Programming and Data Structures

Week 5 Assignment

Reminder: All work must be your own!

Question 1) Use the following selection sort algorithm to answer the questions below:

```
void swap(int *xp, int *yp) {
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

// A function to implement selection sort
void selectionSort(int arr[], int n) {
	// One by one move boundary of unsorted subarray
	for (int i = 0; i < n-1; i++) {
		// Find the minimum element in unsorted array
		int min_idx = i;
		for (int j = i+1; j < n; j++) {
			if (arr[j] < arr[min_idx]) {
				min_idx = j;
				swap(&arr[min_idx], &arr[i]);
			}
		}
	}
}
```

a) Identify the straight-line code in the above algorithm. You can describe or underline.

I have underlined the straight-line code in the above algorithm.

b) Fill in the following table that counts the number of times that the innermost piece of code will be executed.

Iteration #	Value of i	# of executions
1	0	n-1
2	1	n-2
3	2	n-3
…		
n-1	n-2	n-(i+1)
n	n-1	0

c) Sum the last column of the table and simplify as much as you can.

The number of executions of the innermost piece of code is equal to $\sum_{i=1}^{n-1} i$. For example, an array of size 8 would execute the innermost piece of code 7+6+5+4+3+2+1 = 28 times.

d) Based on your answer to **c** what is the runtime of the algorithm?

We know that $\sum_{1}^{n} i = \frac{n*(n+1)}{2}$. Substituting "n-1" for "n" yields $\frac{n(n-1)}{2}$. The expression simplifies to $\frac{1}{2}(n^2 - n)$. Dropping the constant "$\frac{1}{2}$" and the lower-ordered term "n", the runtime of the selectionSort algorithm is $O(n^2)$.

Question 2) For each code snippet, state its runtime in terms of N, you can assume that the '...' represents straight line code.

```
a) for (int i = N; i >= 0; i -= 4) { … }
```

$$\frac{0 - N}{-4} = \frac{1}{4} * N$$

$$Runtime = O(N)$$

```
b) for (int i = 1; i < N; i *= 5) { … }
```

$$Loop\ will\ run\ until\ 5^k \geq N$$
$$\log_5(5^k) = \log_5 N$$
$$k = \log_5 N$$
$$Runtime = O(\log(N))$$

```
c) for (int i = 0; i < N; i++) {
       for (int j = N; j > 0; j /= 2) { … }
   }
```

$$Outer\ loop: \frac{N - 0}{1} = O(N)$$
$$Inner\ loop\ will\ run\ until \frac{N}{2^k} \leq 0$$
$$\log_2(2^k) = k = \log_2 N$$
$$Inner\ loop: O(\log(N))$$

$Since\ the\ two\ loops\ have\ no\ dependencies\ on\ variables, the\ total\ runtime\ is\ the\ product\ of\ the\ runtime\ of\ each\ loop.$

$$Total\ Runtime = O(N * \log(N))$$

```
d) for (int i = 0; i < N; i++) {
       for (int j =N; j > i; j--) { … }
   }
```

Iteration	i	Inner Iterations
0	0	N
1	1	N-1
2	2	N-2
k	k	N-k
N	N	N-N = 0

$$\sum_{k=0}^{N}(N-k) = \frac{N(N+1)}{2} = \frac{1}{2}(N^2+N)$$

$$\textit{Removing the constant and lower ordered term, the runtime is } O(N^2)$$

```
e) for (int i = 1; i < N; i*=2) {
       for (int j = 0; j < i; j++) { … }
   }
```

Iteration	i	Inner Iterations
0	1	1
1	2	2
2	4	4
k	2^k	2^k
$Log_2(N)$	$2^{Log_2 N} = N$	N

$$\sum_{k=0}^{Log_2 N} 2^k = \frac{2^{Log_2 N+1}-1}{2-1} = \frac{2N-1}{1} = 2N-1$$

$$\textit{Removing the constant terms yields a runtime of } O(N)$$

Question 3) For each of the following function pairs (f & g), give an M and x_0 that holds that f(x) ∈ O(g(x)). You do not need to write a proof of such, just state an M and x_0 that the formula holds for. For some M and x0, f(x) <= M * g(x), for all x > x_0. For all questions, it is true that f(x) = O(g(x)).

Hint: Consider setting the two formulas equal and solving for x.

a) f(x) = 100x + 10
 g(x) = 5x

M = 21
X_0 = 2

b) f(x) = 10x
 g(x) = 1/2 x^2

M = 1
$X_0 = 20$

c) $f(x) = 1000x^2$
$g(x) = x^3$

M = 1
$X_0 = 1000$

Question 4) For each of the following function pairs, use the limit rule to determine which of the following options best applies:

i) $f(x) \in O(g(x))$

ii) $f(x) \in \Omega(g(x))$

iii) $f(x) \in \theta(g(x))$

a) $f(x) = 3x^2$
$g(x) = 15x^2$

$$\left(\frac{3x^2}{15x^2}\right) = \frac{3}{15} = \lim_{x\to\infty}\left(\frac{3}{15}\right) = \frac{3}{15} = \frac{1}{5} = C$$

Therefore, iii) $f(x) \in \theta(g(x))$

b) $f(x) = x^2$
$g(x) = 3x^3$

$$\left(\frac{x^2}{3x^3}\right) = \frac{1}{3x} = \lim_{x\to\infty}\left(\frac{1}{3x}\right) = \frac{1}{\infty} = 0 = C$$

Therefore, i) $f(x) \in O(g(x))$

c) $f(x) = \log_2(x)$
$g(x) = \log_3(x)$

$$\lim_{x\to\infty}\left(\frac{\log_2 x}{\log_3 x}\right) = \frac{\infty}{\infty}$$

Apply L'Hopital's Rule.
Say $y = \log_2 x$ which is f(x) (the numerator). Then $2^y = x$.

$$y * \ln(2) = \ln(x)$$

$$\frac{dy}{dx} * \ln(2) = \frac{1}{x}$$

$$\frac{dy}{dx} = \frac{1}{x(\ln(2))}$$

The same can be done for the denominator g(x). Then we have:

$$\frac{\frac{1}{x * \ln(2)}}{\frac{1}{x * \ln(3)}} = \frac{\ln(3)}{\ln(2)} = C$$

Therefore, iii) f(x) ∈ θ(g(x))

d) f(x) = x * log(x)
g(x) = 5x

$$\lim_{x\to\infty}\left(\frac{x\log(x)}{5x}\right) = \frac{\log(x)}{5} = \frac{\infty}{5} = \infty = C$$

Therefore, ii) f(x) ∈ Ω(g(x))

e) f(x) = $2^{\log2(x)}$
g(x) = 2x

$$\lim_{x\to\infty}\left(\frac{2^{\log_2 x}}{2x}\right) = \frac{x}{2x} = \frac{1}{2} = C$$

Therefore, iii) f(x) ∈ θ(g(x))