CSE 4/587 - Practice HW

Question 1 - Spark vs MapReduce

a. List 2 benefits that spark has over MapReduce
b. Name one other technology in the Hadoop ecosystem that improves programmer productivity with MapReduce
c. In one sentence explain the primary way fault-tolerance is achieved in MapReduce
d. In one sentence explain the primary way fault-tolerance is achieved in Spark
e. Explain the difference between a transformation and an action in Spark
f. Explain the difference between a narrow dependency and a wide dependency in Spark
    i. Name one transformation that results in a narrow dependency, draw a DAG
    ii. Same as above for wide dependency

Spark - Code Example 1

```
lines = sc.textFile(sys.argv[1]).map(lambda r: r[0])
K = int(sys.argv[2])
convergeDist = float(sys.argv[3])

data = lines.map(parseVector).cache()
kPoints = data.takeSample(False, K, 1)
tempDist = 1.0

while tempDist > convergeDist:
  closest = data.map(
    lambda p: (closestPoint(p, kPoints), (p, 1)))
  pointStats = closest.reduceByKey(
    lambda p1_c1, p2_c2: (p1_c1[0] + p2_c2[0], p1_c1[1] + p2_c2[1]))
  newPoints = pointStats.map(
    lambda st: (st[0], st[1][0] / st[1][1])).collect()

  tempDist = sum(np.sum((kPoints[iK] - p) ** 2) for (iK, p) in
newPoints)
  for (iK, p) in newPoints:
    kPoints[iK] = p
```

a. Given the above spark application, draw the lineage graph DAG for the RDD **newPoints**
    i. Include nodes for all intermediate RDDs, even if they are unnamed
b. Identify in the above code one instance of:
    i. A transformation that results in a wide dependency
    ii. A transformation that results in a narrow dependency
    iii. An action
c. How many "jobs" will the above code run?
d. Based on your DAG, determine how it is broken up into stages (state the number of stages, and name the transformations in each stage)
e. What algorithm is the above code an implementation of?

Spark - Code Example 2

```
lines = sc.textFile(file)
  links = lines.map(lambda urls: parseNeighbors(urls)) \
               .groupByKey()
               .cache()
  N = links.count()
  ranks = links.map(lambda u: (u[0], 1.0/N))

  for i in range(iters):
    contribs = links.join(ranks) \
                    .flatMap(lambda u: computeContribs(u[1][0],
u[1][1]))

    ranks = contribs.reduceByKey(lambda a,b: a+b) \
                    .mapValues(lambda rank: rank * 0.85 +
0.15*(1.0/N))
  return ranks
```

a. Given the above spark application, draw the lineage graph DAG for the RDD **newPoints**
    i. Include nodes for all intermediate RDDs, even if they are unnamed
b. Identify in the above code one instance of:
    i. A transformation that results in a wide dependency
    ii. A transformation that results in a narrow dependency
    iii. A transformation that may result in a narrow dependency OR a wide dependency
    iv. An action
c. How many "jobs" will the above code run?
d. Based on your DAG, determine how it is broken up into stages (state the number of stages, and name the transformations in each stage)
e. What algorithm is the above code an implementation of?