```
scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD
scala> import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.SparkSession
scala> import org.apache.spark.graphx._
import org.apache.spark.graphx._
scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD
scala> val spark = SparkSession.builder.appName("GitHub Community Detection").master("local[*]").getOrCreate()
24/11/28 19:10:41 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@5e418db4
scala> val sc = spark.sparkContext
sc: org.apache.spark.SparkContext = org.apache.spark.SparkContext@fd255bb
scala> val edgesRaw =
sc.textFile("/Users/parasdhiman/Desktop/assmt/BDA/assmt4/git_web_ml/musae_git_edges.csv").filter(line =>
!line.startsWith("id_1"))
edgesRaw: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[96] at filter at <console>:113
scala> val edges: RDD[Edge[Int]] = edgesRaw.map { line =>
     |   val parts = line.split(",").map(_.trim)
     |   Edge(parts(0).toLong, parts(1).toLong, 1)
     | }
edges: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[Int]] = MapPartitionsRDD[97] at map at
<console>:113
scala> val nodesRaw =
sc.textFile("/Users/parasdhiman/Desktop/assmt/BDA/assmt4/git_web_ml/musae_git_target.csv").filter(line =>
!line.startsWith("id"))
nodesRaw: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[100] at filter at <console>:113
scala> val vertices: RDD[(VertexId, Int)] = nodesRaw.map { line =>
     |   val parts = line.split(",").map(_.trim)
     |   (parts(0).toLong, parts(2).toInt)
     | }
vertices: org.apache.spark.rdd.RDD[(org.apache.spark.graphx.VertexId, Int)] = MapPartitionsRDD[101] at map at
<console>:113
scala> val graph = Graph(verticesRDD, edgesRDD)
graph: org.apache.spark.graphx.Graph[String,Int] = org.apache.spark.graphx.impl.GraphImpl@4d4f2da
scala>
scala>     // Initialize random community assignments
scala>     val initialGraph = graph.mapVertices { case (id, _) => Random.nextLong() }
initialGraph: org.apache.spark.graphx.Graph[Long,Int] = org.apache.spark.graphx.impl.GraphImpl@2946e1b2
scala> val initialGraph = graph.mapVertices { case (id, _) => Random.nextLong() }
initialGraph: org.apache.spark.graphx.Graph[Long,Int] = org.apache.spark.graphx.impl.GraphImpl@65bfc4cb
scala>
scala>     // AGM-based community detection
scala>     val maxIterations = 10
maxIterations: Int = 10
scala>     val agmGraph = (0 until maxIterations).foldLeft(initialGraph) { (currentGraph, _) =>
     |       val updatedVertices = currentGraph.aggregateMessages[Map[Long, Int]](
     |         triplet => {
     |           triplet.sendToSrc(Map(triplet.dstAttr -> 1))
     |           triplet.sendToDst(Map(triplet.srcAttr -> 1))
     |         },
     |         (a, b) => (a.keySet ++ b.keySet).map(k => k -> (a.getOrElse(k, 0) + b.getOrElse(k, 0))).toMap
     |       )
     |
     |
You typed two blank lines.  Starting a new command.
scala> val initialGraph = graph.mapVertices { case (id, _) => Random.nextLong() }
```

```
initialGraph: org.apache.spark.graphx.Graph[Long,Int] = org.apache.spark.graphx.impl.GraphImpl@717dabe
scala>
scala>     // AGM-based community detection
scala>     val maxIterations = 10
maxIterations: Int = 10
scala>     val agmGraph = (0 until maxIterations).foldLeft(initialGraph) { (currentGraph, _) =>
     |       val updatedVertices = currentGraph.aggregateMessages[Map[Long, Int]](
     |         triplet => {
     |           triplet.sendToSrc(Map(triplet.dstAttr -> 1))
     |           triplet.sendToDst(Map(triplet.srcAttr -> 1))
     |         },
     |         (a, b) => (a.keySet ++ b.keySet).map(k => k -> (a.getOrElse(k, 0) + b.getOrElse(k, 0))).toMap
     |       )
     |
     |
You typed two blank lines.  Starting a new command.
