一、运行流程和结果

1. 将程序打包成 jar 包，将 NewInstance.txt 文件导入到 hdfs 文件系统根目录下

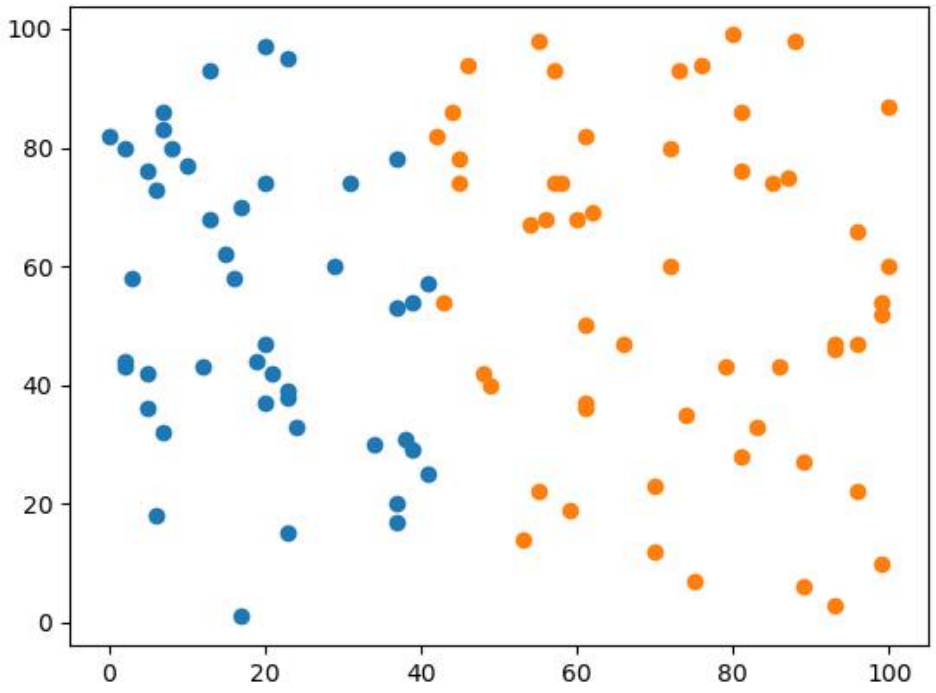
```
E:\南京大学\计金\大三上\金融大数据处理技术\第9周\可视化>hadoop fs -put NewInstance.txt /
2020-11-09 10:49:55,852 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteH
ostTrusted = false

E:\南京大学\计金\大三上\金融大数据处理技术\第9周\可视化>hadoop fs -ls /
Found 1 items
-rw-r--r--   1 luzhongtian supergroup        584 2020-11-09 10:49 /NewInstance.txt
```

2. 输入 hadoop jar KMeans-1.jar <簇个数> <迭代次数> <inputpath> <outputpath>，例如 hadoop jar KMeans-1.jar 2 3 /NewInstance.txt /2-3 为两个簇、迭代 3 次。

