Problem 3-1

Ans a-)
Now observing the limits

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{9n}{5n3} \qquad \text{and} \qquad \lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{5n3}{9n}$$

$$= \lim_{n\to\infty} \frac{9}{5n2} \qquad = \lim_{n\to\infty} \frac{5n2}{9}$$

$$= 0 \qquad = \infty$$

from these limits we can make the following conclusions

$f(n) \in O(5n^3)$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is true]

$f(n) \notin \Omega(5n^3)$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ is not true]

$f(n) \notin \Theta(5n^3)$ [$\because 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \notin \omega(5n3)$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ is not true]

$f(n) \in o(5n3)$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ is true].

and,

$g(n) \notin O(9n)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is not true]

$g(n) \in \Omega(9n)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} > 0$ is true]

$g(n) \notin \Theta(9n)$ [$\because 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is not true]

$g(n) \in \omega(9n)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ is true]

$g(n) \notin o(9n)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$ is not true].

Ans 5.1

We have:

$$f(n) = 9n^{0.8} + 2n^{0.3} + 14\log(n) \quad \text{and} \quad g(n) = \sqrt{n}$$

Now,

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

$$= \lim_{n\to\infty} \frac{9n^{0.8}}{\sqrt{n}} + \frac{2n^{0.3}}{\sqrt{n}} + \frac{14\log(n)}{\sqrt{n}}$$

$$= \lim_{n\to\infty} 9n^{0.8-0.5} + 2n^{0.3-0.5} + \frac{14\log(n)}{\sqrt{n}}$$

$$= \lim_{n\to\infty} 9n^{0.3} + 2n^{-0.2} + \frac{14\log(n)}{\sqrt{n}}$$

$$= \lim_{n\to\infty} 9n^{0.3} + \lim_{n\to\infty} 2n^{-0.2} + \lim_{n\to\infty} \frac{14\log(n)}{\sqrt{n}}$$

$$= \lim_{n\to\infty} 9n^{0.3} + 0 + \lim_{n\to\infty} \frac{14 \cdot 1/n}{\frac{1}{2\sqrt{n}}} \quad [\text{Using L'Hopital's rule}]$$

$$= \lim_{n\to\infty} 9n^{0.3} \lim_{n\to\infty} \frac{28}{\sqrt{n}}$$

$$= \infty + 0$$

$$= \infty$$

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right)^{-1}$$

$$= \left(\lim_{n\to\infty} \frac{f(n)}{g(n)}\right)^{-1}$$

$$= (\infty)^{-1}$$

$$= 0$$

From the limits above, we can conclude that

From the limits above, we can conclude that

$f(n) \notin O(\sqrt{n})$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \in \Omega(\sqrt{n})$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ is true]

$f(n) \notin \Theta(\sqrt{n})$ [$\because 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \in \omega(\sqrt{n})$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ is true]

$f(n) \notin o(\sqrt{n})$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ is not true.]

and

$g(n) \in O(9n^{0.8} + 2n^{0.3} + 14\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is true]

$g(n) \notin \Omega(9n^{0.8} + 2n^{0.3} + 14\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} > 0$ is not true]

$g(n) \notin \Theta(9n^{0.8} + 2n^{0.3} + 14\log(n))$ [$\because 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is not true]

$g(n) \notin \omega(9n^{0.8} + 2n^{0.3} + 14\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ is not true]

$g(n) \in o(9n^{0.8} + 2n^{0.3} + 14\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$ is true]

Ans c.)

We have

$f(n) = \frac{n^2}{\log(n)}$ and $g(n) = n\log(n)$

Now,

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

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

$$= \lim_{n\to\infty} \frac{\frac{d}{dn} n}{\frac{d}{dn} (\log(n))^2} \quad \text{[using L'Hopital's rule]}$$

$$= \lim_{n\to\infty} \frac{1}{2\log(n) \cdot \frac{1}{\ln(2)} \cdot \frac{1}{n}}$$

$$= \lim_{n\to\infty} \frac{n \cdot \ln(2)}{\log_2(n)}$$

$$= \infty$$

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

$$= \lim_{n\to\infty} \frac{[\log_2(n)]^2}{n}$$

$$= \lim_{n\to\infty} \frac{\frac{d}{dn} [\log_2(n)]^2}{\frac{d}{dn} n} \quad \text{[using L'Hopital rule]}$$

$$= \lim_{n\to\infty} \frac{2\log_2(n) \cdot \frac{1}{n \cdot \ln(2)}}{1}$$

$$= \lim_{n\to\infty} \frac{2\frac{\ln(n)}{\ln(2)} \cdot \frac{1}{\ln(2)\, n}}{1}$$

$$= \lim_{n\to\infty} 2\,\frac{\ln(n)}{n\cdot(\ln(2))^2}$$

$$= 2 \cdot 0 = 0$$

Observing the limits we can conclude that:

$f(n) \notin O(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \in \Omega(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ is true]

$f(n) \notin \Theta(n\log(n))$ [$\because 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \in o(n\log(n))$ [$\because \lim_{n\to\infty} = \infty$ is true]

$f(n) \notin \omega(n\log(n))$ [$\because \lim_{n\to\infty} = 0$ is not true]

and,

$g(n) \in O(n^2/\log n)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is true]

$g(n) \notin \Omega(n^2/\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} > 0$ is not true]

$g(n) \notin \Theta(n^2/\log(n))$ [$\because 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is not true]

$g(n) \in o(n^2/\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$ is true]

$g(n) \notin \omega(n^2/\log(n))$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ is not true]

Ans d.)

$f(n) = (\log(3n))^3$ and $g(n) = g\log(n)$

Now

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

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

$$= \lim_{n\to\infty} \frac{(\log(3))^3 + 3\cdot(\log(3))^2 \cdot \log(n) + 3\cdot\log(3)\cdot(\log(n))^2 + (\log n)^3}{g\log(n)}$$

$$= \lim_{n\to\infty} \frac{(\log(3))^3}{g\log(n)} + \frac{3(\log(3))^2}{g} + \frac{3\log(3)\cdot\log(n)}{g} + \frac{(\log(n))^2}{g}$$

$$= \lim_{n\to\infty} \frac{(\log(3))^3}{g\log(n)} + \lim_{n\to\infty} \frac{(\log(3))^2}{3} + \lim_{n\to\infty} \frac{\log(3)\cdot\log(n)}{3} + \lim_{n\to\infty} \frac{\log(n)^2}{g}$$

$$= 0 + \frac{\log(3)^2}{3} + \infty + \infty$$

$$= \infty$$

Then,

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \left(\frac{f(n)}{g(n)}\right)^{-1}$$

$$= \left(\lim_{n\to\infty} \frac{f(n)}{g(n)}\right)^{-1}$$

$$= \frac{1}{\lim_{n\to\infty} \frac{f(n)}{g(n)}}$$

$$= \frac{1}{\infty} = 0$$

$\therefore$

Now, from the limits obtained, we can conclude that:

$f(n) \notin O(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \in \Omega(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ is true]

$f(n) \notin \Theta(n\log(n))$ [$\because 0 < \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ is not true]

$f(n) \notin o(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ is not true]

$f(n) \in \omega(n\log(n))$ [$\because \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ is true]

and

$g(n) \in O((\log(3n))^3)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$ is true]

$g(n) \notin \Omega((\log(3n))^3)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} > 0$ is not true]

$g(n) \notin \Theta((\log(3n))^3)$ [$\because 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)}$ is not true]

$g(n) \in o((\log(3n))^3)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)}$ ue]

$g(n) \in \omega((\log(3n))^3)$ [$\because \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$ is true]

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ is not tr

$< \infty$

Problem 3.2

a) Implement a selection sort:

-> Implementation is inside selection_sort.cpp

b.) Show that selection sort is correct:

```
->      selection_sort(A,last)
        for i = 1 to last
                pos = I;
                smallest = A[i]
                for j = i to last
                        if A[j] < smallest
                                pos = j
                                smallest = A[j]
                copy = A[i]
                A[i] = smallest
                A[j] = copy
```

Chosen loop invariant: The A[1..i-1] subarray is always sorted.

Initialization: Before entering the outer loop the value of i is 1. Thus, there is no subarray A[1..i-1] and hence can be considered as an empty array. Since an empty array has no elements it can be considered as a sorted array.

Maintenance: As soon as the program flow enters the loop the smallest element amongst the remaining elements is searched for. In fact, the first time when this is done, the smallest element in the array is placed in the first position of the array, as if the first element in the unsorted array is the smallest in the entire array then it ends up being placed at the beginning of the array, and if it is not the smallest element, then the rest of the elements are searched for the smallest element and that element is placed at the beginning of the array. The value of i increases to 2, and since the sub-array A[1..i-1] contains only one element it is already sorted.

After the value of i has been increased to 2 the algorithm searches for the smallest element in the array whilst ignoring the element placed at the beginning of the array. Since the first element was the smallest element, the search ends up in finding the second smallest element. This element is then exchanged with the second element, which ends up putting the second-smallest element in the array immediately after the smallest element. So, when the value of i increases to 3, the subarray A[1..i-1(i.e 2)] is already sorted.

