Question 3

L'algorithme de Floyd que nous avons vu en classe donne la longueur du plus court chemin entre toutes les paires de noeuds mais ne permet pas de retrouver les plus courts chemins. Montrer comment modifier l'algorithme afin qu'il mémorise l'information permettant de retrouver efficacement (temps $O(n)$) le plus court chemin entre deux noeuds quelconque du graphe.

Nous savons que l'algorithme de Floyd est :

$$D_{\text{k}}[i,j] = Min(D_{\text{k-1}}[i.j].D_{\text{k-1}}[i,k] + D_{\text{k-1}}[k,j])$$

Or, afin de mémoriser l'information sur le plus court chemin entre deux noeurs quelconque du graphe, il faut ajouter l'équation suivante :

$$P_k[i,j] = \begin{cases} P_{k-1}[k,j] & \text{si } D_k[i,j] \neq D_{k-1}[i,j] \\ P_{k-1}[i,j] & \text{sinon} \end{cases}$$

Cette matrice contient le plus court chemin entre deux noeuds quelconque. Cependant, afin d'obtenir ce chemin, il faut ajouter cet algorithme à la matrice P ainsi complétée :

Algorithme 1 : Trouver le plus court chemin avec Floyd

Données : $P[1 \dots n, 1 \dots n], entier\ i, entier\ j$
Résultat : *tableau d'entier chemin*

```
1 si P[i][j] = null alors
2 |   retourner
3 fin
4 tant que i ≠ j faire
5 |   j ← P[i][j]
6 |   chemin.append(j)
7 fin
```

Note : la donnée i est le noeud de départ et la donnée j est le noeud de fin