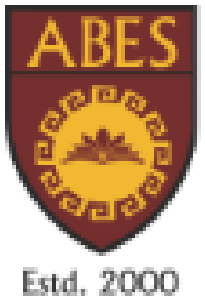


ABES
Engineering College
College Code-032

Data Structure Training

Algorithm & Complexity

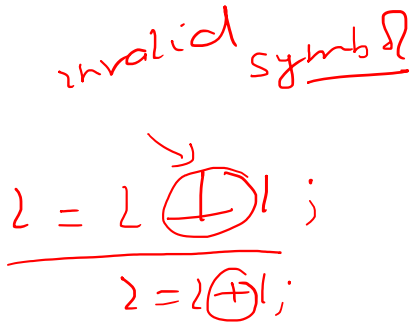
Department of CSE, CS & IT

ABES Engineering College, Ghaziabad

Data Structure Training(Algo&Complexity)

Algorithm-

1-Finite set of steps to solve a problem is called algorithm.
2-Here steps means instruction which contain fundamental operators.



Characteristics Of fundamental Instruction-

1-Definiteness-

Every fundamental instruction must be definite without any ambiguity.

```
i=i  1
i=i+1;
```

2-Finiteness-

Every instruction must be terminate within finite amount of time.

```
i=1;
while(1)
{
    i=i+1;//not allowed in algorithm since executing infinitely.
}
```

Infinite loop

break;

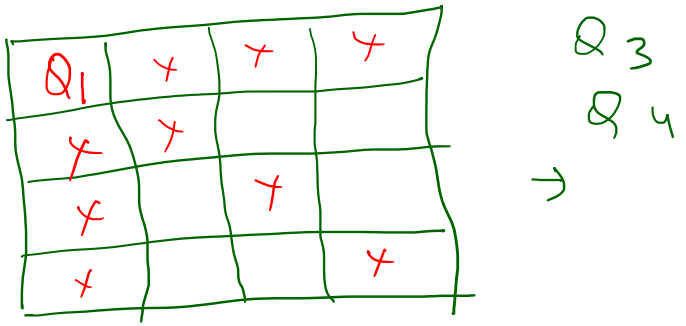
3- Every instruction must accept atleast 0 input and provides atmost 1 output.

Data Structure Training(Algo&Complexity)

Steps For Solving Any Problem-

1-Identifying Problem Statement

4 Queen Problem



2-Identifying Constraints

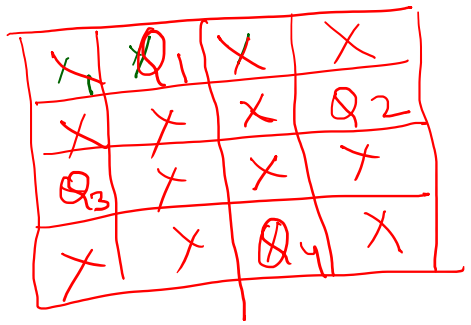
Data Structure Training(Algo&Complexity)

Steps For Solving Any Problem-

3-Design Logic-

Depends on the characteristics of the problem we can choose any one of the following design strategy for design logic.

a)Devide & Conquer
b)Greedy Method
c)Dynamic Programming
d)Branch & Bound
e)Backtracking etc……..



4-Validation-

Most of the algorithm validated by mathematical induction.

5-Analysis-

Process of comparing two algorithms with respect to time,space,number of register,network bandwidth etc is called analysis.

Data Structure Training(Algo&Complexity)

Types Of Analysis-

Priory Analysis-

1-Analysis done before executing

X=x+1;