scala>
scala> val maxIterations = 10
maxIterations: Int = 10
scala>     val agmGraph = (0 until maxIterations).foldLeft(initialGraph) { (currentGraph, _) =>
     |       val updatedVertices = currentGraph.aggregateMessages[Map[Long, Int]](
     |         triplet => {
     |           triplet.sendToSrc(Map(triplet.dstAttr -> 1))
     |           triplet.sendToDst(Map(triplet.srcAttr -> 1))
     |         },
     |         (a, b) => (a.keySet ++ b.keySet).map(k => k -> (a.getOrElse(k, 0) + b.getOrElse(k, 0))).toMap
     |       )
     |
     |       currentGraph.outerJoinVertices(updatedVertices) { case (_, oldAttr, newAttrOpt) =>
     |         newAttrOpt match {
     |           case Some(newAttr) => newAttr.maxBy(_._2)._1
     |           case None => oldAttr
     |         }
     |       }
     |     }
agmGraph: org.apache.spark.graphx.Graph[Long,Int] = org.apache.spark.graphx.impl.GraphImpl@67e8b61e
scala> // Evaluate modularity
scala>     val communities = agmGraph.vertices.map(_._2).distinct().collect()
communities: Array[Long] = Array(-3789950034135947739, 6661397685359818011, 3384032944393845118,
-3374758099156163740, 2843236671570357348, 3891528763791861767, 1992839733098964865,
-374798850542459953, 8384977230408201975, 3206063710765581490, 6360088892093494150,
5827804368923597997, 8936234278016317059, 8158607921740125360, -3052908954122504141,
-5402706623429668553, -1696411624962883722, 5950922849129585813, -7791360925934903143,
-2597653101460354875, -3310883775683056845, -1692491843631063355, 3046080757674199704,
8568335285994171171, 1493888806445489191, -4467186166493969592, -3835039650079766614,
-6519420168380756246, -5999237072826511431, 4294635214892744190, 1759629845126040741,
4325528367996316810, 2591313611404180725, -5512542376931740696, 4541585026...
scala>     val modularity = communities.map { community =>
     |       val subgraph = agmGraph.subgraph(vpred = (_, attr) => attr == community)
     |       val internalEdges = subgraph.edges.count()
     |       val totalEdges = agmGraph.edges.count()
     |       internalEdges.toDouble / totalEdges
     |     }.sum
modularity: Double = 0.9984014006775017
scala>
scala>     println(s"Detected Communities: ${communities.mkString(", ")}")
```

```
Detected Communities: -3789950034135947739, 6661397685359818011, 3384032944393845118,
-3374758099156163740, 2843236671570357348, 3891528763791861767, 1992839733098964865,
-374798850542459953, 8384977230408201975, 3206063710765581490, 6360088892093494150,
5827804368923597997, 8936234278016317059, 8158607921740125360, -3052908954122504141,
-5402706623429668553, -1696411624962883722, 5950922849129585813, -7791360925934903143,
-2597653101460354875, -3310883775683056845, -1692491843631063355, 3046080757674199704,
8568335285994171171, 1493888806445489191, -4467186166493969592, -3835039650079766614,
-6519420168380756246, -5999237072826511431, 4294635214892744190, 1759629845126040741,
4325528367996316810, 2591313611404180725, -5512542376931740696, 4541585026717216949,
3250085886073718461, 4317597094297862354, -5848258671648042126, 604739379760274 97, -2935698684339404166,
2930258129533918195, 1642553172797557595, 8621700643098392212, -4157110721288352663,
8129857175731749646, 2287675294797141837, 381815873705300688, 1804187086424756023,
-1574798612320613202, -2145923250028553667, -5783291497298793423, -1481437685375608123,
3508910147591329049, 3473398070465905640, 927902989898246332, 3745460917930392888, 2739238765660507491,
2000691267660352753, 761989430417971365, 2078751282734885079, 5025645194903635016, 7784349494286569799,
7486943276640275883, -984716206936472800, 7212833351693 1280, 1184018096968921905, -2787302430312137370,
585298586048869932, 6754331386942781632, -7395647656318804695, 3080976064995526827,
-7920503305036319403, 350532013330879588, -2070698688573303028, -3781362464127568791,
3048711816305378893, 8564436813713371627, 3331364456714894022, 3828417925364115924,
9048420144674113359, 8934891606637069990, 7362714239707068798, 8384629688841772800, 203214595728555243,
891862732579312694, -258520954156271368, -7358976508674498029, -8711255640804874648,
2414815498681240908, 6774080591453696966, -6638374962648238471, 8888798684537763249,
2115532981785178024, -2466092914471246603, 4240945775333727502, 2753785828734367298,
1557999567898026821, -1761395314157449568, -7007675533615315655, -5380293199976370531,
8290715527057915729, -1296464899792469202, 6385283229138799949, 8359034556057968588,
7256055576786791936, 40324228820449984, 8333304065971329573, 7577575321686934899, 2542636809913035852,