3. 运行结束后输入 hadoop fs -get /2-3/points 2-3，将结果导出来

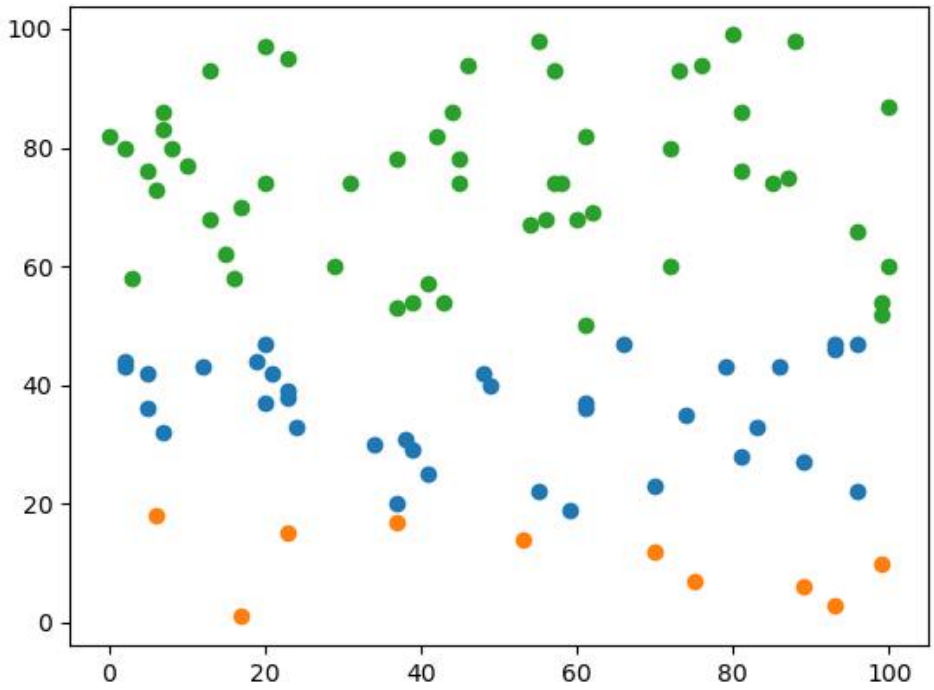
4. 运行可视化.py，生成聚类图像，如 2-3，详细聚类效果见第三部分



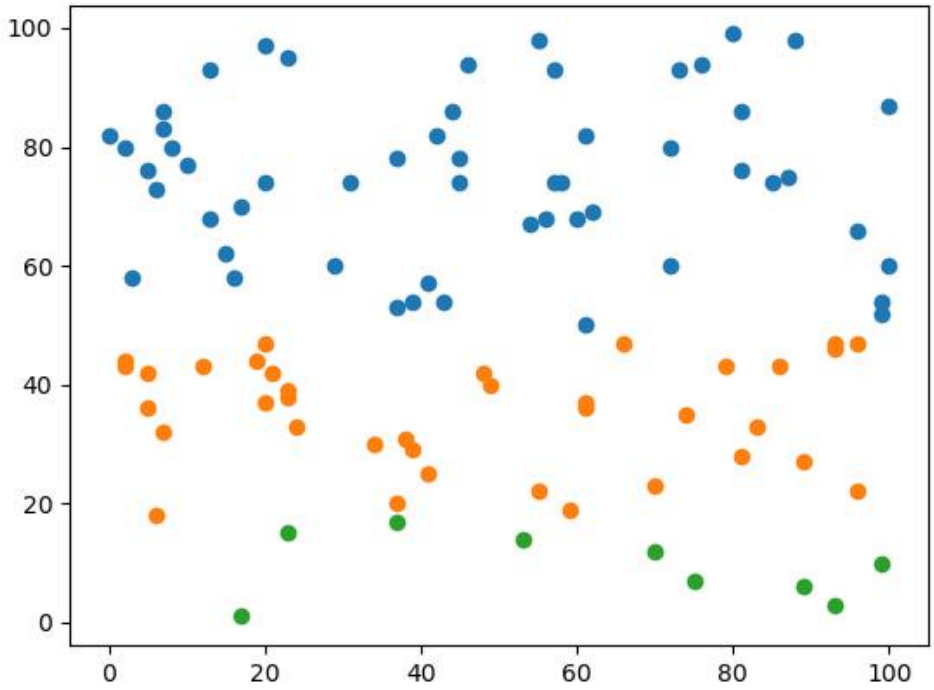
二、设计思路与所遇问题

首先我构造了两个自定义数据类型 Point 和 Cluster，Point 类包含点的信息，Cluster 类包含簇的信息，然后主体分为两个 mapreduce 程序，一个是迭代计算聚类中心，另一个是根据最后生成的聚类中心，标记每个点所属的簇。初始聚类中心就取前 k 个点。

