Homework 1

Problem 1: Asymptotic Analysis

a) $f(n) = 3n$ & $g(n) = n^3$.
$f \in O(g)$ iff $\exists$ c and $n_0 \mid 0 \leq f(n) \leq cg(n), \forall n \geq n_0$

let $c = 3$, $\therefore 0 \leq 3n \leq 3n^3$
$n_0 = 2$ $\quad 0 \leq 3 \leq 3n^2$
$0 \leq 3 \leq 12$
$\therefore$ $f \in O(g)$

b) $f(n) = 7n^{0.7} + 2n^{0.2} + 13 \log n$ & $g(n) = \sqrt{n} = n^{0.5}$
$f \in \Omega(g)$ iff $\exists$ c & $n_0 \mid 0 \leq cg(n) \leq f(n), \forall n \geq n_0$

let $c = 5$
$n_0 = 1000$
$\therefore 0 \leq 5n^{0.5} \leq 7n^{0.7} + 2n^{0.2} + 13 \log n$
$0 \leq 5 \leq 7n^{0.2} + 2n^{-0.3} + {}^{13}/_{\sqrt{n}} \log n$
for $n_0 = 1000$, $f(n) \approx 7$
$\therefore$ $f \in \Omega(g)$

c) $f(n) = {}^{n^2}/_{\log n}$ and $g(n) = n \log n$
$f \in \omega(g)$ iff $g \in o(f)$. ~~$g \in o(f)$ iff $\exists$ c and n_0 $c > 0$~~,
$g \in o(f)$ iff for any $c > 0$, $\exists$ $n_0 > 0 \mid 0 \leq g(n) \leq cf(n)$,
$\forall n_0 \geq n_0$.

$0 \leq g(n) \leq cf(n)$
$0 \leq n\log n \leq c(n^2/\log n)$
$0 \leq \log n \leq c(n/\log n)$
$0 \leq (\log n)(\log n) \leq cn$

$\forall$ integer values of $c > 0$ and $n > 0$, the statement holds true. As $(\log n)^2$ grows logorithmically & cn linearly.
$\therefore$ $\underline{\underline{f \in \omega(g)}}$

d) $f(n) = (\log 3n)^3$ and $g(n) = 9\log n$
$f \in o(g)$ iff for any $c > 0$, $\exists\, n_0 > 0 \mid 0 \leq f(n) \leq cg(n)$, $\forall\, n \geq n_0$

$\therefore$ $0 \leq (\log(3n))^3 \leq (9\log n) \times c$
$0 \leq (\log(3) + \log n)^3 \leq c(9\log n)$
$0 \leq \log n(\log 3 + 1)^3 \leq c(9\log n)$
$0 \leq (\log 3 + 1)^3 \leq 9c$
$0 \leq 3.223 \leq 9c$
$\therefore$ $\underline{\underline{f \in o(g)}}$

Problem 2

All code is written in C.

