Contents:

Part 1.1

1. True
2. False
3. False
4. True
5. False
6. True
7. False
8. True
9. False
10. True
11. False
12. True
13. True
14. True
15. True

Part 1.2 Explanations:

11. False

Limit Test:

$$\lim_{n \to \infty} \frac{\sqrt{n}+n}{nlogn} => \frac{\frac{n^{0.5}}{n}+1}{logn} => \frac{1}{\sqrt{n}logn}$$

$$\lim_{n \to \infty} \frac{1}{\sqrt{n}logn} \;=\; 0 \neq constant$$

Alternatively:

Since n logn grows faster than n and $\sqrt{n}$, there is no n_0 *and* c that will satisfy the equation:

$f(n) \geq c\, g(n)$

12. True:

For O(n^3)

Pick c = 1000; n$_o$ = 1

$100n^3 + n^2 \leq 1000n^3 \quad \forall n \geq 1$

For Ω (n^3)

Pick c = 1; n$_o$ = 1

$100n^3 + n^2 \geq n^3 \quad \forall n \geq 1$

13. True:

Note that $\frac{n^2}{n} = n$

For O(n)

Pick c = 2; n$_o$ = 1

$\frac{1}{n} + n \leq 2n \quad \forall n \geq 1$

For Ω (n)

Pick c = 0.5; n$_o$ = 1

$\frac{1}{n} + n \;\geq\; \frac{n}{2} \quad \forall n \geq 1$

14. True:

Note that log 16 = 4 (base 2)

For O(n)

Pick c = 5; n$_o$ = 1

$\frac{1}{n^{100}} + 4 \leq 5 \quad \forall n \geq 1$

For Ω (n)

Pick c = 1; n$_o$ = 1

$\frac{1}{n^{100}} + 4 \;\geq\; 1 \quad \forall n \geq 1$

15. True

Note that $logn^2 = 2\ logn$

For O(5n)

Pick c = 100; $n_o = 1$

$n + 2\ logn \leq 500n \quad \forall n \geq 1$

For Ω (5n)

Pick c = 0.1; $n_o = 1$

$n + 2\ logn \geq 0.5n \quad \forall n \geq 1$

Part 2

1)

```
num=0;
for (i = 0; i>= 100n; i++)
        num++;
```

Line 1 assignment = 1 instruction for num

Loop: = 100n iterations, for loop instruction and num increment per loop (2 instructions per loop)

Loop fail = 1 instruction

Total: T(n) $= 1 + \sum_{i=1}^{100n} 2 + 1$

$= 2 + 200n$

$= O(2 + 200n)$

$= O(n)$

2)

```
 num=0;
for (i = n*n*n; i<= 0; i=i-4)
        Num++;
```

Line 1 assignment = 1 instruction for num

Loop: = $\frac{n*n*n}{4}$ iterations, for loop instruction and num increment per loop (2 instructions per loop)

Loop fail = 1 instruction

Total: T(n) $= 1 + \sum_{i=1}^{\frac{n*n*n}{4}} 2 + 1$

$= 2 + \frac{n^3}{2}$

$= O(2 + 0.5n^3)$

$= O(n^3)$

3)

```
 num=0;
```

```
for (i = n; i<= 0; i=i/2)
        num = num + n;
```

Line 1 assignment = 1 instruction for num

Loop: $log_2 n$ iterations, for loop instruction and num increment per loop (2 instructions per loop)

Loop fail = 1 instruction

Total: T(n) $= 1 + \sum_{i=1}^{log_2 n} 2 + 1$

$= 2 + 2 \log n$

$= O(2+2 \log n)$

$= O(\log n)$

4)

```
num=0;
for (i = 1; i<=100; i++)
        for (j = 1; j<=10000; j=j*2)
                num = num + i;
```

Line 1 assignment = 1 instruction for num

Outer Loop: 100 iterations, 1 for loop instruction per iteration, 1 instruction if loop fails

Inner Loop: $log_2 10000$ iterations

For loop instruction + num increment per loop (2 instructions per iteration)

Inner loop fail = 1 instructions

Total: T(n) $= 1 + \sum_{i=1}^{100} (\sum_{j=1}^{log_2 10000} 2) + 1$

= 1 + some constant

= O(some constant)

= O(1)

5)

```
num=0;
for (i = 1; i<=n; i++)
        for (j = 1; j<=i; j++)
                num = num + i;
```

Line 1 assignment = 1 instruction for num

Outer Loop: n iterations, 1 for loop instruction per iteration, 1 instruction if loop fails

Inner Loop: i iterations

For loop instruction + num increment per loop (2 instructions per iteration)

Inner loop fail = 1 instructions

Total: T(n)

$$= 1 + \sum_{i=1}^{n}(\sum_{j=1}^{i} 2) + 1$$

$$= 2 + 2(1 + 2 + 3 + 4 + \ldots + n)$$

$$= O(2 + 2(\frac{(n+1)n}{2}))$$

$$= O(n^2 + n)$$

$$= \mathbf{O(n^2)}$$

6)

```
num=0;
for (i = 1; i<=n; i=i*2)
        for (j = 1; j<=n; j=j+4)
                num = num + i;
```

