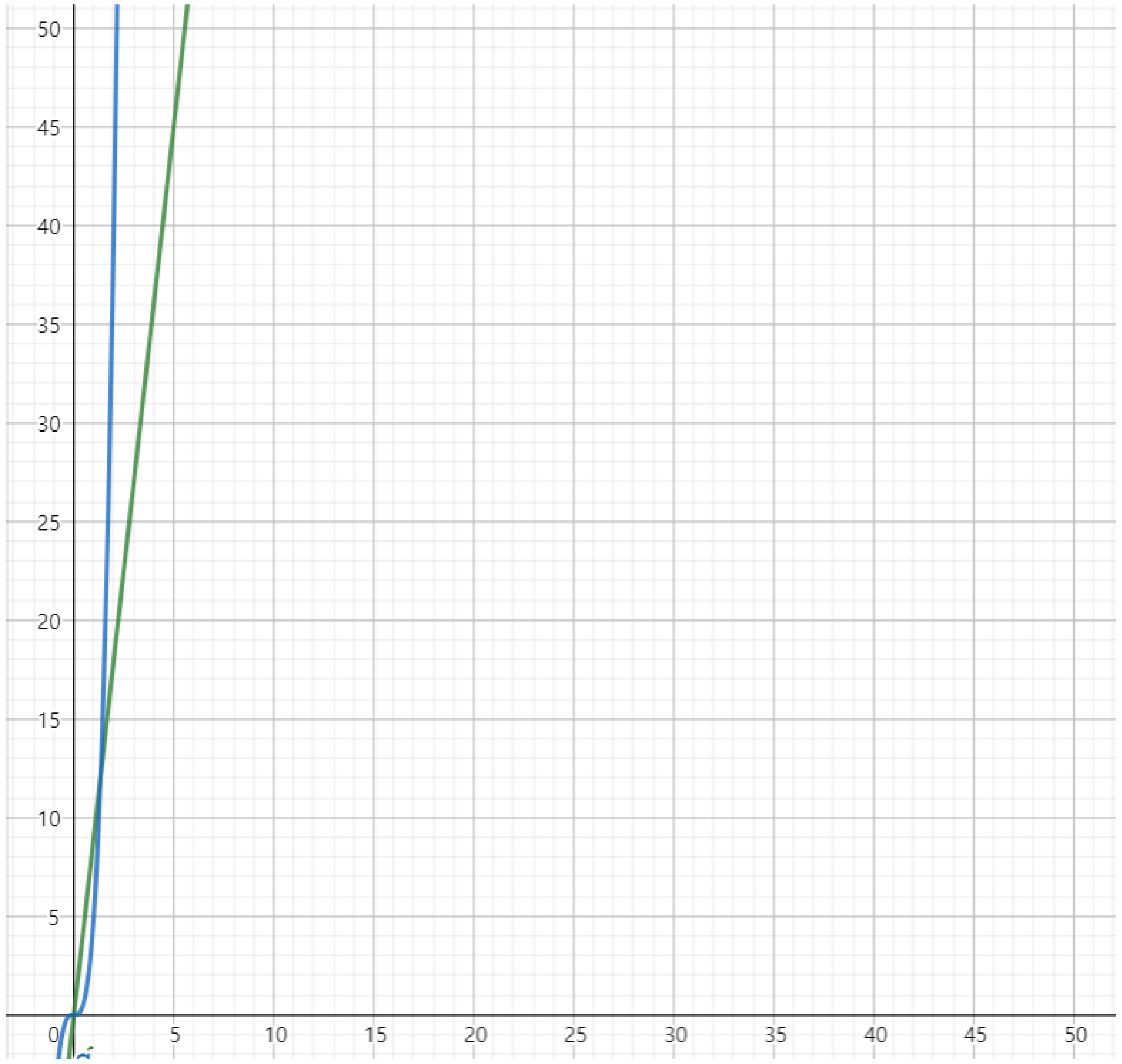
HOMEWORK 3 _ ANSWER SHEET

Name: Dongwook Lee

Problem 3.1 _ Asymptotic Analysis

(a) (2 points) $f(n) = 9n$ and $g(n) = 5n^3$



For all constant $\forall C > 0$, There always exist 3 intersections between f(n) and C*g(n);

