CSC345 Discussion 4

Going over the quiz, Big O

Problem 1

Problem 1. (6 points) Determine the truth value of the following statements if the domain of each variable consists of all integers $\mathbb{Z}$

(a) $\forall x \exists y \quad x + y = 1$

(b) $\forall x \exists y \quad x + y = 0$

(c) $\exists x \forall y \quad x + y = 0$

(a) True

(b) True

(c) False (Suppose y = 1 − x then x + y = x + 1 − x = 1 != 0)

Problem 2

Problem 2. (4 points) $O(n + n^2)$ has a _____ runtime complexity.

a. linear-quadratic b. exponential c. quadratic

Answer: C

Why?

First recall the definition of Big O

For $\mathbf{T}(n)$ a non-negatively valued function, $\mathbf{T}(n)$ is in set $\mathrm{O}(f(n))$ if there exist two positive constants c and n_0 such that $\mathbf{T}(n) \leq cf(n)$ for all $n > n_0$.

So,

$n^2 + n = O(n^2)$ Let $n_0 = 1$

We want to show that $n^2 + n \leq c * n^2$

So we have,

$n^2 + n \leq n^2 + n^2 = 2(n^2)$

Thus,

let $c = 2$

Problem 3

Problem 3. (10 points) In Java, the ArrayList resizing formula is $NewCapacity = 1.5 \times OldCapacity$. The capacity of a given ArrayList is N, what is the best and worst case of time complexity when adding a new element to this ArrayList? Use Θ notation for your answer.

Answer:

Best Case: $\Theta(1)$
Worst Case: $\Theta(n)$

Problem 4

Problem 4. (10 points) We want to print elements of a linked list in reverse order in linear time. Show pseudo code for your answer and justify your time complexity.

```
stk = stack()
while(node != null) {
        stk.push(node)
        Node = node.next
}
while(!stk.isEmpty())
        print(stk.pop())
```

Definitions

Ω(n)

For $\mathbf{T}(n)$ a non-negatively valued function, $\mathbf{T}(n)$ is in set $\Omega(g(n))$ if there exist two positive constants c and n_0 such that $\mathbf{T}(n) \geq cg(n)$ for all $n > n_0$.[1]

O(n)

For $\mathbf{T}(n)$ a non-negatively valued function, $\mathbf{T}(n)$ is in set $\mathrm{O}(f(n))$ if there exist two positive constants c and n_0 such that $\mathbf{T}(n) \leq cf(n)$ for all $n > n_0$.

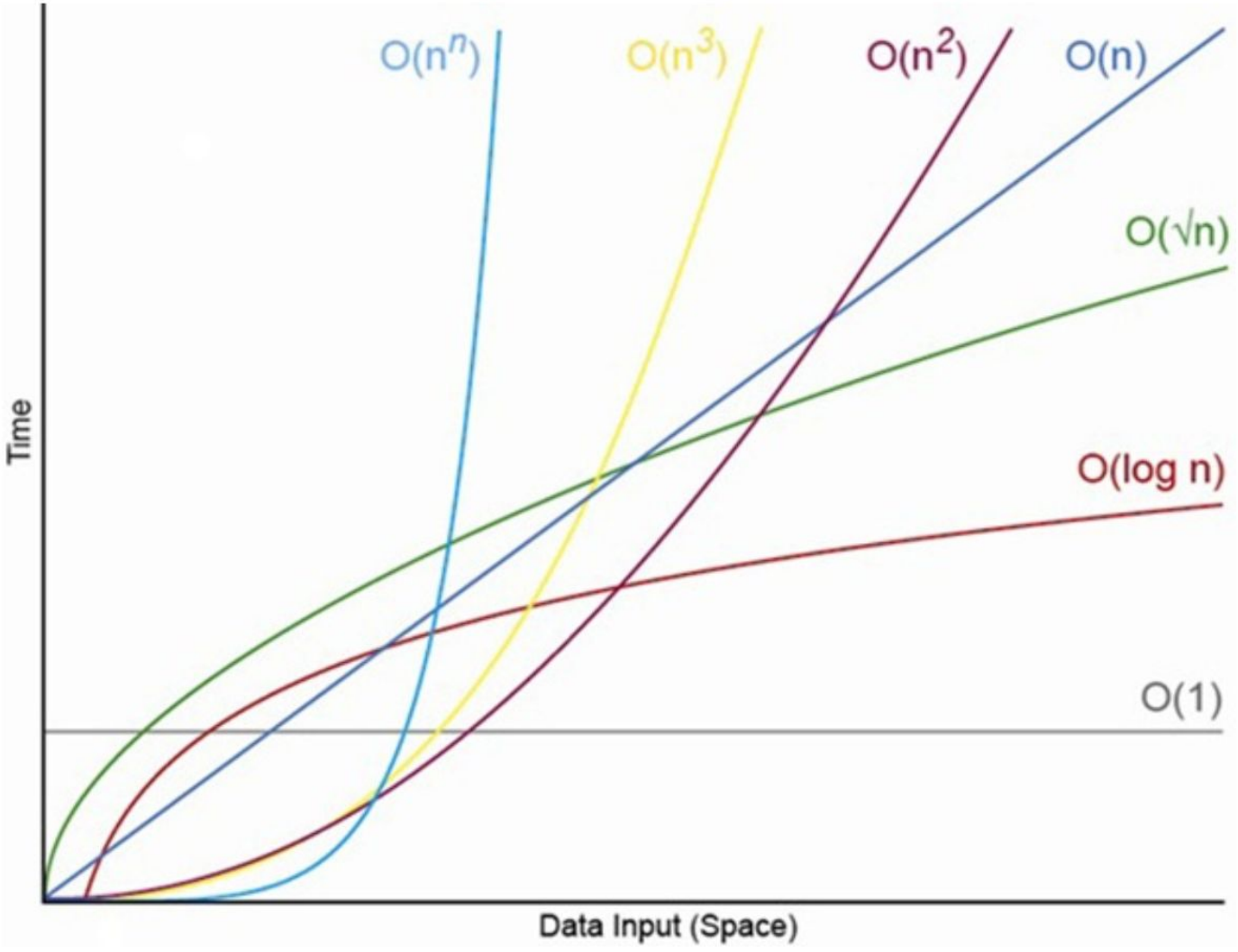
What about Θ

$\Theta(n)$

When the upper bound and the lower bound are the same by a constant factor. Thus, an algorithm is said to be $\Theta(h(n))$ if it is in $O(h(n))$ and it is in $\Omega(h(n))$.

What we are looking for is the ***growth rate*** of the algorithm. Their is a change in cost as the input size changes.

This is different then best and worst case. Best and worst case deal with a definitive input size.


O(n^n)
O(n^3)
O(n^2)
O(n)
O(√n)
O(log n)
O(1)
Time
Data Input (Space)

Practice

Calculate the running time for

num = 1

Assignment is an $\Theta(1)$ operation.

Practice

Calculate the running time for

```
sum = 0
for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
                sum++;
```

This is an $\Theta(n^2)$ operation.

Practice

Calculate the running time for

```
sum = 0
for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
                sum++;
sum2 = 0
for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
                sum2++;
```

This is an Θ(n^2) operation.