EXERCISE 1.0

In [18]:
```
rdd = sc.textFile('/data/students/bigdata_internet/lab2/word_frequency.tsv')
newRDD = rdd.map(lambda e : (e.split('\t')[0], int(e.split('\t')[1])))
sample = newRDD.takeSample(True, 5)
```

1.0.1 Can you draw 5 samples from the input RDD? Which command do you use?

In [2]:
```
print(sample)
```

```
[("'loose'", 1), ('identify', 369), ('fruit-flavored-rope-like-candy', 1), ('seru
m', 39), ('machineIt', 1)]
```

1.0.2 Now pick the first 5 words in order of frequency

In [7]:
```
orderedRDD = newRDD.top(5, lambda e : e[1])
print(orderedRDD)
```

```
[('the', 1630750), ('I', 1448619), ('and', 1237250), ('a', 1164419), ('to', 99797
9)]
```

1.0.3 How many words does the file contain?

In [6]:
```
print(newRDD.count())
```

```
339819
```

1.0.4 Is `word_frequency.tsv` a folder or a file? It is a folder containing three items (_SUCCESS(empty),part-00000,part-00001).

EXERCISE 1.1

In [8]:
```
prefix = 'ho'
filteredRDD = newRDD.filter(lambda e: e[0].startswith(prefix))
```

1.1.1 How many lines are left?

In [9]:
```
print(filteredRDD.count())
```

```
1519
```

In [19]:
```
maxFreqRDD = filteredRDD.map(lambda e : int(e[1]))
maxFreq = maxFreqRDD.takeOrdered(1, lambda n: -1*n)[0]
```

1.1.2 How frequent is the most frequent word of the selected sample (i.e., the maximum value of `freq` in the lines obtained by applying the filter)?

In [11]:
```
print(maxFreq)
```

```
36264
```

1.1.3 Report the code of 3 different ways to solve the task number 1.1.2 (we only want the frequency, i.e., a number and not a tuple/list)

maxFreq = maxFreqRDD.reduce(lambda e1, e2 : max(e1,e2))

maxFreq = maxFreqRDD.top(1)[0] It could also be done by using max on the frequencies.

EXERCISE 1.2

In [14]:
```
temp = int(70/100*int(maxFreq))
filteredRDD2 = filteredRDD.filter(lambda line: line[1]>temp)
```

Out[14]:
```
[('hot', 32944), ('how', 36264)]
```

EXERCISE 1.3

1.3.1 Count the number of selected lines and print this number on the standard output.

In [15]:
```
filteredRDD2.count()
```

Out[15]:
```
2
```

1.3.2 Save the selected words (without frequency) in an hdfs output folder. Every line should contain a single word and ends with a semicolumn (`;`).

In [15]:
```
out=filteredRDD2.map(lambda l: l[0]+';')
```

Out[15]:
```
['hot;', 'how;']
```

In [ ]:
```
outputPath= "/user/s307735/lab2Es1" #change path to save again
out.saveAsTextFile(outputPath);
```

EXERCISE 2

In [ ]:
```
from pyspark import SparkConf, SparkContext
conf = SparkConf().setAppName("My app")
sc = SparkContext(conf = conf)

import sys
import time

start = time.time()


name_file=sys.argv[0]
print(f"Name of the file is {name_file}")
prefix=sys.argv[1]
print(f"Il prefisso è:{prefix}")
outputPath=sys.argv[2]
print(f"L'output folder in cui salvare il file è:{outputPath}")


inputPath="/data/students/bigdata_internet/lab2/word_frequency.tsv"
rdd=sc.textFile(inputPath)
samplesRDD=rdd.map(lambda l: (l.split('\t')[0],int(l.split('\t')[1])))
hoWordsRDD = samplesRDD.filter(lambda line: line[0].startswith(prefix))
hoWordsRDD_fin = hoWordsRDD.map(lambda e : e[0] + '-' + str(e[1]) )
hoWordsRDD_fin.saveAsTextFile(outputPath)

stop = time.time()
print(f"Program took {stop - start} seconds to run")
```

2.1 Run your script locally and in the cluster (--master option). How much time do the two modes require to run? Is there a difference? Can you give a plausible explanation?

Yarn: between 12.04 seconds and 9.56 seconds (on multiple tries)
local: 2.63
Running a PySpark script locally can take less time than running it in cluster mode. Locally, the time taken to transfer data over the network can be avoided, which can help to speed up the processing time.

2.2 In this application, would caching an RDD increase the performance? If yes, which RDD would you cache?
If we analyze the tree that has rdd as root, there is only one action. So, it is not necessary to cache this application, however, the only plausible candidate could be 'rdd'.

BONUS TASK - EXERCISE 3

In [20]:
```
RDD = sc.textFile('/data/students/bigdata_internet/lab2/finefoods_text.txt')
newRDD = RDD.flatMap(lambda e : (e.split(' ')))
newRDD2 = newRDD.filter((lambda x: x != ''))
```

3.1 How many words (with repetitions) does it contain? Consider a word all the characters between spaces (elements found with `split()` method)

In [19]:
```
num =newRDD2.count()
print(num)
```

```
[Stage 12:==============================>                          (1 + 1) / 2]
45444841
```

3.2 Report the code to obtain the word frequency file starting from the original file.

In [26]:
```
freq=newRDD2.flatMap(lambda line:map(lambda w: (w, 1), line.split(' '))).reduceByKe
freq.take(5)
```

Out[26]:
```
[('have', 338996),
 ('bought', 46988),
 ('several', 19688),
 ('of', 792000),
 ('Vitality', 252)]
```