Study Write-Up: Bubble Sort

When it comes to sorting algorithms, bubble sort is quite simple. It works as the name suggests: smaller values of the given data set “bubble” up to the top to sort the data. Here is a simple visualization:

Bubble sort is quite simple to understand. It works like so: starting from the beginning of the list, every adjacent pair gets compared. Swap their position if they are not in the right order (i.e., the latter one is smaller than the former one). After each iteration, one less element (the last one) is needed to be compared until there are no more elements left to be compared. Therefore, as mentioned earlier, the algorithm is dubbed “bubble sort” because the list is sorted as the smaller values “bubble” to the top.

Here is a simple implementation of bubble sort in Ruby:

```ruby
def bubble_sort(array, length)
  # The loop that iteratively goes through the array and sorts it
  array.each do |element|
    # Some pretty self-explanatory variables
    current_element = array.index(element)
    next_element    = current_element + 1

    # The first check is to make sure that the current element is not the last
    # because the last element has no next element to look to, so the program will crash

    if array[current_element] != array.last

      # The following bit of conditional logic compares the current element to the next
element and
      # replaces the current one with the next one if the next one is smaller
      # (so that it bubbles up the index chain)

      if array[next_element] < array[current_element]
        array.insert(current_element, array.delete_at(next_element))
      end
    end
```

```
  end

  # This bit of logic makes sure that the method recurs if it
  # has not iterated through the array for it's length of times

  if length == 0
    print array
  else
    bubble_sort(array, length - 1)
  end
end

array_to_be_sorted = [4,3,78,2,0,2] # insert your array to be sorted
bubble_sort(array_to_be_sorted, array_to_be_sorted.length)
```

As you can see, bubble sort is an easy algorithm to use for sorting. Unfortunately, it does have its drawbacks, chief among them being that it is slow and impractical for most problems when compared to other sorting algorithms. Another problem is that it can occasionally have some out-of-place values that are nearly in position. Nonetheless, it is a good algorithm to show beginners the concept of sorting and is used for that purpose to this day.