$$f(n) = \underbrace{3n^3}_{} + 6n^2 + n + 5 \quad \Rightarrow \quad O(n^3)$$

$O(n^2)$

$n = 1000 \rightarrow 5$ seconds

$9x$

$3x$

$n = 3000 \rightarrow 45$ seconds

?

[1, 2, 3, 4, 5]

1 + 2 + 3 + 4 + 5

list $\quad n \Rightarrow$ length of the list

```
def stddev(data):
    result = 0.0                              1
    average = mean(data)                      n    sum(data) / len(data)   nt ops
    for elem in data:                                                 1    1
        result += (elem - average) ** 2       n (3)
    return math.sqrt(result / (len(data) - 1))     4
```

$1 + n + 3n + 4 \qquad \cancel{4}n + \cancel{5} \Rightarrow O(n)$

```
def stddev2(data):
    result = 0.0                              2
    for elem in data:
        result += (elem - mean(data)) ** 2    n     n (n + 3)
    return math.sqrt(result / (len(data) - 1))     4
```

$1 + n(n+3) + 4 \qquad n^2 + 3n + 5 \Rightarrow O(n^2)$

What is the big-O asymptotic bound of

$$f(n) = \cancel{3}n^2 + \cancel{4n + 2}$$

A. O(n)
B. O(n^2)
C. O(n^2 + n)
D. O($3n^2$ + 4n + 2)

Answer: B
With big-O we only retain the term with the highest growth rate and drop all constants.

Which of the following functions has the same big-O asymptotic bound as

$$f(n) = \cancel{2}n^3 + \cancel{5n + 2}$$

$O(n^3)$

A. $2n^2 + 5n + 2$

B. $n^3 + 2\cancel{n}^2$ $O(n^3)$

C. $n(2n^3 + 5n + 2)$

D. 2

Answer: B

With big-O we only retain the term with the highest growth rate and drop all constants.

```
numbers = list(range(100))
the_max = max(numbers)
```

What is the time complexity of the Python `max` function (example above):

A. O(1)
B. O(100)
C. O(n)
D. O(n log n)
E. O(n^2)

Answer: C

To compute the max, we need to examine all n items in the list. The time complexity is independent of any particular input, i.e., we describe it in terms of the size of the input as it grows arbitrarily large.

```
for i in range(n):
    for j in range(n):
        val = 0.0
        for k in range(n):
            val += x[i][k] * y[k][j]
        res[i][j] = val
```

$f(n) \sim \overset{i}{n}(\overset{j}{n}(1 + \overset{k}{n}(2) + 1))$

1

$O(n^3)$

The above code implements matrix multiply on two *n* x *n* matrices (stored as lists of lists). What is the time complexity of this algorithm?

A. O(1)
B. O(n)
C. O(n log n)
D. $O(n^2)$
E. $O(n^3)$

Answer: E
Since the 3 loops are nested, we will do n * n * n multiplications for $O(n^3)$

$$
\begin{array}{r}
437 \\
-256 \\
\hline
181 \\
-128 \\
\hline
53 \\
-32 \\
\hline
21 \\
-16 \\
\hline
5 \\
-4 \\
\hline
1
\end{array}
$$

$437 = 256 + 128 + 32 + 16 + 4 + 1$

$2^8 \quad 2^7 \quad 2^5 \quad 2^4 \quad 2^2 \quad 2^0$

110110101

$$
\begin{array}{r}
111 \\
10111 \\
+\ 00101 \\
\hline
11100
\end{array}
\qquad
\begin{array}{ll}
16+4+2+1 & \Rightarrow 23 \\
4+1 & \Rightarrow 5 \\
\hline
16+8+4 & \Rightarrow 28
\end{array}
$$

$$
\begin{array}{r}
0111 \\
\cancel{1}0001 \\
-\ 01010 \\
\hline
00111
\end{array}
\qquad
\begin{array}{l}
\Rightarrow 17 \\
\Rightarrow 10 \\
\Rightarrow 7
\end{array}
$$

What is the minimum number of bits (binary digits) to represent the decimal number 32?

A. 3
B. 4
C. 5
D. 6
E. 7

$2^5 \rightarrow 32$

100000

Answer: D
32 is 2^5, that is a 1 at index 5. Index 5 is the 6th bit. 5 bits can represent the 32 numbers from 0-31 inclusive, but not 32.

How many distinct numbers can be represented in 5 bits (binary digits) ?

A. 15
B. 16
C. 31
D. 32
E. 63

Answer: D
Since 2^5 equals 32, 5 bits can represent 32 distinct numbers.

What is the result of adding 01101 and 00100?

A. 01001
B. 01101
C. 10000
D. 10001
E. 10101

16 8 4 2 1
11
01101 13
+ 00100 4
10001 17 ✓

Answer: D
Equivalent to 13+4 = 17

11
01101
00100

10001

```
0.125 + 0.25 <= 0.375
```

Will the above expression evaluate to True (as expected)?

A. Yes
B. Possibly, I would have to try it out
C. No

Answer: A
Since those values are uniquely representable as floating points numbers, i.e. as 1.0*2^-3, 1.0*2^-2 and 1.5*2^-2, there is no error in the computation and so we will know the expression will evaluate to True.