2-Principle-

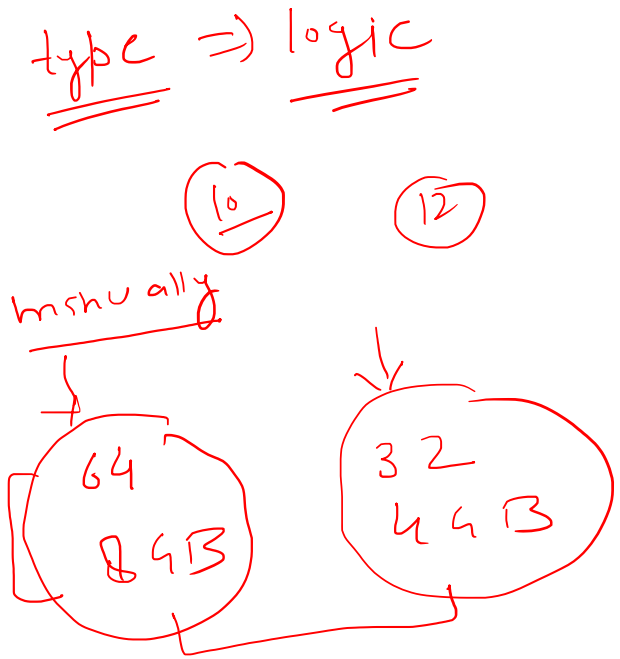
Frequency count of fundamental instruction.

Since x=x+1 being carried out only one time so its complexity os O(1) order of 1

3-It provides estimated values.

4-It provides uniform values.

5-It is independent of CPU,O/S & system architecture



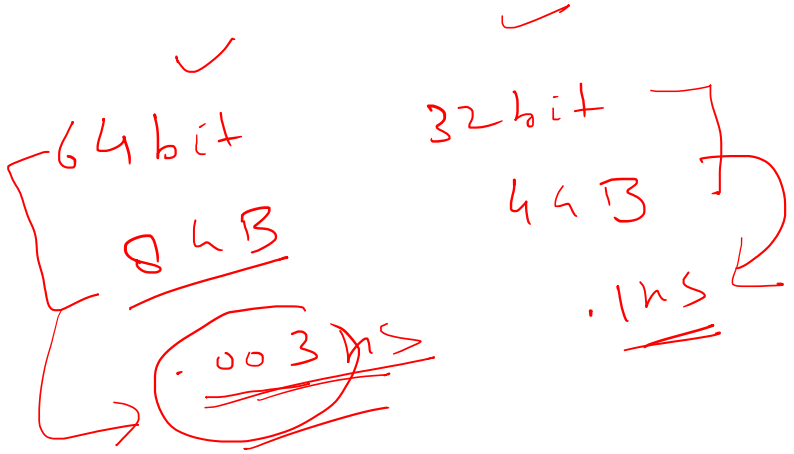
Data Structure Training(Algo&Complexity)

Types Of Analysis-

Posterior Analysis-

1-Analysis done after executing

X=x+1;



3-It provides exact values.

4-It provides non uniform values.

5-It is dependent of CPU,O/S & system architecture

6- Implementation → C, C++, Java, Python

7-Testing And Debugging

Data Structure Training(Algo&Complexity)

Time Complexity

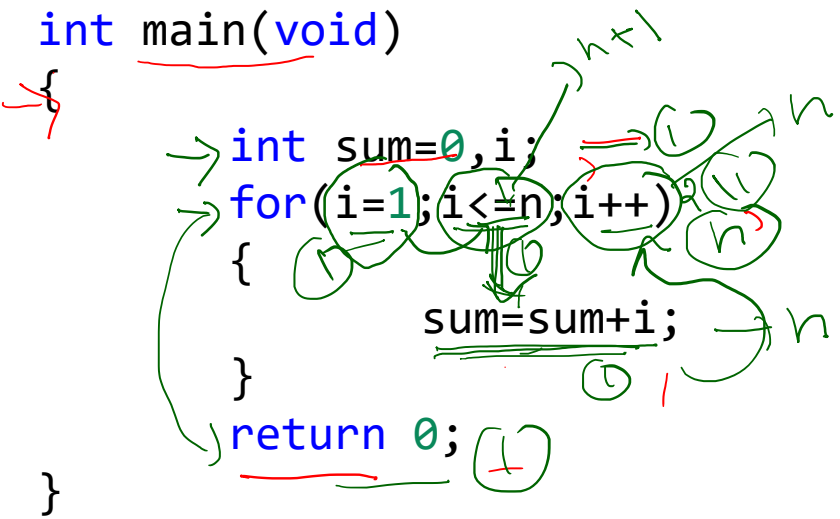
1-Simple for loop ✓
2-Nested for loop ✓
3-if-else ✓
4-Recursive Algorithm ✓

