ADS 3

P3.1

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

$(f_n = 9n) \in O(n)$
$(g(n) = 5n^3) \in O(n^3)$
g grows faster than f as for $n \to \infty$ f will never be even close to g

so g is strictly upper bound for f

$\Rightarrow f(n) \in o(g(n)) \Rightarrow f \in O(g(n))$
$\Rightarrow g(n) \in \omega(f(n)) \Rightarrow g_n \in \Omega(f_n)$

b - $f(n) = 9n^{0,8} + 2n^{0,3} + 14\log(n)$ / $g(n) = \sqrt{n}$

$f(n) \in O(n^{0,8})$ "because $9n^{0,8}$ is the dominant term"
$g(n) \in O(n^{0,5})$ " $n^{0,5} = \sqrt{n}$ "

$n^{0,8}$ is more dominant than $n^{0,5}$ so f grows faster than g / $(n \to \infty)$ (g never reaches f)

$\Rightarrow g(n) \in o(f(n))$, $g(n) \in O(f(n))$
$f(n) \in \omega(g(n))$, $f(n) \in \Omega(g(n))$

c - $f(n) = \dfrac{n^2}{\log(n)}$ / $g = n \log(n)$

with this one it is not specifically clean wich one is the dominant.

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

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \frac{n \log n}{\frac{n^2}{\log n}} = \frac{(\log(n))^2}{n} = 0$$

$\Rightarrow g(n) \in o(f(n))$, $g(n) \in O(f(n))$
$f(n) \in \omega(g(n))$, $f(n) \in \Omega(g(n))$

d - $f(n) = (\log(3n))^3$ / $g(n) = 9 \log(n)$

$f(n)$ is obviously growing faster than $g(n)$ for the same reason.

$\Rightarrow g(n) \in o(f(n))$, $g(n) \in O(f(n))$
$f(n) \in \omega(g(n))$, $f(n) \in \Omega(g(n))$

P3.2/

a- implementation of selection sort

P3_1.cpp > ...

```cpp
#include <algorithm>

    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

    void selection_sort(int a[], int n){
        for (int i = 0; i < n; i++){
            int min = i;
            for (int j = i+1; j < n; j++){
                if (a[j] < a[min]){
                    min = j;
                }
            }
            std::swap(a[i], a[min]);
        }
    }
```

(PS: done on VS code using C++ :)
the whole code is in "selectionsort.cpp"

b-

in the selection sort the loop invariant is that after each iteration the left side of the array is always sorted, in a increasing manner, to prove that:

Initialisation:

before the first iteration the left portion of the array contains 0 element which can be translated in it being trivialy alreandy sorted.

Maintenance:

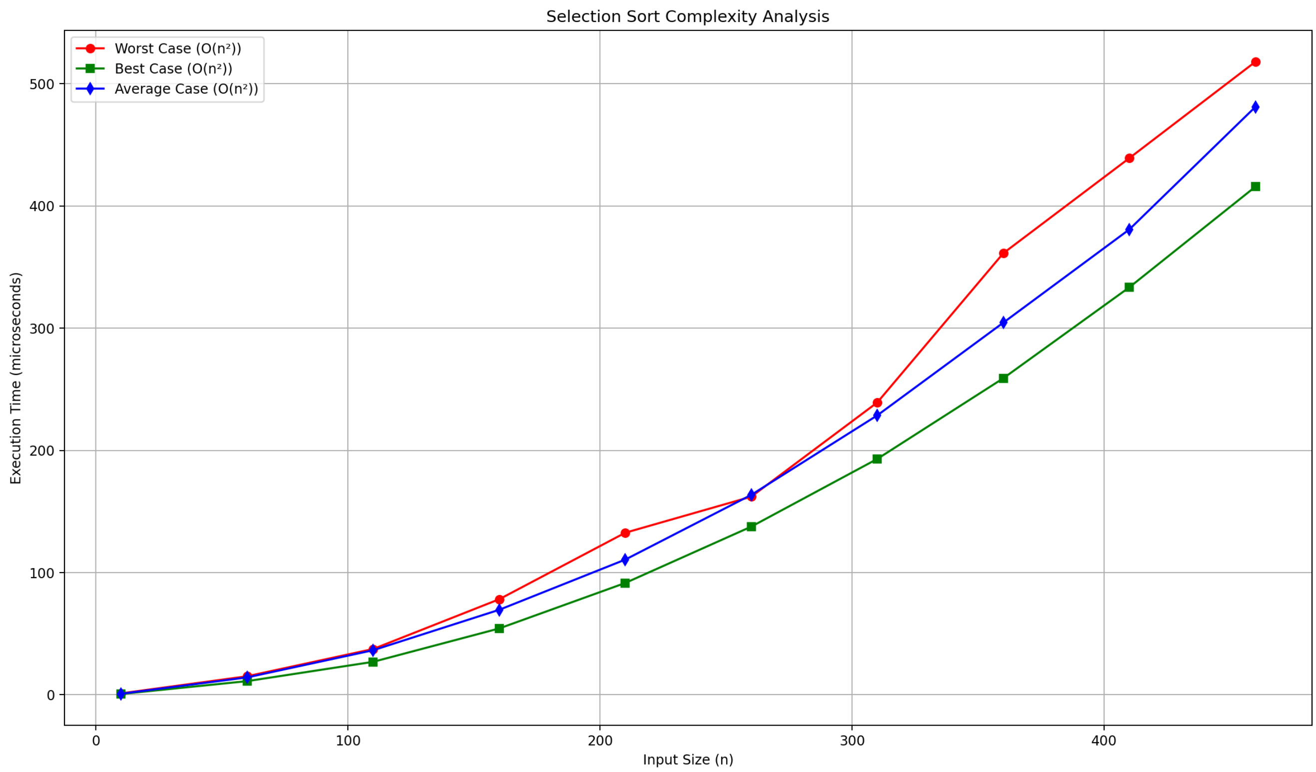
here we need to show that each iteration maintains the invariant, at iteration i an the smallest element in the right portion is alwas selected and sorted, put in the right position of the sorted portion of the array, which means that after each iteration the the invariant is maitained.

Termination:

here, at this point the number of iteration i is equal to n, number of element wich means that the left portion of the array is equal to the whole array which is now sorted.

c- } all seen in the cpp file
d- }

e- plotting.py


Selection Sort Complexity Analysis
Worst Case (O(n²))
Best Case (O(n²))
Average Case (O(n²))
Execution Time (microseconds)
Input Size (n)
0
100
200
300
400
500