

Testing an Algorithm for Efficiency

Textbook

Testing an Algorithm



After creating an algorithm, it's important to make sure it will do what it's designed to do with a variety of possible inputs. One way programmers test their algorithms is through an [empirical analysis](#).

Empirical Analysis

An [empirical analysis](#) is experimentation with a variety of inputs to observe what the program does.

Let's take a look at this example. We have a list of different temperatures (in Fahrenheit) and we are seeing if they are above or below the freezing point, which is 32 degrees Fahrenheit.

```
1 temperature = [80, 50, 100, 20, 15, 95]
2
3 for x in temperature:
4     if x > 32:
5         print(str(x) + " is above freezing")
6     else:
```

```
7 print(str(x) + " is below freezing")
```

Try it!

Let's do some [empirical testing](#) on our algorithm. What would happen if we added some unusual numbers to the list? Would it still work?

```
1 temperature = [80, 50, 100, 20, 15, 95, -10, 0, 32, -10000, 5829274]
2
3 for x in temperature:
4     if x > 32:
5         print(str(x) + " is above freezing")
6     else:
7         print(str(x) + " is below freezing")
```

Try it!

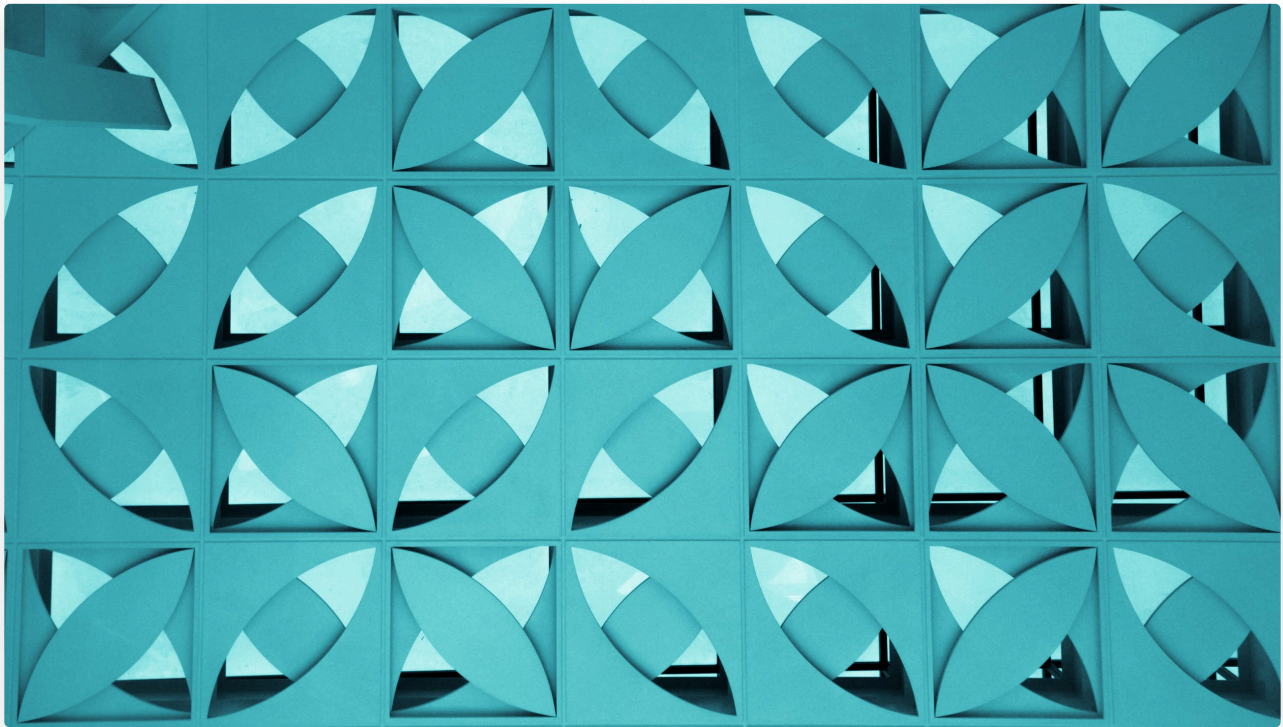
After adding the numbers to the list, we find that our algorithm still works. The number 32 is an interesting addition. What do we want the program to do AT the freezing point? Upon further consideration, we might modify our algorithm to look like this.

```
1 temperature = [80, 50, 100, 20, 15, 95, -10, 0, 32, -10000, 5829274]
2
3
4 for x in temperature:
5     if x > 32:
6         print(str(x) + " is above freezing")
7     elif x == 32:
8         print(str(x) + " is the freezing point")
9     else:
10        print(str(x) + " is below freezing")
```

Try it!