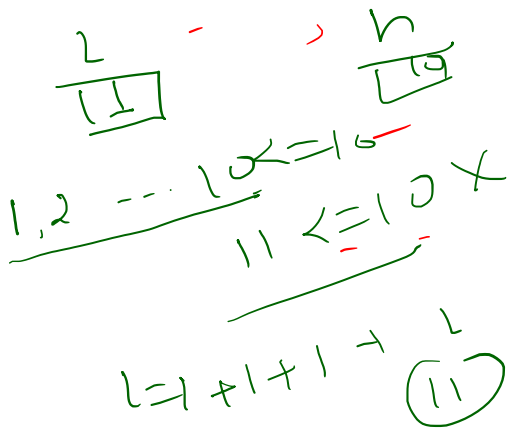
Data Structure Training(Algo&Complexity)

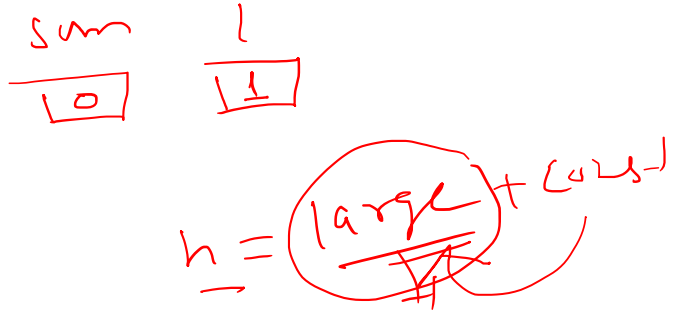
Simple for loop

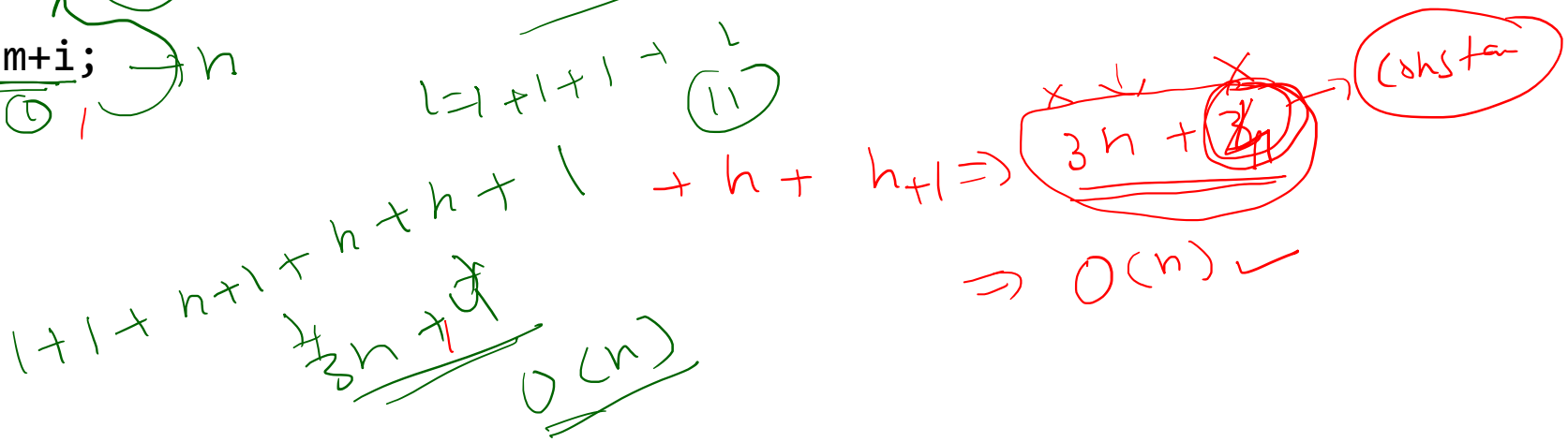
Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
    int sum=0,i;
