ADS Homework 3

Name - Tewodros Adane
Email - tadane@jacobs-university.de

Problem 3.1 - Asymptotic Analysis

(a) $f(n) = 9n$ and $g(n) = 5n^3$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{9n}{5n^3} = 0 \qquad \lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{5n^3}{9n} = \infty$$

Therefore, the pair belong to the following relations $f \in o(g)$, $f \in O(g)$, $g \in \Omega(f)$, and $g \in \omega(f)$.

(b) $f(n) = 9n^{0.8} + 2n^{0.3} + 14logn$ and $g(n) = \sqrt{n}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{9n^{0.8}+2n^{0.3}+14logn}{\sqrt{n}} = \infty \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\sqrt{n}}{9n^{0.8}+2n^{0.3}+14logn} = 0$$

Therefore, the pair belong to the following relations $f \in \Omega(g)$, $f \in \omega(g)$, $g \in o(f)$, and $g \in O(f)$.

(c) $f(n) = n^2/logn$ and $g(n) = nlogn$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n^2/logn}{nlogn} = \frac{n}{(logn)^2} = \infty \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{nlogn}{n^2/logn} = \frac{(logn)^2}{n} = 0$$

Therefore, the pair belong to the following relations $f \in \Omega(g)$, $f \in \omega(g)$, $g \in o(f)$, and $g \in O(f)$.

(d) $f(n) = (log(3n))^3$ and $g(n) = 9logn$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{(log(3n))^3}{9logn} = \infty \qquad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{9logn}{(log(3n))^3} = 0$$

Therefore, the pair belong to the following relations $f \in \Omega(g)$, $f \in \omega(g)$, $g \in o(f)$, and $g \in O(f)$.

Problem 3.2 - Selection Sort

(a) The implementation is done in java

```java
static void selectionSort(int[] array) {
   for (int i = 0; i < array.length - 1; i++) {
       int currentMinInd = i;
       for (int j = i + 1; j < array.length; j++) {
           if (array[j] < array[currentMinInd]) {
               currentMinInd = j;
           }
       }
       if (currentMinInd != i) {
           int temp = array[i];
           array[i] = array[currentMinInd];
           array[currentMinInd] = temp;
       }
   }
}
```

(b) Given an unsorted array `A[0...n-1]` of length `n` unsorted integers.

Loop invariant: the element `A[currentMinInd]` is less than or equal to `A[i...j-1]`. This is always true at the beginning because `currentMinInd = i` and `j = i + 1` in the inner for loop, therefore, `A[i...j-1]` is the same as `A[i]`. For later iterations, this holds because `currentMinInd` is updated to have the index of the smallest int in the array. As the program exits the inner loop the elements at `currentMinInd` are swapped to position `i`, this builds up the subarray `A[0...i]` to the left of the `i` that is sorted. The loop terminates at `i==n` at this point all elements are to the left of `i` and are in sorted order.

(c) The array generation code for all cases is included as methods in the *SelectionSort.java* file.

Case A: The most swaps are done when the largest item is the first element and the rest of the elements are added to the right of the first element in increasing order. The number of swaps is n - 1