Testing different instances of the problem help to determine if the algorithm works well or not. A problem is a general description of a task that can (or cannot) be solved algorithmically. An [instance](#) of a problem also includes specific input. For example, sorting is a problem; sorting the list (2,3,1,7) is an instance of the problem.

Algorithmic Efficiency



Just like most problems in life, there are many ways to reach a solution to the problem. Different algorithms that produce the same solution often have different efficiencies. Some might take a long time or require a lot of resources to get the right answer. Others might be quicker or use less resources.

In programming, [efficiency](#) is an estimation of the amount of computational resources used by an algorithm. Algorithms that take more steps or more time are often less efficient than those with fewer steps. An algorithm's efficiency is determined through formal or mathematical reasoning.

An algorithm's efficiency can be informally measured by determining the number of times a statement or group of statements executes.

Linear vs Binary Search

Let's look at how [linear search](#) compares with [binary search](#). Often, binary search cuts down the search steps significantly because the program stops looking after it finds what it's looking for. Therefore, binary search is often more efficient than linear searches.

Categorizing Run Times

Computer algorithms are useful because they can work through large data sets very quickly. As you may have experienced, waiting for a program to load can be frustrating. So we are continually looking for ways to speed up the processing speed of our programs. In order to do this, we need to calculate how long an algorithm takes to execute. This is often called the "[run time](#)."

We categorize different algorithms depending on how long they take to run as the size of data increases. Does it slow down as the data size increases? Does it speed up? Does it stay the same?

We can categorize the run time by counting how many steps it takes to complete the algorithm.

