TP8 - Correction

Découverte

```
scala> val tweetsFile = sc.textFile("posAndNeg")

scala>tweetsFile.count()

scala> tweetsFile.foreach(println)

scala> tweetsFile.take(1)

scala> var tweetsRdd = tweetsFile.flatMap(x => x.split(","))

scala> tweetsRdd.map(x => (x(0), 1)).reduceByKey(_+_).collect()

scala> tweetsPos = tweetsRdd.filter(x => x.contains("4"))

scala> tweetsNeg = tweetsRdd.filter(x => x.contains("0"))

scala> tweetsRdd.flatMap(x => x(5).split(" ").map(x=>(x,1))).reduceByKey(_+_).collect()
```

Compter les degrés

```
graphRdd.flatMap{
case(int, outList) =>
outList.map(dst => (int, 1))
}.reduceByKey((x, y) => x + y)
```

Voisins communs :

```
graphRdd.flatMap{
case(in, outList) =>
outList.map(dst => ((math.min(in, dst.toInt), math.max(in, dst.toInt)),outList))
}.reduceByKey((x, y) => x.intersect(y))
```

Connected component

```
scala> val graphFile = sc.textFile("deuxComposantes")
scala> val graphRdd=graphFile.flatMap(x => Seq((x.split(";")(0).toInt, x.split(";")(1).toInt),(x.split(";")(1).toInt, x.split(";")(0).toInt) ))
scala> var graphAdj=graphRdd.groupByKey()
scala> graphAdj.collect()
scala> var labelRdd=graphAdj.map{ case(id, neighbors) => (id, id) }
scala> labelRdd.collect()
for (i <- 0 until 5) {
val contributions = graphAdj.join(labelRdd).flatMap {
case (pageId, (g, l)) =>
g.map(dest => ( (math.min(pageId, dest),math.max(pageId, dest)), math.min(math.min(pageId, dest), l)))
}.reduceByKey(math.min)
labelRdd=contributions.flatMap{
case ((src, dst), label) => Seq((src, label), (dst, label))
}.reduceByKey(math.min)
}
```

PageRank :

```
var rankRdd = rankFile.map(x => (x.split(";")(0).toInt, x.split(";")(1).toDouble))
val graphRdd=graphFile.map(x => (x.split(";")(0).toInt, x.split(";").slice(1,x.split(";").length)))
for (i <- 0 until 10) {
val contributions = graphRdd.join(rankRdd).flatMap {
case (pageId, (graphRdd, rankRdd)) =>
graphRdd.map(dest => (dest.toInt, rankRdd / graphRdd.size))
}
rankRdd = contributions.reduceByKey(_ + _).mapValues(0.15 + 0.85 * _)
}
```

Détection de communauté par Label propagation

```
labelRdd=labelFile.map(x => (x.split(";")(0).toInt, x.split(";")(1).toInt))
```

```
for (i <- 0 until 50) {
val contributions = graphRdd.join(labelRdd).flatMap {
case (pageId, (g, l)) =>
g.map(dest => ((dest.toInt, l), 1))
}.reduceByKey(_ + _).map {
case ((node, label), nbLabel) =>
(node, (label, nbLabel))
}.reduceByKey((x1, x2) => if (x1._2 > x2._2) x1 else x2)
labelRdd=contributions.map{
case (pageId, (label, nbLabel)) =>
(pageId, label)
}
}
```

Sur Deux Cliques :

```
val graphFile = sc.textFile("deuxCliques")
val graphRdd=graphFile.flatMap(x => Seq((x.split(";")(0), x.split(";")(1)),(x.split(";")(1), x.split(";")(0)) ))
val graphAdj=graphRdd.groupByKey()
var labelRdd=graphAdj.map{ case(id, neighbors) => (id, id) }

for (i <- 0 until 5) {
val contributions = graphAdj.join(labelRdd).flatMap {
case (pageId, (g, l)) =>
g.map(dest => ((dest, l), 1))
}.reduceByKey(_ + _).map {
case ((node, label), nbLabel) =>
(node, (label, nbLabel))
}.reduceByKey((x1, x2) => if (x1._2 > x2._2) x1 else x2)
labelRdd=contributions.map{
case (pageId, (label, nbLabel)) =>
(pageId, label)
}
}
labelRdd=labelRdd.map{case (id, label) => (label, id)}
var communities = labelRdd.groupByKey()
communities.saveAsTextFile("Communities_DeuxCliques")
```