Example - for an array of 5 elements

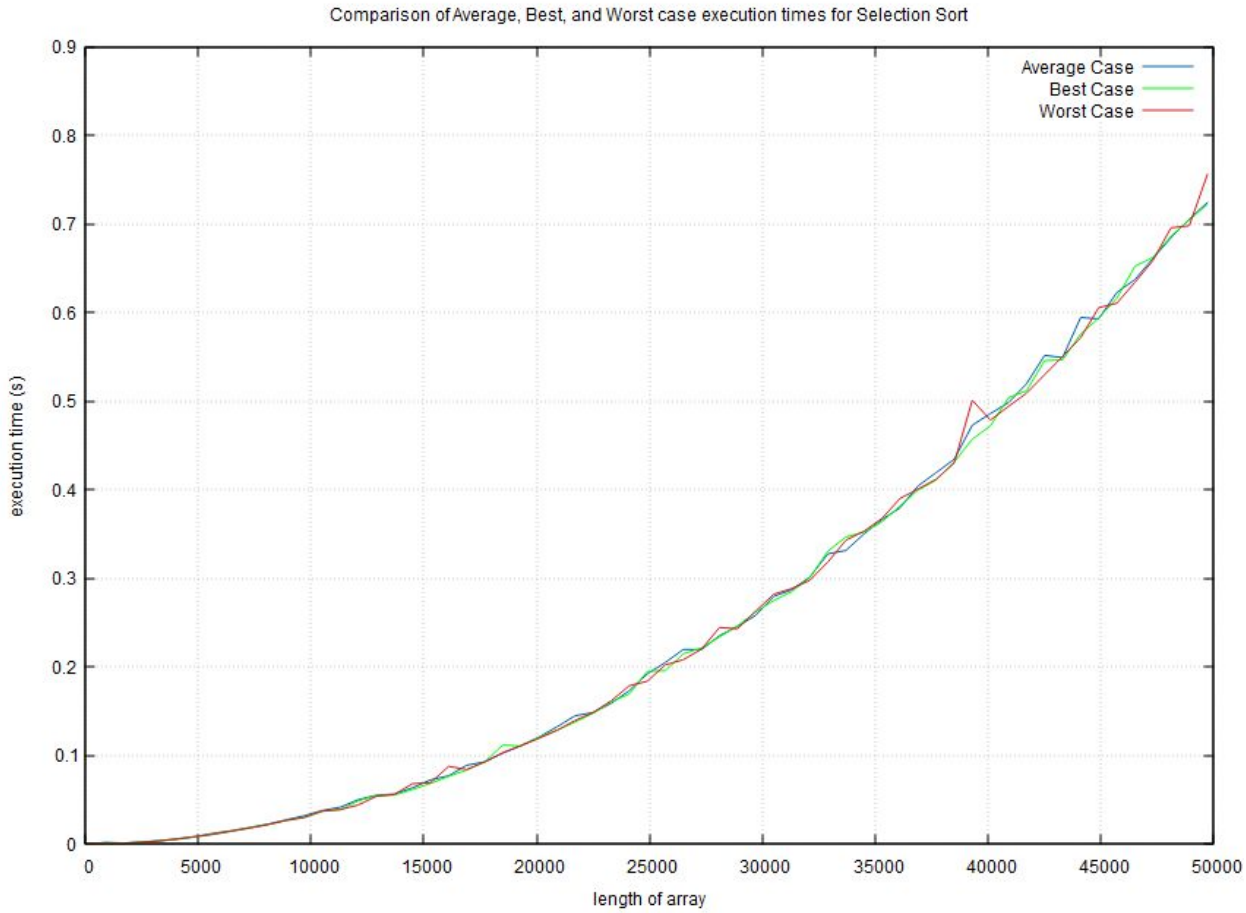
[5,1,2,3,4]→[1,5,2,3,4]→[1,2,5,3,4]→[1,2,3,5,4]→[1,2,3,4,5]

After four swaps every element is sorted.

Case B: The least number of swaps is achieved when the list is initially sorted. Because there is nothing to swap.

Random Case: The random case generator was implemented by creating an array of a given length and filling it with random values.

(d) When the *SelectionSort.java* is run. The program creates arrays for all three cases with an increasing length starting from 100 to 50000 with an interval of 800. For all cases at each specific length sorting is done 5 times and the average time is taken. Then the program writes the length and corresponding time to 3 different files, that are used to generate the following graph.



(e) For an array of length n, no matter if the input takes the least number of swaps or the most number of swaps, the outer for loop is gonna be run n times and the inner for loop is going to be run starting from n times at the beginning to only once at the end. Therefore the sum of that progression is going to be $\frac{n(n+1)}{2}$, this shows that no matter the type of input Selection sort belongs to $O(n^2)$. We can also see this from the graph above the best, average, and worst case all lie more or less on the same line. The graph has some fluctuations because the computer is not only sorting but the operating system also has some background processes running.