UNIVERSITÉ D'ORLÉANS **M1 Info**

Informatique Année **2021 - 2022**

Contrôle : Programmation Parallèle - 2h

Soit un ensemble de navires dont on a relevé la trajectoire entre un port de départ et d'arrivée sur une année (plusieurs voyages par navire). Chaque trajectoire est simplement le relevé d'une position (x, y) tous les pas de temps (1h par exemple) lors du voyage. On souhaite étudier la similarité de ces trajectoires afin de les classer en sous ensembles et analyser ainsi les différentes familles de chemins empruntés.

On considère qu'on a généré N trajectoires dont les longueurs peuvent être différentes car la durée du voyage n'est pas toujours la même pour les navires.

Pour calculer la similarité entre deux trajectoires T_1 et T_2 de longueur respective n_1 et n_2 on utilise la distance de Fréchet discrète. La distance de Fréchet entre deux trajectoires consiste à calculer la plus petite longueur nécessaire à une laisse pour qu'un promeneur et son chien puissent avancer ensemble chacun sur une des 2 trajectoires comme illustré Figure 1. Une trajectoire est une ligne brisée (courbe polygonale) et représentée par une suite de points. Le

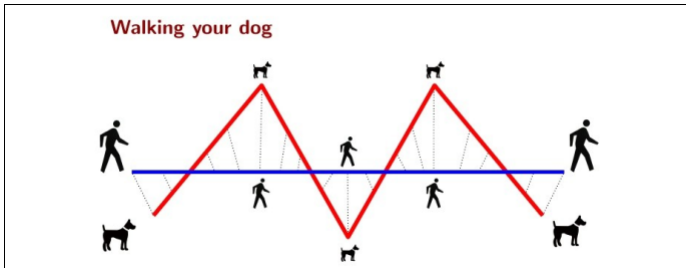

Figure 1: Distance de Fréchet par l'exemple du promeneur et de son chien

principe du calcul de cette distance repose sur le calcul d'une matrice de distances MD qui permet de stocker les distances entre tous les couples de points de deux trajectoires. Ainsi si $pts_i^T = (x_i^T, y_i^T)$ désigne le ième point de la trajectoire T,

$$MD_{ij} = dist(pts_i^{T_1}, pts_j^{T_2}) = \sqrt{(x_i^{T_1} - x_j^{T_2})^2 + (y_i^{T_1} - y_j^{T_2})^2}$$

À partir de cette matrice, le calcul de la distance de Fréchet consiste à choisir la manière de parcourir la matrice de distances de la case MD_{00} en haut à gauche à $MD_{(n_1-1)(n_2-1)}$ en bas à droite sans retour en arrière. Ce parcours décrit comment le promeneur et le chien se déplacent sur leur trajectoire. Si pour chaque parcours on calcule la plus grande distance entre 2 points (il s'agit de la longueur de la laisse pour ce parcours) la distance de Fréchet est donnée par le minimum de toutes ces distances.

Soit l'exemple illustré Figure 2 avec un promeneur qui effectue une trajectoire composée de 7 points pendant que son chien suit une autre trajectoire composée de 6 points. La première ligne de la matrice MD correspond à la longueur de la laisse entre le promeneur qui reste sur le premier point de sa trajectoire alors que le chien avance sur sa trajectoire.

MD			chien							
			num	0	1	2	3	4	5	6
			x	0,3	3,2	3,8	5,2	6,5	7	8,9
promeneur			y	1,6	3	1,8	3,1	2,8	0,8	0,6
num	x	y								
0	0,2	2		0,412	3,162	3,606	5,120	6,351	6,905	8,812
1	1,5	2,8								
2	2,3	1,6								
3	2,9	1,8								
4	4,1	3,1								
5	5,6	2,9								
6	7,2	1,3								
7	8,2	1,1								

Figure 2: Une matrice de distances avec un promeneur sur une trajectoire de 7 points et son chien sur 6 points.

