	Routeur	Interface 1 : `192.168.3.2/24` Interface 2 : `192.168.7.2/24` Interface 3 : `192.168.9.1/24` Interface 4 : `192.168.10.1/24`
R4	Routeur	Interface 1 : `192.168.5.2/24` Interface 2 : `192.168.6.1/24`
R5	Routeur	Interface 1 : `192.168.8.2/24` Interface 2 : `192.168.9.2/24` Interface 3 : `192.168.11.1/24` Interface 4 : `192.168.12.1/24`
R6	Routeur	Interface 1 : `192.168.4.2/24` Interface 2 : `192.168.6.2/24` Interface 3 : `192.168.10.2/24` Interface 4 : `192.168.11.2/24` Interface 5 : `192.168.13.1/24` Interface 6 : `192.168.14.1/24`
R7	Routeur	Interface 1 : `192.168.12.2/24` Interface 2 : `192.168.13.2/24` Interface 3 : `192.168.15.1/24` Interface 4 : `192.168.16.1/24`
R8	Routeur	Interface 1 : `192.168.14.2/24` Interface 2 : `192.168.15.2/24` Interface 3 : `192.168.17.1/24`
R9	Routeur	Interface 1 : `192.168.16.2/24` Interface 2 : `192.168.17.2/24` Interface 3 : `192.168.18.1/24`
P1	Portable	`192.168.1.10`
P2	Portable	Non fourni
P3	Portable	Non fourni
P4	Portable	Non fourni

1) En utilisant les adresses IP des différentes interfaces et des ordinateurs portables, en déduire une adresse possible pour le portable P2.

Solution : L'ordinateur P2 se trouve dans le même réseau que P1. Il est donc dans le réseau d'adresse `192.168.1.0/24`. Il peut donc avoir n'importe quelle adresse entre `192.168.1.2` et `192.168.1.254`, à l'exception de `192.168.1.10`.

2) Donner l'adresse du réseau local L2 ainsi que le nombre d'adresses possibles pour les ordinateurs portables P3 et P4.
Solution : Le réseau L2 a pour adresse `192.168.18.0/24`. Il faut enlever cette adresse, celle de l'interface 3 de R9 et l'adresse de diffusion `192.168.18.255`. Il reste donc 253 adresses possibles.

Partie B
Le graphe G, représenté ci-dessous, schématise l'architecture du réseau de la société. Les sommets représentent les routeurs et les arêtes représentent les liaisons.

