TASK

Interview Preparation: Big O Notation

Visit our website

Copyright © 2021 HyperionDev. All rights reserved.

Introduction

WELCOME TO THE INTERVIEW PREPARATION: BIG O NOTATION TASK!

A key skill that a potential employer will most likely be looking for in a web developer is not only the ability to write code but also the ability to analyse code. A developer should be able to evaluate whether or not code is written efficiently and determine how to improve the efficiency of the code. Big O Notation is used to evaluate and describe the efficiency of code. In this task, we will be discussing Big O Notation.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

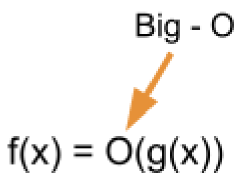
The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

Copyright © 2021 HyperionDev. All rights reserved.

WHAT IS BIG O NOTATION?

Big O notation, also called Landau's Symbol, is used in maths, complexity theory, and computer science. The O refers to "order of magnitude". Order of magnitude refers to the rate of growth of a function. An example of Big O notation in use is shown below:



Don't be intimidated by the maths. You don't have to understand the maths completely to be able to benefit from using Big O Notation. In this task, we will see why we use Big O notation and some examples of how it is applied. However, before continuing with this task, you need a basic understanding of what is meant by the terms function and algorithm in the context of Big O notation. The definitions from the **Oxford dictionary** are shown below:

Word	Definition
Function	"A basic task of a computer, especially one that corresponds to a single instruction from the user." **OR in mathematics** "A relation or expression involving one or more variables. 'the function (bx + c)'"
Algorithm	"A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer."

In this task, when we speak about 'algorithm analysis' we will be referring to the analysis of a piece of code that is used to solve a particular problem. Therefore, in this task, an algorithm can be a computer function such as the JavaScript functions that we have used and created in this Bootcamp. When we refer to functions in this task though, we will generally be referring to mathematical functions as described in the second definition of the word function in the table above.

HyperionDev Copyright © 2021 HyperionDev. All rights reserved.

WHY DO WE USE BIG O NOTATION?

Big O notation is used for algorithm analysis. As you are well aware, to be a good software developer, you must be able to do more than write code that simply works. You want to be able to write code that works as effectively and efficiently as possible. Thus, you must be able to analyse a piece of code concerning its performance in terms of speed (i.e. how fast the algorithm is) and resources (i.e. how much memory is required to execute the code). We use Big O notation to do this. The best way to understand Big O notation is to see it in action. In this task, we will use Big O notation as we consider some important functions used in algorithm analysis.

SEVEN FUNCTIONS OF ALGORITHM ANALYSIS

There are seven main functions (bear in mind that in this context, we are referring to mathematical functions, not programming functions) that are used to measure the complexity and performance of an algorithm. Each of the functions is shown in the image below. Each function describes how an algorithm's complexity changes as the size of the input to the algorithm changes. As you can see from the image below, some functions (e.g. O log n or O(1)) describe algorithms where the efficiency of the algorithm stays the same regardless of how big the input to the algorithm becomes. Other functions (e.g. O(n!) and O(2n)), on the other hand, describe a situation where, as the size of the input to the algorithm increases, the complexity of the algorithm greatly increases and the efficiency of the algorithm greatly decreases.

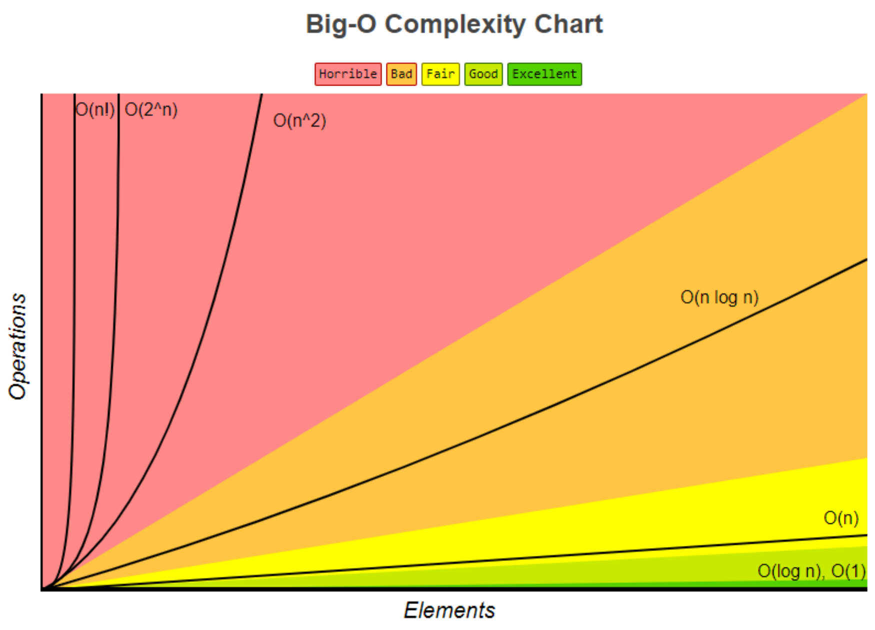


Image source: ***http://bigocheatsheet.com/***

Copyright © 2021 HyperionDev. All rights reserved.