Line 1 assignment = 1 instruction for num

Outer Loop: log n iterations, 1 for loop instruction per iteration, 1 instruction if loop fails

Inner Loop: n/4 iterations

For loop instruction + num increment per loop (2 instructions per iteration)

Inner loop fail = 1 instructions

Total: T(n)

$$= 1 + \sum_{i=1}^{logn}(\sum_{j=1}^{\frac{n}{4}} 2) + 1$$

$$= 2 + \sum_{i=1}^{logn}(\frac{n}{2})$$

$$= O(2 + (\frac{n}{2}\,logn))$$

$$\mathbf{= O(n\ logn)}$$

7)

```
for (i = 1; i<= 2*n; i++)
        num++;
for (j = 0; j<=n*n; j++)
        for (i=0; i<= n; i++)
                num++;
```

Loop1: 2n iterations, 1 for loop instruction per iteration, 1 instruction inside loop

Loop2:

Outer: n^2 iterations

Inner: n iterations

For loop instruction + num increment per loop (2 instructions per iteration)

Inner loop fail = 1 instructions

Total: T(n) $= \sum_{i=1}^{2n} 2 + \sum_{i=1}^{n^2} (\sum_{j=0}^{n} 2) + 1$

$= 4n + 2n^3$

$= O(2n^3 + 4n)$

= O(n^3)

8)

```
num=0;
        for (i = 0; i<n-1; i++)
                for (j = 0; j< i * i – 1; j++)
                        num = num + i
```

Loop:

Outer: n-1 iterations

Inner: i^2 iterations

For loop instruction + num increment per loop (2 instructions per iteration)

Inner loop fail = 1 instructions

Intuition: we can think of this loop as a triple nested for loop, with j = 1 to i, and k = 1 to i. (we'll drop the constants for the sake of convenience)

We see that the pattern looks like $(1^2 + 2^2 + 3^2 + \ldots + n^2) = \sum_{a=1}^{n} a^2 = \frac{n(n+1)(2n+1)}{6}$

https://trans4mind.com/personal_development/mathematics/series/sumNaturalSquares.htm if you're interested in how

Total: T(n) $= 1 + \frac{n(n+1)(2n+1)}{6}$

= O(n^3)

9)

```
 for (i = n-1; i > 0; i––) {
        MPos = i;
        for (j = 0; j < i; j++) {
                if (a[j] > a[MPos])
                        MPos = j;
        }
        swap(i, MPos); // has three instructions
}
```

Loop Outer: n -1 iterations

Does a swap worth 3 instructions every time

Inner: i iterations

For loop instruction + assignment/if check per loop (3 instructions per iteration)

Total: T(n) $= 1 + \sum_{i=1}^{n-1}(3 + \sum_{j=0}^{i} 3)$

$= 1 + \sum_{i=1}^{n-1} 3 + \sum_{i=1}^{n-1}\sum_{j=1}^{i} 3$

$= 1 + 3(n-1) + 3(\frac{n(n-1)}{2})$

$= O(3n - 2 + \frac{3(n^2-n)}{2})$

= **$O(n^2)$**

10)

```
for (i = n; i > 0; i = i/2) {
        for (j = i; j > 0; j = j/2) {
                //constant time operations
        }
}
```

Outer: log n iterations

Inner: log n iterations

For loop instruction + some constant operation (2 instructions per iteration)

Total: T(n) = **O(log n * log n)**

Part 3: LinkedList and Sorted ArrayList

1) Reversing the list
    a) ArrayList: Ω (n)
        i) Iterate from start to middle, and swap elements with the end, takes n/2
    b) LinkedList: Ω (n)
        i) Can be done with 2-3 pointers (based on implementation), just reversing the prev and next for each node. Update the head at the end
2) Adding Value to the end of the list
    a) ArrayList: Ω (n)
        i) Assume worse case, need to increase the size of the array, so create a new array and move everything over
    b) LinkedList: Ω (n)
        i) There is no tail in the Doubly LinkedList so we need to iterate to the end before we can add.
3) Removing the value from the list at a given index
    a) ArrayList: Ω (n)
        i) Need to shift everything over
    b) LinkedList: Ω (n)
        i) Need to iterate to that given index
4) Removing the first value from the List
    a) ArrayList: Ω (n)
        i) Need to shift everything over
    b) LinkedList: Ω (1)
        i) Remove from head is constant, just move head to head.next and remove the links to the old head
5) Determining whether the list contains some value v
    a) ArrayList: Ω (log n)
        i) Use binary search to find the value since the array is sorted
    b) LinkedList: Ω (n)
        i) Need to iterate through, since there is no efficient way to do a binary search

Part 4

Plot for LinkedList should be a linear search, with roughly linear time Ω (n)
Plot for ArrayList should be a binary search, with roughly Ω (log n) time