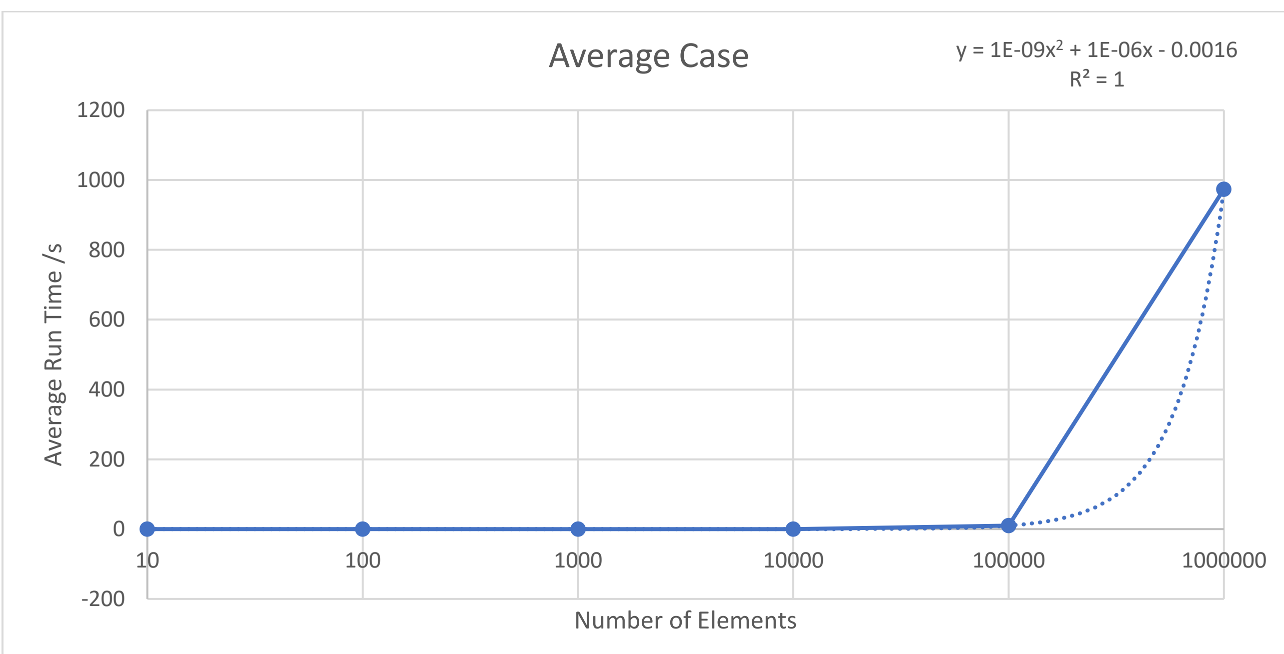
a) Selection sort implementation:

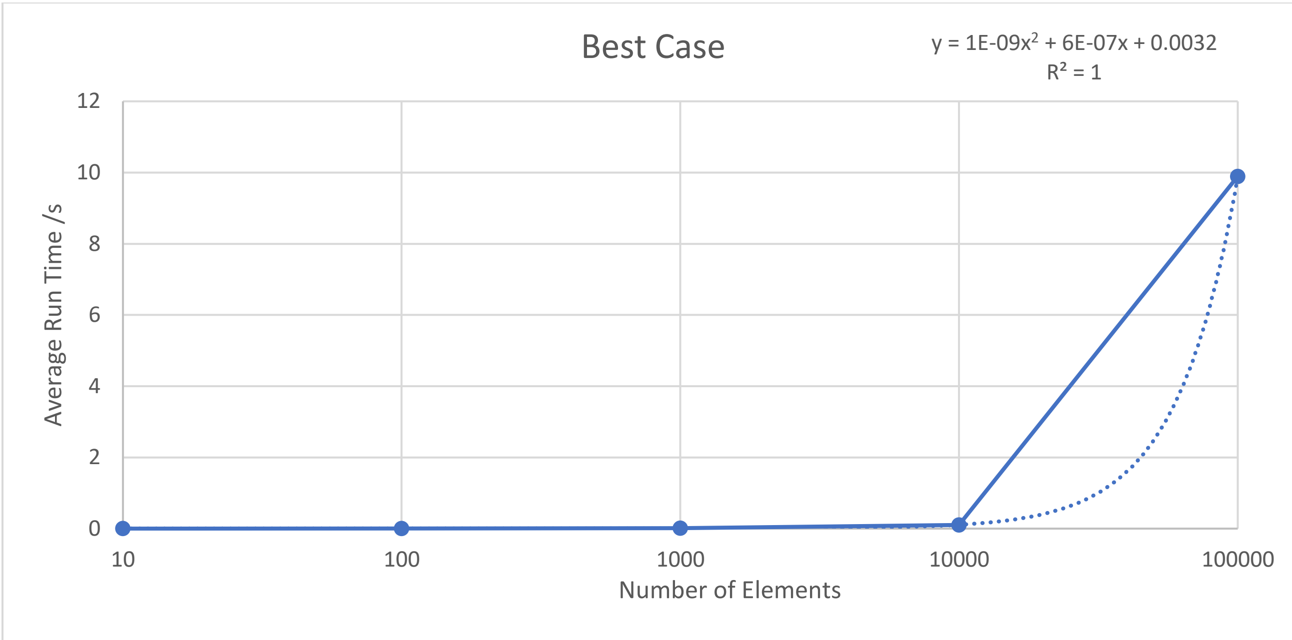
```
for(i = 0; i < SIZE; i++)
  {
//Setting the current index to the minimum
    min = i;
    //Running from end of array towards current position
    for(j = i; j < SIZE; j++)
        {
          //Always replacing min with a smaller element*/
          if(arr[j] < arr[min])
            min = j;
        }
    //Swap function to switch the elements
    swap(&arr[i], &arr[min]);
  }
```