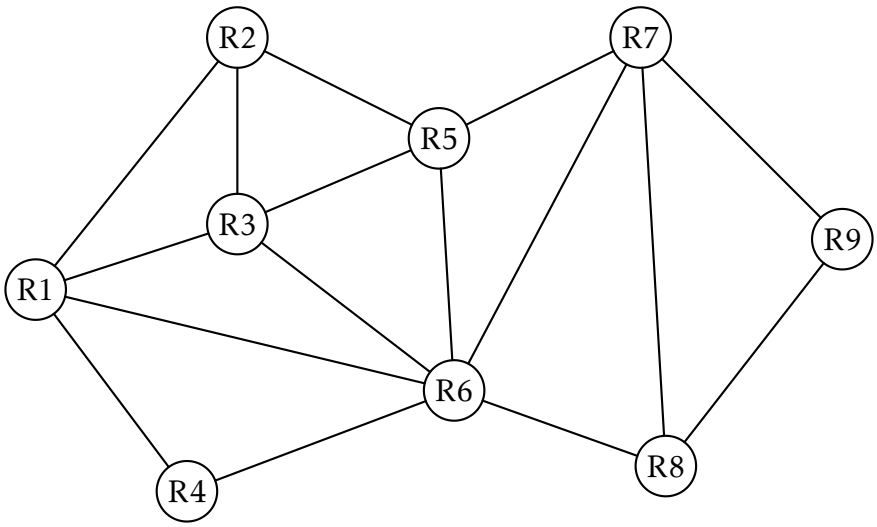


Figure 2. Graphe non pondéré

3) Donner l'implémentation Python des listes d'adjacence de ce graphe à l'aide d'un dictionnaire dont les clés sont les sommets et les valeurs la liste des sommets adjacents du sommet clé. On nomme `G` ce dictionnaire. On notera `'R1'`, `'R2'`... les noms des sommets.
Afin de faciliter la notation, on s'autorise à écrire chaque couple clé/valeur sur une nouvelle ligne.
Solution :

```
G = {'R1': ['R2', 'R3', 'R4', 'R6'],
     'R2': ['R1', 'R3', 'R5'],
     'R3': ['R1', 'R2', 'R5', 'R6'],
     'R4': ['R1', 'R6'],
     'R5': ['R2', 'R3', 'R6', 'R7'],
     'R6': ['R1', 'R3', 'R4', 'R7', 'R8'],
     'R7': ['R5', 'R6', 'R8', 'R9'],
     'R8': ['R6', 'R7', 'R9'],
     'R9': ['R7', 'R8']}
```

On suppose que le protocole de routage RIP est utilisé.
4) Recopier et compléter, en rajoutant autant de lignes que nécessaire, la table de routage simplifiée suivante du routeur R1. On ne mettra que les routeurs comme destination.

Destination	Suivant	Nombre de sauts
R2	R2	1
R3		

Solution :

Destination	Suivant	Nombre de sauts
R2	R2	1
R3	R3	1
R4	R4	1
R5	R3	2
R6	R6	1
R7	R6	2
R8	R6	2
R9	R6	3

5) L'ordinateur P1 envoie un paquet de données à l'ordinateur P3. Donner l'un des chemins empruntés par le paquet ainsi que le nombre de sauts.
Solution : Le paquet peut passer par les routeurs R1 – R6 – R8 – R9, ce qui fait 3 sauts, ou 5 si on compte les passages entre les réseaux locaux et les routeurs.

La société doit vérifier l'état physique de la fibre optique installée sur le réseau. Un robot inspecte toute la longueur de la fibre optique afin de s'assurer qu'elle ne présente pas de détérioration apparente.
On appelle `M` la matrice d'adjacence du graphe de la figure 2. Les sommets sont rangés par ordre croissant des numéros des routeurs (R1, R2, ..., R9). Par exemple, on mettra un 1 en `M[0][1]` parce qu'il y a une liaison entre R1 et R2, et un 0 en `M[0][6]` parce qu'il n'y a pas de liaison directe entre R1 et R7.

6) Donner l'écriture en Python de cette matrice d'adjacence sous la forme d'une liste de listes.
Solution :

```
'     R1 R2 R3 R4 R5 R6 R7 R8 R9'
M = [[0, 1, 1, 1, 0, 1, 0, 0, 0], #R1
     [1, 0, 1, 0, 1, 0, 0, 0, 0], #R2
     [1, 1, 0, 0, 1, 1, 0, 0, 0], #R3
     [1, 0, 0, 0, 0, 1, 0, 0, 0], #R4
     [0, 1, 1, 0, 0, 1, 1, 0, 0], #R5
     [1, 0, 1, 1, 1, 0, 1, 1, 0], #R6
     [0, 0, 0, 0, 1, 1, 0, 1, 1], #R7
     [0, 0, 0, 0, 0, 1, 1, 0, 1], #R8
     [0, 0, 0, 0, 0, 0, 1, 1, 0]] #R9
```

Le **degré d'un sommet** est le nombre d'arêtes dont ce sommet est une extrémité.

7) Recopier et compléter les lignes 3, 5 et 6 de le fonction `degre` qui prend en paramètre la matrice d'adjacence d'un graphe donné sous forme d'une liste de listes et qui renvoie la liste des degrés de tous les sommets du graphe rangés dans le même ordre que les sommets de la matrice d'adjacence.

```
1 def degre(MATRICE):
2     d = []
3     for ligne in MATRICE:
4         cpt = 0
5         for v in ligne:
6             cpt = cpt + v
7         d.append(cpt)
8     return d
```

8) Donner la liste renvoyée par `degre(M)`.
Solution : On obtient `[4, 3, 4, 2, 4, 6, 4, 3, 2]`.

On appelle **chaîne eulérienne** d'un graphe non orienté un chemin qui passe une et une seule fois par toutes les arêtes du graphe. Un graphe est **connexe** s'il y a un chemin permettant de relier n'importe quel couple de sommets. Un grpahe connexe admet une chaîne eulérienne si et seulement si le graphe possède, au plus, deux sommets de degré impair.