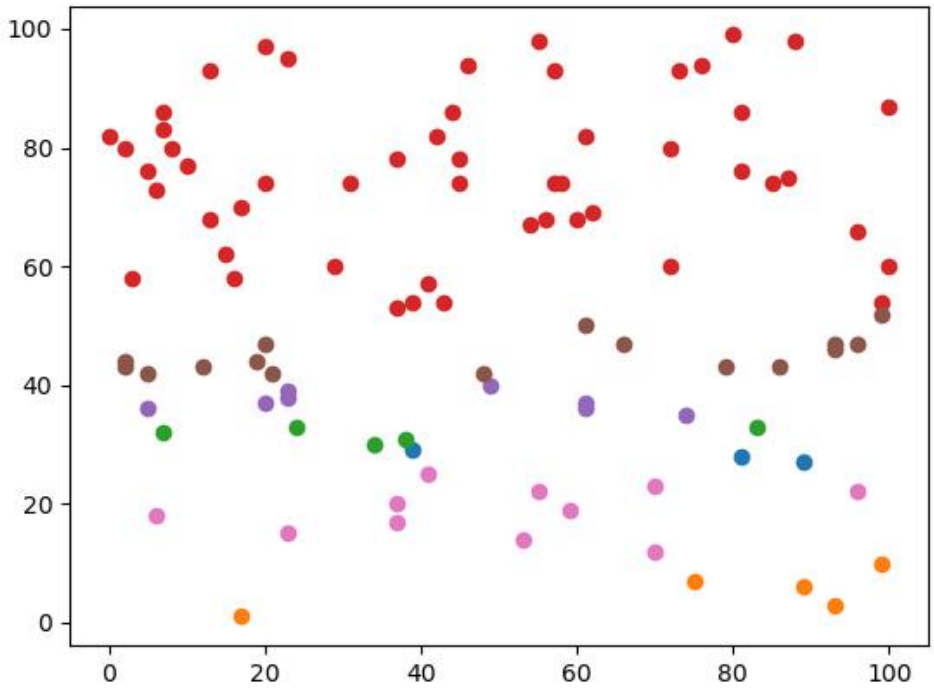
一开始我自己写了个程序，尝试 3 个聚类，迭代 5 次，发现跑出来的结果是这样的



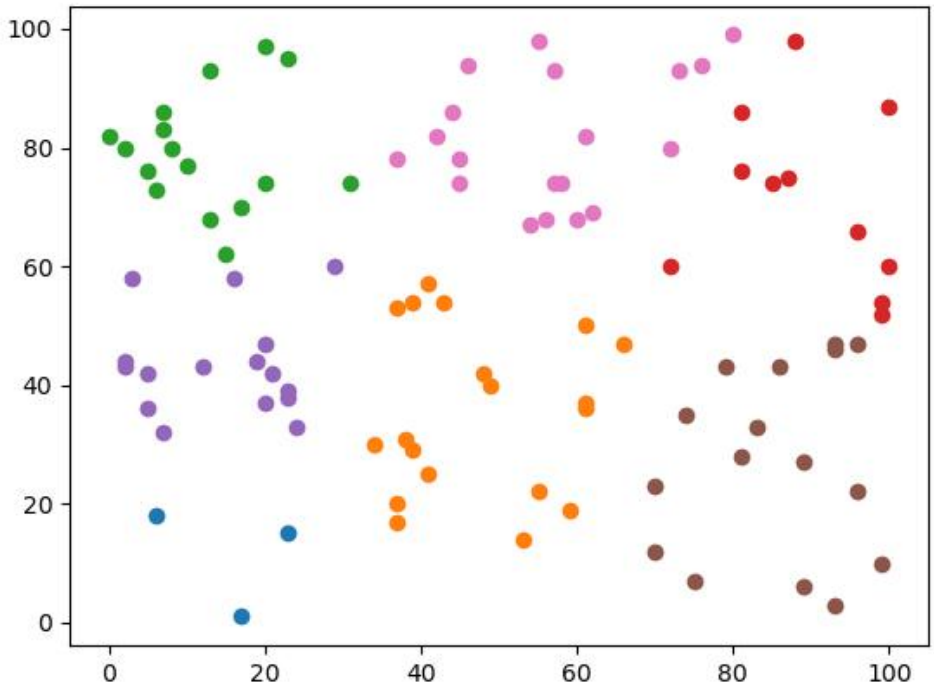
这明显是不对的，我估计是迭代次数不够，把迭代次数改为了 10 次，结果还是如此



把簇的个数改为 7，结果仍然是呈横条状分布



我用示例程序运行 7 个簇，结果是比较符合预期的



我估计是迭代求中心的部分写错了，因为自己写的非常乱，而且很繁杂，就对照树上的示例程序，把初始生成聚类中心、迭代计算中心的代码进行了替换，但结果还是如此。我又将最后标记每个点的代码改成了示例程序，结果还是错误。

