

Part 0:

Link for introduction: <https://github.com/illinoistech-itm/agupta110>

Link for image: <https://github.com/illinoistech-itm/agupta110/issues>

Part 1:

Dataset 1: Year 1990

```
1990.gz mv 1990.gz all
vagrant@vagrant-ubuntu-trusty-64:~$ mv 1990.gz all
vagrant@vagrant-ubuntu-trusty-64:~$ time -p .max_temperature.sh
.max_temperature.sh: command not found
real 1.67
user 0.48
sys 0.13
vagrant@vagrant-ubuntu-trusty-64:~$ time -p ./max_temperature.sh
1990 607
real 15.51
user 14.11
sys 1.34
vagrant@vagrant-ubuntu-trusty-64:~$ free -m
              total        used        free      shared    buffers     cached
Mem:           2001         311        1690           0          24         164
-/+ buffers/cache:         122        1879
Swap:              0              0              0
vagrant@vagrant-ubuntu-trusty-64:~$ apt-get install sysstat

update-alternatives: using /usr/bin/sar.sysstat to provide /usr/bin/sar (sar) in auto mode
Processing triggers for libc-bin (2.19-0ubuntu6.9) ...
Processing triggers for ureadahead (0.100.0-16) ...
vagrant@vagrant-ubuntu-trusty-64:~$ mpstat
Linux 3.13.0-107-generic (vagrant-ubuntu-trusty-64)      01/26/2017      _x86_64_      (1 CPU)

04:31:02 PM CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
04:31:02 PM all     0.65    0.52    1.80    0.48     0.00    0.03    0.00    0.00    0.00   96.53
vagrant@vagrant-ubuntu-trusty-64:~$
```

Dataset 2: Year 1990 and 1992

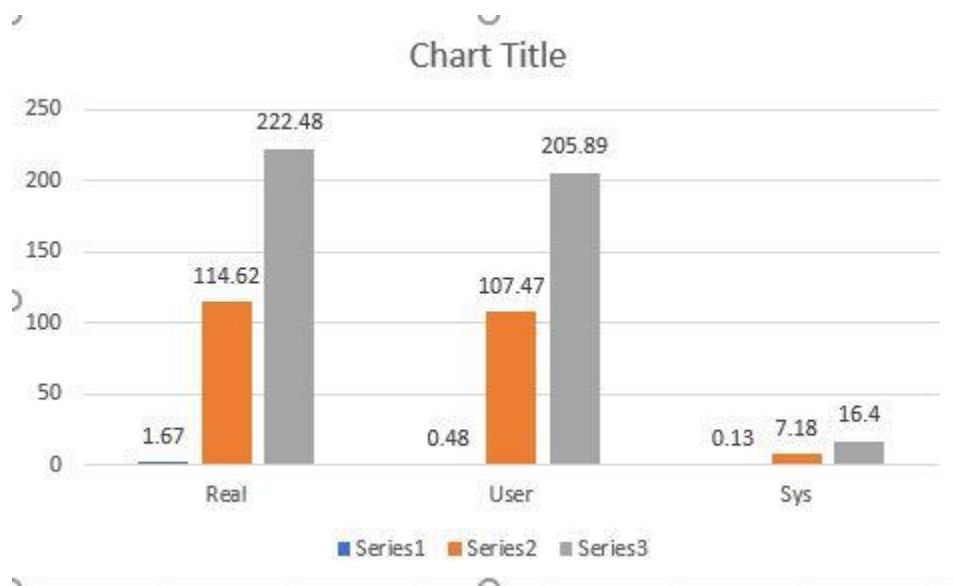
```
vagrant@vagrant-ubuntu-trusty-64:~$ mv 1992.gz all
vagrant@vagrant-ubuntu-trusty-64:~$ time -p ./max_temperature.sh
1990 607
1992 605
real 114.62
user 107.47
sys 7.18
vagrant@vagrant-ubuntu-trusty-64:~$ free -m
              total        used        free      shared    buffers     cached
Mem:           2001        1251         749           0          27        1075
-/+ buffers/cache:         148        1853
Swap:              0              0              0
vagrant@vagrant-ubuntu-trusty-64:~$ mpstat
Linux 3.13.0-107-generic (vagrant-ubuntu-trusty-64)      01/26/2017      _x86_64_      (1 CPU)

04:38:45 PM CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
04:38:45 PM all     1.66    0.49    1.85    0.46     0.00    0.03    0.00    0.00    0.00   95.52
vagrant@vagrant-ubuntu-trusty-64:~$
```

Dataset 3: All years

```
vagrant@vagrant-ubuntu-trusty-64:~$ mv 1993.gz all
vagrant@vagrant-ubuntu-trusty-64:~$ time -p ./max_temperature.sh
1990    607
1991    607
1992    605
1993    567
real 222.48
user 205.89
sys 16.40
vagrant@vagrant-ubuntu-trusty-64:~$ free -m
              total        used        free      shared  buffers   cached
Mem:           2001         1934          67           0          1       1784
-/+ buffers/cache: 148        1853
Swap:              0           0           0
vagrant@vagrant-ubuntu-trusty-64:~$ mpstat
Linux 3.13.0-107-generic (vagrant-ubuntu-trusty-64)      01/26/2017      _x86_64_      (1 CPU)
04:45:52 PM CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
04:45:52 PM all     3.50    0.47    1.98    0.45    0.00    0.03    0.00    0.00    0.00   93.56
vagrant@vagrant-ubuntu-trusty-64:~$
```

