TP16: Floyd-Warshall

Dans tout le TP, on suppose que les graphes n'ont pas de cycles négatifs, et sont sans boucles.

Exercice 1: Floyd-Warshall

Soit $G = (S, A, w)$ un graphe orienté pondéré. On suppose que $S = \{0, 1, \dots, n-1\}$. On rappelle que la **matrice** des distances de G est une matrice D de dimensions $n \times n$ telle que pour $i, j \in [\![0, n-1]\!]$, D_{ij} est la longueur du plus court chemin de i à j (ou $+\infty$ si aucun tel chemin n'existe).

En OCaml, la valeur flottante `Float.infinity` représente $+\infty$.

Q1. Créer un fichier OCaml `floyd_warshall.ml`. Y définir le type OCaml suivant pour représenter les graphes pondérés par **matrice d'adjacence**:

```
1 type graphe = float array array
```

On rappelle l'existence de la fonction `Array.create_matrix`:

```
1 (* Array.create_matrix n m x crée une matrice de n lignes et m colonnes,
2    dont toutes les cases valent x. *)
3 Array.create_matrix : int -> int -> 'a -> 'a array array
```

Q2. Écrire une fonction `distances : graphe -> float array array` calculant la matrice des distances d'un graphe, à l'aide de l'algorithme de Floyd-Warshall.

Q3. (Facultatif) Écrire une deuxième version de la fonction `distances` utilisant un espace $\mathcal{O}(n^2)$ en ne gardant que les deux dernières matrices calculées en mémoire à chaque étape.

Exercice 2: Reconstruction de chemin

Voyons comment reconstruire des plus courts chemins dans le graphe à partir des matrices successives générées par l'algorithme de Floyd-Warshall.

Soient $i, j \in [\![0, n-1]\!]$ et $k \in [\![0, n]\!]$.
Notons C_{ij}^k la longueur d'un plus court chemin de i à j dont les sommets internes sont tous d'indice au plus $k-1$. On sait que pour $k \geq 1$:

$$C_{ij}^{k+1} = \min(C_{ij}^k, C_{ik}^k + C_{kj}^k)$$

Ainsi, pour construire un plus court chemin de i à j ne passant que par des sommets d'indice $\leq k$:

- Si $C_{ij}^k < C_{ik}^k + C_{kj}^k$, alors on construit un PCC de i à j ne passant que par des sommets d'indice $\leq k-1$
- Sinon, on construit un PCC de i à k ne passant que par des sommets d'indice $\leq k-1$, notons le C_1. Puis, on construit un PCC de k à j ne passant que par des sommets d'indice $\leq k-1$, notons le C_2. On concatène alors les deux chemins $C_1.C_2$.

Le procédé décrit ci-dessus est récursif: on le complète avec le cas de base: pour construire un chemin de i à j ne passant par aucun sommet intermédiaire, on renvoie le chemin $i \to j$ si (i, j) est un arc, et on échoue sinon.

Q1. Modifier la fonction `distances` pour qu'elle renvoie **toutes** les matrices $C^0, \ldots, C^n$ construites par l'algorithme de Floyd-Warshall, sous forme d'un tableau de matrices.

Q2. Écrire une fonction récursive de reconstruction:

```
(* construire c i j k
   Entrée:
   - c tableau des matrices calculées par Floyd-Warshall
   - i, j deux indices de sommets
   - k un numéro de matrice (k = 0, 1, ... n avec n la taille du graphe)
   Sortie:
   l liste de sommets formant un plus court chemin de i vers j ne passant
     que par des sommets de 0, ..., k-1 *)
construire : float array array array -> int -> int -> int -> int list
```

Q3. En déduire une fonction `plus_court_chemin : graphe -> int -> int -> int list` calculant un plus court chemin d'un sommet à un autre dans un graphe.