$9n = C*5n^3 \Rightarrow C*5n^3 - 9n = 0 \Rightarrow n(C*5n^2 - 9) = 0 \Rightarrow$ **Zeroes** $= -\sqrt{(9/(C*5))}, 0, \sqrt{(9/(C*5))}$

And for $\forall n > n_0$, where n_0 = (our right-most intersection point) $= \sqrt{(9/(C*5))}$;

we can figure out by Test Point Insertion that

- $f(\sqrt{(10/(C*5))}) = 9*\sqrt{(10/(C*5))} = \sqrt{(162/C)}$
- $C*g(\sqrt{(10/(C*5))}) = C*5*(\sqrt{(10/(C*5))})^3 = \sqrt{(200/C)}$

$\Rightarrow f(n) < C*g(n)$ for $\forall C > 0$

$\therefore$ $f \in O(g)$ & $g \in \Omega(f)$

To check whether the bounds are tight or not, We can put limit to the expressions f/g and g/f.

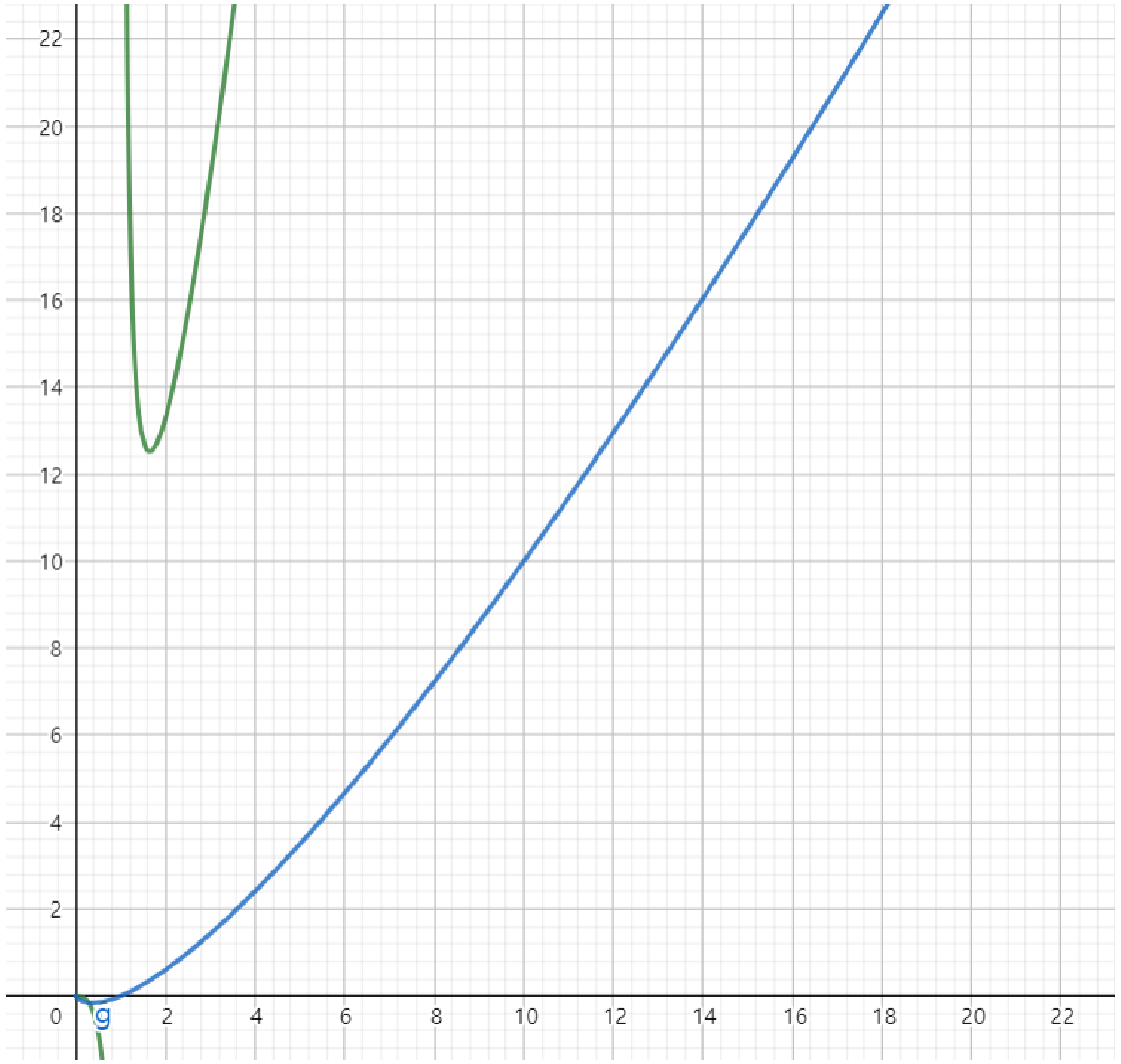
$\lim_{n\to\infty} f(n)/g(n) = \lim_{n\to\infty} (9*n)/(5*n^3) = 0$