    for(i=1;i<=n;i++)
    {
        sum=sum+i;
    }
    return 0;
}
```







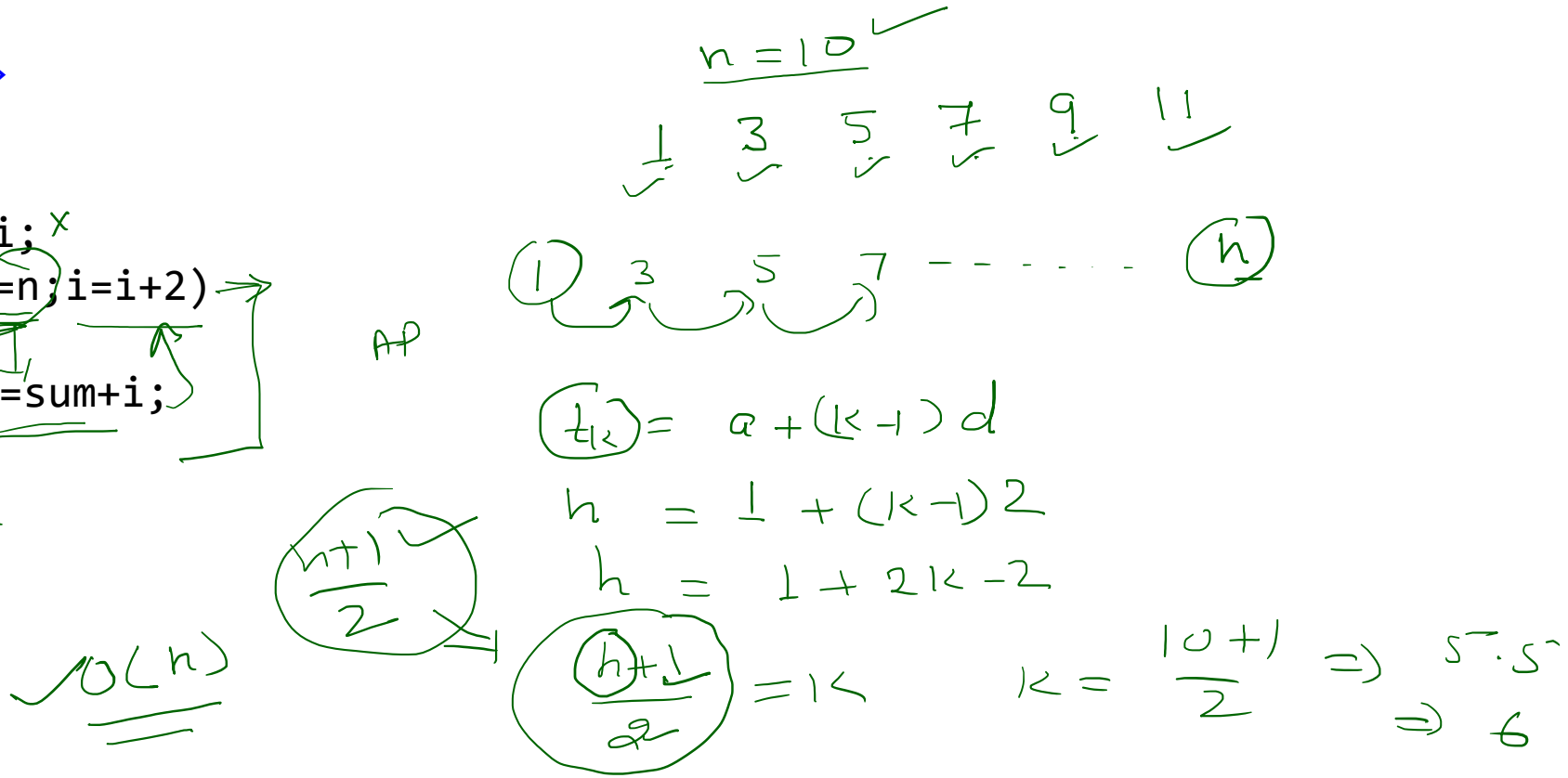


Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=1;i<=n;i=i+2)
        {
                sum=sum+i;
        }
        return 0;
}
```

