Assignment - 4
Analysis of Algorithms

Problem #1

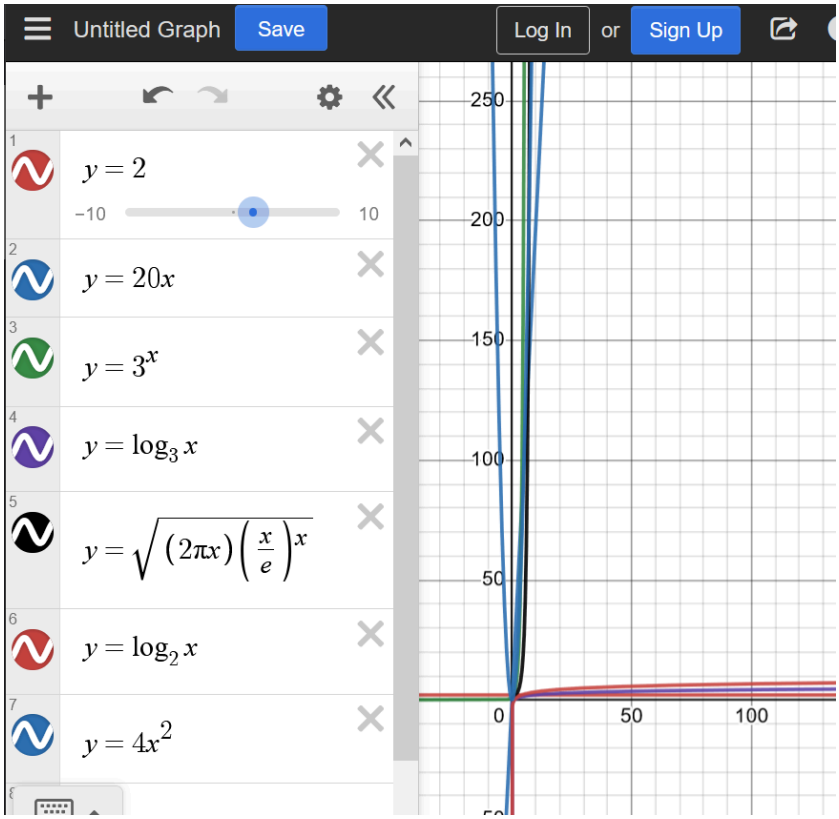
1) Arrange the following expressions by growth rate from slowest to fastest.

$4n^2$, $\log_3 n$, $n!$, 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 $n! \approx \sqrt{(2\pi n)}\ (n/e)^n$

SOLUTION:



Visually, we can see differences in which will grow faster. However we know from class the speeds of each type of curve:

Best time complexitities:

Method	Rating
constant	**Ideal**
logarithmic	**Ideal, most common**
linear	Mid
linearithmic	Mid
quadratic	Worse
cubic	Even Worse
exponential	**TERRIBLE**

And as we want to sort from slowest to fastest, we get the following answer:

Slowest							**Fastest**
2	**log2n**	**log3n**	**n^2/3**	**3n**	**20n**	**4n^2**	**n!**

Problem #2

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

(a) Suppose that a particular algorithm has time complexity $T(n) = 3 \times 2^n$, 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?

(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?

(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?

SOLUTION:

Each instance of n becomes 64n for this problem as it is asking for inputs available, not time complexity. If it was time complexity, it would be 1/64 n.

A) We can process 6 more inputs each second, so:

   n + 6 inputs

B) We can process 8x the inputs each second, so:

   8n inputs

C) We can process 64x the inputs each second, so:

   64n inputs

Problem #3

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?

n n^2 n^3 2^n

SOLUTION:

The size input that can be provided will be whatever the size the competitors can within an hour.

So n becomes **100n**

n^2 becomes **10n**

n^3 becomes **4.64n**

2^n becomes **n + 6.64**

Problem #4

4) 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.