g(n) is a non-tight upper bound of f(n) $\therefore$ $f \in o(g)$

$\lim_{n\to\infty} g(n)/f(n) = \lim_{n\to\infty} (5*n^3)/(9*n) = \infty$

f(n) is a non-tight lower bound of g(n) $\therefore$ $g \in \omega(f)$

(c) (2 points) $f(n) = n^2 / \log(n)$ and $g(n) = n * \log(n)$



$\lim_{n\to\infty} f(n)/g(n)$

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

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

$= \infty$

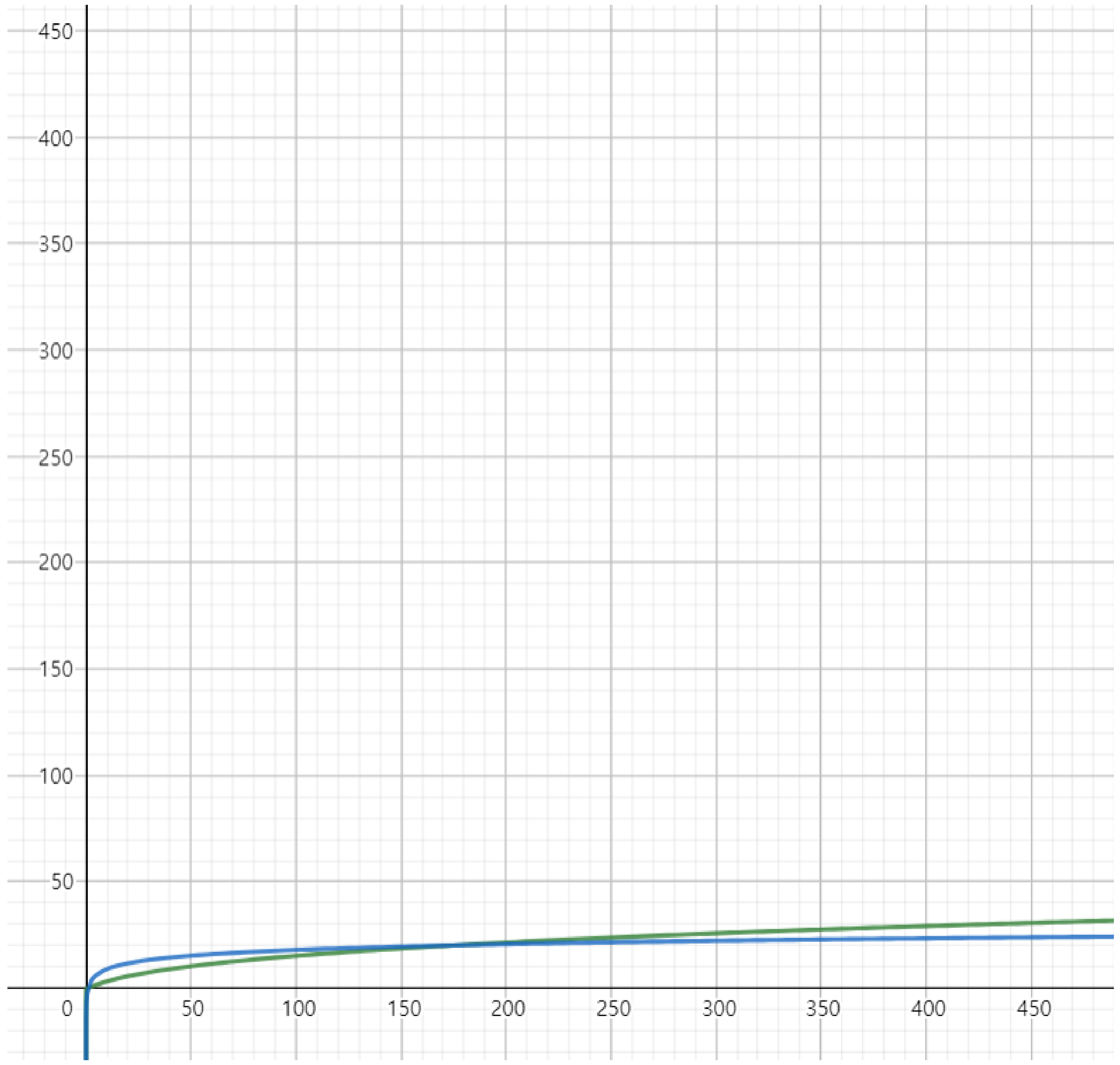
$\therefore f \in \omega(g)$

$\therefore f \in \Omega(g)$

$\therefore g \in o(f)$

$\therefore g \in O(f)$