La figure 3 illustre la matrice de distance complètement remplie et également quel est le parcours (les cases sur fond jaune) qui permet de déterminer la distance de Fréchet finale soit 1.697. Ce parcours peut être lu comme le promeneur et le chien commencent au début de la trajectoire puis le promeneur avance d'un point alors que le chien ne bouge pas. Ensuite tous les 2 avancent d'un point et ainsi de suite.

			chien						
			num	0	1	2	3	4	5	6
			x	0,3	3,2	3,8	5,2	6,5	7	8,9
promeneur			y	1,6	3	1,8	3,1	2,8	0,8	0,6
num	x	y								
0	0,2	2		0,412	3,162	3,606	5,120	6,351	6,905	8,812
1	1,5	2,8		**1,697**	1,712	2,508	3,712	5,000	5,852	7,720
2	2,3	1,6		2,000	1,664	1,513	3,265	4,368	4,768	6,675
3	2,9	1,8		2,608	1,237	0,900	2,642	3,736	4,220	6,119
4	4,1	3,1		4,085	0,906	1,334	1,100	2,419	3,701	5,412
5	5,6	2,9		5,457	2,402	2,110	0,447	0,906	2,524	4,022
6	7,2	1,3		6,907	4,346	3,437	2,691	1,655	0,539	1,838
7	8,2	1,1		7,916	5,349	4,455	3,606	2,404	1,237	0,860

Figure 3: La matrice de distances complète avec le parcours (cases sur fond jaune) permettant de définir la distance de Fréchet.

Pour calculer la distance de Fréchet à partir de la matrice de distances il existe un algorithme qui évite de calculer tous les parcours possibles mais on ne rentrera pas dans les détails. Cependant il est possible de réduire les $n_1 \times n_2$ calculs de distances dans la matrice MD en garantissant que cet algorithme fonctionne. Pour cela, on utilise comme référence le parcours de la quasi diagonale. Comme la matrice MD est rectangulaire on parle de quasi diagonale et ça correspond à la diagonale principale plus les cases supplémentaires pour parcourir les points restants sur la trajectoire la plus longue. Ce parcours correspond donc au promeneur et au chien qui avancent d'un point à chaque étape et au promeneur ou au chien qui termine sa trajectoire.

Soit d_d la plus grande distance de la quasi diagonale, il est clair que la distance de Fréchet est inférieure ou égale à d_d. Donc lorsqu'on calcule le reste de la matrice de distance on peut arrêter le calcul d'une colonne sous la diagonale ou d'une ligne au-dessus de la diagonale, dès que la distance est supérieure à d_d[1]. Cette optimisation est illustrée Figure 4 avec les cases en gris foncé qui indiquent la dernière distance calculée sur une ligne ou une colonne et les cases en gris clair qui indiquent les distances qui ne seront pas calculées avec cette optimisation. La quasi diagonale est marquée par les cases en fond jaune et la valeur de la laisse pour ce parcours est indiquée en gras.

			chien						
			num	0	1	2	3	4	5	6
			x	0,3	3,2	3,8	5,2	6,5	7	8,9
promeneur			y	1,6	3	1,8	3,1	2,8	0,8	0,6
num	x	y								
0	0,2	2		0,412	3,162	3,606	5,120	6,351	6,905	8,812
1	1,5	2,8		1,697	1,712	2,508	3,712	5,000	5,852	7,720
2	2,3	1,6		2,000	1,664	1,513	3,265	4,368	4,768	6,675
3	2,9	1,8		2,608	1,237	0,900	**2,642**	3,736	4,220	6,119
4	4,1	3,1		4,085	0,906	1,334	1,100	2,419	3,701	5,412
5	5,6	2,9		5,457	2,402	2,110	0,447	0,906	2,524	4,022
6	7,2	1,3		6,907	4,346	3,437	2,691	1,655	0,539	1,838
7	8,2	1,1		7,916	5,349	4,455	3,606	2,404	1,237	0,860

Figure 4: Illustration de l'optimisation de la quasi diagonale pour notre exemple.

