Programming and Data Structures

Week 5 Assignment

Reminder: All work must be your own!

"What' s big O for this?"

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
		for (int j = i+1; j < n; j++) {
			if (arr[j] < arr[i]) {
				swap(&arr[j], &arr[i]);
			}
		}
	}
}
```

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

```
			if (arr[j] < arr[i]) {
				swap(&arr[j], &arr[i]);
			}
```

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-2	n-3	2
n-1	n-2	1

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

$\sum_{1}^{n-1} i$=1+2+3⋯ (n−3) + (n−2) + (n−1) =1/2 (n−1) *n

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

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) { … }

# of times looping through that loop
O(n)
```

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

O($\log_5 n$)

```
c) for (int i = 0; i < N; i++) {
       for (int j = N; j > 0; j /= 2) { … }
   }
       Inner doesn't depend on i
       O(n*log(n))
```

```
d) for (int i = 0; i < N; i++) {
       for (int j =N; j > i; j--) { … }
   }
Does depend on outer. Show table.
```

Iteration	I	J	# of executions
1	0	N	n-1
2	1	n-1	n-2
3	2	n-2	n-3
4	3	n-3	n-4

O(n^2)

E

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

Does depend on outer. Show table.
I number of executions.
Going up in iterations, get closer to n with the outer loop BC multiplying by 2.
Log involved with *2
Smaller than n*logn

i is always exceeding j; 2^(n-1) vs. n-1

Iteration	I	J	# of executions
1	1	0	1
2	2	1	2
3	4	2	4
4 5 6 … $2^{log2(n)}$	8 16 32 64	3	8 … 2^(n-1)

O($\log_2 n$)

G(x) is always an upper bound for f(x
G(x)*m = f(x)
Set equal and solve for x. continue to get solution.

Question 3) For each of the following function pairs (f & g), give an M and x_0 that holds that f(x) $\in$ 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

Answer:

X_0= 1

m=22

100x+10 <=110x for all x > 1

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

answer:

X_0= .1

m=200

10x<=100x^2 for all x > .1

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

answer:

M= 100,000

X_0= .01

1000x^2 <= 100,000x^3 for all x >.01

Either big O, big omega, or all three.

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

1/5 which is between 0 and infinity;

$f(x) \in O, \ \Omega, \text{and} \ \theta(g(x))$

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

1/(3x) which goes to 0;

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

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

from change of base formula, $\frac{\frac{\log(x)}{\log(2)}}{\frac{\log(x)}{\log(3)}} = \frac{\log(x)}{\log(2)} * \frac{\log(3)}{\log(x)} = \frac{\log(3)}{\log(2)} \approx 1.5849$; between 0 and infinity

$f(x) \in O, \ \Omega, \text{and} \ \theta(g(x))$

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

log(x)/5 which goes to +infinity; $f(x) \in \Omega(g(x))$

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

½ which is a constant greater than 0; $f(x) \in O, \ \Omega, \text{and} \ \theta(g(x))$