Likewise, when the outer loop reaches its n^{th} iteration, elements in the sub-array A[1..n-1] comprise of the smallest n-1 elements in sorted order and the algorithm searches for the n^{th} smallest element to be placed after the A[1..n-1] subarray. Thus leading to the A[1..n-1] subarray always being sorted.

Termination: When the value of i is 'last', then the algorithm will search for the smallest element within the part of the array whose index runs from last to last. Since it is an empty sub-array, it finds no element at all and puts the last element found in the unsorted array at the end of the sorted array consisting of the first $(last-1)^{th}$ smallest elements. The last element must be larger than all of these elements as it is the reason why it has not been incorporated into the A[1..(last-1)] subarray, leaving the A[1..last] array sorted.

(c) Generate random input sequences of length n as well as sequences of length n that represent Case A and Case B for the Selection Sort algorithm. Case A: the case which involves the most swaps (Hint: it is not a decreasingly ordered array). Case B: the case with the least swaps. Briefly describe how you generated the sequences (e.g., with a random sequence generator using your chosen language)

The algorithm makes the maximum number of swaps, hence leading to the worst possible behaviour of the algorithm when the given array(whose elements are randomly generated by the function below) has its largest element in the first position and when the element onwards are already sorted. The above function generates such an array by sorting the array first and then shifting the index of all the elements of the array by 1. Then the first element of the array is assigned the largest value that the array used to hold through the copy of that element, which was made prior to the shifting of the index.

The best possible behaviour of the algorithm is achieved when it makes the least number of swaps, and that is achieved when the array given to the algorithm is already sorted.

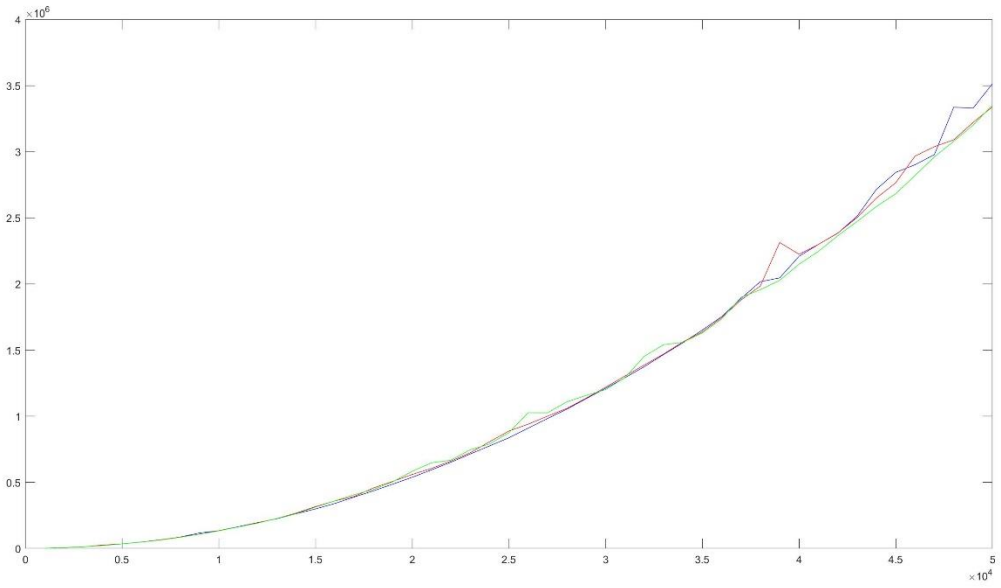
D. Run the algorithm on the sequences from (c) with length n for increasing values of n and measure the computation times. Plot curves that show the computation time of the algorithm in Case A, Case B, and an average case for an increasing input length n. Note that in order to compute reliable measurements for the average case, you have to run the algorithm multiple times for each entry in your plot. You can use a plotting tool/software of your choice (Gnuplot, R, Matlab, Excel, etc.).

Case A: Red

Case B: Green

Average case: Blue

->



5. Interpret the plots from (d) with respect to asymptotic behavior and constants.

The curves for all the cases appear to belong to the set $\theta(n^2)$. This is because the searching process of the smallest element in the algorithm repeats itself in the pattern n, n-1, n-2…3,2,1with every iteration of the outer loop. Hence for the given input this step repeats itself n(n+1)/2 times, and since this is the term of highest order of n present in the sum of computations done in the algorithm, the degree of the expression which gives the sum of computation of the algorithm is $n^{2.}$