DrillComparingOrdersOfGrowth-1

Tom Bohbot

September 2020

1 Ordering Running Times

1.1 n^2

1. Double the input size:
1) When the input size, x, is doubled it will take this program $(2x)^2$ to run.
2) $(2x)^2 = 4x^2$
3) grows by a factor of 4

2. Increase the input size by one:
1) When the input size, x, is increased by one it will take this program $(x + 1)^2$ to run.
2) $(x + 1)^2 = x^2 + 2x + 1$
3) grows by an additional $2x + 1$

1.2 n^3

1. Double the input size:
1) When the input size, x, is doubled it will take this program $(2x)^3$ to run.
2) $(2x)^3 = 8x^3$
3) grows by a factor of 8

2. Increase the input size by one:
1) When the input size, x, is increased by one it will take this program $(x + 1)^3$ to run.
2) $(x + 1)^3 = x^3 + 3x^2 + 3x + 1$
3) grows by an additional $3x^2 + 3x + 1$

1.3 $100n^2$

1. Double the input size:
1) When the input size, x, is doubled it will take this program $(100(2x))^2$ to run.
2) $100(2x)^2 = (100)(4x)^2 = 400x^2$
3) $400x^2 / 100x^2 = 4$

4) grows by a factor of 4

2. Increase the input size by one:
1) When the input size, x, is increased by one it will take $100(x + 1)^2$ to run.
2) $100(x + 1)^2 = 100(x^2 + 2x + 1) = (100x^2 + 200x + 100)$
3) grows by an additional 200x + 100

1.4 nlogn

1. Double the input size:
1) When the input size, x, is doubled it will take this program 2xlog2x to run.
2) 2xlog2x = 2xlogx + 2xlog2 3) grows by a factor of 2 and an additional 2xlog2. If the log's base is 2, then it grows by a factor of 2 and an additional 2x.

2. Increase the input size by one:
1) When the input size, x, is increased by one it will take (x + 1)log(x+ 1) to run.
2) (x + 1)log(x + 1) = xlog(x + 1) + log(x + 1).
3) grows by an additional one unit in the first log plus log(x + 1).

1.5 2^n

1. Double the input size:
1) When the input size, x, is doubled it will take this program $(2)^{2x}$ to run.
2) $(2)^{2x} = (2)^x * (2)^x$
3) grows by a multiplicative factor of $(2)^x$

2. Increase the input size by one:
1) When the input size, x, is increased by one it will take this program 2^{n+1} to run.
2) $2^{n+1} = 2^n * 2^1 = 2^n * 2$
3) grows by a multiplicative factor of 2

2 Really Understanding Order-of-Growth

2.1 n^2

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
3) n^2 = 36 trillion

4) n = 6,000,000 or 6 million.
5) The maximum input size is 6,000,000.

2.2 n^3

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
3) n^3 = 36 trillion
4) n = 33,019.3
5) The maximum input size is 33,019.

2.3 $100n^2$

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
3) $100n^2$ = 36 trillion
4) n^2 = 360 billion
5) n = 600,000
5) The maximum input size is 600,000.

2.4 nlogn

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
3) nlogn = 36 trillion
4) Through plugging and guessing, (2,889,069,820,989)log(2,889,069,820,989) = (2,889,069,820,989)*12.46075804 $\approx$ 35.99 trillion. Any number larger than 2,889,069,820,989 is larger than 36 trillion. When approaching the correct number I observed that the log did not change from 12.46075804, so I set an equation being X*12.46075804 = 36 trillion. I quickly found my result after that.
5) Therefore, the maximum input size is 2,889,069,820,989.

NOTE: This is assuming log base 10.
Source for calcuator: https://dqydj.com/log-base-10-calculator/

2.5 2^n

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
3) $2^n = 36$ trillion
4) Through plugging and guessing, 2^{45} is approximately 35.18 trillion. Therefore, n cannot exceed 45 as it would then be much higher than 35.18 trillion.
5) The maximum input size is 45.

2.6 2^{2^n}

Largest Input Size n to compute results in an hour assuming a computer that can do 10^{10} operations per second:
1) Maximum operations per minute = $60(10^{10}$ = 600,000,000,000 or 600 billion.
2) Maximum operations per hour = 60 x maximum operations per minute = 36,000,000,000,000 or 36 trillion.
4) Since we know that the largest input size for 2^n is 45, we only need to calculate for $2^n = 45$.
5) $2^n = 45$
6) n $\approx$ 5
7) Let n = 5 as n must be a whole number.
8) 2^{2^5} is approximately 4.29 billion.
9) If 2^{2^6} was plugged in, then the result we be much higher than 36 trillion. Therefore, n cannot exceed 5
10) The maximum input size is 5.