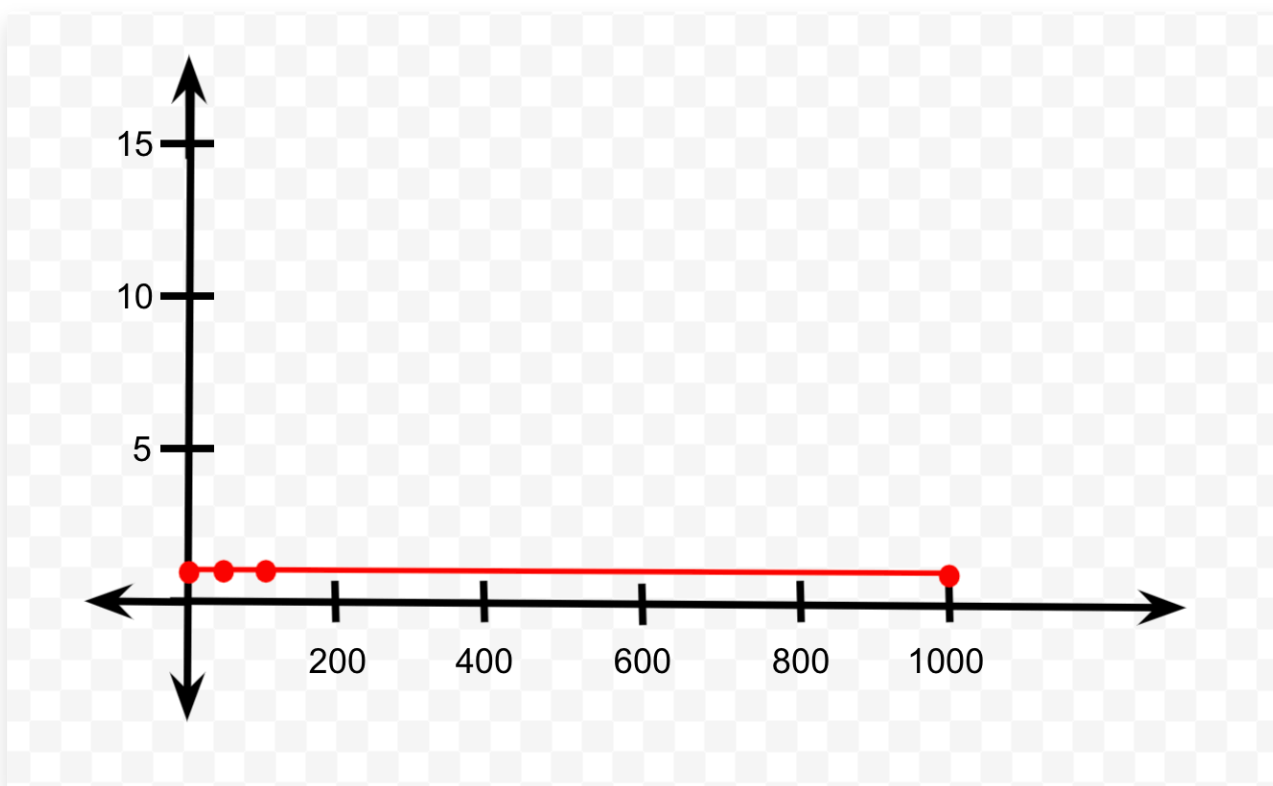
Constant Time

Some algorithms take the same amount of time to run no matter how much data it is processing. Let's visualize this effect with a table.

Number of Data Points	Steps
1	1

10	1
100	1
1000	1

Let's see what this would look like as a graph.



As you might be thinking, an algorithm that runs at constant time would be highly useful. No matter how much data it's processing, it still runs at the same speed. Most algorithms don't work this way and have different run times. Let's explore some other ways that algorithms behave.

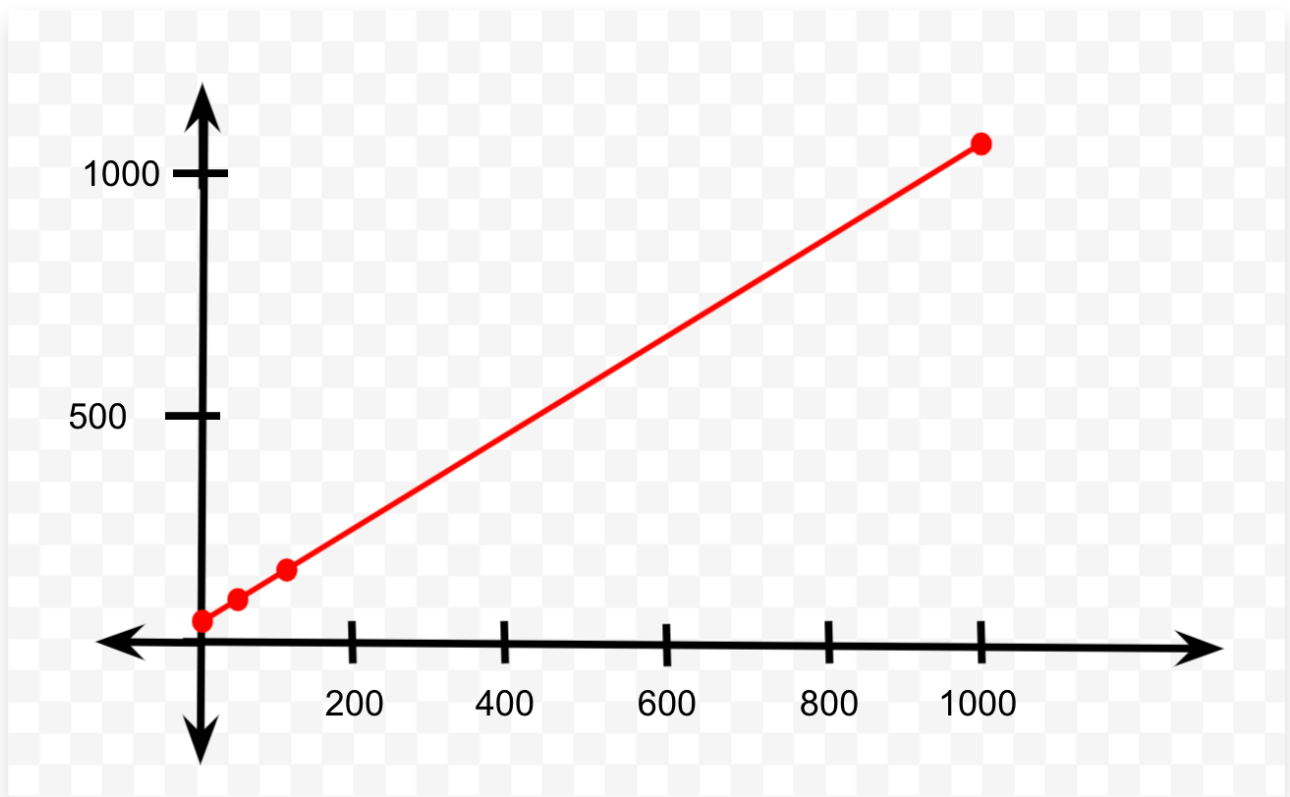
Linear Run Times

For an algorithm that runs with a linear run time, the number of steps increases proportionally to the number of data points.

