Домашняя работа №3. Дедлайн 25 марта в 17:59

Поиск кратчайшего пути в графе

В этом задании вам предстоит реализовать алгоритм поиска кратчайшего пути в графе, используя алгоритм Breadth-first search. Алгоритм BFS, в отличие от алгоритма Depth-first search, исследует все возможные пути от целевой вершины одновременно.

Данные

Вам дан слепок графа социальной сети Twitter в формате

user_id<TAB>follower_id

Датасет представлен в двух частях (далее указаны пути в HDFS):

- /datasets/twitter/twitter_sample_small.tsv (660 КБ, 54485 ребер, 54161 вершина)
- /datasets/twitter/twitter.tsv (64 МБ, 5432909 ребер, 100000 вершин)

Задача

Необходимо реализовать программу на Spark, которая будет искать кратчайший путь (кратчайшие пути) между двумя заданными вершинами графа.

Заметьте, что в графе могут быть циклы, поэтому наличие механики останова по достижению max_path_length очень важна.

Результат

Программа должна сохранять полученные кратчайшие пути в HDFS в формате CSV. Например, если необходимо найти кратчайший путь между вершинами 1 и 99, то возможный вывод программы может быть таким:

1,5,14,45,99
1,5,16,88,99

Никаких пробелов в строчках с ответами быть не должно, ни между числами, ни до, ни после.

Вариант задания

Вам необходимо найти кратчайший путь между вершинами 12 и 34. Датасет twitter_sample_small.tsv дан для экспериментов. Чекер будет проверять правильный ответ на датасете twitter.tsv. Для самопроверки приводим правильные ответы на меньших датасетах: twitter_sample_small.tsv - 12,422,53,52,107,20,23,274,34

Оформление задания

Создайте в вашем приватном репозитории подпапку projects/3. Поместите туда программу shortest_path.py, которую чекер будет запускать следующим образом:

```
PYSPARK_PYTHON=/opt/conda/envs/dsenv/bin/python spark3-submit \
    --master yarn \
    --name checker \
    projects/3/shortest_path.py 12 34 /datasets/twitter/twitter.tsv hw3_output
```

Ваша программа должна принимать 4 параметра:

- начальный узел
- конечный узел
- путь к датасету
- путь к директории с ответами

Проверка

Запустите чекер командой **`checker.sh 3`**.

Чекер запустит ваш скрипт и сравнит ответы (пути между вершинами) с референсными. Решение зачитывается, если и ваши ответы правильные и их число совпадает с числом правильных ответов. Если у вас в файле больше ответов, чем надо, то выдается соответствующее сообщение об ошибке. Если у вас меньше правильных ответов, чем в референсном решении, то выдается сообщение об ошибке. Для ориентации, чекер выдаст в лог количество ответов в вашем файле. Количество ответов в референсном файле не выдается.

Добейтесь успешного прохождения чекера (PASSED 1)

Особенности и замечания

1. Заметьте, что формат графа инвертирован, поэтому исходными вершинами ребер будут значения во втором столбце

2. Ответ на самом большом датасете отличается от ответов на маленьком как по значениям, так и по длине

3. Вам могут понадобиться какие-нибудь нестандартные функции. Не забывайте, в первую очередь, обратиться к документации по Spark 3.3.0, установленном на кластере.

4. Для самопроверки вы можете пользоваться сторонними библиотеками. Однако финальное решение не должно использовать никаких сторонних библиотек.

5. Ваша программа не должна работать более 5 минут даже на самом большом датасете. Если программа работает более 5 минут, то стоит подумать как ее оптимизировать (например, не обсчитывать заведомо длинные пути)

Как обрабатывать параметры программы в Python?

Параметры командной строки можно получить в программе на Python из переменных: sys.argv[1], sys.argv[2] и т.д. Заметьте, что sys.argv[0] - это имя самой запускаемой программы.

Как инициализировать SparkSession в отдельной программе?

```
from pyspark import SparkContext, SparkConf

conf = SparkConf()
sc = SparkContext(appName="Pagerank", conf=conf)
```