Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Primer Semestre de 2023

IIC 2440 – Procesamiento de Datos Masivos
Tarea 2

1. Enunciado

En esta tarea vamos a programar algoritmos de Graph Analytics pensando en ejecutarlos en un entorno distribuido. En concreto, vamos a programar los algoritmos de Page Rank y Single Source Shortest Path utilizando la estructura RDD de PySpark, junto con las funciones asociadas a este tipo de objetos.

1.1. Page Rank en un entorno distribuido

Una opción para computar Page Rank en un entorno distribuido es programarlo como un algoritmo en que los nodos se comunican mensajes entre ellos. Por ejemplo, veamos el grafo de la Figura 1.

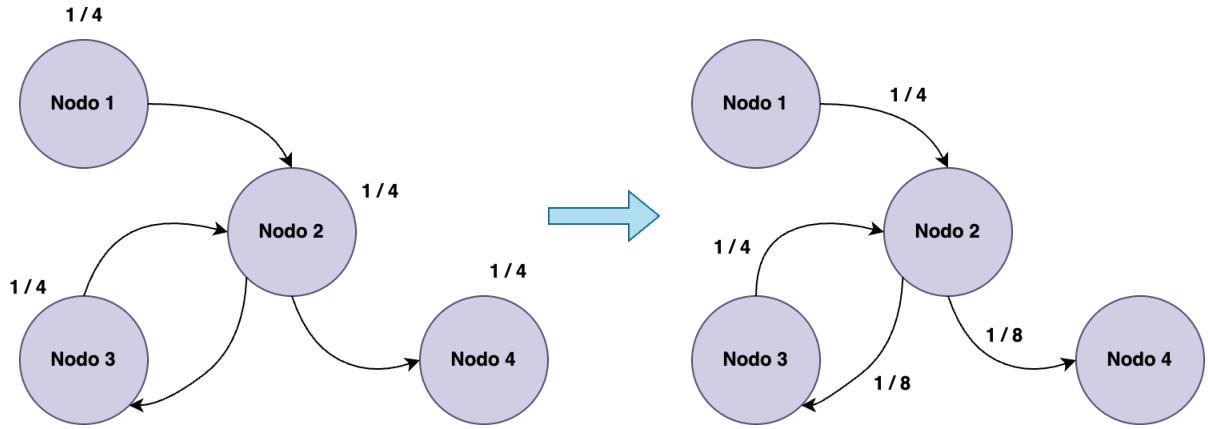

Figura 1: Page Rank calculado como nodos comunicando mensajes a sus vecinos.

En este grafo, todos los nodos parten con el mismo Page Rank que es igual a $\frac{1}{4}$, que equivale a 1 dividido en el número de nodos. Luego, cada nodo envía a sus vecinos el Page Rank correspondiente, donde notamos que si un nodo tiene más de un vecino, su Page Rank "se reparte" en partes iguales. Luego, al recibir todos los mensajes, cada nodo debe hacer una operación de tipo reduce en la que sumamos todos los mensajes recibidos para calcular el Page Rank recibido por cada nodo. Finalmente, para actualizar el valor de Page Rank de cada nodo, recordemos que hay que multiplicar la suma de los mensajes por el damping factor d, además de sumar $\frac{(1-d)}{n}$ en cada nodo.

Así, considera que partes con un RDD que representa una lista con los nodos y un RDD que representa una lista de tuplas indicando las aristas dirigidas. En el grafo del ejemplo tendríamos:

```
nodes = [1, 2, 3, 4]
edges = [(1, 2), (2, 3), (2, 4), (3, 2)]
```

Con esto en cuenta, tienes que resolver el siguiente problema.

Problema 1. Programa el algoritmo de Page Rank siguiendo la estrategia de más arriba. En concreto, deberías seguir los siguientes pasos:

1. Prepara un RDD que tenga cada nodo con su Page Rank inicial. Luego, haz una función que prepare el mensaje que cada nodo va a enviar. Pprobablemente quieras almacenar estos valores como otro RDD.
2. Escribe una función que se haga cargo del intercambio de mensajes entre nodos. Esta función envía los mensajes a los nodos correspondientes y se hace cargo del merge de los mensajes recibidos por cada nodo. Debe retornar un RDD que para cada nodo diga cuál es el mensaje final recibido.
3. Haz una función que actualice el valor de Page Rank para cada nodo considerando el damping factor. Probablemente quieras hacer una función que tome el *output* del punto anterior y lo procese.
4. Itera los pasos correspondientes por un número máximo de iteraciones, o hasta que la diferencia entre dos iteraciones del valor de Page Rank sea mínima.

