Homework 1

Yousef Suleiman | Due: Jan 23

Question 1

$$42n \lg n < 6n^2$$
$$42(1) \lg (1) = 0 < 6(1)^2 = 6$$
$$42(2) \lg (2) = 84 \not< 6(2)^2 = 24$$
$$42(10) \lg (10) = 1395.5 \not< 6(10)^2 = 600$$
$$42(20) \lg (20) = 3630.4 \not< 6(20)^2 = 2400$$
$$42(30) \lg (30) = 6182.7 \not< 6(30)^2 = 5400$$
$$42(40) \lg (40) = 8940.8 < 6(40)^2 = 9600$$

Excluding $n = 1$, the inequality doesn't hold until n is more than some value between 30 and 40.

$$42(35) \lg (35) = 7540 \not< 6(35)^2 = 7350$$
$$42(37) \lg (37) = 8095.5 < 6(37)^2 = 8214$$
$$42(36) \lg (36) = 7816.9 \not< 6(36)^2 = 7776$$

The inequality doesn't hold for values of n in $[1, 36]$ meaning insertion sort beats merge sort for input sizes between 1 and 36.

Question 2

(1)

`count` will increment as follows:

$$\text{count} : [1, 1 * 3, 3^2, 3^3, 3^4, \ldots, 3^k]$$

where $3^k \geq n$ and k would be the number of times the statement is executed.

$$k = \log_3 n$$
$$T(n) = O(\log_3 n)$$

(2)

For this problem, I am going to assume the initial assignment is `j = 1` as `j = 0` will result in an infinite loop.

Focusing on the inner loop, `j` will increment as follows:

$$j : [1, 1 * 2, 2^2, 3^3, \dots, 2^k]$$

where $2^k \geq n$ and $k = \lg n$ such that the inner loop has a time complexity of $O(\lg n)$. The outer loop will have a time complexity of $O(n * \lg n)$ as it runs the inner loop n times.

$$T(n) = O(n \lg n)$$

(3)

The inner loop can only execute once each iteration of the outer loop as the initial value of `j = i` and the condition is `j ⩽ i` so when `j++` is executed, the inner loop breaks after its first iteration.

The outer loops iterates $O(n)$

$$T(n) = O(n)$$

Question 3

	n	n^2	2^n
n^n	Ω	Ω	Ω
$n \log n^4$	Ω	O	O
$(n-2)!$	Ω	Ω	Ω
$2^{\log n}$	O	O	O

Explanation

- n^n grows faster than all 3
- $n \log n^4$ grows faster than only n
- $(n-2)!$ grows faster than all 3
- $2^{\log n}$ grows slower than all 3

Question 4

```
1  Merge(A, p, q, r)
2      n1 = q - p + 1
3      n2 = r - q
4      let L[1 .. n1] and R[1 .. n2] be new arrays
5
6      /* copy A subarrays to L and R */
7      for i = 1 to n1
8          L[i] = A[p + i - 1]
9      for j = 1 to n2
10         R[j] = A[q + j]
11
12     /* compare into A */
13     l = 1
14     r = 1
15     a = 1
16     while l < length(L) and r < length(B)
17         if L[l] < R[r]
18             A[a] = L[l]
19             l++
20         else
21             A[a] = R[r]
22             r++
23         a++
24
25     /* this will only run if either L or R have any left */
26     while l < length(L)
27         A[a] = L[l]
28         l++
29         a++
30     while r < length(R)
31         A[a] = R[r]
32         r++
33         a++
```