Comparison:



Series 1: Dataset 1

Series 2: Dataset 2

Series 3: Dataset 3

Since Dataset 1 has only 1 year's records, its execution time is the least. Whereas since Dataset 2 has 2 year's records, its execution time is more than that of dataset 2. Dataset 3 has records of 4 years so its dataset is the biggest hence its execution time is the highest.

Part 2.

My code is not running.

I broke down the components into various sub parts. I used the internet to see the code.

These are as follows:

1. I first made a connection to the database. I used the internet to see the code.
Reference: <https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm>
2. After that used the concept of FileInputStream and GZIP input stream to read the file.
Reference: <http://www.java2s.com/Code/Java/File-Input-Output/Readsomedatafromagzipfile.htm>
3. Then I used substring concept using the explanation of the data from the book and found the indices of the substring as taught in the class. I used the below explanation to decide the indices.

Example 2-1. Format of a National Climatic Data Center record

```
0057
332130 # USAF weather station identifier
99999 # WBAN weather station identifier
19500101 # observation date
0300 # observation time
4
+51317 # latitude (degrees x 1000)
+028783 # longitude (degrees x 1000)
FM-12
+0171 # elevation (meters)
99999
V020
320 # wind direction (degrees)
1 # quality code
N
0072
1
00450 # sky ceiling height (meters)
1 # quality code
C
N
010000 # visibility distance (meters)
1 # quality code
N
9
-0128 # air temperature (degrees Celsius x 10)
1 # quality code
-0139 # dew point temperature (degrees Celsius x 10)
1 # quality code
10268 # atmospheric pressure (hectopascals x 10)
1 # quality code
```

Book: Hadoop: The Definitive Guide, 4th Edition Storage and Analysis at Internet Scale, By Tom White

4. I finally read the internet and used `preparedstmt.setString()` to insert and finalise the statements.

Reference:

<http://www.java2s.com/Code/JavaAPI/java.sql/PreparedStatementsetStringintparameterIndexStringx.htm>

Some screenshots:

1) Database creation

```
mysql> create database arjweek2;
Query OK, 1 row affected (0.00 sec)

mysql> connect arjweek2
Connection id:      44
Current database: arjweek2

mysql> exit
Bye
vagrant@vagrant-ubuntu-trusty-64:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 45
Server version: 5.5.54-0ubuntu0.14.04.1 (Ubuntu)
```

2) Schema creation

```
mysql> connect arjweek2;
Connection id:      50
Current database: arjweek2

mysql> create table arjweek2(ul1 varchar(125),usafwsi varchar(125),
-> wban varchar(125),Year varchar(125),DateMonth varchar(125),hour varchar(125),ul2 varchar(125),latitude varchar(125),
-> longitude varchar(125),ul3 varchar(125),elevation varchar(125),ul4 varchar(125),ul5 varchar(125),
-> wind_direction varchar(125),qc1 varchar(125),ul6 varchar(125),ul7 varchar(125),ul8 varchar(125),
-> sky_cieling varchar(125),qc2 varchar(125),ul9 varchar(125),ul10 varchar(125),visibity varchar(125),
-> qc3 varchar(125),ul11 varchar(125),ul12 varchar(125),Temp varchar(125),
-> qc4 varchar(125),dew_point varchar(125),qc5 varchar(125),atm_pressure varchar(125),qc6 varchar(125));
Query OK, 0 rows affected (0.07 sec)
```

3) Error message

```
vagrant@vagrant-ubuntu-trusty-64:~$ vim q2code.java
vagrant@vagrant-ubuntu-trusty-64:~$ CLASSPATH=$CLASSPATH:/usr/share/java/mysql.jar
vagrant@vagrant-ubuntu-trusty-64:~$ export CLASSPATH
vagrant@vagrant-ubuntu-trusty-64:~$ echo $CLASSPATH
:/usr/share/java/mysql.jar
vagrant@vagrant-ubuntu-trusty-64:~$ javac q2code.java
vagrant@vagrant-ubuntu-trusty-64:~$ java q2code
Exception in thread "main" java.sql.SQLException: Parameter index out of range (32 > number of parameters, which is 31).
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:1086)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:989)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:975)
    at com.mysql.jdbc.SQLException.createSQLException(SQLException.java:920)
    at com.mysql.jdbc.PreparedStatement.checkBounds(PreparedStatement.java:3796)
    at com.mysql.jdbc.PreparedStatement.setInternal(PreparedStatement.java:3778)
    at com.mysql.jdbc.PreparedStatement.setString(PreparedStatement.java:4599)
    at q2code.main(q2code.java:103)
```