Big Data Management (MIRI/MDS/BDMA)

Final Exam

Family name:_____ Given name: _____

1. (13 points) Name the three characteristics of **fragmentation** that make it correct, and for each of them give an example of fragmentation schema where it is **not fulfilled**.

   (a) Completeness

   Với một dataset và một tập fragments, mọi data item đều có thể đc tìm thấy ở ít nhất một fragment. Data ko bị mất khi fragmenting.

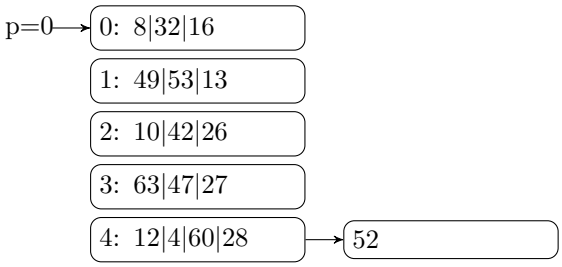
   (b) Disjiontess

   Với một dataset và một tập fragments, ko có redundancy nào (i.e. unnecessary repetitions) trong defined fragments.

   (c) Reconstruction

   Với một dataset và một tập fragments, dataset gốc có thể đc khôi phục từ fragments bằng Relational operators.

2. (20 points) Given the **linear hash** underneath with $f(x) = x$ (i.e., we directly apply the module to the keys), and a capacity of four keys per bucket, indicate if it corresponds to a state **valid or not**. If valid, give a possible insertion order leading to it. If not, clearly explain why.



3. (14 points) Let's consider that the CPU time to process a key-value pair is always zero (both in the map and the reduce functions), as well as the time to retrieve a disk block, and we can send control messages between machines without any cost. However, our network bandwidth is 10MB/sec and we only have one processor per machine. We have ten machines, which have a direct connection between them (i.e., sending data from machine A to machine B is always one hop away, for any pair of machines, as depicted in the figure).

   We are launching a MapReduce job with a mapper and a reducer in each of these machines and an input file of key-value pairs of an overall size of 1GB[1] (whose chunks are already uniformly distributed in the ten machines, without any replication and not necessary of default size). The map function generates exactly one key-value pair per call, with exactly the same size as the key-value pair in the input. Assume that the hash function (without any cost) used in the partition phase perfectly maps the values in a uniform way; and also that when two machines are transferring data from one to another, they cannot perform any other data transfer activity in parallel (i.e., while A is transferring data to B, neither A nor B can send or receive anything from anyone else until this transfer is over, A can only send to B and not receive anything from anyone, B can only receive from A and not send anything to anyone, not even A). Supposing that scheduling the data transfer in the cluster of ten machines is perfect and there is not any idle period for any of the machines, which is the **duration of the job**? Explicit any **calculation and assumption** you need to make.

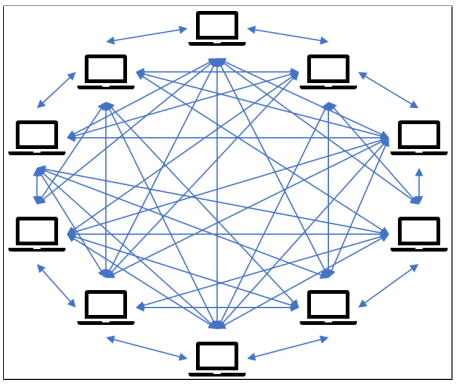


Figure 1: Fully connected architecture

Concept	**Cost**
Process any key-value pair	0s
Retrieve a disk block	0s
Send control messages between machines	0s
Network bandwidth	10MB/s

[1] Assume 1GB=1000MB.

4. (20 points) Assume that "spark" variable is a **Spark** session and the `dataset.csv` contains the two columns "Color" and "Radius". Clearly **identify the problems** you find in the following Spark code and propose some fix to obtain the expected result.

```
df0= spark.read.csv("dataset.csv", \
                    header='true', \
                    inferSchema='true', \
                    sep=',')
df1 = df0.select("Color", "Radius")
df2 = df1.withColumn("Pi", 3.141592)
df3 = df2.withColumn("Area", df2.Radius*df2.Radius*df2.Pi)
df4 = df3.groupBy("Color").max("Area")
df5 = df4.sort("Color")
```

.....

.....

.....

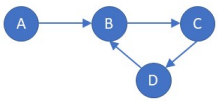
5. (20 points) **Modify** the following **MapReduce code** in Python so that given a file with key-value pairs representing the edges of a graph[2] (i.e., the key is the label of the first node and the value is the label of the second), it identifies all pairs of nodes connected by a path of length two (if two nodes are connected multiple times, they will be repeated in the output). Do not need to optimize it in any way, just make it work as expected.

```
def map(k,v):
        return [(k,(k,v))]

def reduce(k, lv):
        outgoing = []
        incoming = []
        for v in lv:
                if k == v[0]:
                        outgoing.append(v[1])
                elif k == v[1]:
                        incoming.append(v[0])
        retValue = []
        for o in outgoing:
                for i in incoming:
                        retValue.append(i+"-2->"+o)
        return retValue
```

Example of Input

Key	Value
A	B
B	C
C	D
D	B



Expected output

A	-2->	C
D	-2->	C
B	-2->	D
C	-2->	B

.....

.....

.....

[2] Without reflexive edges like (A,A).

6. (13 points) Identify and briefly explain the two **problems** of taking a "**Load First, Model Later**´´ approach. Propose a solution or mitigation action for each of them.

   (a) .....

   Problem :

   Risk của ph/ph này là các files có thể bị massively accumulated ko theo thứ tự, KQ là tạo ra Data Swamp, mà ở đó việc tìm relevant data cũng rất khó.

   Solution/Mitigation action :

   Giải pháp là tạo some organizzation of files và semantiacally annotate chúng với metadata.

   (b) .....

   Problem :

   When data is ingested without a predefined schema, there is a higher risk of inconsistencies and errors in the data.

   Solution/Mitigation action :

   Implement robust data validation and cleaning processes as part of the ingestion pipeline.