# Algorithm: Min/Max/Average Algorithm

## Resources & Acknowledgements

The following materials were utilized to create this document:

https://en.wikipedia.org/wiki/Time\_complexity#Linear\_time

## Description

Imagine you are given a list of unordered numbers, and you need to find the min, max, and average of the set of numbers. Solving this problem is very similar to using the Linear search algorithm.

Unfortunately, if the list is unordered we need a brute force style of algorithm to solve. In fact, in some instances it may make sense to first sort the list, but this time we will work with the given list as is.

It is important to reiterate this algorithm belongs in the class of “brute-force” algorithms. It is not an efficient method.

## Preliminary Analysis

We should think about a general approach before we begin programming the solution for this problem.

#### Min/Max Search

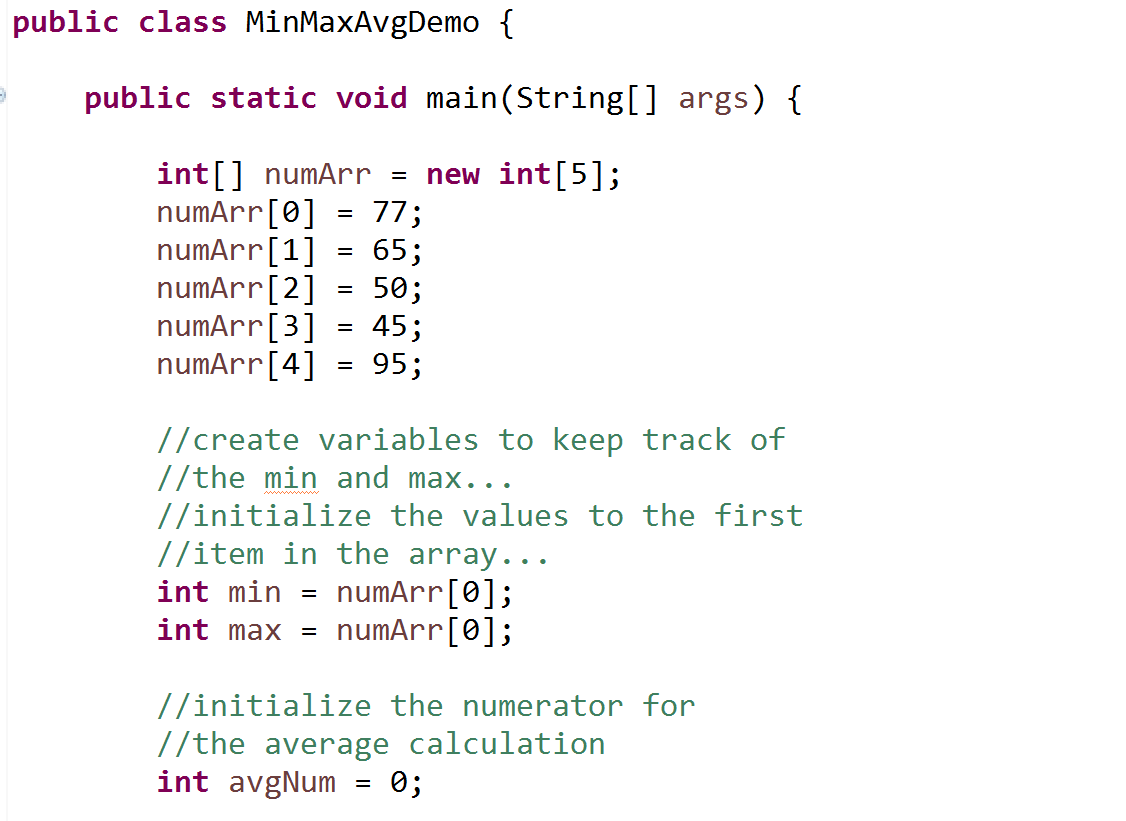
Unfortunately, we need to look at every value in the unordered set to determine if it is the min or max. We realize we don’t need to iterate the set two separate times, but instead, we can look for the min/max values by processing the set only once.

