Keshav Narang

September 27, 2021

**Data Structures and Algorithms - Analysis of Algorithms**

1. Arrange the following expressions by growth rate from slowest to fastest.

   $4n^2$ , $\log_3 n$, n!, $3^n$ , 20n, 2, $\log_2 n$, $n^{2/3}$

   Use Stirling's approximation in for help in classifying n!

   Stirling's approximation states that $n! \approx \sqrt{(2\pi n)}\,(n/e)^n$

   2, $\log_3 n$, $\log_2 n$, $n^{2/3}$, 20n,$4n^2$, $3^n$, n!

2. Estimate the number of inputs that could be processed in the following cases:

   a. Suppose that a particular algorithm has time complexity T(n) = 3 x $2^n$, and that executing an implementation of it on a particular machine takes t seconds for n inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in t seconds?

      The machine can do n + 6 inputs in t seconds ($2^{n+6}$ = 64t).

   b. Suppose that another algorithm has time complexity T(n) = $n^2$, and that executing an implementation of it on a particular machine takes t seconds for n inputs. Now suppose that we are presented with a machine that is 64 times as fast. How many inputs could we process on the new machine in t seconds?

      The new machine can do 8n inputs in t seconds ($8n^2$ = 64t).

   c. A third algorithm has time complexity T(n) = 8n. Executing an implementation of the algorithm on a particular machine takes t seconds for n inputs. Given a new machine that is 64 times as fast, how many inputs could we process in t seconds?

      The new machine can do 64n inputs in t seconds (512n = 64t).

3. A hardware vendor claims that their latest computer will run 100 times faster than that of their competitor. If the competitor's computer can execute a program on input of size n in one hour, what size input can vendor's computer execute in one hour for each algorithm with the following growth rate equations?

    a. n

       100n (n x 100)

    b. $n^2$

       10n (n x $100^{1/2}$)

    c. $n^3$

       4.64n (n x $100^{1/3}$)

    d. $2^n$

       6.64n (n x $\log_2 100$)

4. Using the definition of big-Oh, show that 1 is in O(1) and that 1 is in O(n).

   The definition of big-Oh states that if a function f(n) is less than or equal to a constant c, times another function g(n) that is greater than or equal to f(n) from an input size $n_o$ to infinity, f(n) belongs to g(n).

   In the first problem, f(n) = 1 and g(n) = 1, and for all n, f(n) <= g(n), so 1 is in O(1).

   In the second, f(n) = 1 and g(n) = n, so for all n greater than or equal to 1, f(n) <= g(n). Once again, f(n) must be a part of g(n), therefore.

5. For each of the following pairs of functions, either f(n) is in O(g(n)), f(n) is in Ω(g(n)), or f(n) = Θ(g(n)). For each pair, determine which relationship is correct. Justify your answer.

    a. $f(n) = \log n^2$; $g(n) = \log n + 5$.

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

b. $f(n) = \sqrt{n}$; $g(n) = \log n^2$

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

c. $f(n) = \log^2 n$; $g(n) = \log n$.

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

d. $f(n) = n$; $g(n) = \log n$.

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

e. $f(n) = n \log n + n$; $g(n) = \log n$.

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

f. $f(n) = \log n^2$; $g(n) = (\log n)^2$

lim $n\to\infty$ $f(n)/g(n)$ is zero; $f(n) = O(g(n))$

g. $f(n) = 10$; $g(n) = \log 10$.

lim $n\to\infty$ $f(n)/g(n)$ is 10; $f(n) = \Theta(g(n))$

h. $f(n) = 2^n$; $g(n) = 10n^2$

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

i. $f(n) = 2^n$; $g(n) = n \log n$.

lim $n\to\infty$ $f(n)/g(n)$ is infinity; $f(n) = \Omega(g(n))$

j. $f(n) = 2^n$; $g(n) = 3^n$

lim $n\to\infty$ $f(n)/g(n)$ is 2/3; $f(n) = \Theta(g(n))$

k. $f(n) = 2^n$; $g(n) = n^n$

lim $n\to\infty$ $f(n)/g(n)$ is zero; $f(n) = O(g(n))$

6. Determine $\Theta$ for the following code fragments in the average case. Assume that all variables are of type int.

a.

```
a = b + c;
d = a + e;
```

1

b.

```
sum = 0;
for (i=0; i<3; i++)
        for (j=0; j<n; j++)
                sum++;
```

3n

c.

```
sum=0;
        for (i=0; i<n*n; i++)
                sum++;
```

$n^2$

d.

```
for (i=0; i < n-1; i++)
        for (j=i+1; j < n; j++) {
                tmp = AA[i][j];
                AA[i][j] = AA[j][i];
                AA[j][i] = tmp;
        }
```

$\frac{1}{2} n^2$

e.

sum = 0;

i. for (i=1; i<=n; i++)

1. for (j=1; j<=n; j*=2)

a. sum++;

½ $n^2$

f.

sum = 0;

for (i=1; i<=n; i*=2)

for (j=1; j<=n; j++)

sum++;

½ $n^2$

g. Assume that array A contains n values, random takes constant time, and sort takes n log n steps.

```
for (i=0; i<n; i++) {
    for (j=0; j<n; j++)
        A[i] = util.random(n);
        sort(A);
    }
```

$n^3\log(n)$

h. Assume array A contains a random permutation of the values from 0 to n - 1.

sum = 0;

for (i=0; i<n; i++)

```
        for (j=0; A[j]!=i; j++)
                sum++;
```

$n^2$

i. 
```
sum = 0;
if (EVEN(n))
        for (i=0; i<n; i++)
                sum++;
else
        sum = sum + n;
```

n/2