Constant function

The Big O notation for a constant function is O(1). This is used to describe an algorithm that will always execute in the same time (or space) regardless of the size of the input to that algorithm. In the example below, as you can see, `AddFruit()` will work as efficiently regardless of the size of the array that is passed to it. It will always take the same amount of time to add an item to an array.

```
let fruits = ['Apple', 'Banana'];

function AddFruit(fruits){
let newLength = fruits.push('Orange');
return newLength;
}

console.log("The fruit array now contains " + AddFruit(fruits) + " items");
```

Compare this with the linear function described next.

Linear function

The Big O notation for a linear function is O(N), where N represents the size of the input set. A linear function describes an algorithm whose performance will grow linearly as the size of the input to that algorithm grows. In other words, an O(N) algorithm with an input of a dataset with 100 items (N = 100) will be 100 slower than if the input to that algorithm was a dataset with 1 item (N = 1). See the example below. The number of items in a loop directly affects how long it will take to execute that loop.

Notice that, in the example below, the item being searched for is the first item in the array. This doesn't affect the fact that this is O(N) because Big O notation always assumes the worst-case scenario where the algorithm will execute the maximum number of times.

```
let fruits = ['Apple', 'Banana'];

function SearchArray(){
  let itemFound = false;
  fruits.forEach(function(item, index, array) {
    if (item === "Apple"){
      itemFound = true;
    }
```

Copyright © 2021 HyperionDev. All rights reserved.

```
  });
  return itemFound
}
```

Exponential functions

The Big O notation for an exponential function is O(2^n). It describes a function which will grow exponentially as the size of the input set grows. Each time the size of the input increases, the growth of the function doubles. As you can see in the graph above, this type of algorithm's efficiency decreases significantly as the input set increases. The code below shows an example of this sort of algorithm. It uses a recursive function (in other words a function that calls itself to solve a problem) to calculate all the numbers in the Fibonacci sequence.

```
function fibonacci(num) {
  if (num <= 1) return 1;
  return fibonacci(num - 1) + fibonacci(num - 2);
}
```

A Fibonacci sequence is a group of numbers where you determine the next number in the sequence by adding the previous two numbers in the sequence together.

Fibonacci numbers: 1, 1, 2, 3, 5, 8, 13, 21, 34 ...

The Fibonacci sequence can be described with the following formula:

Fn = Fn-1 + Fn-2   where the seed/original values are F1 = 1 and F2 =1.

BEFORE YOU START THE COMPULSORY TASK

As you research Big O notation don't be overwhelmed by the maths. What is important to you as a web developer is not being able to completely understand and apply all the maths associated with complexity theory, but rather to understand how Big O notation is used to describe and calculate the performance of your code.

In this compulsory task, you are again going to be asked to create a blog post. Remember that this task aims to prepare you to find a job. Since these interview preparation tasks are meant to help you explain your knowledge of a topic to a

Copyright © 2021 HyperionDev. All rights reserved.

prospective employer, a blog is a good format to help you get used to explaining technical concepts to others. Keep your blog post conversational, as if you were explaining the topic to someone during a job interview.

Compulsory Task 1

Follow these steps:

Create the following code:

- In this task, you have been given an example of a recursive function that can be used to determine the numbers in the Fibonacci sequence. Use this code to create a JavaScript program (called **recursiveFibonacci.js**) that displays the first 5 numbers in the Fibonacci sequence.
- Now create another JavaScript program (called **myFibonacci.js**) that writes the first 5 numbers in the Fibonacci sequence but without using a recursive function.

Create a blog post called "Big O notation basics for web developers" that includes the following:

- Explain what Big O notation is and why web developers should know about it. **Recommended reading** and **recommended reading 2**.
- Explain what a quadratic function (O(n^2)) is. Include an example of JavaScript code that is O(n^2) in your explanation. Explain, using the code you have given, how the efficiency of the code is affected as the input set of the algorithm increases. **Recommended reading**.
- Compare a linear search with a binary search algorithm. Give code examples of each searching algorithm in JavaScript. Which searching algorithm is more efficient? In very basic terms, how would you use Big O notation to describe each type of searching algorithm? **Recommended reading 1** and **recommended reading 2**.
- Describe what the Fibonacci sequence is. Use the code you created in the first part of this compulsory task to show different algorithms you can use to find the numbers in the Fibonacci sequence. Use Big O notation to compare the efficiency of each algorithm. Which algorithm is the more efficient solution to the problem — the recursive function or the one that you wrote?

HyperionDev Copyright © 2021 HyperionDev. All rights reserved.

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.

Copyright © 2021 HyperionDev. All rights reserved.