Let's look at a chart of a linear run time.

Number of Data Points	Steps
1	1
10	10
100	100
1000	1000

Let's see a linear run time as a graph.



As you can see, linear algorithms increase in time to run just as quickly as the number of steps increase.

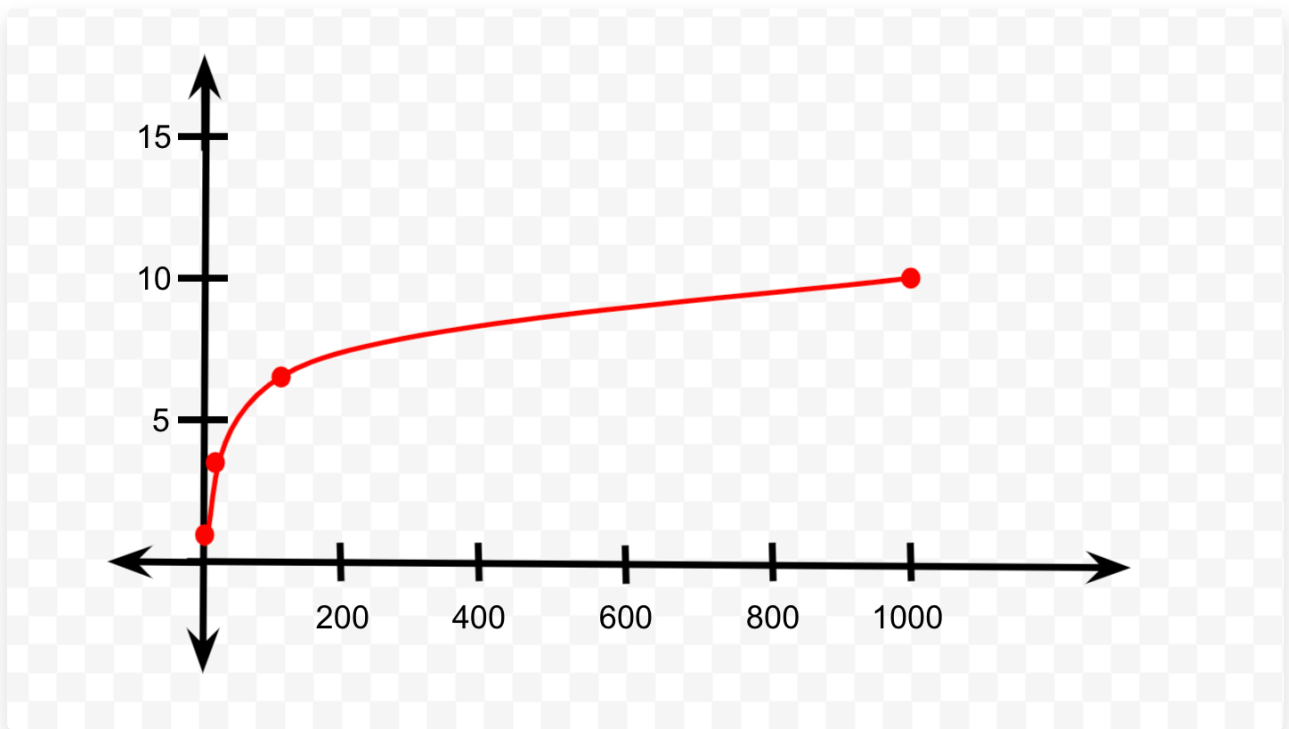
Logarithmic Run Times

When an algorithm runs in logarithmic time, it increases proportionally to the logarithm of the input size. An example of a logarithmic function is binary searching.

Let's look at a chart of a logarithmic run time.

Number of Data Points	Steps
1	1
10	4
100	7
1000	10

Let's see it as a graph.



