Andrew Covarrubias (axc554)

Questions

Question 1:
What are the results for each of the programs? In your report, include the program name, and copy/paste the output.

Example1:

```
Example1
"C:\Program Files\Java\jd
35   208  169  212  160
298  302  312  299  305
459  462  439  450  445
596  614  588  601  586
744  730  762  772  755
906  901  916  905  920
1030 1029 1034 1053 1066
1195 1190 1179 1190 1179
1328 1327 1329 1320 1320
```

Example2:

```
Example2
"C:\Program Files\Java\jdk1.8.0_101
40     40     181    155    209
611    592    598    590    602
1341   1335   1342   1336   1335
2441   2588   2419   2596   2379
3758   3725   3863   3692   4105
6046   5661   5552   5583   5579
7592   7631   7608   7581   7775
9668   9544   9661   9486   9900
12398  12579  12654  12557  12414
```

Example3:

```
Example3
"C:\Program Files\Java\jdk1.8.
45    41    167   172   214
599   601   600   612   598
1343  1342  1342  1344  1340
2394  2382  2387  2380  2395
3741  3814  3914  3935  3966
5836  5587  5452  5942  5733
7501  7560  7318  7271  7252
9532  10089 9704  10359 10014
12418 12671 12425 12618 12603
```

Example4:

```
Example4
"C:\Program Files\Java\jdk1.8.0_101\bin\java"
750     657     663     650     660
1066    1077    1058    1063    1060
1705    1717    1763    1767    1750
2772    2791    2776    2791    2797
4479    4497    4473    4481    4526
7457    7533    7519    7225    7344
11834   11888   12382   12285   12437
19547   19001   19528   19229   19395
31270   31446   33758   32907   32586
```

Question 2:
Explain (in complete sentences) how each algorithm compares quantitatively. How much faster does the time grow for each algorithm as compared to the others?

The four different examples each use different algorithms which all vary in speed.

In order of speed Example1 is the fastest and Example4 is the slowest, the order being Example1, Example2, Example3, Example4 and the time growth is the reverse Example4 grows the quickest and Example1 grows the slowest.

Example1 has a linear run time of n, where each value is run through one time due to the single loop

Example2 has a squared run time of n^2, where each value is run through two times because of the nested for loops. Example2 will grow at a rate of Example1 squared

Example3 has a cubed run time of n^3. This is due to it being a combination of Example1 and Example2. It has both a single for loop which runs through all the values and a nested for loop which runs through each value twice. As a result the run time can be shown by the following equation $n^3 = n^2 * n$. Example3 will grow at a rate of Example3 cubed.

Example4 has an exponential run time of 2^n meaning it will undoubtedly take the longest. It's run time comes from the recursive method it calls itself. Meaning for a given n value the previous two n values are required and the process repeats itself for however large the n value is.

Question 3:
What do you notice about the results that might be unexpected?

For the Example1 and Example2 it seemed to be that the first few values were always lower than the rest of the values. It seemed to repeat, where the first few trials of the initial n values would be lower than the later trials. For instance Example1 starts with 35 and the rest of the values are above 150.

Question 4:
Offer a possible explanation for the unexpected results in Question (3) above.

After thinking about it for a while I could not really think of a reasonable explanation for the unexpected results. The only thing that might have gone wrong was the way I used for loops to print all the trials with different n values. I tried playing around with the placement of the loops and time counting, but nothing seemed to work.