Exercice 3: Diamètre

Définition 1. Soit $G = (S, A, w)$ un graphe orienté pondéré. Pour $u, v \in S$, on note $d(u, v)$ la distance de u à v, c'est à dire le poids d'un plus court chemin de u à v, ou $+\infty$ s'il n'existe pas de chemin de u à v.
Le **diamètre** de G est la longueur du plus long des plus courts chemins entre tout couple de sommets. Autrement dit:

$$\textbf{diam}(G) = \max\{d(u, v) \mid u, v \in S\}$$

En particulier, le diamètre d'un graphe vaut $+\infty$ s'il n'est pas fortement connexe.

Dans cet exercice, on considère des graphes pondérés par des **entiers**. On suppose de plus que les poids des graphes manipulés sont assez petits pour pouvoir considérer que 10^6 vaut $+\infty$:

```c
#define INFINI 1000000
```

Q1. Comment calculer le diamètre d'un graphe à l'aide de l'algorithme de Floyd-Warshall ?

Q2. Créer un fichier C, et y recopier le type suivant pour les graphes représentés par matrice d'adjacence:

```c
typedef struct {
  int n;
  int** mat; // mat[i][j] = poids de l'arc (i, j)
} graphe_t;
```

Q3. Implémenter une fonction C calculant le diamètre d'un graphe.

Définition 2. Le modèle de graphe aléatoire d'Erdős et Rényi permet de générer des graphes selon la loi de probabilité suivante. Pour $n \in \mathbb{N}$ et $p \in [0; 1]$, le graphe $G(n, p)$ est un graphe **non-orienté non pondéré** à n sommets, sans boucles, où chacune des $\binom{n}{2}$ arêtes existe avec une probabilité p, toutes les arêtes étant indépendantes.

On verra les graphes non-pondérés comme des graphes pondérés où chaque arête a un poids de 1.

Q4. Écrire une fonction `graphe_t* GER(int n, float p)` générant un graphe aléatoire selon le modèle d'Erdős et Rényi. N'oubliez pas que les graphes sont non-orientés, et *pensez à initialiser l'aléatoire dans le main !*

On aimerait déterminer empiriquement le diamètre moyen d'un graphe $G(n, p)$. Plus précisément, à p fixé, on se propose de tracer l'évolution de $\mathbb{E}(\textbf{diam}(G(n, p)))$ en fonction de n.

Q5. Écrire une fonction `void print_diametres(float p, int K, int n_max)`. Cette fonction itère sur $n = 5, 10, 15, \ldots, n_{max}$ et, pour chaque n, créer K graphes aléatoires $G(n, p)$ et calcule leurs diamètres. Pour chaque graphe généré, le programme affichera la valeur de n et le diamètre trouvé sur une ligne:

```
5 2
5 3
10 7
10 3
...
```

Q6. Stocker les valeurs générées avec $K = 5, n_{max} = 100$ et $p = 0, 0.05, 0.1, 0.15 \ldots, 1$ dans un grand fichier texte. Les récupérer dans un programme Python et tracer les nuages de points obtenus pour chaque valeur de p.

Q7. Vérifier empiriquement que le diamètre du graphe $G(n, p)$ semble tendre vers 2 quand $n \to +\infty$.

Exercice 4: Limite du diamètre moyen d'un graphe aléatoire

Montrons le résultat expérimental de l'exercice précédent, à savoir:

$$\lim_{n \to +\infty} \mathbb{P}(\textbf{diam}(G(n,p)) = 2) = 1$$

On rappelle que $G(n, p)$ est un graphe aléatoire non-orienté de n sommets, où chaque arête existe avec une probabilité p, et ce de manière indépendante.

Q1. A quelle condition a t-on que le diamètre d'un graphe vaut 1 ? En déduire la probabilité que le diamètre soit au moins 2, et la limite de cette probabilité quand $n \to +\infty$.

Q2. Soient $u, v \in S$ avec $u \neq v$. Montrer que $\mathbb{P}(d(u, v) > 2) \leq (1 - p^2)^{n-2}$

Q3. Montrer que $\mathbb{P}(\textbf{diam}(G(n,p)) \leq 2)$ tend vers 1, et conclure.