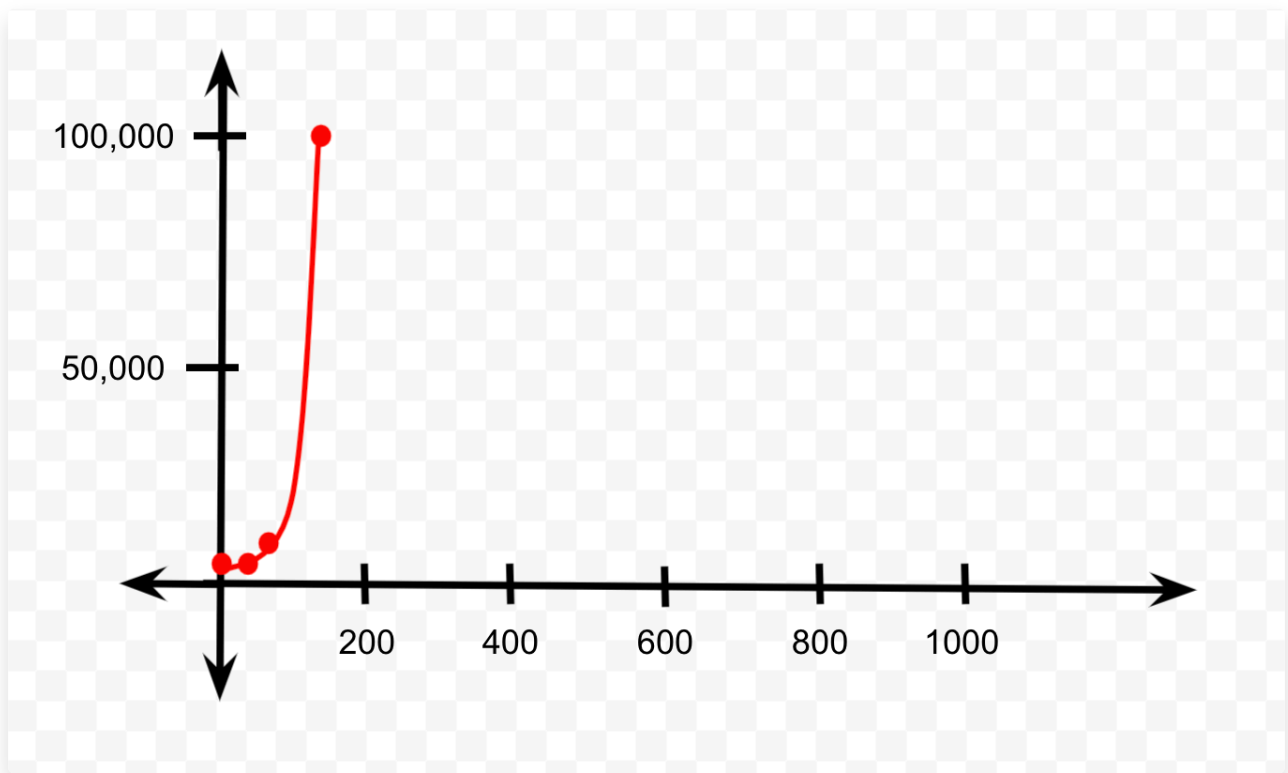
Exponential Run Time

Algorithms that run on an exponential run time increase their time significantly with the addition of more data points.

Let's see a chart with an exponential run time.