Big-O Notation	Comparison Notation	Limit Definition
$f \in o(g)$	$f \lessdot g$	$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 0$
$f \in O(g)$	$f \leqslant g$	$\lim_{x\to\infty} \frac{f(x)}{g(x)} < \infty$
$f \in \Theta(g)$	$f = g$	$\lim_{x\to\infty} \frac{f(x)}{g(x)} \in \mathbb{R}_{>0}$
$f \in \Omega(g)$	$f \geqslant g$	$\lim_{x\to\infty} \frac{f(x)}{g(x)} > 0$
$f \in \omega(g)$	$f > g$	$\lim_{x\to\infty} \frac{f(x)}{g(x)} = \infty$

SOLUTION:

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

Answer is Θ

because the two functions (log) grow at the same rate, so they match the comparison of the listed notation. We can further prove this equality by the limit definition:

$$\lim_{x\to\infty}\left(\frac{\ln(x^2)}{\ln(x+5)}\right)$$

Solution

2

As this equals c in which c>0, it is **Θ**

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

Answer is Ω as:

$$\lim_{n\to\infty}\left(\frac{\sqrt{n}}{\ln\left(n^2\right)}\right)$$

Solution

$$\infty$$

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is Ω as:

$$\lim_{n\to\infty}\left(\frac{\ln\left(2^n\right)}{\ln(n)}\right)$$

Solution

$$\infty$$

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is Ω as:

$\lim_{n\to\infty}\left(\frac{n}{\ln(n)}\right)$

Solution

∞

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is Ω as:

Answer: $\lim_{n\to\infty} \frac{n\ln(2n)}{\ln(n)} = \infty$

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is O as:

$$\lim_{n\to\infty}\left(\frac{\ln(n^2)}{(\ln(n))^2}\right)$$

Solution

0

Which means that it is big O as f(n)/g(n) is zero. Therefore g(n) grows faster than f(n) as n becomes larger and larger.

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

Answer is **Θ**g(n)

as these are both constants, however f(n) is 10 and g(n) is 1, meaning that their limit is a constant equal to 10. This makes sense as they grow at the same rate.

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

Answer is Ω as:

$$\lim_{n\to\infty}\left(\frac{2^n}{10n^2}\right)$$

Solution

$$\infty$$

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is Ω as:

$$\lim_{n\to\infty}\left(\frac{2^n}{n\ln(n)}\right)$$

Solution

$$\infty$$

Which means that it is omega as infinite limit f(n)/g(n) is bigger than 0 and not a constant in all real numbers. Therefore f(n) grows faster than g(n) and matches the case.

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

Answer is O as:

$$\lim_{n\to\infty}\left(\frac{2^n}{3^n}\right)$$

Solution

$$0$$

Which means that it is big O as f(n)/g(n) is zero. Therefore g(n) grows faster than f(n) as n becomes larger and larger. Makes sense as g(n) is clearly faster in growth being the same thing but with a bigger constant.

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

Answer is O as:

$$\lim_{n\to\infty}\left(\frac{2^n}{n^n}\right)$$

Solution

$$0$$

Which means that it is big O as f(n)/g(n) is zero. Therefore g(n) grows faster than f(n) as n becomes larger and larger. It is like the last problem except it will grow WAY faster.

Problem #6 (5 but typo)

6) Determine Θ for the following code fragments in the average case. Assume that all variables are of type **int**.

SOLUTION:

a

θ(1)

as the simple addition operation has addition is a constant time operation despite the values of the numbers

b

Θ(n)

as despite whatever the value of n is, the code runs 3 times. Then the inner loop runs for n times, so we get a total number of iterations being 3*n. As the three runs are constant and n is the only significant amount, then the time complexity is considered n.

C

Θ(n*n) or (n^2)

as the loop runs n*n times, which is considerable for our time complexity. Therefore, as N*n is the number of runs, n*n is our complexity.

D

Θ(n^2)

as almost all elements of the 2D array are iterated over, and the array is size n*n.

E

Θ(n * log(n))

as j doubling runs in logarithmic time of n, and as the outer loop runs n times, it becomes n * the log time.

F

Θ(n * log(n))

for the same reasons as above, except the outer loop and inner are swapped in complexities. It is still them times each other however due to the nature of their looping.

G

Θ(n^2 * log(n))

There are two loops that iterate until n, so n*n. The sort takes n log n and the n is accounted for already, so our end answer is met because constant time is not considered for complexity here.

H

Θ(n^2)

as there is a nested loop that iterates n times, and another one which may iterate n times.

I

Θ(n)

as at the worst case this runs n times and otherwise runs <n times. Both are constant time.