结果只能是我自定义的数据类型有问题，但经过反复查看没有发现错误。

于是我在 mapper 和 reducer 中输出了每一步得到的结果，查看日志，发现了问题所在：在 mapper 部分发送的点是对的

```
Point is:86.0,43.0
onePointCluster is:3,1,86.0,43.0
Point is:5.0,36.0
onePointCluster is:3,1,5.0,36.0
Point is:16.0,58.0
onePointCluster is:3,1,16.0,58.0
Point is:66.0,47.0
onePointCluster is:3,1,66.0,47.0
Point is:20.0,37.0
onePointCluster is:3,1,20.0,37.0
Point is:89.0,27.0
onePointCluster is:3,1,89.0,27.0
Point is:56.0,68.0
onePointCluster is:3,1,56.0,68.0
Point is:21.0,42.0
onePointCluster is:3,1,21.0,42.0
Point is:96.0,22.0
onePointCluster is:3,1,96.0,22.0
Point is:72.0,80.0
onePointCluster is:2,1,72.0,80.0
Point is:99.0,10.0
onePointCluster is:3,1,99.0,10.0
Point is:20.0,74.0
onePointCluster is:2,1,20.0,74.0
Point is:59.0,19.0
onePointCluster is:3,1,59.0,19.0
Point is:70.0,23.0
onePointCluster is:3,1,70.0,23.0
Point is:81.0,86.0
onePointCluster is:2,1,81.0,86.0
Point is:53.0,14.0
onePointCluster is:3,1,53.0,14.0
Point is:72.0,60.0
onePointCluster is:3,1,72.0,60.0
Point is:2.0,80.0
onePointCluster is:1,1,2.0,80.0
Point is:10.0,77.0
onePointCluster is:1,1,10.0,77.0
```

但是 combiner 收到的数据却发生了改变

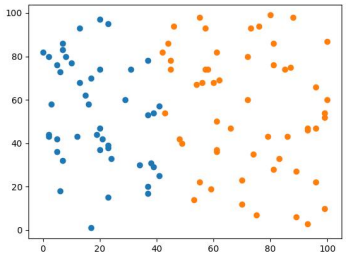
```
cluster is:1,1,4.776077105E-314,5.331378433E-315
cluster is:1,1,4.7758504513E-314,5.331216537E-315
cluster is:1,1,4.776174242E-314,5.331702224E-315
cluster is:1,1,4.243991582E-314,5.332349805E-315
cluster is:1,1,4.7744905297E-314,5.32765484E-315
cluster is:1,1,4.7744905297E-314,5.32749294E-315
cluster is:1,1,4.775656177E-314,5.331945067E-315
cluster is:1,1,4.7744905297E-314,5.33218791E-315
cluster is:1,1,4.775397144E-314,5.332430753E-315
cluster is:1,1,4.7755266604E-314,5.33218791E-315
cluster is:1,1,4.7762713794E-314,5.333402126E-315
cluster is:1,1,4.7759799676E-314,5.330568955E-315
cluster is:1,1,4.7751381114E-314,5.33186412E-315
cluster is:1,1,4.775397144E-314,5.332673596E-315
cluster is:1,1,4.776044726E-314,5.329921374E-315
cluster is:1,1,4.7758504513E-314,5.33324023E-315
cluster is:1,1,4.7747495624E-314,5.329921374E-315
cluster is:1,1,4.7751381114E-314,5.327331047E-315
cluster is:1,1,4.776174242E-314,5.33356402E-315
cluster is:1,1,4.7752676277E-314,5.331621276E-315
combiner emit cluster:1,20,4.7489078135E-314,5.33123273E-315
cluster is:2,1,4.776643739E-314,5.329111897E-315
cluster is:2,1,4.776676118E-314,5.32927379E-315
cluster is:2,1,4.776740876E-314,5.32927379E-315
cluster is:2,1,4.7769351507E-314,5.33364497E-315
cluster is:2,1,4.776643739E-314,5.332026014E-315
cluster is:2,1,4.77696753E-314,5.331702224E-315
cluster is:2,1,4.77696753E-314,5.33324023E-315
cluster is:2,1,4.776708497E-314,5.32975948E-315
cluster is:2,1,4.7771618043E-314,5.32619778E-315
cluster is:2,1,4.777210373E-314,5.333725916E-315
cluster is:2,1,4.77723465 7E-314,5.32587399E-315
cluster is:2,1,4.7770484775E-314,5.331297485E-315
cluster is:2,1,4.776465654E-314,5.330245165E-315
cluster is:2,1,4.777250847E-314,5.331702224E-315
```