(d) (2 points) $f(n) = (\log(3n))^3$ and $g(n) = 9*\log(n)$



$\lim_{n\to\infty} f(n)/g(n)$

$= \lim_{n\to\infty} (\log(3n))^3/(9*\log(n))$

$= \infty$

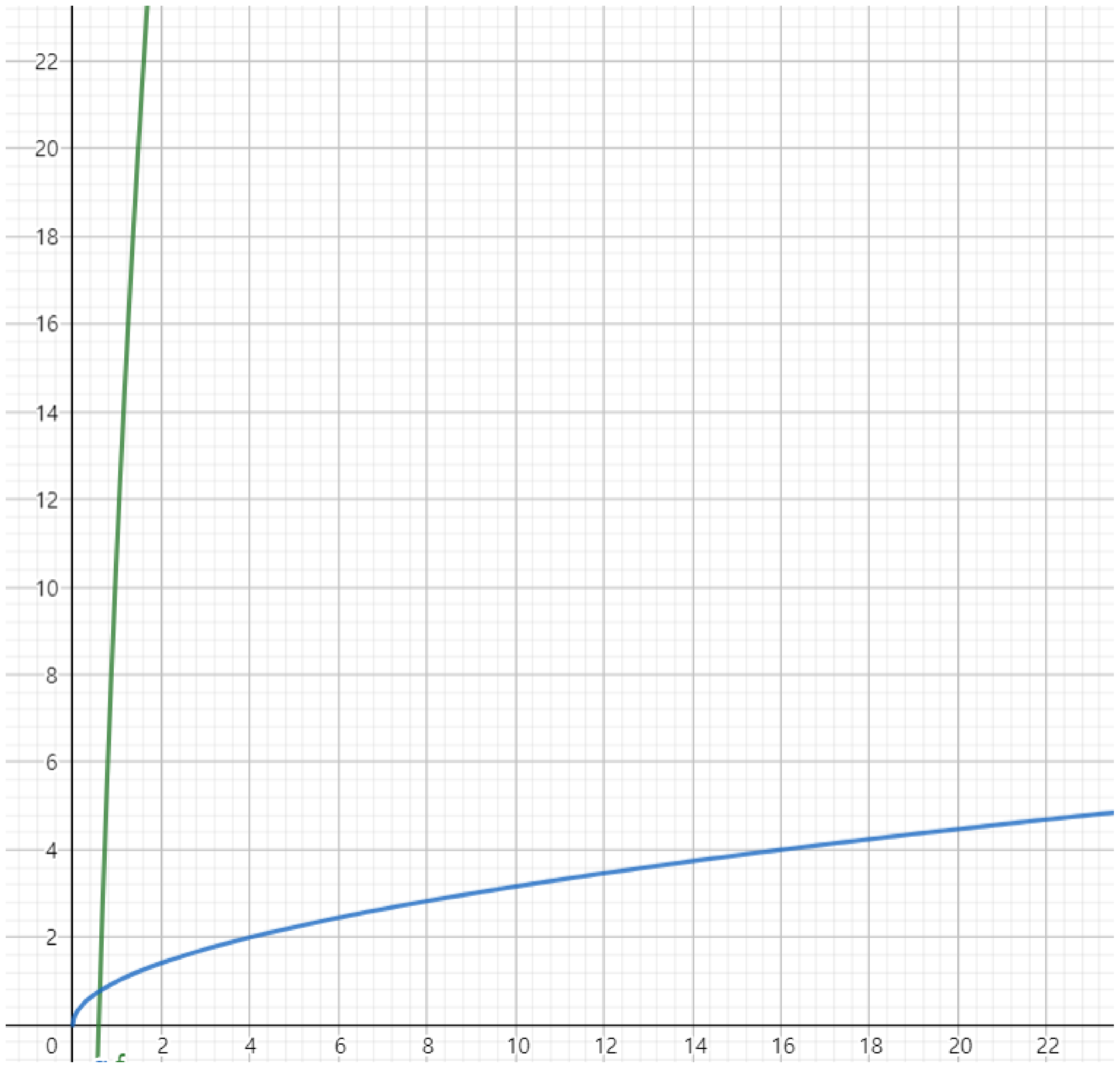
$\therefore f \in \omega(g)$

$\therefore f \in \Omega(g)$

$\therefore g \in o(f)$

$\therefore g \in O(f)$

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



$\lim_{n\to\infty} f(n)/g(n)$

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

$= \infty$

$\therefore f \in \omega(g)$

$\therefore f \in \Omega(g)$

$\lim_{n\to\infty} g(n)/f(n)$

