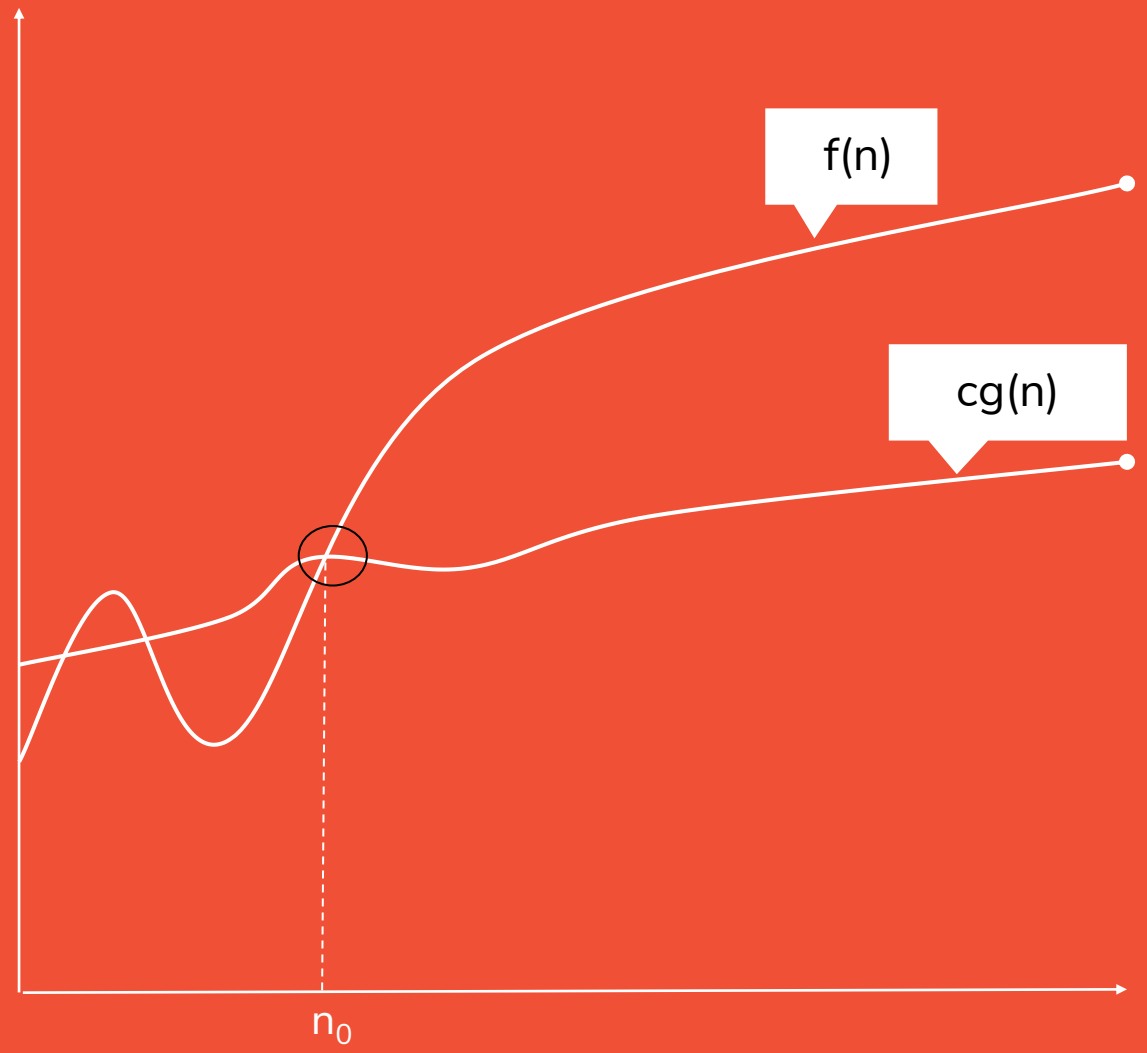