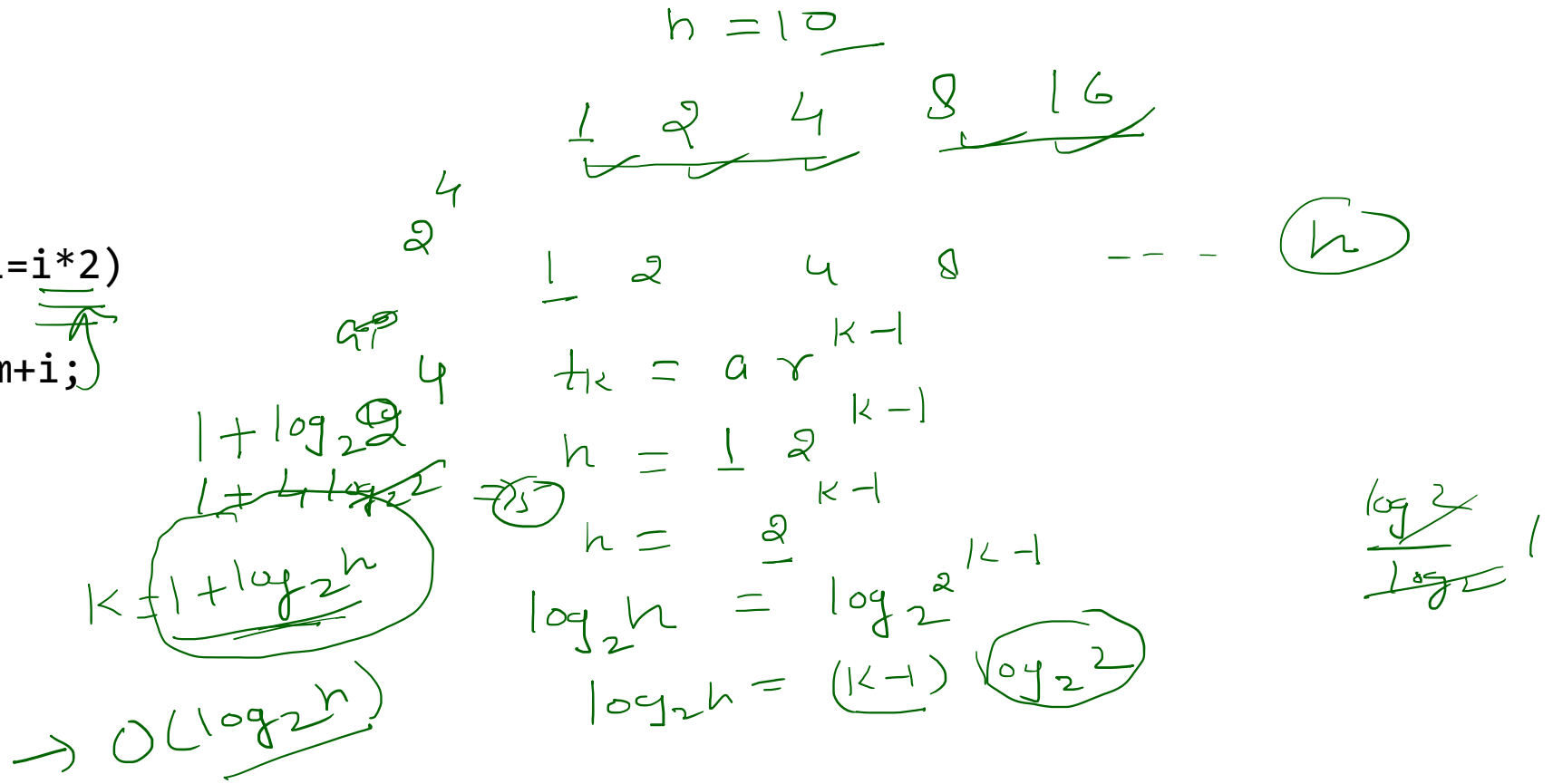


Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=1;i<=n;i=i*2)
        {
                sum=sum+i;
        }
        return 0;
}
```

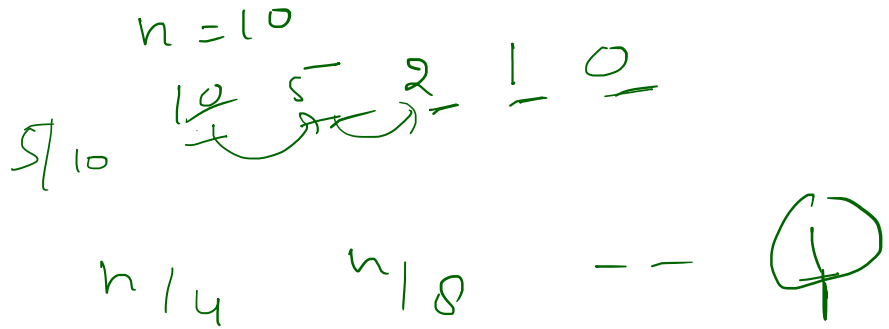


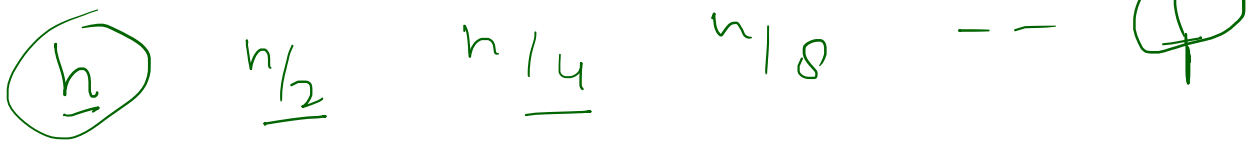
Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i;
        for(i=n;i>0;i=i/2)
        {
                sum=sum+i;
        }
        return 0;
}
```





$t_k = a\,r^{k-1}$