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

$= 0$

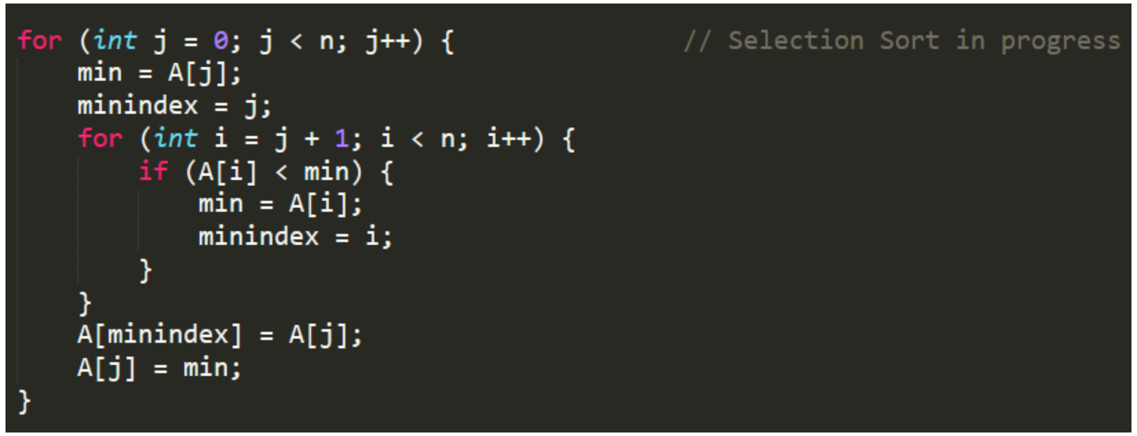
$\therefore g \in o(f)$

$\therefore g \in O(f)$

Problem 3.2 Selection Sort

(a) (2 points) Implement Selection Sort

(b) (3 points) Show that Selection Sort is correct (Hint: consider the loop invariant).



```
for (int j = 0; j < n; j++) {                    // Selection Sort in progress
    min = A[j];
    minindex = j;
    for (int i = j + 1; i < n; i++) {
        if (A[i] < min) {
            min = A[i];
            minindex = i;
        }
    }
    A[minindex] = A[j];
    A[j] = min;
}
```

