

TP PC : algorithme de Floyd-Warshall

On donne l'algorithme de Floyd-Warshall : c'est un algorithme de recherche de PCC entre tout couple de sommets dans un graphe pondéré supposé sans circuit de poids négatif.

Algorithm 1 Algorithme de Floyd-Warshall

```
1 : Entrée : un graphe orienté  $G = (V, E)$  avec poids  $w(u, v)$  sur les arêtes
2 : Sortie : la matrice  $D$  des plus courts chemins entre tous les sommets
3 : Initialiser  $D[i, j] \leftarrow w(i, j)$  si  $(i, j) \in E$ , sinon  $+\infty$ 
4 :  $D[i, i] \leftarrow 0$  pour tout  $i \in V$ 
5 : for  $k = 1$  to  $n$  do
6 :   for  $i = 1$  to  $n$  do
7 :     for  $j = 1$  to  $n$  do
8 :        $D[i, j] \leftarrow \min(D[i, j], D[i, k] + D[k, j])$ 
9 :     end for
10 :   end for
11 : end for
```

Il s'agit bien d'un algorithme programmation dynamique : si on sait aller du point i au point j en n'utilisant que des sommets entre 0 et $k - 1$, on vérifie si ce ne serait pas plus judicieux de se rendre d'abord au sommet k en partant de i puis de faire un trajet de k à j . Il faut alors choisir la meilleure option.

Les graphes pondérés sont donnés sous la forme de matrices de pondération : le coefficient d'indice (i, j) indique le poids de l'arc $i \rightarrow j$ si cet arc existe et vaut $+\infty$ sinon. Par convention dans ce TP, on indique que la pondération (i, i) vaut 0 mais certains auteurs préfèrent mettre cette distance à $+\infty$.

Question 1.

On donne

```
1 # Exemple 'd'utilisation'
2 INF = float('inf')
3 graph = [
4     [0, 3, INF, 5],
5     [2, 0, INF, 4],
6     [INF, 1, 0, INF],
7     [INF, INF, 2, 0]
8 ]
```

Dessiner le graphe correspondant.

Question 2.

Écrire la fonction `floyd_warshall(graph)` qui prend en paramètre un graphe donné sous forme de matrice de pondérations et retourne deux matrices `(dist, pred)` de mêmes dimensions :

- `dist[i][j]` = longueur d'un plus court chemin de `i` à `j`
- `pred[i][j]` = prédécesseur de `j` sur un plus court chemin de `i` à `j`

```

1 dist, pred = floyd_warshall(graph)
2
3 print("Matrice des plus courts chemins (distances) :")
4 for row in dist:
5     print(row)
6
7 print("\nMatrice des prédécesseurs :")
8 for row in pred:
9     print(row)

```

Matrice des plus courts chemins (distances) :

```

[0, 3, 7, 5]
[2, 0, 6, 4]
[3, 1, 0, 5]
[5, 3, 2, 0]

```

Matrice des prédécesseurs :

```

[None, 0, 3, 0]
[1, None, 3, 1]
[1, 2, None, 1]
[1, 2, 3, None]

```

Question 3.

Écrire la fonction `reconstruire_chemin(pred, i, j)` qui prend en paramètres une matrice de prédécesseurs et deux sommets `i` et `j`. La fonction renvoie sous forme de liste un PCC du premier au second.

```

1 print("\nExemple : chemin de 0 à 2")
2 print(reconstruire_chemin(pred, 0, 2))

```

Exemple : chemin de 0 à 2
[0, 3, 2]