9) En utilisant le résultat de la question précédente et en admettant que le graphe est connexe, indiquer si le robot peut parcourir l'ensemble du réseau en suivant les fibres optiques et en empruntant chaque fibre optique une et une seule fois.
Solution : Puisqu'il n'y a que 2 sommets qui ont un degré impair (R2 et R8), il existe un chemin qui passe une unique fois par chaque fibre.

Partie C
Le poids sur chaque arête représente la bande passante en megabits par seconde (Mb/s) de chaque liaison.

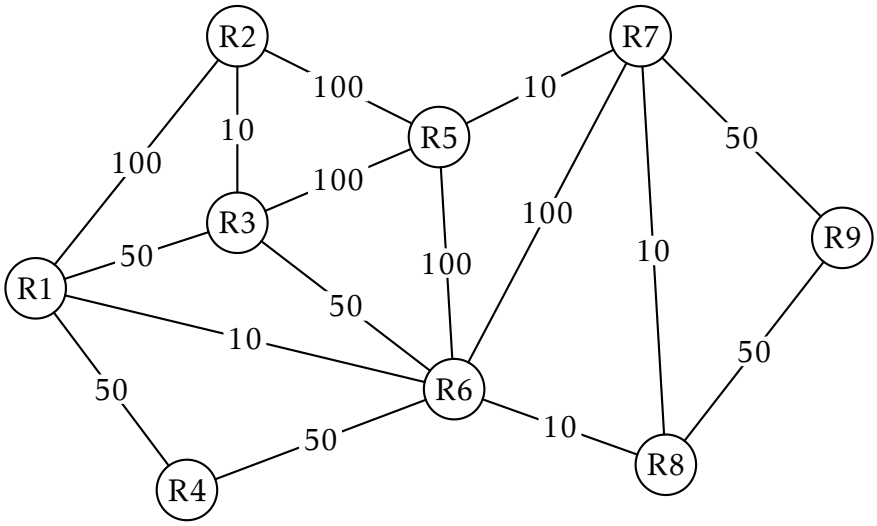


Figure 3. Graphe pondérée

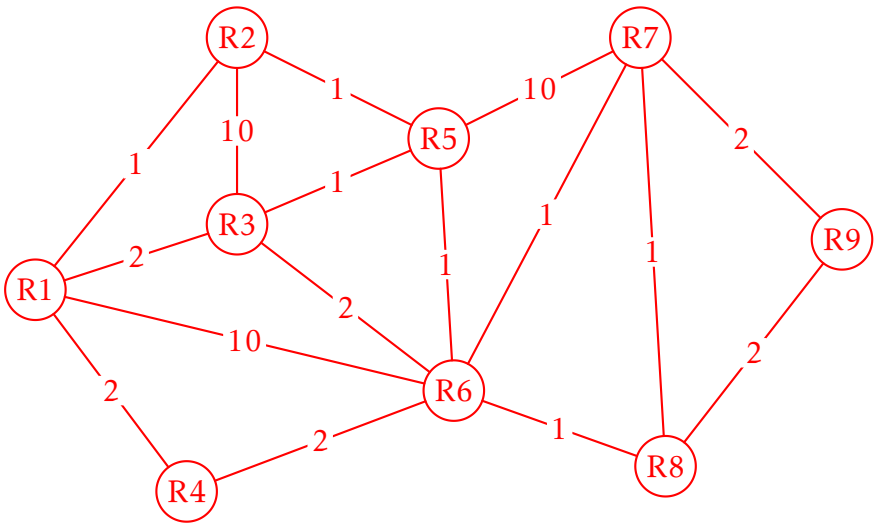
Dans cette partie, on utilise le protocole de routage OSPF. Pour calculer le coût d'une liaison, on utilise la formule :

$$C = \frac{10^8}{BP}$$

où BP est la bande passante en bits par seconde.

10) Déterminer la route qui sera empruntée par le paquet pour aller de l'ordinateur P1 à l'ordinateur P3. Préciser le coût de ce trajet.

Solution : On obtient les coûts suivants :



On peut prendre la route R1 – R2 – R5 – R6 – R8 – R9, pour coût total de 6.