| | | |
|---|---|---|
| **CSE 7350/5350** | **Homework Set # 1** | **Due date:** |
| **D.W. Matula** | **Fall 2013** | **17 Sept 2013** |

# Measuring Algorithm Efficiency

**Total Points: 70**

**Reading:**

**(1) Chapters 1, 2 and 3 (pp 5-64) of text (Cormen,et.al., 3rd edition, 2009);**
**(2) Preface and pp 1-33 of Computers and Intractability(Garey and Johnson)**

**Grading Policy:**

***Unless otherwise stated all problems are 10 points each. Partial credit is available on all problems.***

**1. [Algorithm Complexity] (15 points)**

Discuss in 3-4 pages the issues involved in implementation independent measurement of the space and time complexity of algorithms. Include an analysis from your current perspective of the following fundamental issues:

a. list five basic elements of space measurement and a problem illustrating each

b. polynomial/exponential growth in time or space measurement

c. absolute/relative growth time measurement

d. upper/lower asymptotic bounds

e. worst/average case analysis

f. deterministic/probabilistic algorithm

g. deterministic (solution algorithm) vs nondeterministic (verification algorithm)

h. ~~verification by "zero knowledge proof" (interactive algorithm)~~

Note: This exercise is designed to familiarize you with basic concerns in the analysis of algorithms and deep comprehension is not expected at this time. This same question later in the course should elicit a more comprehensive response.

**2. [Measuring Growth Rates]**

a. Text problem 1-1, p. 14 measuring f(n) in nanoseconds.

b. Text problem 3-2, p. 61. Do only for Big-Oh, Big-Omega, and Big-Theta.

c. Follow the instructions for text problem 3-3 (a) for the following functions:

| | | | | |
|---|---|---|---|---|
| $(5/2)^n$ | $n^2$ | (lg n)! | $lg^2 n$ | 2(constant) |
| $n2^n$ | $2^n$ | $2^{lg\ n}$ | n! | n / lg n |

**3. [Problem Statements]**

Express the following five loosely described problems carefully in { Instance, Question } form as utilized in ***"Computers and Intractability"***. For each problem discuss the best time and space complexity you are aware of for *solving* the problem (from scratch) along with a few words naming or describing the method.

(i) Finding the median of n = 2 k + 1 integers.

(ii) Finding the 2 largest and 2 smallest of n integers.

(iii) Determining that a graph is bipartite.

(iv) Determining that a list of *n* numbers has no duplicates.

For the following just describe a verification algorithm and the method and efficiency of checking the answer using the verification algorithm. Problem numbers and pages refer to Gary and Johnson.

(v) Determining that the maximum number of edge disjoint paths between vertices *v* and *w* in a graph is less than *k*.

(vi) Clique of size J (see page 47 of Gary and Johnson).

(vii) Set Packing: SP3 (p. 221).

**4. [Estimation]**

For a random geometric graph, G(n, r), estimate the average degree of a vertex:

(a) at least distance r from the boundary,

(b) on the boundary (convex hull),

and estimate the time (big Oh) of determining all edges employing:

(c) all vertex pairs testing,

(d) the line sweep method,

(e) the cell method.

**5. [Verification by digit sums]**

Refers to "Finite Precision Number Systems and Arithmetic", page 13.

(i) Problem 1.4.3 a, b and d-h.

(ii) Problem 1.4. 4.

**6. [Implementation Testing] (15 points)**

You are to implement and test a program for summing 1/x as x runs over all approximately eight million (23 fraction bits) single precision floating point numbers in the interval [1, 2). You are to do this on a server, PC (or Mac) of your choice. You are first asked to predetermine estimates of your implementation's computation time (architecture and compiler dependant), result accuracy (algorithm and round-off dependant), and result value (real valued sum estimate). Specifically:

a. Discuss the computational environment for your tests, including the compiler, operating system, machine MHz and cycle times for appropriate instructions and whether pipelining affects your execution time.

b. Predetermine an estimate of the time utilizing single precision for the variables for all computations from any documentation you can find from the hardware manufacturer and/or compiler and system provider.

c. Predetermine a rough estimate of the exact sum (hint: how many terms are being added and how large is an "average term").

d. Predetermine an estimate of the accuracy. Single precision computation should be done in round to nearest mode as provided by standard C implementations. By accuracy of the sum we mean the difference between the rounded sum of rounded values compared to the exact sum of exact values.

e. Give the measured running time and the computed result for your implementation. Compare the results with your estimates of running time and approximate sum. Compute the sum in double precision and single precision and compare to give a reasonable value for the accuracy of the single precision sum. Compare your result with another student's results that might have performed the sum in a different order. Can you explain the size of the approximation error?