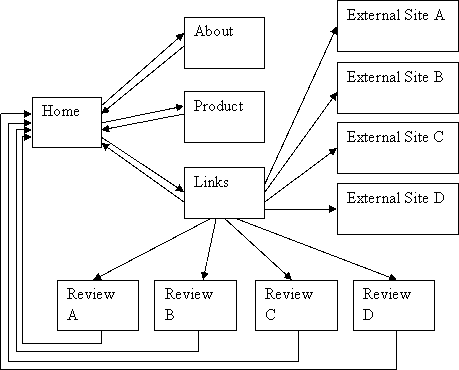
Sample



**2)**

Class MAPPER

Method MAP(score s, linkList ll)

For all link l ϵ linklist ll do

EMIT(link l, score s / len(linklist ll))

Class REDUCER

Method REDUCE(link l, scores [s1, s2, …])

sum <- 0

For all score s ϵ scores [s1, s2, …] do

sum <- sum + s

EMIT(l, sum \* 0.85 + 0.0125)

**3)**

**NOTE:** I tried using looping techniques and had odd results. When I tried a FOR loop to iterate a fixed number of times, IntelliJ would freeze. When I tried using a WHILE loop to iterate until the change was less than ε, it took a substantial number of iterations to reach a stable answer. However, if I just hard-coded chained iterations, the execution time was very fast and I still achieved an accurate and stable answer. That is why much of the code below is repetitive instead of in a looping structure.

package ICE7

import org.apache.spark.{SparkConf, SparkContext}

object ICE7 {

def main(args: Array[String]) {

// administration

System.setProperty("hadoop.home.dir", "C:\\winutils")

val config = new SparkConf()

.setAppName("ICE7")

.setMaster("local[\*]")

val sc = new SparkContext(config)

var pr = sc.textFile("src/main/scala/ICE7/PageRanks.txt").map(x => (x.split('|')(0).split(' ')(0),

(x.split('|')(0).split(' ')(1).toDouble, x.split('|')(1))))

val n = pr.count()

val d = 0.85

/\*

var check1 = pr.values.keys.sum().toDouble

var check2 = -1.0

while (((check1-check2)\*(check1-check2)) > 0.1) {

pr = pr.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

check2 = check1

check1 = pr.values.keys.sum()

}

\*/

pr = pr.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

.flatMap(x => (x.\_2.\_2.split(" ").map(y => (y, ((x.\_2.\_1) / x.\_2.\_2.split(" ").length)))))

.union(pr.map(x => (x.\_1, 0)))

.reduceByKey(\_ + \_).map(x => (x.\_1, x.\_2 \* d + (1 - d)/n))

.join(pr).map(x => (x.\_1, (x.\_2.\_1, x.\_2.\_2.\_2)))

pr.saveAsTextFile("src/main/scala/ICE7/output")

}

}