$1 = n\left(\frac{1}{2}\right)^{k-1}$

$1 = \frac{n}{2^{k-1}} \Rightarrow$

$2^{k-1} = n$

$\log_2 2^{k-1} = \log_2 n$

$(k-1) = \log_2 n$

$k = 1 + \log_2 n$

$k = 1 + \log_2 2^4$

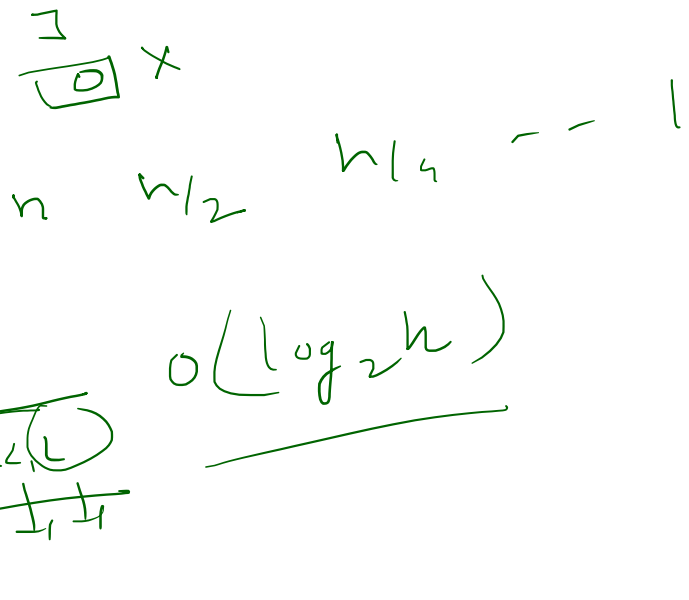
$= 5$

Data Structure Training(Algo&Complexity)

Simple for loop

Find total comparison and time complexity of following code and let val(j) denote value stored in variable j.after termination of loop what is the value of val(j)