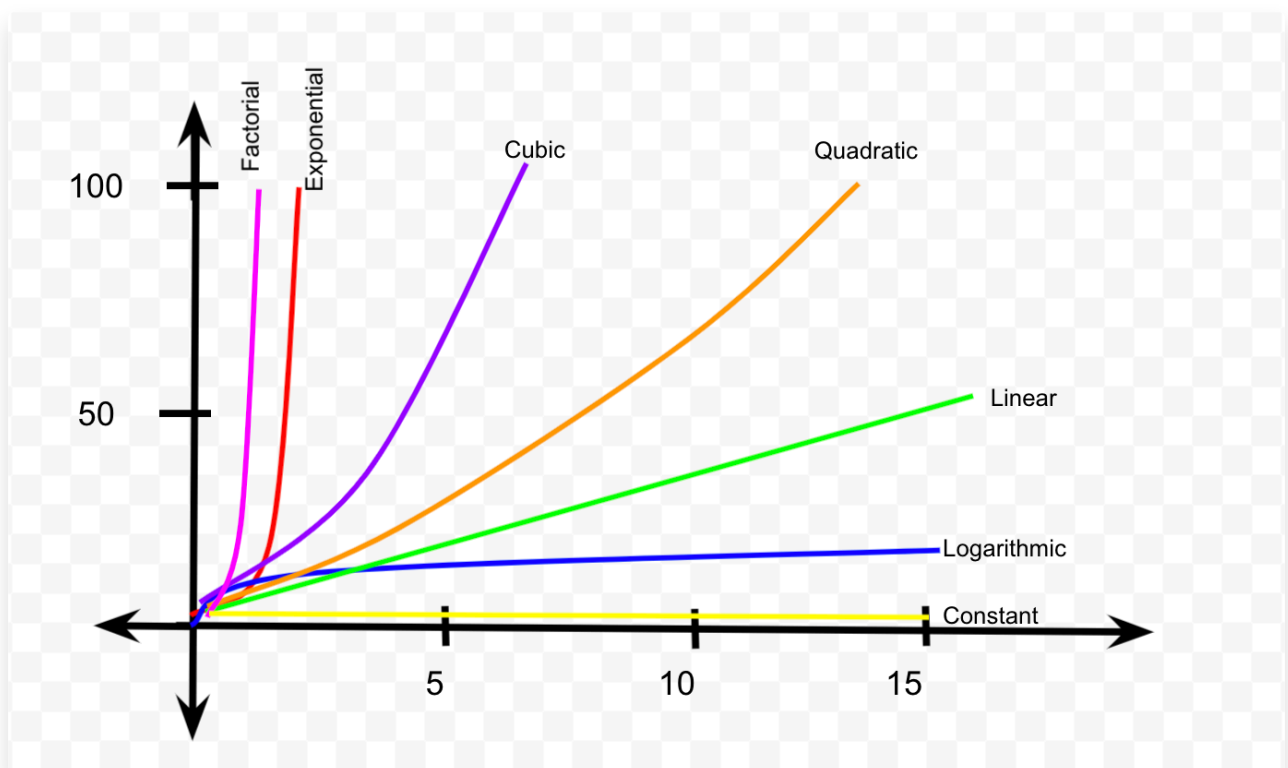
Number of Data Points	Steps
1	10
10	100
100	1000
1000	10000

Let's see an exponential run time as a graph.



Other Run Times

We won't walk through a table for each of the following run times, but we will put them into the chart for comparison purposes.

