Algorithms and Data Structures
Assignment 1

Dragi Kamov

Problem 1

(a) $f(n) = 3n \qquad g(n) = n^3$

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

$f(n) = o(g(n))$

$g(n) = \omega(f(n))$

(b) $f(n) = 7n^{0.7} + 2n^{0.2} + 13\log n$

$g(n) = \sqrt{n} = n^{0.5}$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{7n^{0.7} + 2n^{0.2} + 13\log n}{n^{0.5}} = \infty$$

$f(n) = \omega(g(n))$

$g(n) = o(f(n))$

(c) $f(n) = n^2/\log n \qquad g(n) = n\log n$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\frac{n^2}{\log n}}{n\log n} = \frac{n^2}{n\log^2 n} = \frac{n}{\log^2 n} = \infty$$

$\lim_{n\to\infty} \frac{n}{n} = 1$

$f(n) = \omega(g(n))$

$g(n) = o(f(n))$

(d) $f(n) = (\log(3n))^3$

$g(n) = g\log n$

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

$f(n) = \omega(g(n))$

$g(n) = o(f(n))$

Problem 2

a) "sel_sort.cpp"

b) Selection sort works in the way that it starts from the begining of the array, takes that element and checks for a value lower than the one that it is checking for and also that value has to be the smallest in the rest of the array. For example if the program is on position i, it takes that value of that position and goes through $[i+1, n]$ looking for the minimum value, and then it is swarping those two values, thus sorting the

array as it is going (loop invariant)

If we try to prove it by induction:

Induction base:
Let's say that we are on position i=0, the selection sort will start going through the rest of the array [i+1, n] and let's say that it finds the minimum value at position j and then it will swap them. Whilst continuing on the i+1 position and checking from [i+2, n].

Induction hypothesis:
Let's assume that from an array a part till the j position is sorted so the selection sort for the j position would search through [j+1, n] for the minimum value and then swap them.

Induction step:
So the array is sorted till j+1 and the selection sort would continue to do the same, looking for the minimum value and swap elements till the end of the array where we would have a sorted array.

c) I am generating random input sequences with "randomized.cpp" using the function "rand()" and writing the times that it needs to do the selection sort for that array in the "random.txt" file.

The best case would be already sorted array. The code for the best case is in "best.cpp" and the times are written in "best.txt".

The worst case would be an array sorted in descending order. The code for the worst case is "worst.cpp" and the times are in "worst.txt"

e) The red line represents the worst case, the blue line represents the best case and the random case is represented by the yellow line.

$r(n) = w(b(n))$
$r(n) = w(y(n))$
$y(n) = w(b(n))$

$b(n) = o(r(n))$
$b(n) = o(y(n))$
$y(n) = o(r(n))$

$r(n) \rightarrow$ worst
$b(n) \rightarrow$ best
$y(n) \rightarrow$ random