El programa debe retornar cada nodo junto a su valor final de Page Rank.

1.2. Single Source Shortest Path

Uno de los algoritmos más famosos en el análisis de grafos es el que nos permite encontrar el camino óptimo entre dos nodos. Cuando hablamos de Single Source Shortest Path, es cuando queremos partir de un nodo y descubrir los caminos más cortos hacia todos los nodos que son alcanzables desde el nodo inicial.

Problema 2. Siguiendo la misma estrategia de arriba, en esta pregunta tienes que desarrollar un algoritmo con las funciones básicas de RDDs para computar los costos de los caminos más cortos partiendo de un nodo inicial **pensando en que pasas mensajes entre nodos**. Al igual que en el punto anterior, la idea es que el algoritmo que implementas se pueda ejecutar en un entorno distribuido. El output del código debe ser cada nodo, junto al costo asociado a llegar a ese nodo, considerando que partimos de un nodo inicial i indicado por el usuario. Además, en este caso el RDD de las aristas se ve de la siguiente forma:

```
edges = [(1, 2, 10), (2, 3, 3), (2, 4, 24), (3, 2, 1)]
```

En donde el tercer elemento indica el costo de la arista. Ahora, te dejamos una idea del algoritmo para que entiendas a grandes rasgos los pasos.

1. Escoge el nodo inicial, este nodo tiene costo acumulado 0 y todos los demás tienen costo acumulado infinito.
2. En cada iteración, cada nodo comunica el costo acumulado a sus vecinos. Cada nodo recibe este costo, sumado con el costo de atravesar la arista.
3. Para hacer merge de todos los mensajes dejamos el mínimo de todos los costos. Así, actualizamos cada nodo con el costo mínimo recibido solo si es menor al costo acumulado que ya tenía ese nodo.
4. Si en dos iteraciones el costo en llegar para cada nodo no cambia, entonces nos detenemos.

1.3. Una estrategia general

Ahora que ya captaste la idea, queremos que nos cuentes cómo crees que se ve una estrategia general para hacer cómputos para análisis de grafos en entornos distribuidos. Estos algoritmos se basan en la idea de que los nodos comunican mensajes a los vecinos. Así, la idea que propongas debe tener en cuenta al menos:

1. La preparación de los nodos.
2. La regla para pasar mensajes entre nodos.

3. La definición de funciones para hacer merge de varios mensajes.

4. Las condiciones de término de un algoritmo.

5. La forma de actualizar las propiedades de los nodos.

Piensa que quieres desarrollar una librería que permita hacer este tipo de cómputos para grafos arbitrarios, y que puede ayudar a resolver otro tipo de problemas.

Problema 3. En esta parte tienes que crear un video que debe durar 5 minutos, y en él nos tienes que contar cómo se ve una estrategia que generalice estos algoritmos de paso de mensajes entre nodos, y como programarías los dos algoritmos previos (Page Rank y SSSP) usando tu estrategia general.

2. Detalles académicos

Esta tarea debe resolverse en grupos de dos personas, aunque puede ser resuelta en forma individual. El formato de entrega consta de los siguientes archivos:

- Un link a un repositorio público en Github donde se encuentre todo el código necesario para correr los códigos de las dos primeras preguntas. Debes subir dos *notebooks*, uno por cada pregunta, que se puedan probar en Google Colab. **Importante**: nos interesa que nos muestres que tu código sea escalable, por lo que debes probarlos con grafos de tamaño considerable. Debes darnos las instrucciones de como correr todo en Google Colab en el Readme del repositorio.
- El link del video de la tercera pregunta. Este debe ser un video en youtube que debe explicar lo solicitado en la pregunta

Fechas. La fecha de entrega de la tarea es el 28 de Junio, a las 13:00 hrs.