ADS Assignment 1

Drishti Mahajan

Problem 1.1

a> $f(n) = 3n$ and $g(n) = n^3$

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

Since, $\lim = 0$, $f = o(g)$ and this implies $f = O(g)$.

And, $\lim = 0 \Rightarrow f \neq \Omega(g)$.

$\lim \neq \infty \Rightarrow f \neq \omega(g)$. Thus, $f \neq \Theta(g)$.

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

Since $\lim \neq 0$, $g \neq o(f)$ and $g = \Omega(f)$, and $\lim = \infty \Rightarrow g \neq O(f)$ and $g = \omega(f)$. Thus, $g \neq \Theta(f)$

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

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

As $\lim \neq 0$, $f \neq o(g)$ and $f = \Omega(g)$.

And $\lim = \infty \Rightarrow f = \omega(g)$ and $f \neq O(g)$.

So, $f \neq \Theta(g)$

$$\Rightarrow \lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{1}{\infty} = 0$$

As limit $= 0 \Rightarrow g = o(f)$ and $g \neq \Omega(f)$.

As limit $\neq \infty \Rightarrow g = O(f)$ and $g \neq \omega(f)$.

Hence, $g \neq \Theta(f)$

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

$$\lim \frac{f(n)}{g(n)} = \frac{n^2/\log n}{n \log n} = \frac{n}{(\log n)^2} = \infty$$

$$\lim \frac{g(n)}{f(n)} = \frac{n \log n}{\frac{n^2}{\log n}} = \frac{(\log n)^2}{n} = 0$$

So, $\underline{f = \omega(g)}$; $\underline{f = \Omega(g)}$

$\underline{f \neq o(g)}$; $\underline{f \neq O(g)}$. $\underline{f \neq \Theta(g)}$

And, $\underline{g = o(f)}$; $\underline{g = O(f)}$

$\underline{g \neq \omega(f)}$; $\underline{g \neq \Omega(f)}$. $\underline{g \neq \Theta(g)}$

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

$$\Rightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{(\log 3n)^3}{9 \log n} = \frac{\left(\frac{\ln 3n}{\ln 10}\right)^3}{9 \frac{\ln n}{\ln 10}}$$

$$= \frac{1}{9(\ln 10)^2} \times \lim \frac{\ln (3n)^3}{\ln n}$$

$$= \frac{1}{\cancel{9}_3 \times 2 \ln 10} \times \lim \frac{\cancel{3} \ln 3n}{\ln n} = \frac{1}{6 \ln 10} \lim_{n \to \infty} \frac{\ln 3n}{\ln n}$$

$$= \frac{1}{6 \ln 10} \times 1 = \frac{1}{6 \ln 10}$$

As $\lim \neq 0$, $\underline{f \neq o(g)}$ and $\underline{f = \Omega(g)}$

As $\lim \neq \infty$, $\underline{f = O(g)}$ and $\underline{f \neq \omega(g)}$

So, $\underline{f = \Theta(g)}$

Then,

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

$$= \frac{9 \frac{\ln n}{\ln 10}}{\left(\frac{\ln 3n}{\ln 10}\right)^3}$$

$$= 9 (\ln 10)^2 \cdot \frac{\ln n}{\ln (3n)^3}$$

$$= 9 \times 2 \ln 10 \cdot \frac{\ln n}{3 \ln 3n}$$

$$= 6 \ln 10 \cdot \lim_{n \to \infty} \frac{\ln n}{\ln 3n}$$

$$= 6 \ln 10$$

So, according to same logic,

$\underline{g \neq o(f)}$; $\underline{g = O(f)}$; $\underline{g \neq \omega(f)}$; $\underline{g = \Omega(f)}$.

And, $\underline{g = \Theta(f)}$.

Problem 1.2

a) Implementation of selection sort in C (code: 1.2.c) ⇒ source code

Pseudocode:

```
array [1..n]
1. for i = 1 to n-1
2.     min = i
3.         for j = i+1 to n
4.             if A[j] < A[min]
5.                 min = j
6.         swap A[i] with A[min]
```

b) Loop invariant:

(i) Initialization - Prior to inner loop, sub array $A[1..j-1] = A[1]$. It contains only one element (first one). Since it is a trivial case, it is sorted.

(ii) Maintenance -
At line 2, min has index to smallest element of sub-array ~~before entering loop~~. If any element $A[j]$ is lesser than the min element, min index changes & values are swapped. At start of each iteration j, the sub-array $A[1..j-1]$ always contains a sorted list. as each iteration finds minimum of unsorted list $A[j..n]$.

(iii) Termination: During termination of loop j, min has element less than or equal to elements in sub array $A[i..n]$. ⇒

Since $j = n+1$ upon termination. This helps to find next smallest element in unsorted list, while keeping elements in sorted list in order. And, moving smallest element in correct location becomes possible with swap.