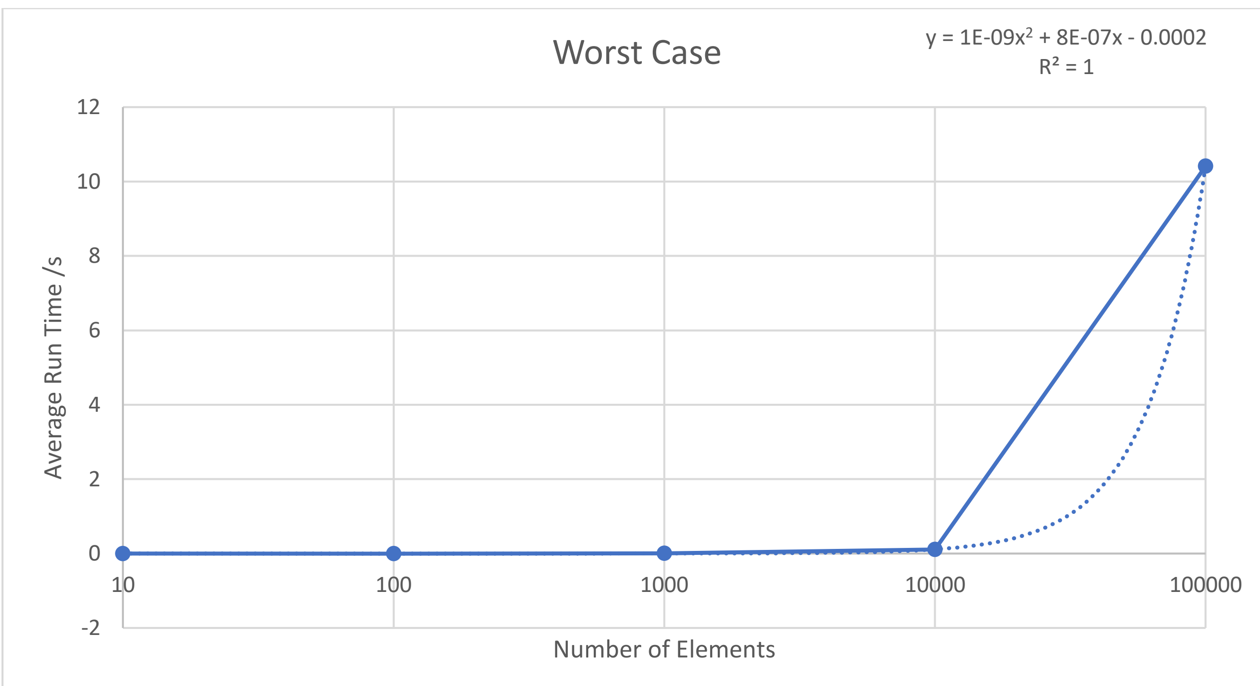
b) Loop invariant: at the end of every iteration of the loop, the subarray arr[0…i] contains the smallest elements of the arr[0…SIZE], but in sorted order.

c) Random numbers were generated programmatically using the rand function and the time on the machine running the code. Find attached C file.

d) Below are the graphs containing the run times for the selection sort. The points were calculated from the average of 10 trails for each number of elements. Find the excel sheet with the raw data in the .zip file.







e) Best case: $f(n) = 10^{-9}n^2 + 6 \times 10^{-7}n + 0.0032$
Average case: $f(n) = 10^{-9}n^2 + 6 \times 10^{-6}n - 0.0016$
Worst case: $f(n) = 10^{-9}n^2 + 8 \times 10^{-6}n - 0.0002$

These functions gave the best R^2 value, showing that the algorithm has polynomial time, of degree 2 (note that polynomial function of degree 3 also gave R^2 value of 1).
Getting rid of machine dependent constants we get a function $\Theta(n^2)$ for this algorithm.