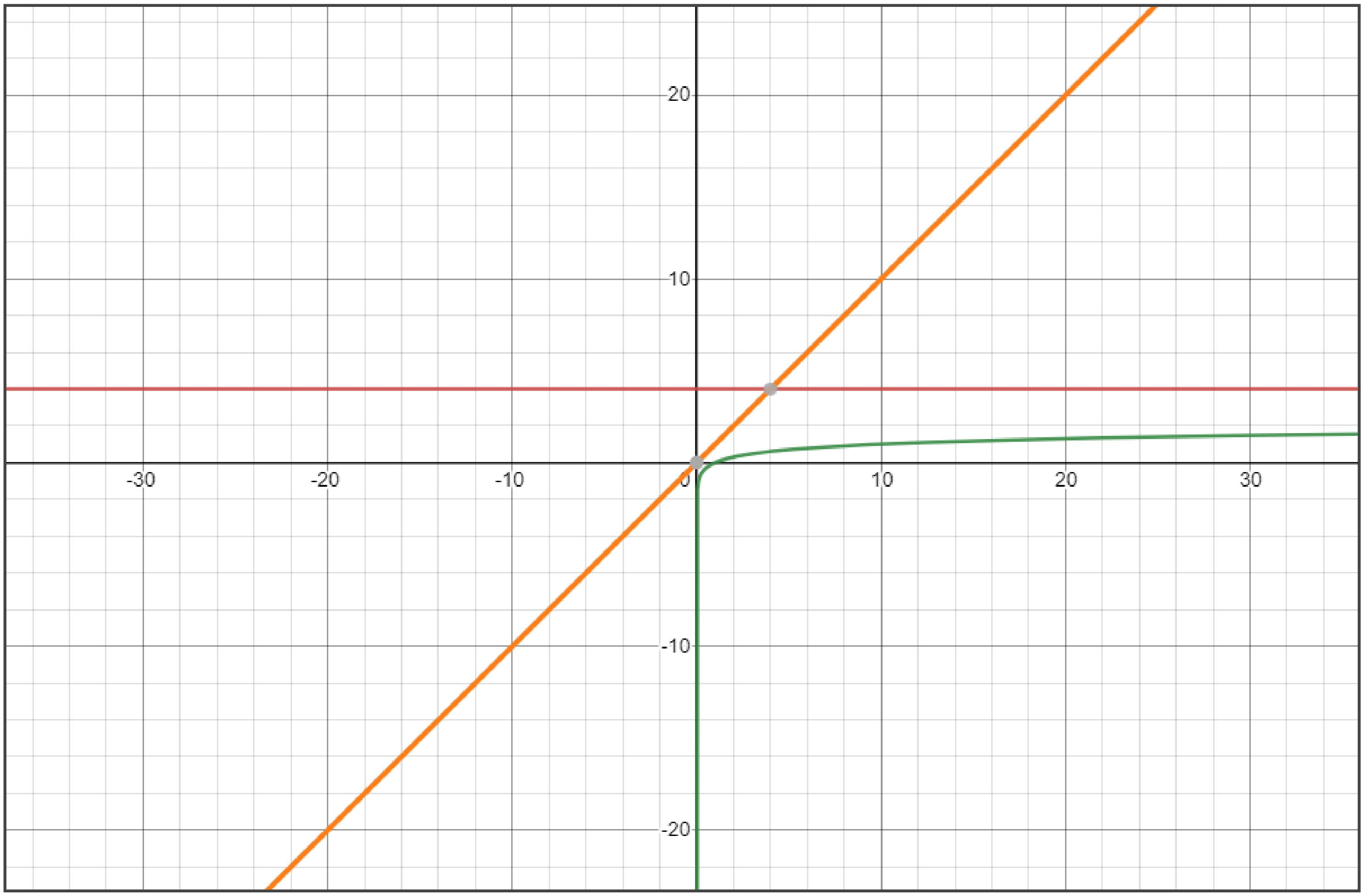
Enter Note Title

Algorithm Complexity:

TIME COMPLEXITY

1.Time taken by program to execute is never called Time complexity
2.Time complexity is a function that gives us relationship about the time as size of input grows.
3.Time always varies from machine to machine
4.Always look for worst case complexity.
5.We always care about large size of data(input).
6.Lets analyise by graphs:
$f(t) = c$,$f(t) = t$,$f(t) = \log(t)$;



1

$y = x$

Enter Note Title

Skip constants

e.g $O(8N^3) = O(N^3)$.

9.Parameters to represent complexity:

$BIG - OH(O)$:

It means algorithm time complexity does not exceed upper bound

e.g $O(N^3)$ means worst case complexity does not exceed N^3.

Lets say we have a function $f(n) = O(g(n))$.

$$\therefore \lim_{n->\infty} = \frac{f(n)}{O(g(n))} < \infty$$

e.g let $f(n) = (3N^4 + 2N^2)$.

$O(N) = N^4$.

$$\lim_{n->\infty} \frac{3N^4 + 2N^2}{N4}$$

$= 3$

Big Ω(omega):

Means TIme complexity never less than lower bound

e.g if complexity is $o(N^3)$.

It means complexity in algorithm is never below than $N^{3.}$.

θ(Theta Notation):

Means both upper bound and lower bound is same

or $0 < \lim_{n->a} \frac{f(n)}{g(n)} < \infty$.

Little o(o):

It means complexity is strictly lower than upper bound

e.g $o(N^3)$ means complexity is always lowers than $O(n^3)$.

It differes from bigo in a ways that bigo is not strictly lesser than upper bound it may be equal also.

Here, $\lim_{n->\infty} \frac{f(n)}{g(n)} = 0$.

TRY example.

Little omega ω:

It means if $f(n) = \omega(g(n))$

then $f > g$

or it means complexity will be strictly graeter than lower bound.

Space complexity:

It means extra space used by an algorithm

e.g we require any array or vector or any map ,set etc during program .we create extra space

consider the following program:

$for(int\ i = 0;i < n;i++)n\ times$

$for(int\ j = 1;j <\ = k;j++)ktimes$

$k++;$

HERE,

$\text{T.C} = O(N \times K);$

S.C: $O(K)$.

We always prefer use of BIG – oh in expressing complexity.