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=n;j=0;i>0;i=i/2,j=j+i)
        {
                sum=sum+i;
        }
        return 0;
}
```

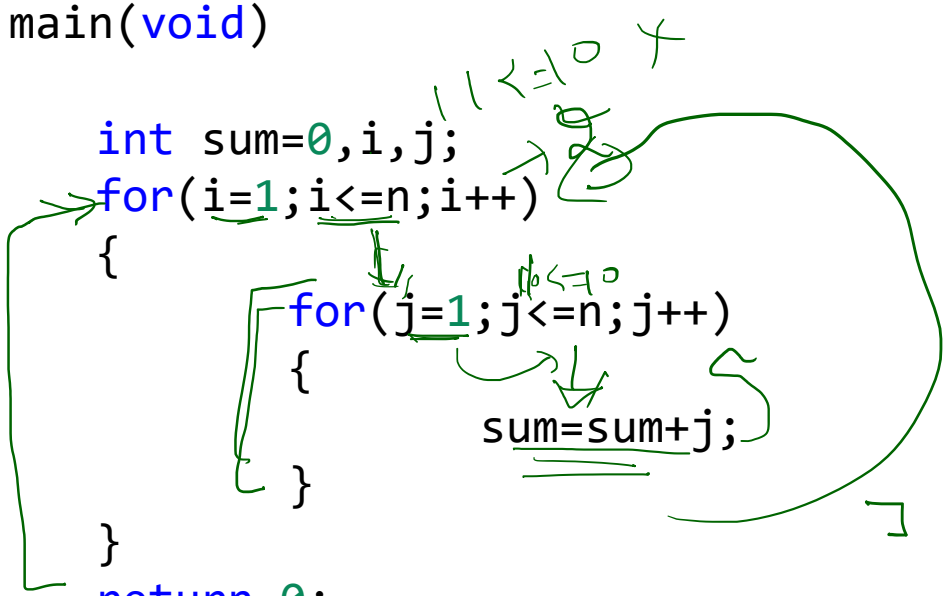


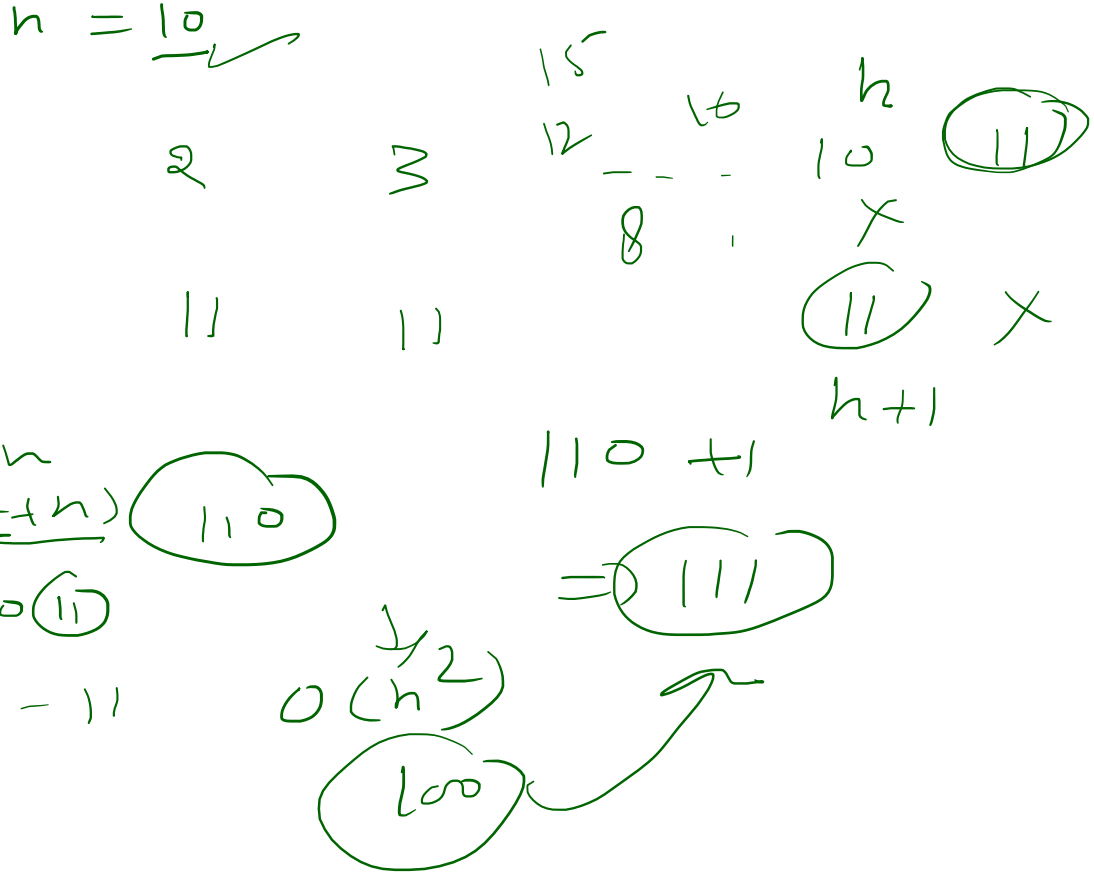
Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
    int sum=0,i,j;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            sum=sum+j;
        }
    }
    return 0;
}
```



