Algorithms | Introduction

The best way to give an insight into something new is by answering some obvious questions and we are going to do the same. So, let's get into this and know what really an algorithm is?

What is an algorithm?

An algorithm is a technique to solve a well-defined problem. It can also be defined as a set of precise **steps** to solve a problem.

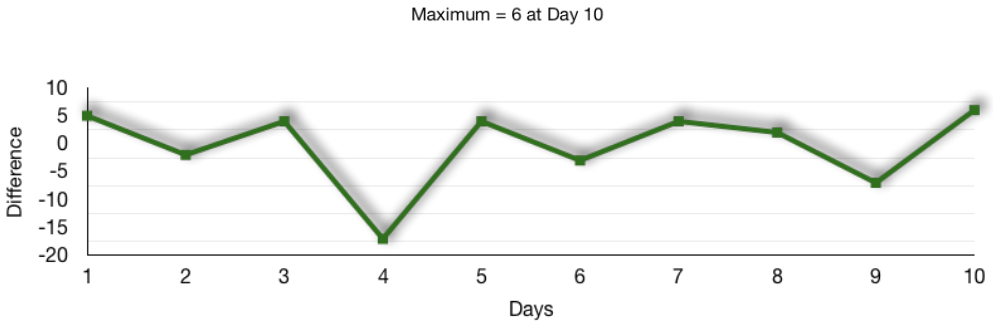
Suppose, we have to make an analysis of the stock market and we have data for one month. Our analysis needs us to find the day on which a maximum profit could be made. Let's assume that the investment will be done at the start of the day and it will be sold at the end. Of course, this isn't entirely a practical problem, at least with the assumptions we have made but it will make you understand what basically an algorithm is.

Day	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Price at Start of Day	45	50	48	52	35	39	36	40	42	35
Price at End of Day	50	48	52	35	39	36	40	42	35	41

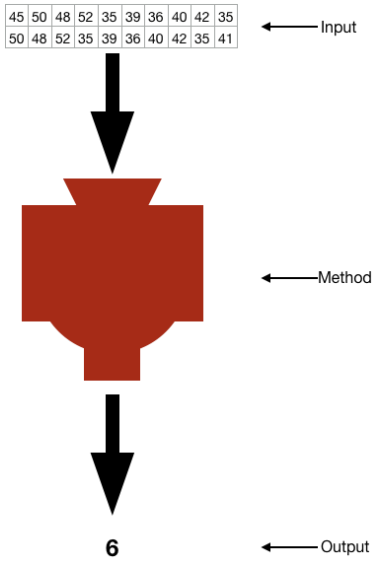
So, we have a problem with the data for one month and this is the **input** to our problem. Our problem also has some constraints about when a stock can be purchased and when it will be sold. And the expected **output** is the maximum profit.

To solve this problem, we would first calculate the difference between the prices of the stocks at the end of the day and at the start of the day. Then we will calculate the maximum among these differences and that would be our output.

Day	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Price at Start of Day	45	50	48	52	35	39	36	40	42	35
Price at End of Day	50	48	52	35	39	36	40	42	35	41
Difference	5	-2	4	-17	4	-3	4	2	-7	6



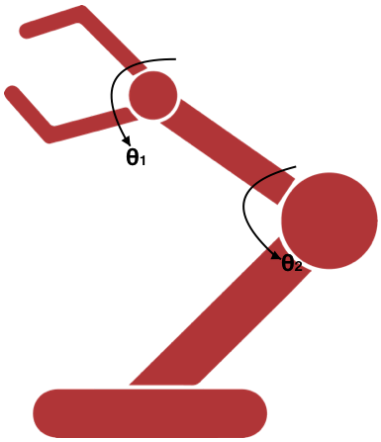
So, we have basically developed a **method** to solve a **specific problem** and our method takes an **input (data of one month)** and gives us an **output (the day of maximum profit)** and this is an ***algorithm***.



So, you are now clear with the idea of what is an algorithm. So, let's answer one more question.

Is algorithm related only to the Computer Science and do I really need to know how to code to learn it?

No, algorithms are not only related to Computer Science but in our world, it is the computer which handles and processes a very large amount of data. Even talking for non-Computer Science domains, algorithms are just well-defined steps to solve a problem and developing an algorithm is the work of a brain, so no coding is required here. We develop algorithms to calculate the position of a robotic arm or to calculate the trajectory of fluid and these are not purely Computer Science domains. And these algorithms are pretty much successful in getting the result if the size of the problem is small.