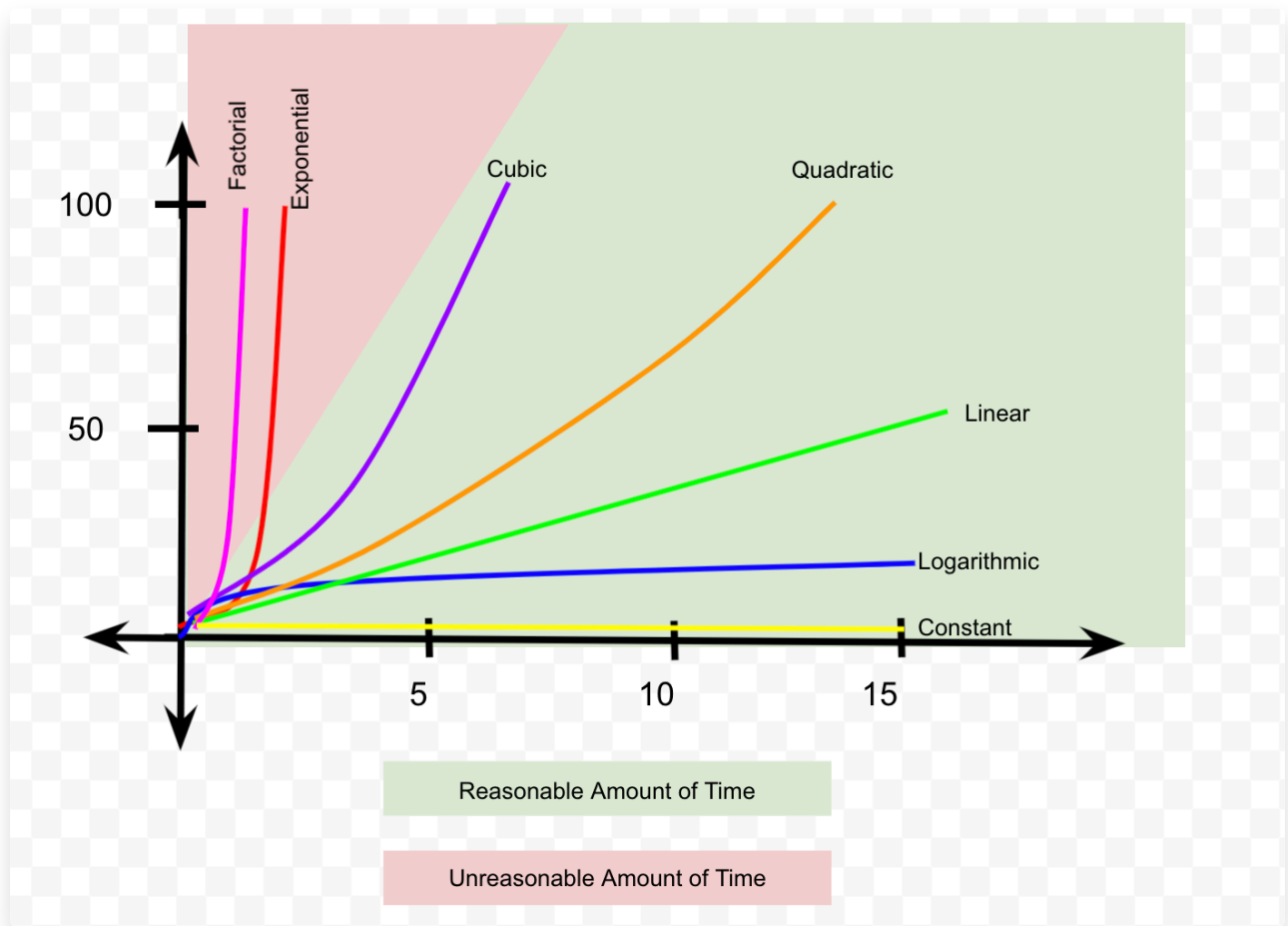


As you can see, different algorithmic efficiencies can dramatically change the amount of time it takes it to run with large sets of data.

Here are the main takeaways from this lesson.

Algorithms with a polynomial efficiency or slower (constant, linear, square, cube, etc.) are said to run in a reasonable amount of time.

Algorithms with exponential or factorial efficiencies are examples of algorithms that run in an unreasonable amount of time.



Summary

Each problem in programming has many different solutions. Since there are so many ways to get the right answer, we need ways to help us choose the best correct answer.

We can run an [empirical analysis](#) on algorithms to determine if the algorithm works with many different kinds of input. We can also assess the algorithmic efficiency of the solution. [Algorithmic efficiency](#) is an estimation of the amount of computational resources used by an algorithm.

Algorithms with a polynomial efficiency or slower (constant, linear, square, cube, etc.) are said to run in a reasonable amount of time.

Algorithms with exponential or factorial efficiencies are examples of algorithms that run in an unreasonable amount of time.

Critical Thinking Questions

1. Imagine you've developed a program that helps a local library find books. If you only test it with books that are currently on the shelf, what types of "unusual numbers" (or unusual situations) might you be missing, and why would testing those specific scenarios be critical before the library uses your program?
2. Consider two different approaches to solving the same problem, both of which give the correct answer. One approach involves many more steps and complex calculations as the amount of data grows. Why might a programmer choose the "less efficient" approach in some situations, even knowing it takes more resources?
3. A new social media platform needs an algorithm to suggest new friends to users. If this algorithm has an "exponential run time," what practical problems would the platform likely face as its number of users rapidly increases?

Questions (8)

1. Which of the following tests a variety of inputs to see what the result might be.

MULTIPLE CHOICE

Choose the correct answer:

- A. Empirical Analysis
- B. Algorithmic Analysis
- C. Efficiency Analysis
- D. Testing Analysis

2. True or False: Each problem in programming has only one algorithmic solution.

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

3. Which of the following run times is the most efficient as the number of data points increases?

MULTIPLE CHOICE

Choose the correct answer:

- A. Linear Run Time
- B. Constant Run Time
- C. Logarithmic Run Time
- D. Exponential Run Time

4. Which of the following run times is the least efficient as the data points increase?

MULTIPLE CHOICE

Choose the correct answer:

- A. constant run time
- B. linear run time
- C. logarithmic run time
- D. exponential run time