8604375779997379243, -1720481476802319272, 7189033007724562130, -4710532350141325570,
-7860116891855609756, -54347043647032554, -3997566535866605351, -3522611705621547482,
3025906590270133141, -5466208706621254268, 3846345810882876841, -5991088183257089676,
-1713127076894063802, -2320351611537112987, 4782264208065795311, -1034921578350833791,
2096776360251026409, -4480560572693691078, -4873071371695651472, -6290229748334523757,
-8638850794900219594, 5417841954707803378, 5834287121450262314, -6933609954543527114,
6289129726856337964, 3525440414803012522, 8536986694435489809, -6475926458271884678,
-5851901898985496644, 3666792611149294728, 4570589039500818411, -2515921063648107643,
-4390783126646919127, -6157801764420233704, -8708897316144983450, 1682383894569427580,
8905048988816688446, -2773017628381643605, 2949085468624443585, 2907227254406712405,
-6263086489257228641, 3688430565765786123, -1784532336360754020, 7154718682639114736,
-7236687982290656643, 5411491665672964379, 6744076464371836977, -3014095328019430867,
512483536599119563, -6939676413992529641, 2655519187044189363, 3871444188958779406, 2876559193373607721,
-6160627204708068333, -8576813975842304493, 7247379201539353098, -1558950439075074727,
-3843474639081029182, -8447839188258253825, -8907776728549117326, 6001579039016070164,
-5176309834773966345, -1651487393394835408, -5243903034187178389, 4769595480639480759,
4006995581475320936, 4728570166057478172, 7652630452933513653, -8173222484905655926,
2943154680021720043, -7846411837804865339, -4394778666320833096, 2127943610651920246,
-4981769012562927044, -7592324060866943705, 8415180668662371190, 8200046848855294269,
4683997413901217047, -6575482439468609526, 2895758507931831166, 4068537998635981507,
-2182010856326389538, -7557157541634012137, -3964007358778592108, -2713206864981363500,
4860137283822435178, 1604703900491582413, 2776445903929127265, -4108103018761720692,
-821304640964214293, -7159358806431 60847, -1593941686983762509, -6686798027834325125,
1885400030623916766, -4668422941199165556, -5162310982367034986, -3977907299171073576,
-3757027031990398998, 3687793824450859959, -8252888884920651212, 1783754975243025179,
1633631899930302803, 1927535972326124765, -2013546115289233728, -1539103732903420187,
-8262463067081354415, 3502372704640634554, 3083760111237505952, 8305341683149932354,
1412307829147588495, 4526823573833891636, -5285377642120619836, -5629536718234234477,
8100166184145672813, -2724026529642524052, -6147822595728800837, -1431365661978700840,
5764749695949857408, 9164774063844828713, 8740866554969760351, 7423649845717579865,
3351968716277620035, 3803119495219450458, -2989881986500273312, 2039490060998841578,
9202142391429475590, 263969861870974613, -2524096387114124375, 1710393901586453859, -800134690876783447,
5103792323292062349, -8202976137038302196, -5757158252332818521, -1149187634318015097,
6703142390706243413, 8900818148852106541, -1288390315492942234, -5551273128321190385,
3978472190353808266, 8772491780852104261, 4046119682337389360, -4717101732174603060,
4815127595325721638, -7990798008108890955, -6560913330905684178, -7343026965198539316,
5410832027694027494, -1255646780011539232, 2507743362020829139, 2009524920187604078,
5618267500601121944, -1410371766865729893, -3200745553907518428, -4583043391523129291,
6303004860675521045, -3238743419471716052, 5819476503329775252, -4387865011392555870,
-6723838511365407802, 3077004934387646720, -2947986162275151601, -5556054076193746120,
-5614872152727294982, 7682388034458060291, -576659756030614098, -7537044254300429160,
-7469299754926344230, -2498057213374973633, -1231882480920573273, 6613709935702935671,
-5366330935408133879, -2011443578797267990, -2859892395507425141, 4658968180910845425,
-3922679871780891360, -2381150239332425666, 8442250390510661816, -5351831543853374229,
7321662303917703627, -445444172102396038, 7294502018377757316, -3123007018838855105,
7083687387396500933, -305868503364693319, 3485480626904862804, 5135824074216129124,
-8004286271809109908, -2516566383369350316, -6292704525777781889, -1523832549243811777,
2312441939937486686, -4369034750605372460, -6902347404121916963, -6054387988456045725,
8631874352640074342, -1554175566787536122, -5769304419827584017, -6682608838572929057,
-8763844157246197602, 7849238942767206885, 1423847132997645231, 5502438737657761697,
-6081902596744808083
scala>     println(s"Modularity: $modularity")
Modularity: 0.9984014006775017
scala>
scala>     // Stop Spark Session
scala>     spark.stop()
scala>
```