Problem Set 2: Introduction to Analysis of Algorithms and Running Time Calculations

Introduction to Analysis of Algorithms

Problem 1

Order the following functions in increasing order by growth rate:

$N^3, 2^{N/2}, N \log \log N, N^2, 2/N, N\log^2 N, N \log N, N, N \log\left(N^2\right), N^{1.5}, 2^N, N^2 \log N, 12, \sqrt{N}$

Indicate which functions grow at the same rate.

Problem 2

Alice is a security consultant. Her consulting contract states that she earns a fee of 2 dollar-bucks for the first hour of work and that her hourly fee is squared for each subsequent hour of work, i.e. the fee progresses: 2 dollar-bucks, 4 dollar-bucks, 16 dollar-bucks, and so on.

A. What is Alice's fee for her *N*th hour?
B. Give a big-O estimate for the number of hours of work it would take for Alice's **hourly** fee to reach *D* dollar-bucks.
C. Give a big-O estimate for the number of hours of work it would take for Alice's **total** fee to reach *D* dollar-bucks.

Problem 3

An algorithm running on a particular computer takes 700 microseconds ($1\mu s = 10^{-6} s$) for input size 100. How large a problem (in terms of input size) can be solved in 3 minutes if the algorithm's running time is the following (assume low-order terms are negligible)?
*Give your answer in scientific notation: 1.234e5 (1.234 * 10^5 ≅ 123,400).*
Hint: Compute the constant C based on the given information, then solve for N.

A. linear: $O(N)$
B. linearithmic: $O(N \log_2 N)$
C. quadratic: $O(N^2)$
D. cubic: $O(N^3)$
E. exponential: $O(2^N)$

Problem 4

Programs `Foo` and `Bar` are analyzed and found to have worst case running times no greater than $221N\log_2 N$ and $3N^2$, respectively. Answer the following questions, if possible:

A. Which program has the better guarantee on the running time for values of $N < 100$?
B. Which program has the better guarantee on the running time for values of $N > 10000$?
C. For what value of N are the worst case running times approximately equal?
D. Is it possible that `Bar` will run faster than `Foo` on all possible inputs? Why (not)?

Running Time Calculations

Problem 1

For each of the following three program fragments:

A. Use big-O notation to analyze the running time (i.e. the value of sum).
B. Implement the code in any programming language and give the actual running time for several values of N.
    a. submit evidence you did this, i.e. the code you wrote
C. Compare your analysis with the actual running times.
    a. submit evidence you did this, i.e. tables or plots of data

Fragment 1

```
sum = 0
for i from 1 to n do
     sum = sum + 1
end
```

Fragment 2

```
sum = 0
for i from 1 to n do
     for j from 1 to i do
     sum = sum + 1
     end
end
```

Fragment 3

```
sum = 0
for i from 1 to n do
     for j from 1 to i^2 do
     for k from 1 to j do
          sum = sum + 1
     end
     end
```

```
end
```

Problem 2

For each of the following operations, write out (e.g. in pseudocode) the algorithm **you** typically use for **hand-calculations** and give a big-*O* estimate of the time complexity of that algorithm.:

A. Add two *N*-digit integers
B. Multiply two *N*-digit integers
C. Divide two *N*-digit integers

Problem 3

Use big-*O* to estimate how much time is required to compute $f(x) = \sum_{i=0}^{N} a_i x^i$

A. using naive exponentiation?
B. using fast exponentiation?
C. using the following algorithm?

```
value = 0
for i from n to 0 by -1 do
    value = value * x + a[i]
end
```

Puzzle Problem [optional]

This one nerd-sniped me, so I'm sharing the fun.
This problem is NOT part of the problem set, it is just for fun.

Show that X^{62} can be computed with only eight (8) multiplications.

Extra Challenge:

What is the smallest power *k* such that X^k requires at least eight multiplications?