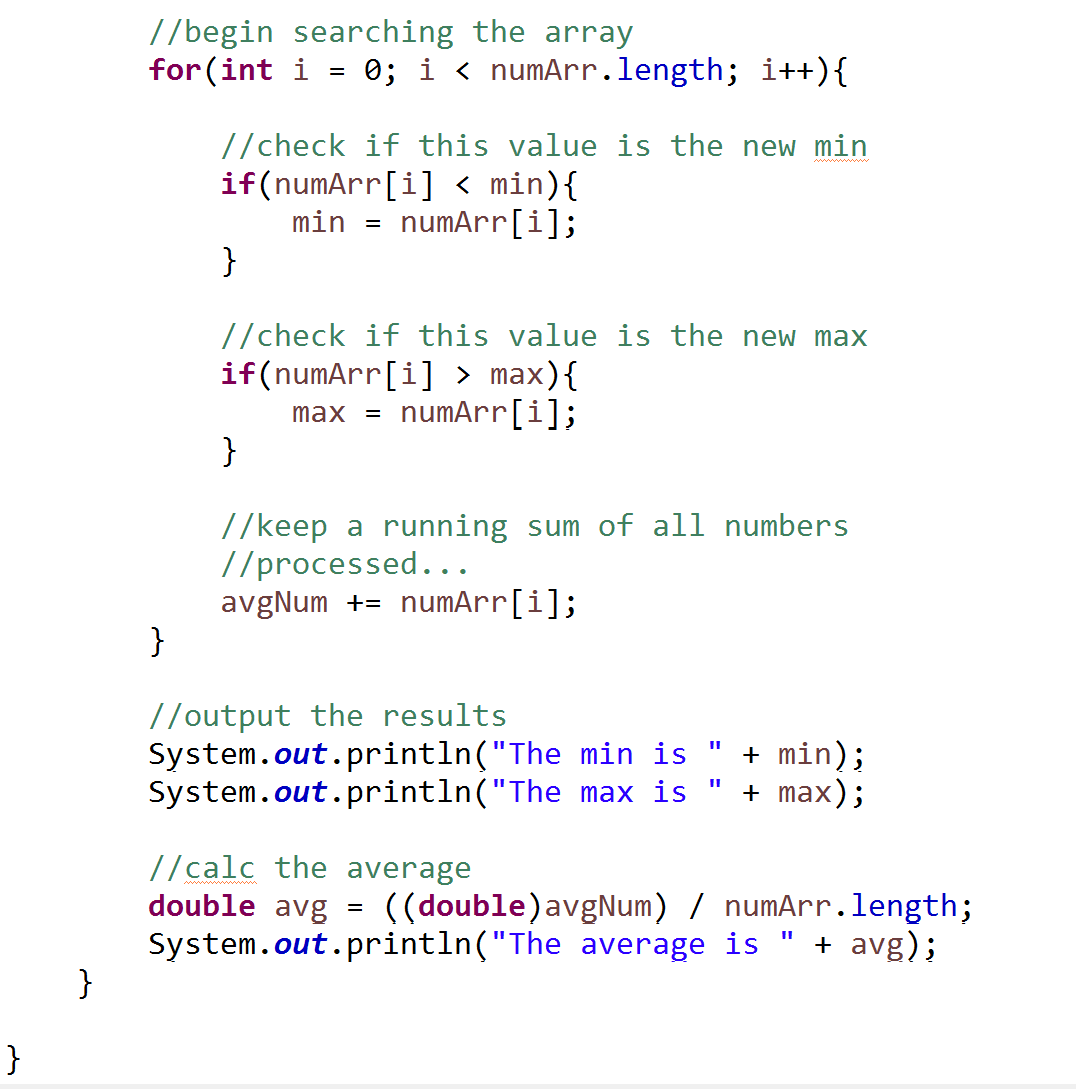
If we begin at the beginning of the array, we will only know min or max for values we have read; ie. There is no way to magically look ahead at the values in the array before we process them. Therefore, we will need to create a variable to track the current max and another variable to track the current min value. We will only know the min and max values, after all items in the array have been processed.

#### Average Calculation

Unfortunately, we need to look at all values here too, since computing the average requires it. Average is defined as the sum of all the values divided by the count of all the values. We will only know the sum after adding all of the values together. We realize we can save some processing by including the calculation of the sum in our loop that looks for the min/max value.

## Java Example





## Performance

Performance is easy to evaluate for this algorithm, since in all cases we need to compare all values in the set.

### Worst-Case, Best-Case, Average-Case

We need to process all items:

***n => O(n)***

[***Wikipedia***](https://en.wikipedia.org/wiki/Time_complexity#Linear_time)***:***

