Traitement du graphe temporel CityBike avec Spark-GraphX

Saison 2022-2023

Composante : ESIR

Spécialité : Technologies de l'information option informatique 3ème année

Module : Data management for big data

Chargée de TP : Maria Massri, maria.massri@irisa.fr

Nous allons découvrir le traitement de graphe en utilisant la librairie GraphX. Cette librairie offre une extension des RDDs et une abstraction de graphe. Les graphes traités par GraphX appartiennent à la famille des graphes dirigés ayant des propriétés sur les noeuds et les relations (Connus sous le nom de Graphes de Propriétés). Le traitement de graphe se fait avec des opérateurs de graphes et une variante de l'API Pregel (Google's Pregel graph system). En plus, GraphX offre des algorithmes de graphes *Built-in* qui facilitent le développement des applications de traitement de graphe et rendent les instructions plus simples et moins verbeuses.

1 Préparation des données

1. Téléchargez le jeux de données en utilisant ce lien : `https://s3.amazonaws.com/tripdata/JC-202112-citibike-tripdata.csv.zip`.
2. Créez un graphe dont les noeuds représentent des stations de vélos et les relations représentent des trajets de vélos entre deux stations. Pensez à créer deux classes `Station` et `Trip` contenants les attributs des stations et trajets, respectivement.

PS : Pensez à convertir les dates en nombre total de millisecondes depuis 1er janvier 1970 (Epoch) avec la fonction timeToLong.

2 Calcul de degré

1. Extrayez le sous-graphe dont les intervalles temporelles de ces trajets (relations) existent entre 05-12-2021 et 25-12-2021. (subgraph)
2. Calculez le nombre total de trajets entrants et sortants de chaque station et affichez les 10 stations ayant plus de trajets sortants et ceux ayants plus de trajets entrants. (aggregateMessages)

3 Proximité entre les stations

1. Trouvez et affichez la station la plus proche de la station **JC013** tel que la distance du trajet (relation) entre les deux stations soit minimale. (aggregateMessage) (Pour le calcul de distance, utilisez la fonction `getDistKilometers`).
2. Trouvez et affichez la station la plus proche de la station **JC013** tel que la durée du trajet (relation) entre les deux stations soit minimale. (aggregateMessage)

4 Plus court chemin (Bonus)

Nous allons calculer les plus courts chemins allants du noeud **JC013** vers tous les autres stations du graphe. Par définition, le plus court chemin est celui dont la somme des attributs de ses relations est minimale. Dans certains cas, nous souhaitons minimiser la distance totale parcourue. Dans autres cas, nous souhaitons minimiser la durée totale écoulée. A vous de choisir l'une de ces deux métriques et comparer le résultat avec celui obtenu par un autre groupe ayant choisit une autre métrique. Pensez à créer une nouvelle classe `StationWithDistance` **OU** `stationWithDuration` qui contient les attributs des stations avec la distance/durée minimale.

1. Calculez, pour chaque station du graphe, la distance **OU** durée minimale du chemin qui le sépare de la station source ayant l'identifiant **JC013**. (Pregel)
2. Affichez les 10 stations les plus proches de la station **JC013** et les distances **OU** durées des chemins qui les séparent.
3. Comparez le résultat obtenu avec les autres groupes.

Nous avons calculer la distance **OU** durée minimale des plus courts chemins. Dans cette partie Bonus, nous demandons d'afficher non seulement la distance minimale mais aussi tous les stations de chaque chemin.

1. Trouvez, pour chaque station du graphe, la distance **OU** durée minimale et les stations du plus court chemin allant de **JC013**.
2. Affichez les 10 plus courts chemins allant de la station **JC013** ainsi que leurs distances **OU** durées et les stations parcourues.