Spark_GraphX

Tutorial para Grafos usando o GraphX do Spark com linguagem Scala, no notebook Zeppelin .

Neste tutorial iremos apresentar alguns exemplos fáceis de análise de grafos usando o GraphX do Spark. Criaremos um grafo simples com cidades do estado do Rio de Janeiro a fim de entender os principais comandos.

```
// Importando as bibliotecas necessárias.
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
import org.apache.spark.graphx.GraphLoader


// Criando um RDD para os vértices. Um número como iD, o nome da cidade e uma característica da cidade.
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Array((0L, ("Magé", "Rota de passagem")), (1L, ("São Gonçalo", "Alto potencial")), (2L, ("Ara
```

```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
import org.apache.spark.graphx.GraphLoader
users: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, (String, String))] = ParallelCollectionRDD[5
74] at parallelize at <console>:71
```

```
// Criando um RDD para as arestas. Nesse exemplo as arestas serão as distâncias entre as cidades.
val relationships: RDD[Edge[Int]] =
  sc.parallelize(Array(Edge(3L, 1L, 29),    Edge(5L, 3L, 141),
                       Edge(2L, 5L, 142), Edge(5L, 1L, 119),
                       Edge(3L, 0L, 63),   Edge(1L, 0L, 40)))
```

```
relationships: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = ParallelCollectionRDD[575] at para
llelize at <console>:69
```

```
// Construindo um grafo
val graph = Graph(users, relationships)
```

```
graph: org.apache.spark.graphx.Graph[(String, String),Int] = org.apache.spark.graphx.impl.GraphImpl@7d107e6a
```

```
// Mostrando o número de Vertices (cidades) contidas no grafo.
val numCidades = graph.numVertices
```

```
numCidades: Long = 6
```

```
//Mostrando o número de estradas ligando as cidades no grafo contruído.
val numEstradas = graph.numEdges
```

```
numEstradas: Long = 6
```

```
//Mostrando as relações entre vértices externos.
graph.outDegrees.collect.foreach(println(_))
```

```
(1,1)
(5,2)
(2,1)
(3,2)
```

```
//Mostrando as relações entre vértices internos.
graph.inDegrees.collect.foreach(println(_))
```

```
(0,2)
(1,2)
(5,1)
(3,1)
```

```
//Mostrando os triangulos formados entre vértices para cada vertice.
graph.triangleCount.vertices.collect.foreach(println(_))
```

```
(4,0)
(0,1)
(1,2)
(5,1)
(2,0)
(3,2)
```

```
//Mostrando o Grafo criado
graph.vertices.collect.foreach(println(_))
```

```
(4,(Rio das Ostras,Balneário))
(0,(Magé,Rota de passagem))
(1,(São Gonçalo,Alto potencial))
(5,(Friburgo,Cidade Turística))
(2,(Araruama,Acessível))
(3,(Rio de Janeiro,Grande centro urbano))
```

```
// Agora usaremos o comando triplet, que mostra o tripleto formado por Vertice + aresta + Vertice, dos relacio
graph.triplets.map(triplet => triplet.srcAttr._1 + " está a " + triplet.attr + " Km de " + triplet.dstAttr._1)
```

```
Rio de Janeiro está a 29 Km de São Gonçalo
Araruama está a 142 Km de Friburgo
Friburgo está a 141 Km de Rio de Janeiro
Friburgo está a 119 Km de São Gonçalo
São Gonçalo está a 40 Km de Magé
Rio de Janeiro está a 63 Km de Magé
```

```
//Descobrindo quais as rotas com distâncias maiores que 100.
graph.edges.filter { case Edge(src, dst, prop) => prop > 100 }.collect.foreach(println)
```

```
Edge(2,5,142)
Edge(5,3,141)
Edge(5,1,119)
```

```
// Vamos supor que a cidade de Magé tem um bloqueio na estrada. Então vamos criar um subgrafo removendo essa c
val validGraph = graph.subgraph(vpred = (id, attr) => attr._1 != "Magé")
```

```
validGraph: org.apache.spark.graphx.Graph[(String, String),Int] = org.apache.spark.graphx.impl.GraphImpl@7323b641
```

```
// Mostrando o subgrafo válido, sem a cidade de Magé.
validGraph.vertices.collect.foreach(println(_))
```

```
(4,(Rio das Ostras,Balneário))
(1,(São Gonçalo,Alto potencial))
(5,(Friburgo,Cidade Turística))
(2,(Araruama,Acessível))
(3,(Rio de Janeiro,Grande centro urbano))
```

```
// Mostrando os tripletos válidos, ou seja, sem a cidade de Magé.
validGraph.triplets.map(triplet => triplet.srcAttr._1 + " está a " + triplet.attr + " Km de " + triplet.dstAtt
```

```
Rio de Janeiro está a 29 Km de São Gonçalo
Araruama está a 142 Km de Friburgo
Friburgo está a 141 Km de Rio de Janeiro
Friburgo está a 119 Km de São Gonçalo
```

```
// Vamos criar um Ranqueamento das cidades, incluindo a cidade de Magé, para ver qual cidade teria mais import
// Usamos o comando PageRank.
val ranks = graph.pageRank(0.0001).vertices
```

```
ranks: org.apache.spark.graphx.VertexRDD[Double] = VertexRDDImpl[769] at RDD at VertexRDD.scala:57
```

```
// Mostraremos o resultado do ranqueamento envolvemdo todos os vértices.
val ranksByUsername = users.join(ranks).map {
  case (id, (username, rank)) => (username, rank)
}
// Mostrando o resultado
println(ranksByUsername.collect().mkString("\n"))
```

```
ranksByUsername: org.apache.spark.rdd.RDD[((String, String), Double)] = MapPartitionsRDD[779] at map at <consol
e>:77
((Rio das Ostras,Balneário),0.4956871986586499)
((Magé,Rota de passagem),1.9444577330494666)
((São Gonçalo,Alto potencial),1.261725293510719)
((Friburgo,Cidade Turística),0.9170213175185022)
((Araruama,Acessível),0.4956871986586499)
((Rio de Janeiro,Grande centro urbano),0.8854212586040134)
```

Pelos resultados apresentados neste exemplo conclue-se que a Cidade de Magé possui o maior valor de ranqueamento, ou seja ela tem mais interrelações do que as outras.
Em segundo lugar está São Gonçalo.