Shorting

By Michael Romero

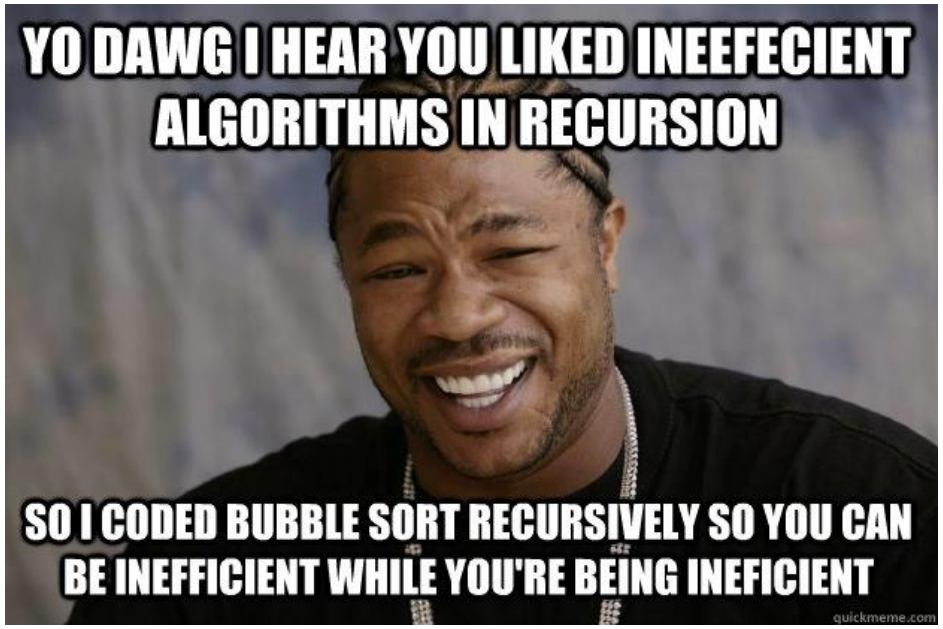
A bad shorting method

```
public int[] bubblesort(int [] array){
        int [] results = new int [array.length];
        for (int i = 0; i < array.length; i++) {
            results[i] = array[i];
        }
        //Shorts the array

        for (int i = 0; i <results.length ; i++) {
            for (int j = 1; j < results.length;
j++) {
                if (results[j-1] > results[j]){
                    int temp = results[j-1];
                    results[j-1] = results[j];
                    results[j] =temp;
                }

            }
//n^2 big o notaition
        }
        return results;
    }
```

A basic bubble short is one of the most inefficient shorting methods that you can use. The big on notation for a bubble sort is n^2. This method of shorting is useful when you have limited space/time and only for small lists that need to be shorted.



Selection short

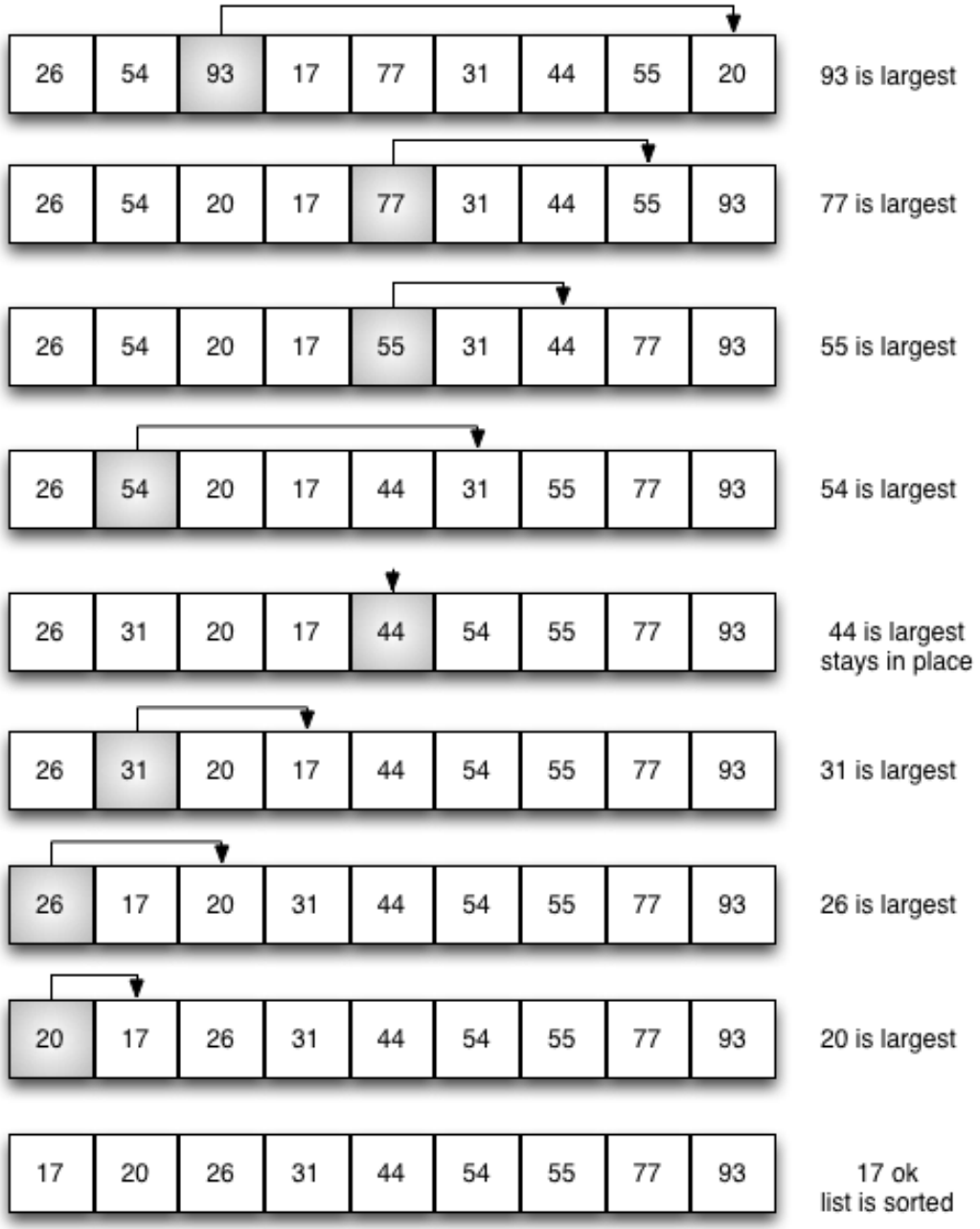
```c
void selectionSort(int arr[], int n)
{
    int i, j, min_idx;
    for (i = 0; i < n-1; i++)
    {
        // Find the minimum element in unsorted array
        min_idx = i;
        for (j = i+1; j < n; j++)
            if (arr[j] < arr[min_idx])
                min_idx = j;

        // Swap the found minimum element with the
first element
        swap(arr[min_idx], arr[i]);
    }
}
void swap(int xp, int yp)
{
    int temp = xp;
xp = yp;
yp = temp;
}
```

Sorting algorithms

Selection sort 2

Selection short finds the largest or smallest value and puts it into the index that it needs to be in. It will start typically at 0 and will move through the array list. Selection sort will have a big o notation of n^2



insertion sort

insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain. The big o notation could be n or n^2.

This short is bad for backwards shorted arrays.