*n is here the number of elements as well as size of the array A

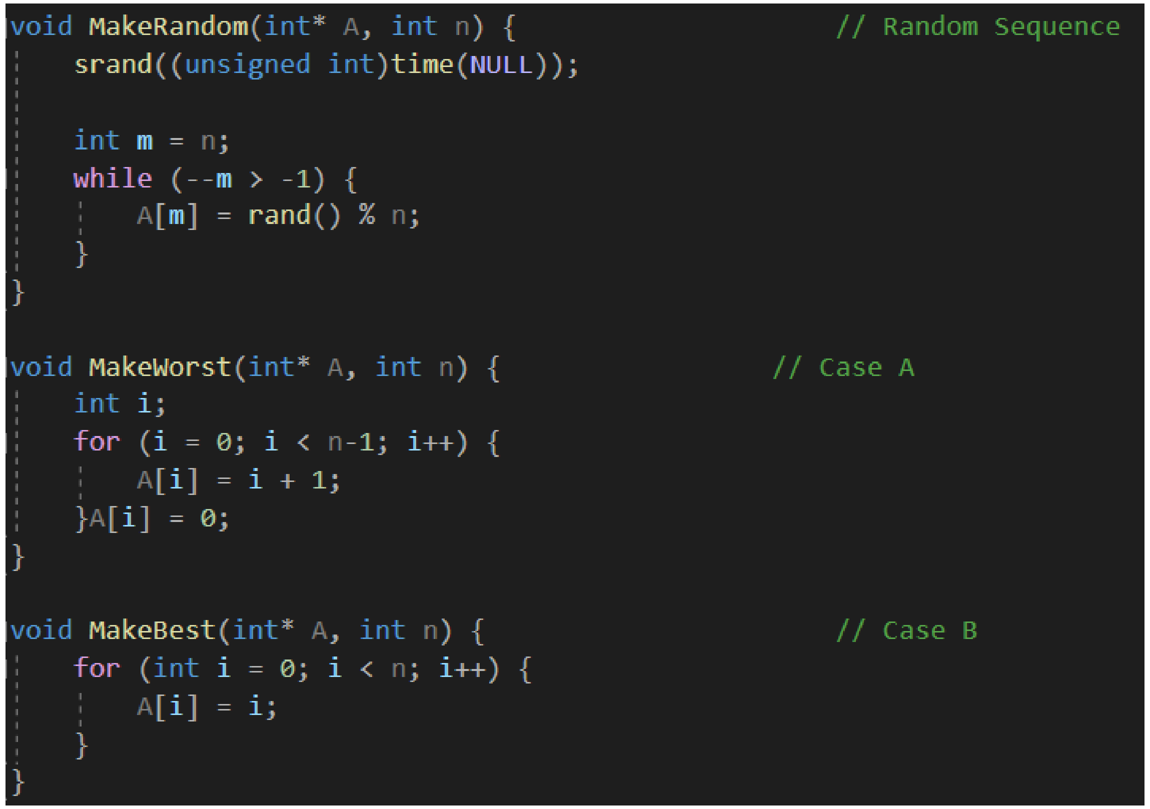
At the beginning and end of each loop iteration, our modified array A consists of same elements with our original array, but in different order.

(c) (3 points) 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).



