COMP9313: Big Data Management

Lecturer: Xin Cao

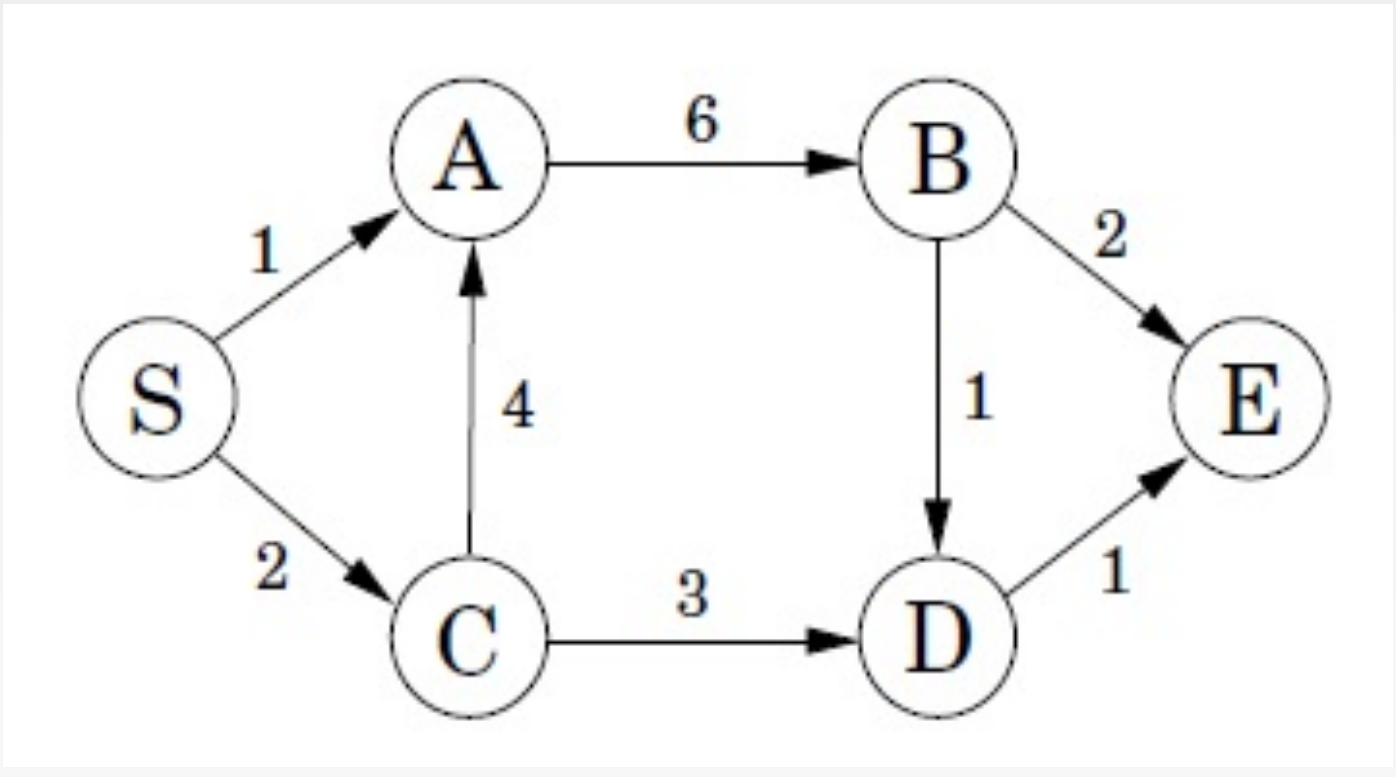
Course web site: http://www.cse.unsw.edu.au/~cs9313/

Question 1 MapReduce

- Assume that you are given a data set crawled from a location-based social network, in which each line of the data is in format of (userID, a list of locations the user has visited <loc1, loc2, …>). Your task is to compute for each location the set of users who have visited it, and the users are sorted in ascending order according to their IDs.

Question 1 MapReduce

- Given the following graph, assume that you are using the single shortest path algorithm to compute the shortest path from node S to node E. Show the output of the mapper (sorted results of all mappers) and the reducer (only one reducer used) in each iteration (including both the distances and the paths).