这让我很困惑，因为经过代码的替换，我的程序和示例程序是几乎一样的了，应该不会有错误。

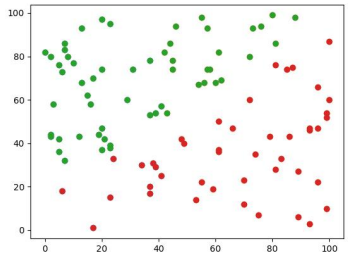
最后，我发现了出错的原因：我自定义的点的 readFields()和 write()不匹配，导致了 mapper 发送和 reducer 接受的数据不同。这是一个很蠢的问题，但我对自定义数据类型的理解不够深入，认为这两部分不是很重要，debug 时甚至没有检查这两部分，还是费了很大周折才找到了这个问题。

三、效果分析

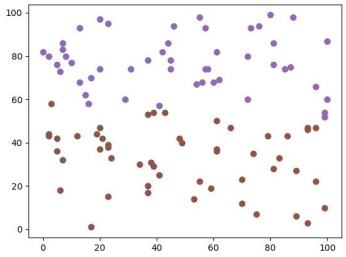
两个簇：（图片下方是迭代次数）



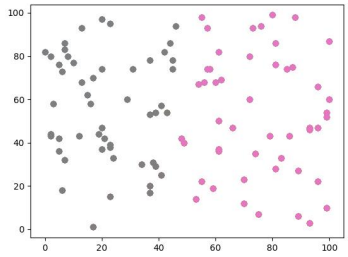
3



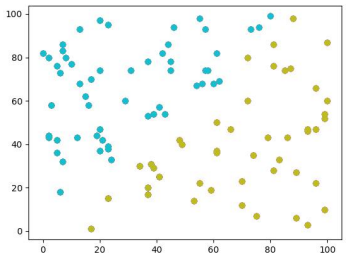
4



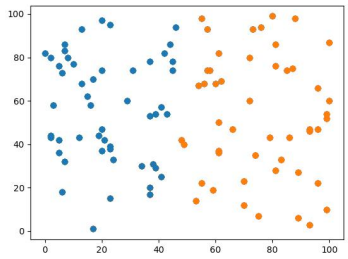
5



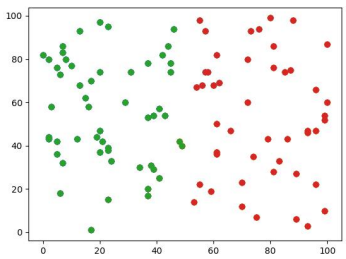
6