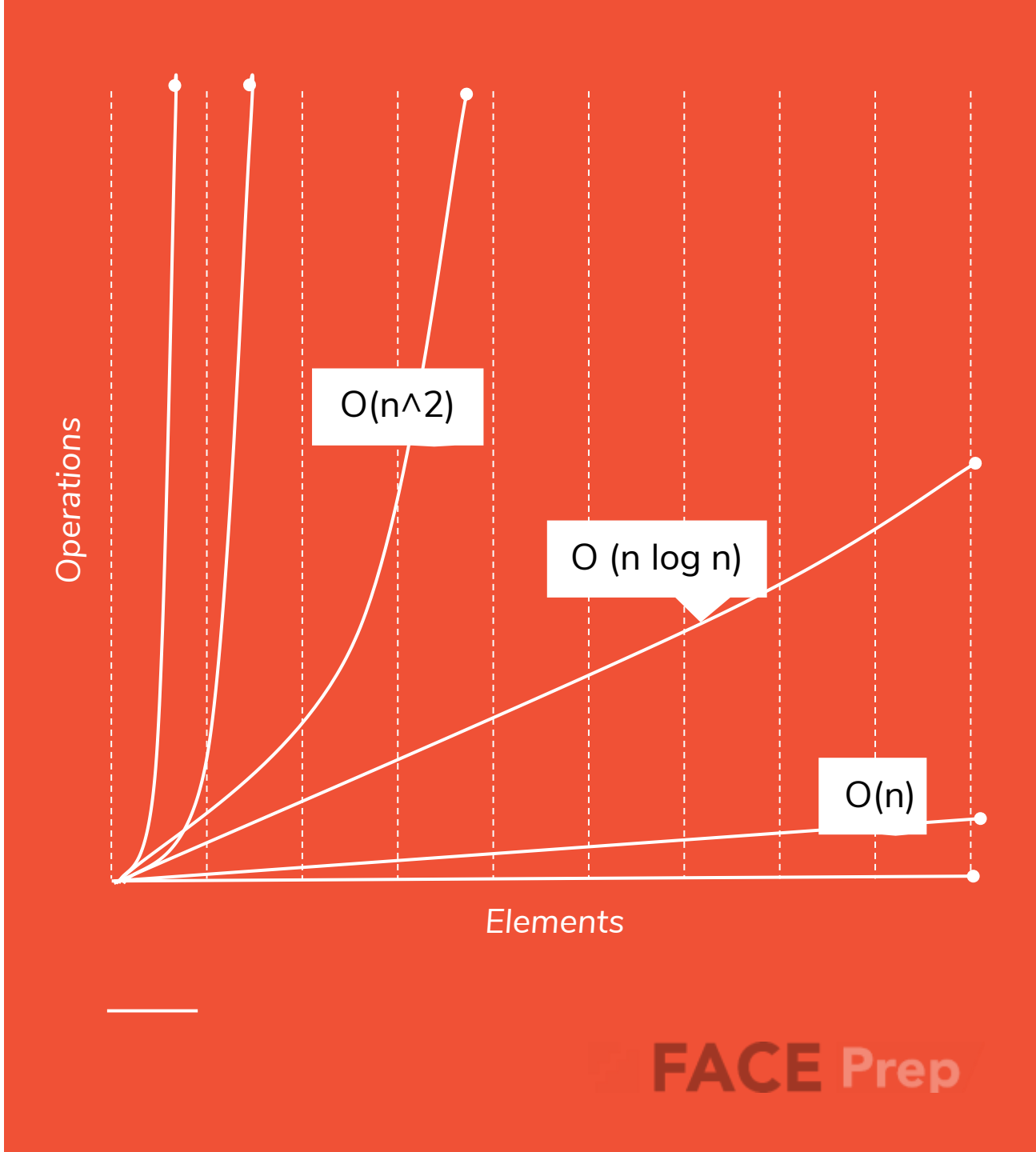

