EXERCICE 1 (6 points)

Cet exercice porte sur la programmation Python, la programmation dynamique, les graphes et les réseaux.

On cherche à lutter contre un virus informatique qui essaie de contourner les protocoles de sécurité en migrant régulièrement vers un autre ordinateur, en choisissant à chaque fois au hasard sa nouvelle cible parmi les ordinateurs accessibles. On cherche à savoir quels ordinateurs protéger afin de lutter de manière la plus efficace possible avec des ressources limitées.

On considère le réseau informatique suivant, composé de 5 ordinateurs numérotés : 0, 1, 2, 3 et 4.

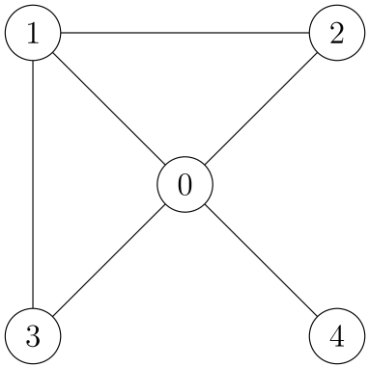

Figure 1. Réseau informatique

1. On représente ce réseau informatique par un graphe que l'on stocke sous forme de listes de voisins. Compléter la définition de la variable `voisins`.

```
voisins = [[1, 2, 3, 4],
           [0, 2, 3],
           [0, 1],
           [...],
           [...]]
```

On ajoute au réseau actuel un sixième ordinateur, numéroté 5. Cet ordinateur n'est accessible que des ordinateurs numérotés 0 et 2.

2. Dessiner le nouveau graphe.

3. Donner la nouvelle définition de la variable `voisins`.

4. Compléter la fonction `voisin_alea` qui prend en paramètre un graphe `voisins` sous forme de listes de voisins et un entier `s` représentant un sommet et qui renvoie un entier représentant un voisin de `s` choisi aléatoirement. On pourra utiliser la fonction `random.randrange(n)` qui renvoie un nombre aléatoire entre `0` inclus et `n` exclus.

```
def voisin_alea(voisins, s):
    ...
```

On donne la fonction `marche_alea` suivante :

```
1 def marche_alea(voisins, i, n):
2     if n == 0:
3         return i
4     return marche_alea(voisins, voisin_alea(voisins, i), n-1)
```

5. Justifier que la fonction `marche_alea` est une fonction récursive.

6. Décrire ce que modélise cette fonction, en rapport avec le contexte de l'exercice.

7. Compléter la fonction `simule` qui simule `n_tests` fois le déplacement d'un virus pendant `n_pas` étapes, démarrant au sommet `i`, et qui renvoie une liste contenant en position `j` le nombre de fois que le virus a terminé son parcours au sommet `j`, divisé par `n_tests`.

```
1 def simule(voisins, i, n_tests, n_pas):
2     results = [0] * len(voisins)
3     ...
```

8. L'appel `simule(voisins, 4, 1000, 1000)` renvoie la valeur suivante : `[0.328, 0.195, 0.18, 0.12, 0.059, 0.118]`. Déduire de ce résultat l'ordinateur du réseau qu'il est le plus rentable de protéger.

Au début, on suppose que le virus n'est présent que sur un ordinateur. À chaque étape, il contamine tous ses voisins non déjà contaminés. On cherche à savoir combien de temps prend ce virus pour se propager à tout le réseau.

9. Un graphe `voisins` représente un réseau, et `s` représente un sommet de départ. Proposer un algorithme pour déterminer le temps, en étape, que met un virus à se propager dans l'intégralité d'un réseau.