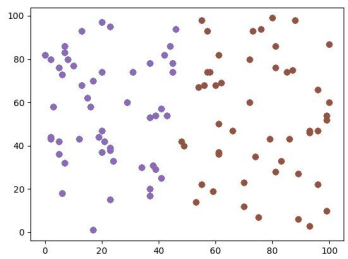


7

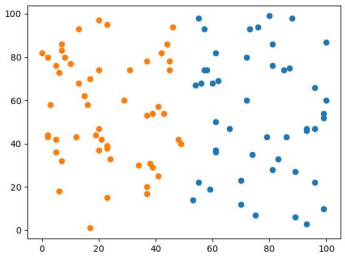


8

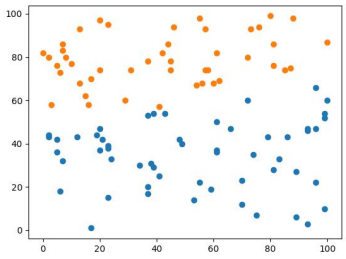




9



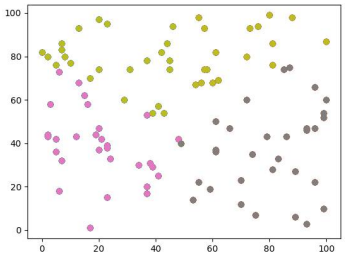
10



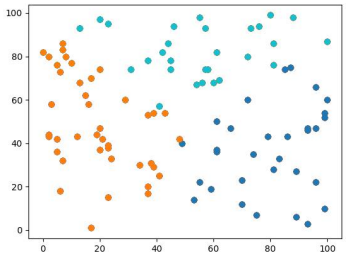
15

20

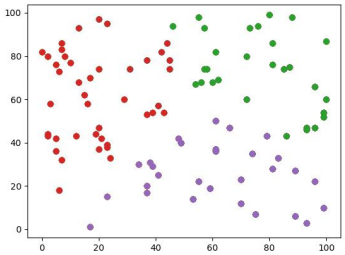
三个簇：



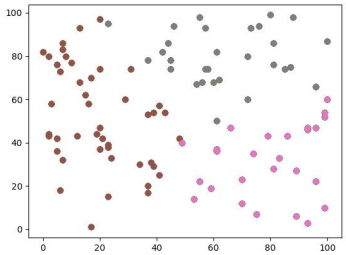
3



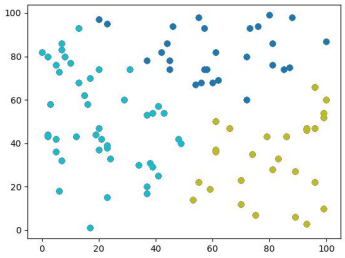
4



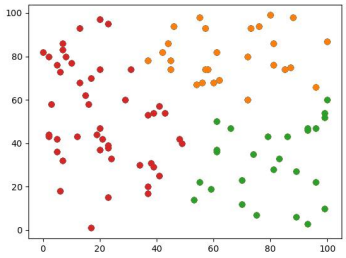
5



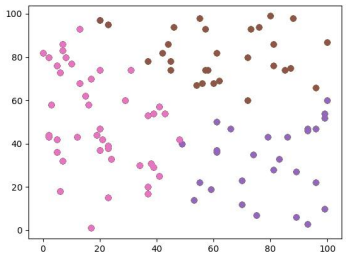
6



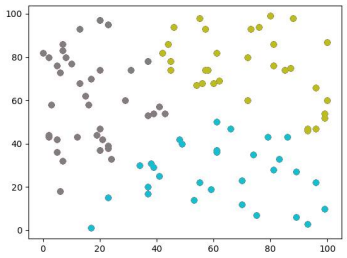