FACE
Prep

Time Complexity



How To

Calculate Time Complexity



FACE Prep

What is Time Complexity?

You can get the time complexity by “counting” the number of operations performed by your code.

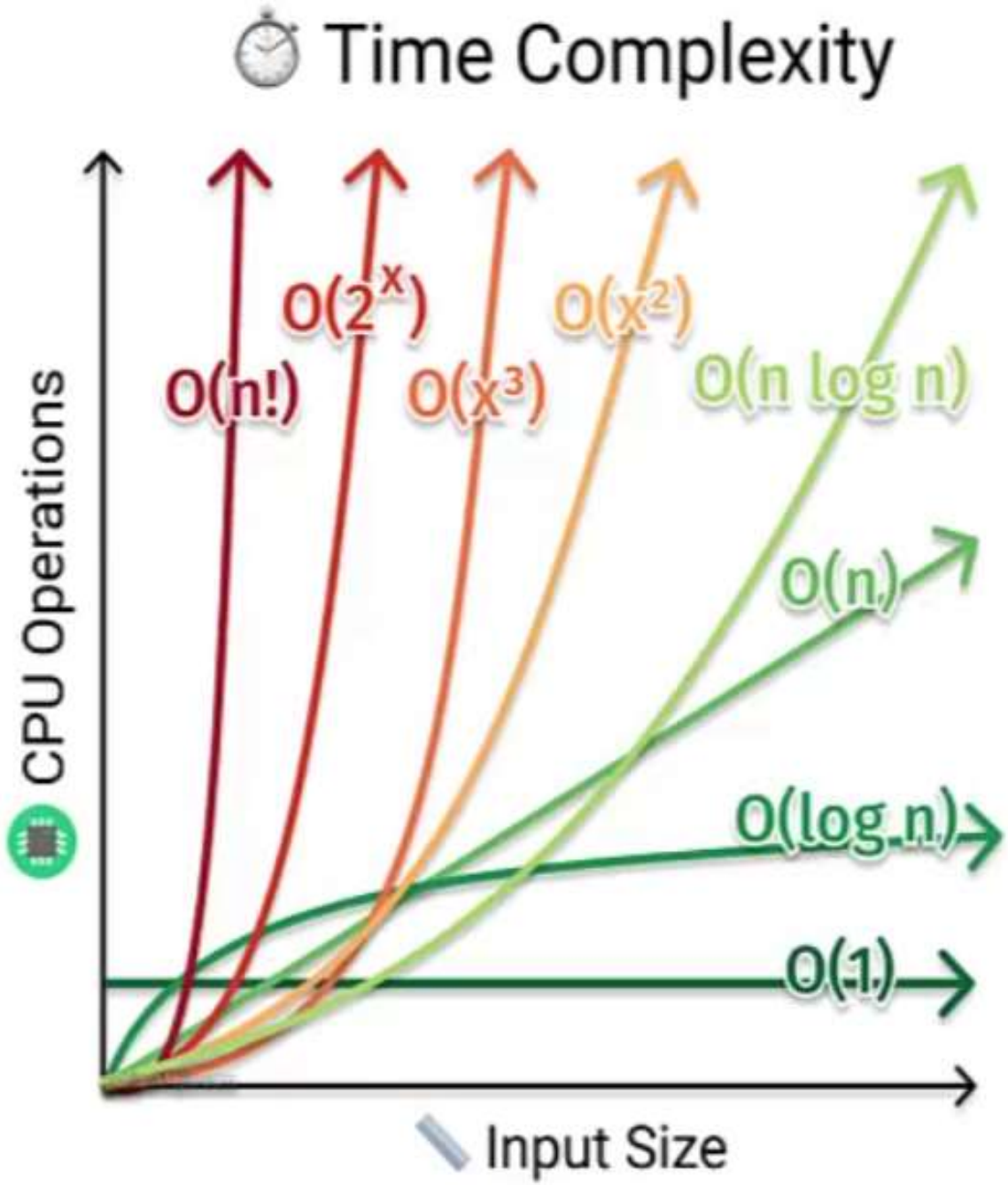
This time complexity is defined as a function of the input size n using Big-O notation. n indicates the input size, while O is the worst-case scenario growth rate function.

We use the Big-O notation to classify algorithms based on their running time or space (memory used) as the input grows.

The O function is the growth rate in function of the input size n.