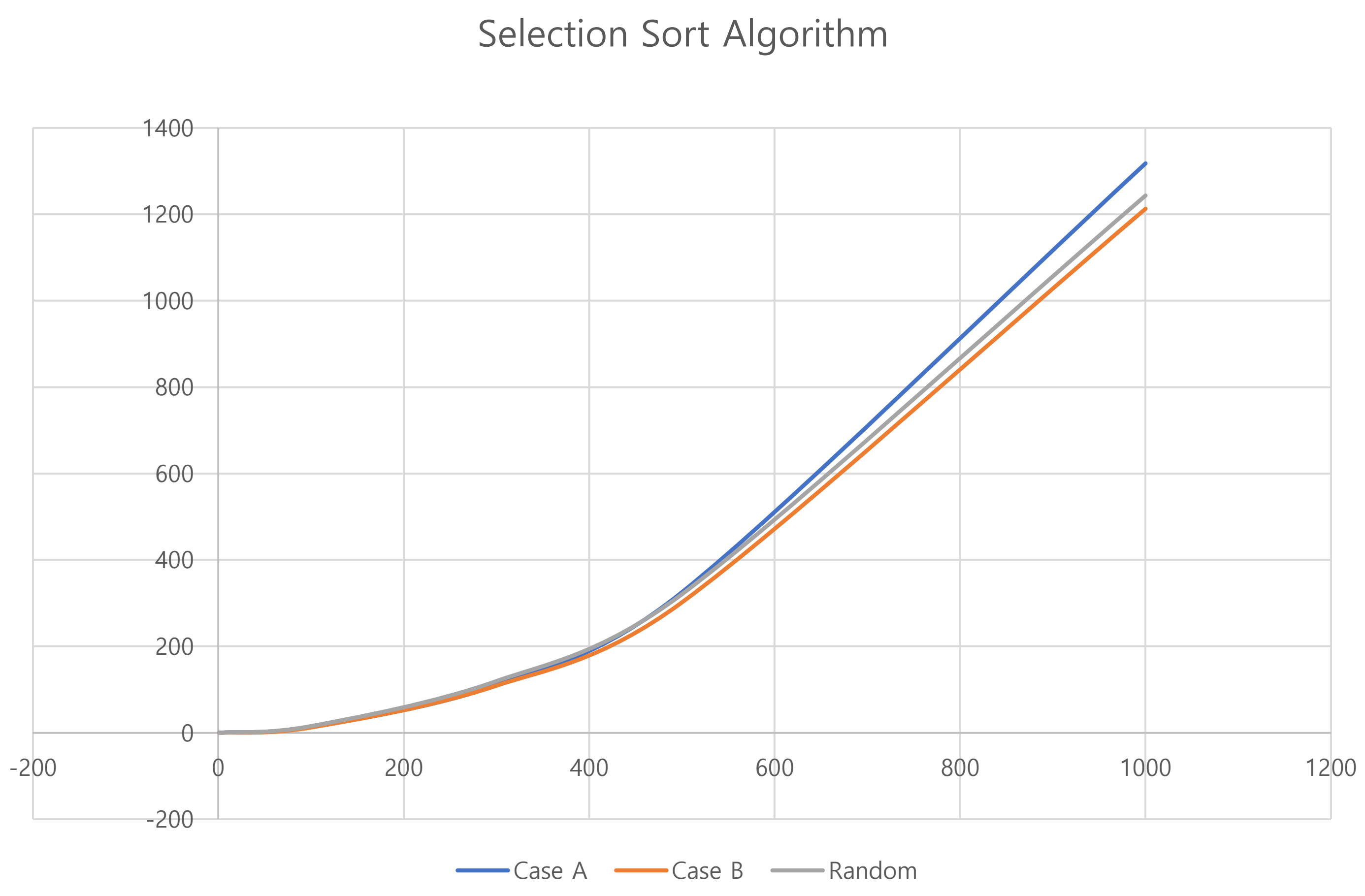
```
void MakeRandom(int* A, int n) {                    // Random Sequence
    srand((unsigned int)time(NULL));

    int m = n;
    while (--m > -1) {
        A[m] = rand() % n;
    }
}

void MakeWorst(int* A, int n) {                 // Case A
    int i;
    for (i = 0; i < n-1; i++) {
        A[i] = i + 1;
    }A[i] = 0;
}

void MakeBest(int* A, int n) {                      // Case B
    for (int i = 0; i < n; i++) {
        A[i] = i;
    }
}
```

To create a random-based sequence, I used a time-based random function srand &rand in c++ programming language.

For Case A, I tried to set every minimum number of such iteration to the end of the array. Thus at the beginning, I put the minimum value to the very end, and others in order that nth minimum goes to the end at (n-1)th iteration.

And for Case B, I just ordered them in ascending order, that there will happen no more swapping in between the elements.

(d) (4 points) 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 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.).



(e) (1 point) Interpret the plots from (d) with respect to asymptotic behavior and constants.

For all of three cases
– worst, best, and randomly created sequence –
the graphs of them seems to be bounded as O(n ^ 2).

And for the bounding constants, the constant for Case A is the largest among those three, and the constant for Case B is the smallest.