TD 2 : Bellman-Ford

On considère un graphe $G = (V, E)$. Étant donné une arrête $e = (u, v) \in E$ on note $poids(e)$ (ou $poids(u, v)$) le poids de l'arrête.

On propose un premier algorithme naïf :

```
début
    pour v ∈ V \ {u} faire
        si v = u alors
            d[v] = 0
        sinon si (u, v) ∈ E alors
            d[v] = poids(u, v)
        sinon
            d[v] = ∞
        fin
    fin
    pour i = 1 à n faire
        pour (v, u) ∈ E faire
            Envoie d au nœud v
        fin
        pour (u, v) ∈ E faire
            Recoit d_v du nœud v
            pour w ∈ V faire
                si d_v[w] + d[v] < d[w] alors
                    d[w] = d_v[w] + d[v]
                fin
            fin
        fin
    fin
fin
```

Algorithme 1 : Algorithme de Bellman-Ford distribuée pour un sommet $u \in V$

On cherche à implémenter cet algorithme en MPI dans le fichier `Bellman_Ford.java` fourni. La code fourni récupère les voisins du sommet correspondant au rang

de la tâche MPI depuis un fichier et fourni le poids des arrêtes correspondantes, dans le dictionnaire `poids`.

1. On considère dans un premier temps que les arcs sont non orientés. Implémentez l'algorithme.
2. Dans le cas d'un graphe orienté, on connaît grâce au dictionnaire `poids` les voisin du sommet correspondant au rang de la tâche, mais on ne sait pas de quels sommets ce dernier est voisin. Proposez une solution permettant à la tâche de récupérer cette information (l'implémentation n'est malheureusement pas possible avec MPJ express).
3. On va vouloir observer ce qu'il se passe si le poids d'une arrête augmente à un moment donné. Implémentez une solution permettant d'arrêter le programme MPI une fois que l'algorithme a convergé (le tableau des distances de chaque nœud est constant sur 2 itérations de la boucle). Pour se simplifier la vie, on pourra considérer une implémentation utilisant des communications ignorant la structure du graphe.
4. Faîtes en sorte que le poids d'un des arcs augmentent avant la convergence initiale de l'algorithme. Comparez le résultat obtenu au résultat attendu (celui qu'on obtiendrait si les poids finaux avaient été fournis initialement à l'algorithme), on devrait observer qu'avec l'algorithme naïf fourni, le changement de poids n'a pas exactement l'impact attendu sur le résultat final.
5. Proposez et implémentez une solution. L'algorithme ainsi obtenue converge-t'il toujours ?
6. (OPTIONNEL) Implémentez la variante split-horizon de l'algorithme de Bellman-Ford pour éviter les routing loop.
7. Similairement, on souhaite proposer une solution dans le cas où un nœud devient inatteignable. Modifiez votre implémentation pour qu'un nœud détecte qu'un de ses voisin ne répond plus et modifie donc la distance vers ce voisin à ∞.