So, why algorithms are studied as a part of Computer Science?

Most of our real-world problem doesn't operate on tens or hundreds of data but on a much larger number like thousands and millions. So, implementing those steps on a dataset which is much larger is not a task of any human and we need to develop code to implement those steps in a computer. Even a computer is going to take a significant amount of time (like days, weeks or even more) if the algorithm we have developed is not good and then we need to optimize our algorithm. So, designing an efficient algorithm is very important for many practical problems.

Do I really need to take a course on algorithm?

Suppose, you are having lunch at your favorite restaurant and you liked the food very much and you want to cook it yourself but you have never cooked anything before. Even though you have never cooked anything but still you know how food is cooked, but is this knowledge is enough to cook the food you ate

(/add_quest

at the restaurant? Possibly, no. But what if you have a strong piece of knowledge in cooking and know how to make a food with a specific taste, texture, etc., can you create the recipe that matches to the food you had tasted? Probably, yes.

The point is if you have a prior knowledge of solving problems (cooking food in this example), then you can tackle a more difficult problem on your own (in this case, cooking the food served at the restaurant). And any course on Algorithms makes you face some problems first so that you are ready to solve further much more difficult problem on your own or to make an algorithm to solve a problem which is entirely new.

We can develop algorithms but is it necessary to optimize it with today's computer speed?

Now, this is a real question, at least if you are a beginner, you think it is. Also, time is not the only thing we are concerned about, we also optimize our algorithm to take less space (memory), programmer's effort, etc. in many cases. The one line answer for these questions would be - we are not provided with a computer with unlimited speed and space, therefore, we need to optimize our approach to solve a problem using a computer.

Even with the high-end computers available in the market, a badly written algorithm can take a significant amount of time like days or weeks or even more. However, an optimized algorithm can finish the same job in minutes or seconds with a much slower computer.

I don't believe you, can you give me an example?

Just try running this code to print the first 10,000 prime numbers.

C **Python** **Java**

```c
#include <stdio.h>

int is_prime(int x) {
  int i;
  int prime = 1;
  for(i=2; i<x; i++) {
    if ((x%i) == 0)
      prime = 0;
  }
  return prime;
}

int main() {
  int count = 0;
  int number = 2;
  while (count<10000) {
    if (is_prime(number)) {
      printf("%d\n",number);
      count++;
    }
    number++;
  }
  return 0;
}
```

So, this course only teaches me some algorithms and expects me to memorize them?

Of course, this course has many examples to explain the concepts behind the different algorithms but this doesn't only end there. Through the entire course, we have focused on the concepts, how a particular algorithm works, and the thought process of coming with the algorithm. At the end of this course, you

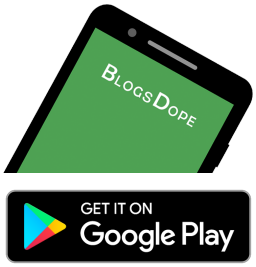
would have enough understanding so that you will be able to come up with a new method to solve a new problem.

Now, you know about the algorithms and are ready to learn how to know which algorithm is good or how to measure optimization of an algorithm. So, let's move to the next chapter.

> “Legacy is a stupid thing! I don’t want a legacy.”
>
> - Bill Gates

(/course/algorithms-analyze-your-algorithm/) NEXT

Download Our App.



(https://play.google.com/store/apps/details?id=com.blogsdope&pcampaignid=MKT-Other-global-all-co-prtnr-py-PartBadge-Mar2515-1)

New Questions

setting up an ide for mac.. **- Cpp**

(/discussion/setting-up-an-ide-for-mac)

Please fill the blanks and help me am stuck **- Java**

(/discussion/fill-the-blanks-and-help-me-am-stuck)

Time complexity of the Python Function **- Python**

(/discussion/time-complexity-of-the-python-function)

How I Can find The Best Target Coupons & Latest Deals Offers? **- Other**

(/discussion/how-i-can-find-the-best-target-coupons-latest-deal)

To Calculate salaries **- Java**

(/discussion/to-calculate-salaries)

How to write a c++ program that will declares a two dimensional array say Char My element [4][5], prompt the user to initialize and display the elemen **- C**

(/discussion/how-to-write-a-c-program-that-will-declares-a-two-)