7



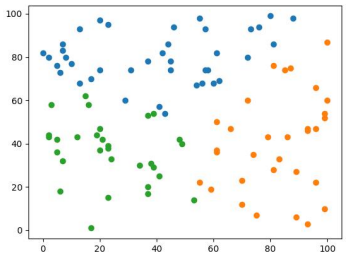
8



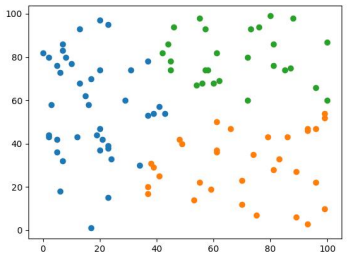
9



10

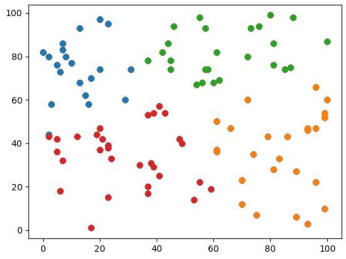


15

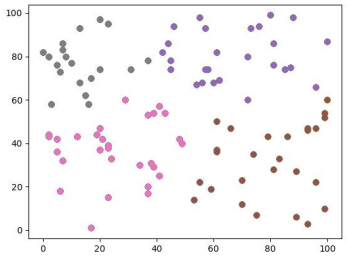


20

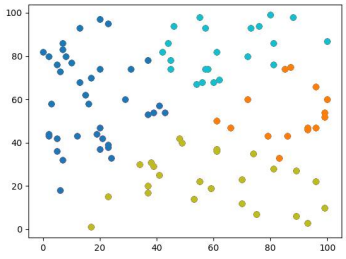
四个簇：



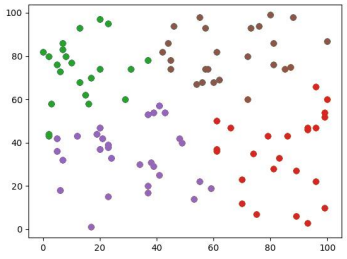
3



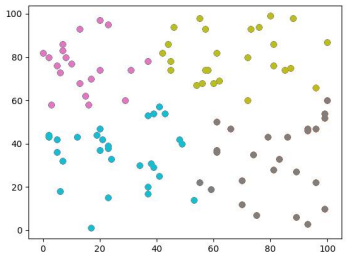
4



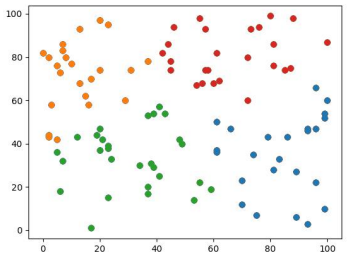
5



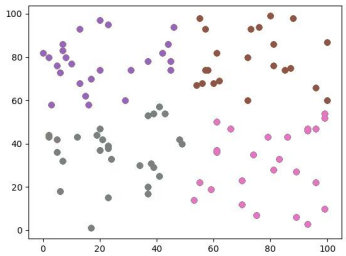
6



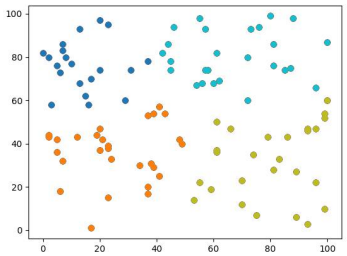
7