[1] On supposera que les lignes et les colonnes de la matrice sont croissantes.

Exercice 1. La similarité de trajectoires : Calcul de la matrice de distances (12pts)

Sans l'optimisation de la quasi diagonale, le calcul de la matrice des distances peut être implémenté de la manière suivante

Code 1: Implémentation de la matrice de distances en séquentiel

```
1  double dist_e(double* v1, double* v2) {
2      return sqrt((v2[0]-v1[0])*(v2[0]-v1[0]) + (v2[1]-v1[1])*(v2[1]-v1[1]));
3  }
4
5  double* MD(double* traj1, int n1, double* traj2, int n2) {
6      double* distances = new double[n1*n2];
7      for (int i=0; i<n1; i++)
8          for (int j=0; j<n2; j++)
9              distances[i*n2+j] = dist_e(traj1+2*i, traj2+2*j);
10         return distances;
11 }
```

1. Proposez une directive OpenMP pour paralléliser ce calcul.
   - Décrivez comment cette directive répartit les calculs sur une équipe de threads.
   - Est ce qu'il y a un intérêt à utiliser la clause **schedule** avec le paramètre **dynamic** ?
2. Donnez une nouvelle version séquentielle de la fonction MD avec l'optimisation de la quasi diagonale.
3. À partir de votre version séquentielle de l'optimisation de la quasi diagonale, indiquez les directives OpenMP à ajouter pour paralléliser les calculs.
   - Si vous utilisez la clause **collapse** expliquez la parallélisation obtenue.
   - Est-il intéressant d'utiliser la clause **schedule(dynamic,x)** ? Si oui quels critères utiliser pour choisir `x` ?
4. Proposez une nouvelle parallélisation en utilisant la directive **#pragma omp task**. Quel est l'intérêt d'utiliser cette directive ?
5. Décrivez une parallélisation possible du calcul de la matrice MD avec l'optimisation de la quasi diagonale pour une architecture à mémoire distribuée sachant qu'un processus `root` dispose des deux trajectoires dont on souhaite calculer la distance de Fréchet. Vous pouvez faire un schéma décrivant la parallélisation en indiquant :
   - les données reçues par chaque processus;
   - les calculs effectués (un morceau de la matrice de distances) par chaque processus;
   - les communications nécessaires et avec quelles routines MPI elles seront réalisées;
   - comment le processus `root` obtient la matrice de distances complète.

   Pour cette partie avec MPI, aucun code n'est demandé mais des explications clairement rédigées et illustrées.

Exercice 2. La similarité de trajectoires : Détermination de la similarité (8pts)

Deux trajectoires sont similaires si leur distance de Fréchet est inférieure à un seuil ϵ donné. A partir du jeu de données initial, les N trajectoires de navires, on souhaite définir une matrice S triangulaire inférieure stricte de taille $N \times N$ telle que S_{ij} avec $1 \leq i \leq N-1$ et $0 \leq j < i$ est vrai si la trajectoire T_i et T_j sont similaires.

Soit une machine à mémoire distribuée de $nprocs$ processeurs où chaque processeur dispose de $ncores$ cœurs. On suppose que le jeu de données est disponible sur un processeur `root`.

1. Comment allez vous répartir les trajectoires sur les différents processeurs ? Sera-t-il nécessaire de communiquer des trajectoires entre les processeurs après cette distribution initiale ?

2. Pour cette machine, il peut être intéressant d'utiliser de la programmation hybride. Dans ce contexte, proposez une parallélisation du calcul de la matrice S sachant que le calcul final de la distance de Fréchet de 2 trajectoires sera fait en séquentiel à partir de la matrice des distances.

3. Quels sont les éléments qui peuvent influencer le temps de calculs (optimisation des communications, efficacité des directives OpenMP, ...) ?