COSC 311 FALL 2016 PROGRAMMING PROJECT #1 Execution Time

Distributed: 9/26/2016 **Due**: 10/5/2016

Problem Statement: Experimentally determine the Big Oh for the problem of n-choose-k without replacement.

Problem approach: Obtain execution time results for three different algorithms implementing a solution to n-choose-k without replacement. Pseudo-code for the solutions is given below.
Fit curves to the results to find a good estimate of the Big Oh.
Determine the errors between the fitted curves and the data.

More detail:
—- Obtain enough data to make a convincing conclusion. For this problem, there are two independent variables: data set size, n, and the number being chosen from the data set, k. The dependent variable is time.

—- Holding n fixed, vary k and obtain execution times.
Holding k fixed, vary n and obtain execution times.

—- In order to make a convincing argument, you need to do the following:
Ensure sufficient variety of data set sizes, n or k,
Take timing information on what you actually need to measure,
Compare the errors of the fitted curves against the actual data.

—- Data set size:
Two data points will suffice to perfectly fit a straight line.
Three data points will suffice to perfectly fit a parabola.
Therefore, you will need significantly more than four data points to measure. At the minimum, you should obtain ten data points for each curve

Algorithms:
(1) Choose k values. Repeat until all values are unique.
(2) Make an array named used[] that corresponds to the n data items. Each time a datum is chosen, mark it in the used[] array. If a chosen data was previously chosen, repeat until you get a new value.
(3) Each time a datum is chosen, swap it with the last element in the data array. Reduce the size of the data array by one each time a datum is chosen.

Range of data set size:
The maximum value of m (m is n or k) should be large enough to successfully acquire timing results within a reasonable amount of elapsed wall-time (say, less than 10 minutes).
Also, the range of m must be such that meaningful measurements are obtained at all data point values. The execution time must be measurable (i.e., not too small).
The range of the data must be wide enough to allow for the fitted curves to show significant errors.

Curve fitting:

Fit the following curves:

f(n) = a m b

g(n) = a m^2 + b

h(n) = a * 2^m + b

k(n) = a * m! + b

where m stands for n or for k.

Ensure your code is correct

Demonstrate your code is correct by producing output of n-choose- for each algorithm. n = 10, k = 5. Repeat this test 5 times for each algorithm.

Plotting:

—- Plot the data with the fitted curves. Show the data points as visible marks, and the fitted curves overlaid on top.

—- The plot must have clearly labeled axes, and there must be a legend giving line shape with corresponding fitted curve.

—- You will need another chart, a bar chart, that compares the errors of the different fits (three bars, one for each fitted curve).

Conclusion:

Give a one paragraph summary giving your conclusion with supporting argument from the data, the plot(s) and the error chart.

Program design:

Each experiment is in a separate class. An experiment comprises all the runs on a particular algorithm, varying n and varying k.

Input to an experiment (may be run-time or file) is given in (n, k) pairs.

Output of each experiment is:

- Name of algorithm
- Date of execution
- Each (n,k) pair immediately followed by measured execution time.
- “Sanity check”: the output of 10 choose 5 (repeated 5 times), appropriately labelled.

Turn in:

Hardcopy of code

All output (three experiments)

Plots and charts

Conclusion: A single page report summarizing the result of the experiments — what is the Big Oh for each algorithm? Support your conclusion by referring to the measurements, the graphs and the errors.

UML

Grade based on:

— Correctness of code, satisfying spec 85%

— Readability, elegance, documentation 15%

How to get timing information?
You may use System.currentTimeMillis() or System.nanoTime(). Note, there are other ways of measuring running time. Any of those are fine – some are better than using the Java System API
Measure the correct thing.