Índice

Listings

1 Grafo multifichero:

En este ejercicio analizamos un grafo formado por multiples archivos y devolvemos todos sus triciclos.

1.1 Unión de rdds:

Primero unimos los rdd:

unión de rdd

```
def independent(rdds, files):
    for i in range(len(rdds)):
        print(f"file: {files[i]}")
```

1.2 Triciclos

Dado el rdd con los nodos y sus listas de adyacencia considerando solo nodos posteriores definimos la función tag que genera la lista de 'exist' y 'pending' para un nodo:

tag

```
def tag(node_adjs): # Función iterativa, no perezosa
    node = node_adjs[0]
    adjs = list(node_adjs[1])
    adjs.sort()
    result = [((node, x), 'exists') for x in adjs]
    for i in range(len(adjs)):
        for j in range(i, len(adjs)):
            result.append(((adjs[i], adjs[j]), ('pending', node)))
    return result
```

Generamos las listas de adyacencia y generamos los triciclos:

tricycles

```
def tricycles(tags):
    return tags\
            .groupByKey()\
            .filter(lambda x: len(x[1]) > 1 and 'exists' in x[1])\
            .flatMap(
                lambda line: map(
                    lambda x: (x[1], line[0][0], line[0][1]),
                    filter(lambda x: not x == 'exists', line[1])
                    )
                )
```

process data

```
def process_data(data):
    edges = get_distict_edges(data)
    node_adjs = get_node_adjs(edges)
    tags = node_adjs.flatMap(tag)
    return tricycles(tags)
```

2 Multiples grafos: