Class Activity 1

31/08/2021

I. Ways to print n^{th} line from a file

We explored four ways to print n^{th} line from a file. Let's look at them.

```
head -5 sample.csv | tail -1
awk 'NR==5 {print $0}' sample.csv
cat sample.csv | awk 'NR==5'
sed -n 5p sample.csv
```

Here, all of them would print the 5^{th} line of a file sample.csv but which one is the best in case of large files containing thousands or millions of lines. To confirm which one works best, we experimented with two csv files. One with $39k$ lines and the second one with $300k$ lines.
1. File with $39k$ lines: First we pick csv file with $39k$ lines and tried to print $35k^{th}$ line and noted the times for all the commands.

```
#!/bin/bash
wc -l sample.csv
time cat sample_39k.csv | awk 'NR==35000'
time head -35000 sample_39k.csv | tail -1
time awk 'NR==35000 {print $0}' sample_39k.csv
time sed -n 35000p sample_39k.csv
```

The time for the commands in the above script is shown as:

```
39828 sample_39k.csv (no. of lines)
real 0m0.015s
real 0m0.028s
real 0m0.013s
real 0m0.013s
```

2. File with $300k$ lines: Now, we pick a csv file having $300k$ lines. We tested all of the commands for $35k^{th}$ and $300k^{th}$ lines and compared these commands on the basis of their time. Below is the script which is made to run the commands and get the time for each command.

```
#!/bin/bash
wc -l sample_300k.csv
time cat sample_300k.csv | awk 'NR==35000'
time head -35000 sample_300k.csv | tail -1
time awk 'NR==35000 {print $0}' sample_300k.csv
time sed -n 35000p sample_300k.csv
```

The time for the commands in the above script is shown as:

```
318628 sample_300k.csv (no. of lines)
real 0m0.120s
real 0m0.026s
real 0m0.101s
real 0m0.089s
```

Now, we check the time for each command for 300k^{th} line of the file.

```
#!/bin/bash
wc -l sample.csv
time cat sample_300k.csv | awk 'NR==300000'
time head -300000 sample_300k.csv | tail -1
time awk 'NR==300000 {print $0}' sample_300k.csv
time sed -n 300000p sample_300k.csv
```

Let's look at the times:

```
318628 sample.csv (no. of lines)
real 0m0.135s
real 0m0.209s
real 0m0.100s
real 0m0.090s
```

Considering the above results, we can say that `sed` command works well for larger files followed by `awk` command. However, an important point to note about `head` and `tail` is that the time of this command depends on the total number of lines in a file and which line we want to access. The other commands would take a similar time for both the cases of accessing a line that is present at the beginning or at the end of the file in a larger file. On the other hand, if we are accessing a line that is at the lower end vs a line that is present at the higher end of the larger file, `head` and `tail` would work differently and take lesser time for the line at the lower end as it doesn't check/ scan the complete file. So, in general, we can say that `head` and `tail` is not the faster ones as compared to the other commands except the case in which we need to get the line that is present at the lower end of the file.