5. Which of the following are considered to have reasonable run times? Select all that apply.

SELECT MULTIPLE

Select all that apply:

- A. constant
- B. linear
- C. logarithmic
- D. cubic
- E. exponential
- F. factorial
- G. quadratic

6. Which of the following are considered to have unreasonable run times? Select all that apply.

SELECT MULTIPLE

Select all that apply:

- A. constant
- B. linear
- C. logarithmic
- D. exponential
- E. cubic
- F. factorial
- G. quadratic

7. How can algorithmic efficiency be informally measured?

MULTIPLE CHOICE

Choose the correct answer:

- A. By determining the number of times a statement executes.
- B. By conducting an empirical analysis
- C. By using formal mathematical reasoning
- D. By counting the lines of code

8. Which type of algorithmic efficiency remains the same regardless of data size?

Choose the correct answer:

- A. Constant time
- B. Linear time
- C. Logarithmic time
- D. Exponential time

Answer Keys & Solutions

Questions

1. Which of the following tests a variety of inputs to see what the result might be.

MULTIPLE CHOICE

Correct Answer:

- | | |
|-------------------------|-------------|
| A. Empirical Analysis | ✓ Correct |
| B. Algorithmic Analysis | ✗ Incorrect |
| C. Efficiency Analysis | ✗ Incorrect |
| D. Testing Anlysis | ✗ Incorrect |

Explanation:

This kind of analysis can be done with a trial and error approach.

2. True or False: Each problem in programming has only one algorithmic solution.

MULTIPLE CHOICE

Correct Answer:

- | | |
|----------|-------------|
| A. True | ✗ Incorrect |
| B. False | ✓ Correct |

Explanation:

Often there are many algorithms that can be correct.

3. Which of the following run times is the most efficient as the number of data points increases?

MULTIPLE CHOICE

Correct Answer:

- | | |
|-------------------------|-------------|
| A. Linear Run Time | ✗ Incorrect |
| B. Constant Run Time | ✓ Correct |
| C. Logarithmic Run Time | ✗ Incorrect |
| D. Exponential Run Time | ✗ Incorrect |

Explanation:

This type of run time is highly desirable.

4. Which of the following run times is the least efficient as the data points increase?

MULTIPLE CHOICE

Correct Answer:

- | | |
|-------------------------|-------------|
| A. constant run time | ✗ Incorrect |
| B. linear run time | ✗ Incorrect |
| C. logarithmic run time | ✗ Incorrect |
| D. exponential run time | ✓ Correct |

Explanation:

If the run time goes up exponentially, that makes it move much slower with a lot of data

5. Which of the following are considered to have reasonable run times? Select all that apply.

SELECT MULTIPLE

Correct Answers:

- | | |
|----------------|-------------|
| A. constant | ✓ Correct |
| B. linear | ✓ Correct |
| C. logarithmic | ✓ Correct |
| D. cubic | ✓ Correct |
| E. exponential | ✗ Incorrect |
| F. factorial | ✗ Incorrect |

G. quadratic

✓ Correct

Explanation:

Only two are considered unacceptable.

**6. Which of the following are considered to have unreasonable run times?
Select all that apply.**

SELECT MULTIPLE

Correct Answers:

A. constant

✗ Incorrect

B. linear

✗ Incorrect

C. logarithmic

✗ Incorrect

D. exponential

✓ Correct

E. cubic

✗ Incorrect

F. factorial

✓ Correct

G. quadratic

✗ Incorrect

Explanation:

Two correct answers. Which two have the steepest curve on the graphs?

7. How can algorithmic efficiency be informally measured?

MULTIPLE CHOICE

Correct Answer:

A. By determining the number of times a statement executes.

✓ Correct

B. By conducting an empirical analysis

✗ Incorrect

C. By using formal mathematical reasoning

✗ Incorrect

D. By counting the lines of code

✗ Incorrect

Explanation:

Algorithms that take more steps or more time are often less efficient than those with fewer steps.

8. Which type of algorithmic efficiency remains the same regardless of data size?

MULTIPLE CHOICE

Correct Answer:

- | | |
|---------------------|-------------|
| A. Constant time | ✓ Correct |
| B. Linear time | ✗ Incorrect |
| C. Logarithmic time | ✗ Incorrect |
| D. Exponential time | ✗ Incorrect |

Explanation:

The time stays constant