An example case of 5 elements:

4 5 3 6 2 0

helps prove that the algorithm works, as well.

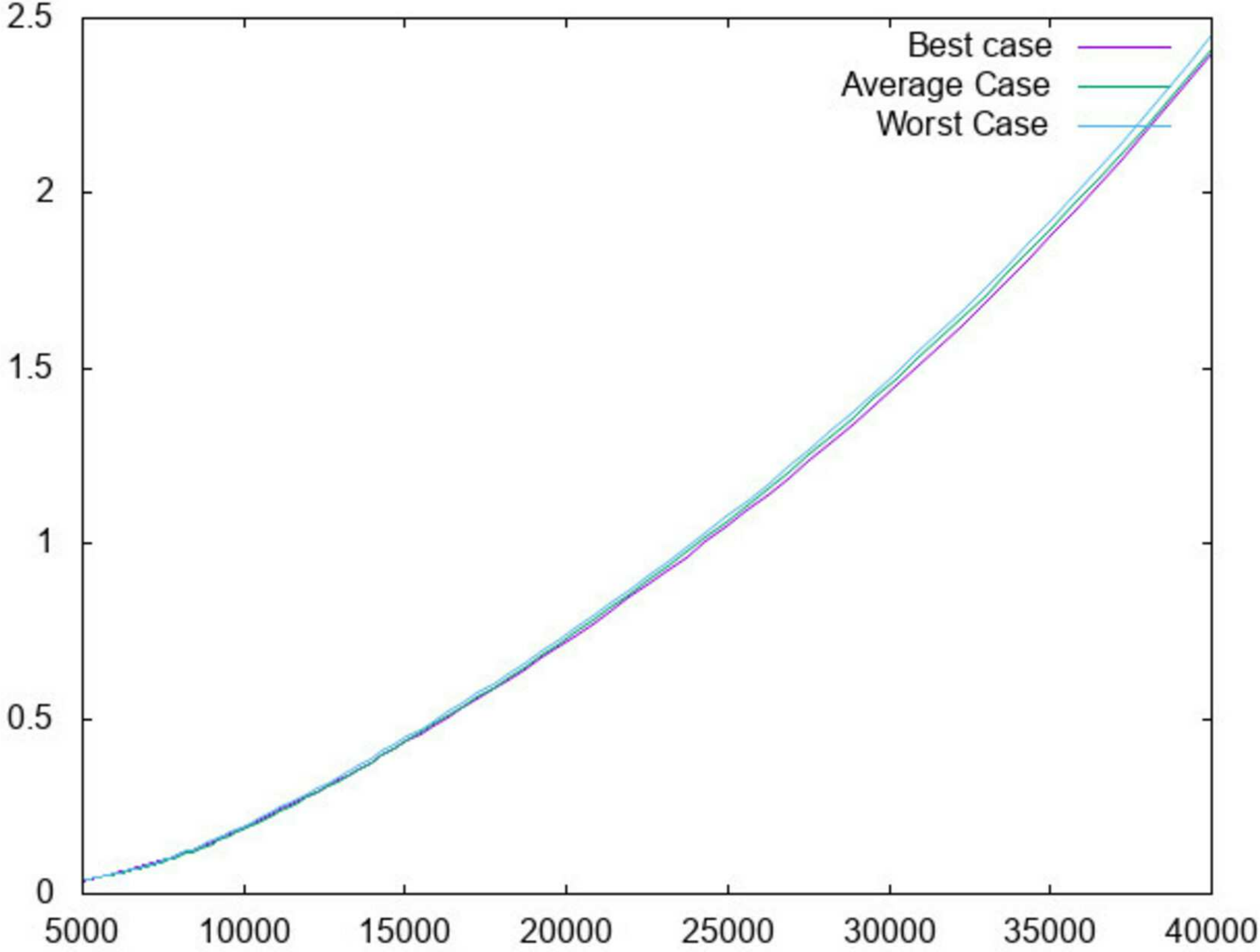
c) Code & explanation in code

d) Computation times were calculated in the code itself. For average case, program was run for 5 times for same n. And then, average time was calculated manually, & graph was plotted via gnuplot. (Graph -> available digitally) (Final data points available too)

e) The data was taken for large numbers n and the graph implies the difference between average, best & worst case is minimal as $n \to \infty$. The cost of selection sort tells us that the number of comparisons $= \sum_{k=2}^{n} k = \frac{n(n+1)}{2}$

It belongs to $O(n^2)$.

The graph clearly demonstrates that the time complexity is quadratic. However, the minimal difference between 3 cases can be reflected to swaps & min assignment.


Best case
Average Case
Worst Case
2.5
2
1.5
1
0.5
0
5000
10000
15000
20000
25000
30000
35000
40000