How To

Calculate Time Complexity



Big O Cheatsheet

Big O Notation	Name	Examples
$O(1)$	Constant	Odd or Even number
$O(\log n)$	Logarithmic	Finding element on sorted array with binary search
$O(n)$	Linear	Find max element in unsorted array
$O(n \log n)$	Linearithmic	Sorting elements in array with merge sort
$O(n^2)$	Quadratic	Sorting array with bubble sort
$O(n^3)$	Cubic	3 variables equation solver
$O(2^n)$	Exponential	Find all subsets
$O(n!)$	Factorial	Find all permutations of a given set/string

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i < n; i++)
      {
            cout << "FACE PREP";
      }
      return 0;
}
```

A) O(n) **B)** O(log n) **C)** O (sqrt n) **D)** O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i <= sqrt(n); i++)
      {
            cout << "FACE PREP";
      }
      return 0;
}
```

A) O(n) **B)** O(log n) **C)** O (sqrt n) **D)** O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i*i <= n; i++)
      {
            cout << "FACE PREP";
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i <= n; i=i*2)
      {
            cout << "FACE PREP";
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      int n, i = 1, s = 1;
      while(s <= n)
      {
            i++;
            s = s + i;
            cout << "FACE PREP";
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      int n;
      while(n > 1)
      {
            n = n / 2;
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i < n; i++)
      {
            for(j = 1; j < n; j++)
            {
                  cout << "FACE PREP";
            }
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i < n; i++)
      {
            for(j = 1; j < n; j++)
            {
                  cout << "FACE PREP";
                  break;
            }
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i < n; i++)
      {
            for(j = 1; j < n; j=j+i)
            {
                  cout << "FACE PREP";
            }
      }
      return 0;
}
```

A) O(n)

B) O(log n)

C) O (n * log n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      int i, j;
      for(i = 1; i < n; i++)
      {
            for(j = 1; j < n; j=j*2)
            {
                  cout << "FACE PREP";
            }
      }
return 0;
}
```

A) O(n)

B) O(log n)

C) O (n * log n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      int i, j;
      for(i = 1; i < n; i++)
      {
            for(j = n; j > 1; j--)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(log n)

C) O (n * log n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
int fun()
{
      int i, j, a = 0, b = 0;

      for(i = 1; i < n; i++)
            a = a + i;
      for(j = 1; j < m; j++)
            b = b + j;

      return 0;
}
```

A) O(n)

B) O(m)

C) O (n + m)

D) O(n * m)

Predict the time complexity for the given below code snippet

```
for(i = 1; i <= n; i++)
{
      for(j = 1; j <= i; j++)
      {
            for(k = 1; k <= 20; k++)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(log n)

C) O (sqrt n)

D) O(n * n)

Predict the time complexity for the given below code snippet

```
for(i = 1; i <= n; i++)
{
      for(j = 1; j <= i*i; j++)
      {
            for(k = 1; k <= (n/2); k++)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(n ^ 2)

C) O (n ^ 3)

D) O(n ^ 4)

Predict the time complexity for the given below code snippet

```
for(i = n/2; i >=1; i--)
{
      for(j = 1; j <= n; j=j*2)
      {
            for(k = 1; k <= n; k=k*2)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(n * n)

C) O (sqrt n)

D) O(n*(log n)^2)

Predict the time complexity for the given below code snippet

```
int fun(int n)
{
      if(n > 1)
      return fun(n - 1);
}
```

A) O(n) **B)** O(log n) **C)** O (n * log n) **D)** O(n * n)

Predict the time complexity for the given below code snippet

```
for(i = n/2; i  <= n; i++)
{
      for(j = 1; j +n/2<= n; j++)
      {
            for(k = 1; k <= n; k=k*2)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(log n)

C) O (n * n)

D) O(n^2 (log n))

Predict the time complexity for the given below code snippet

```
int fun()
{
      int n, ans = 0;
      cin >> n;
      while(n > 0)
      {
            ans += n % 10;
            n /= 10;
      }
      return 0;
}
```

A) $O(\log_2 n)$ **B)** $O(\log_3 n)$ **C)** $O(\log_{10} n)$ **D)** $O(n)$

Predict the time complexity for the given below code snippet

```
for(i = n/2; i >=1; i--)
{
      for(j = 1; j <= n/2; j++)
      {
            for(k = 1; k <= n; k=k*2)
            {
                  cout << "FACE PREP";
            }
      }
}
```

A) O(n)

B) O(log n)

C) O (n * n)

D) O(n^2 (log n))

Predict the time complexity for the given below code snippet

```
int fun()
{
      for(i = 1; i < n; i++)
      {
            for(j = 1; j < sqrt(n); j++)
            {
                  cout << "FACE PREP";
            }
      }
      return 0;
}
```

A) O(n)

B) O(n * sqrt n)

C) O (n * log n)

D) O(n * n)

THANK YOU

FACE Prep