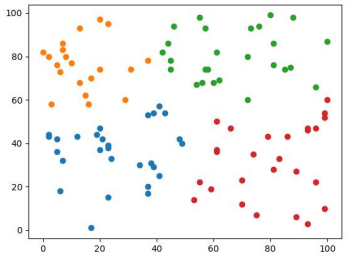
8



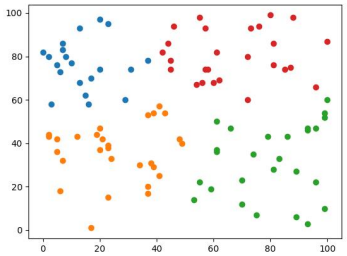
9



10



15

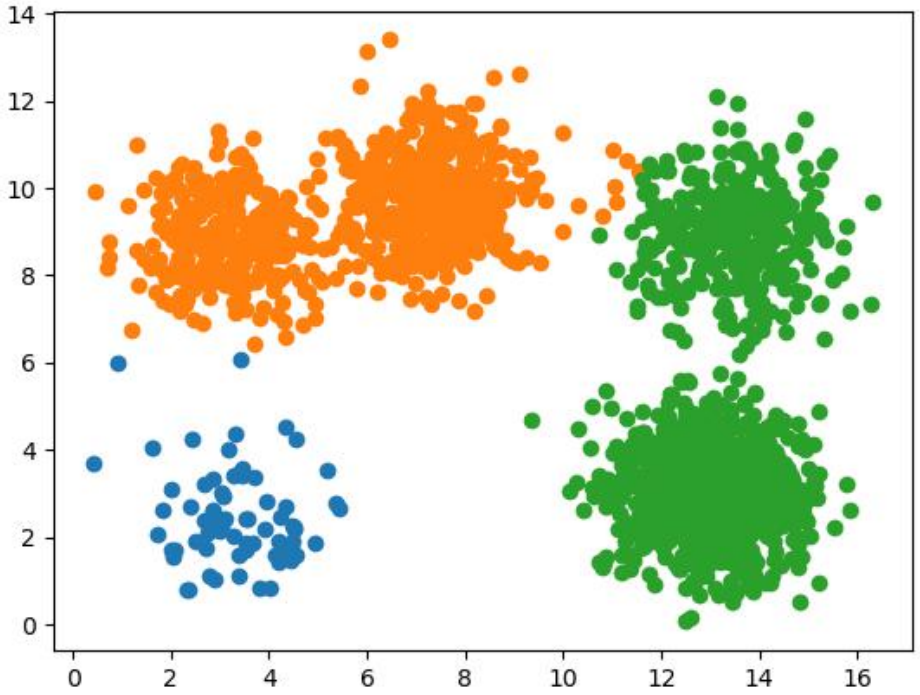


20

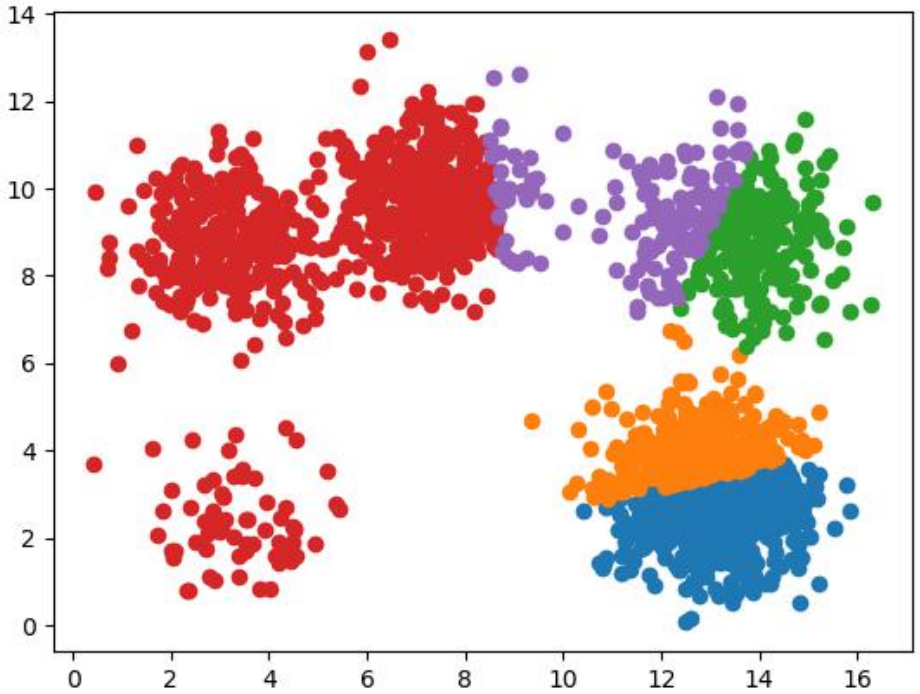
可以看到，簇个数为 3、4 时随着迭代次数不断增加，结果趋于稳定，但是 2 个簇的结果会在上下两部分和左右两部分之间来回晃动，这是由于选取的点均为随机点，没有明显的团簇区分，对 2 个簇来说上下和左右的情况都是合理的。

下面我到网上找了一个有明显聚类的数据，用 mapreduce 程序运行。文件名为 Instance2.txt。

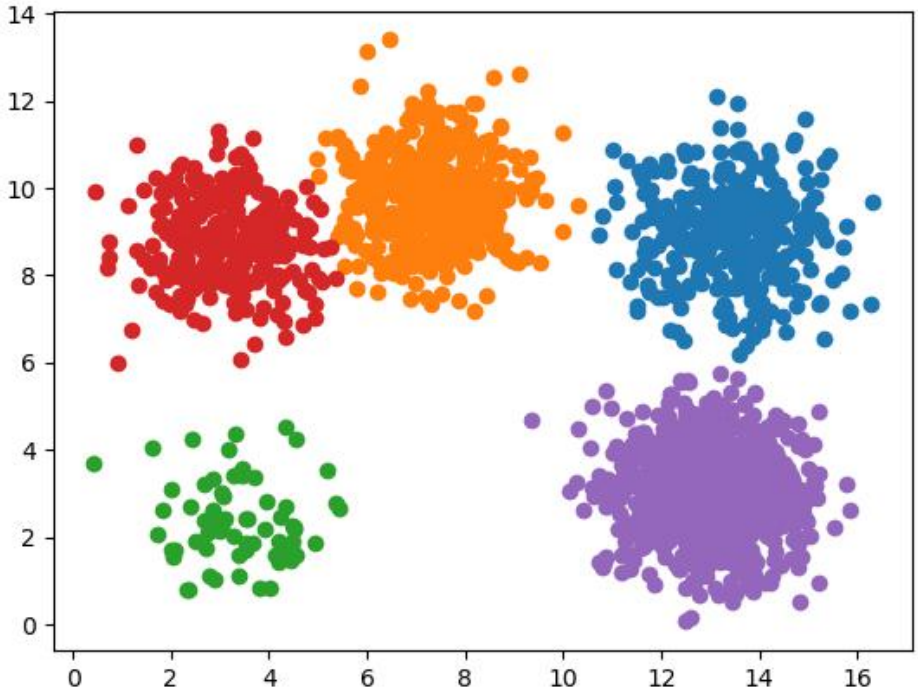
设定簇数为 3，迭代次数 3：



可以看出应该是 5 个簇
接下来设定簇数为 5，迭代 5 次，得到



增加迭代次数到 10 次，得



这样已经得到了正确的聚类，说明程序运行正确。