Question 1 MapReduce

- Assume that in an online shopping system, a huge log file stores the information of each transaction. Each line of the log is in format of "userID\tproduct\t price\t time". Your task is to use MapReduce to find out the top-5 most expensive products purchased by each user in 2016.

local top-5 1:29

Question 1 MapReduce

- Given a large text file, find the top-k words that appear the most frequently.

Question 2 Spark

- Write down the output

a) val lines = sc.parallelize(List("hello world", "this is a scala program", "to create a pair RDD", "in spark"))

```
val pairs = lines.map(x => (x.split(" ")(0), x))
pairs.filter {case (key, value) => key.length <3}.foreach(println)
```

b) val pairs = sc.parallelize(List((1, 2), (3, 4), (3, 9), (4,2)))

```
val pairs1 = pairs.mapValues(x=>(x, 1)).reduceByKey((x,y) => (x._1 + y._1,  x._2+y._2)).mapValues(x=>x._2/x._1)
pairs1.foreach(println)
```

Question 2 Spark

- Given a large text file, your task is to find out the top-k most frequent co-occurring term pairs. The co-occurrence of (w, u) is defined as: u and w appear in the same line (this also means that (w, u) and (u, w) are treated equally). Your Spark program should generate a list of ***k*** key-value pairs ranked in descending order according to the frequencies, where the keys are the pair of terms and the values are the co-occurring frequencies (**Hint:** you need to define a function which takes an array of terms as input and generate all possible pairs).

```
val textFile = sc.textFile(inputFile)
val words = textFile.map(_.split(“ “).toLowerCase)

// fill your code here, and store the result in a pair RDD avgLen

avgLen.foreach(x => println(x._1, x._2))
```

Question 3 Finding Similar Items

- k-Shingles:

Consider two documents A and B. Each document's number of token is O(n). What is the runtime complexity of computing A and B's k-shingle resemblance (using Jaccard similarity)? Assume that comparison of two k-shingles to assess their equivalence is O(k). Express your answer in terms of n and k.

Question 3 Finding Similar Items

- MinHash:

We want to compute min-hash signature for two columns, C_1 and C_2 using two pseudo-random permutations of columns using the following function:

Row	C_1	C_2
0	0	1
1	1	0
2	0	1
3	0	0
4	1	1
5	1	1
6	1	0

$h_1(n) = (3n + 2) \bmod 7$

$h_2(n) = (2n - 1) \bmod 7$

Here, n is the row number in original ordering. Instead of explicitly reordering the columns for each hash function, we use the implementation discussed in class, in which we read each data in a column once in a sequential order, and update the min hash signatures as we pass through them.

Complete the steps of the algorithm and give the resulting signatures for C_1 and C_2.

Question 3 Finding Similar Items

- LSH:

Suppose we wish to find similar sets, and we do so by minhashing the sets 10 times and then applying locality-sensitive hashing using 5 bands of 2 rows (minhash values) each. If two sets had Jaccard similarity 0.6, what is the probability that they will be identified in the locality-sensitive hashing as candidates (i.e. they hash at least once to the same bucket)? You may assume that there are no coincidences, where two unequal values hash to the same bucket. A correct expression is sufficient: you need not give the actual number.

Question 4 Mining Data Streams

quirky solution

- DGIM

Suppose we are maintaining a count of 1s using the DGIM method. We represent a bucket by (i, t), where i is the number of 1s in the bucket and t is the bucket timestamp (time of the most recent 1).

Consider that the current time is 200, window size is 60, and the current list of buckets is: (16, 148) (8, 162) (8, 177) (4, 183) (2, 192) (1, 197) (1, 200). At the next ten clocks, 201 through 210, the stream has 0101010101. What will the sequence of buckets be at the end of these ten inputs?

Question 5 Recommender Systems

- Consider three users u_1, u_2, and u_3, and four movies m_1, m_2, m_3, and m_4. The users rated the movies using a 4-point scale: -1: bad, 1: fair, 2: good, and 3: great. A rating of 0 means that the user did not rate the movie. The three users' ratings for the four movies are: u_1 = (3, 0, 0, -1), u_2 = (2, -1, 0, 3), u_3 = (3, 0, 3, 1)
  - Which user has more similar taste to u_1 based on cosine similarity, u_2 or u_3? Show detailed calculation process.
  - User u_1 has not yet watched movies m_2 and m_3. Which movie(s) are you going to recommend to user u_1, based on the user-based collaborative filtering approach? Justify your answer.