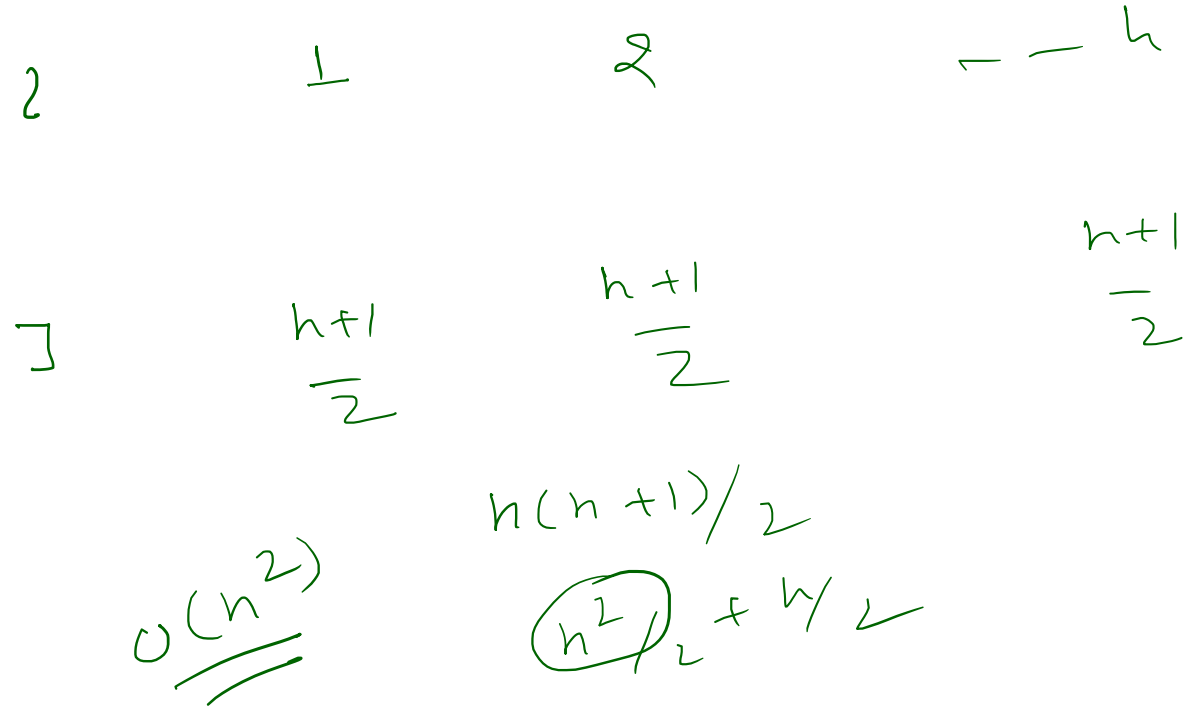


Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i++)
        {
                for(j=1;j<=n;j=j+2)
                {
                        sum=sum+j;
                }
        }
        return 0;
}
```

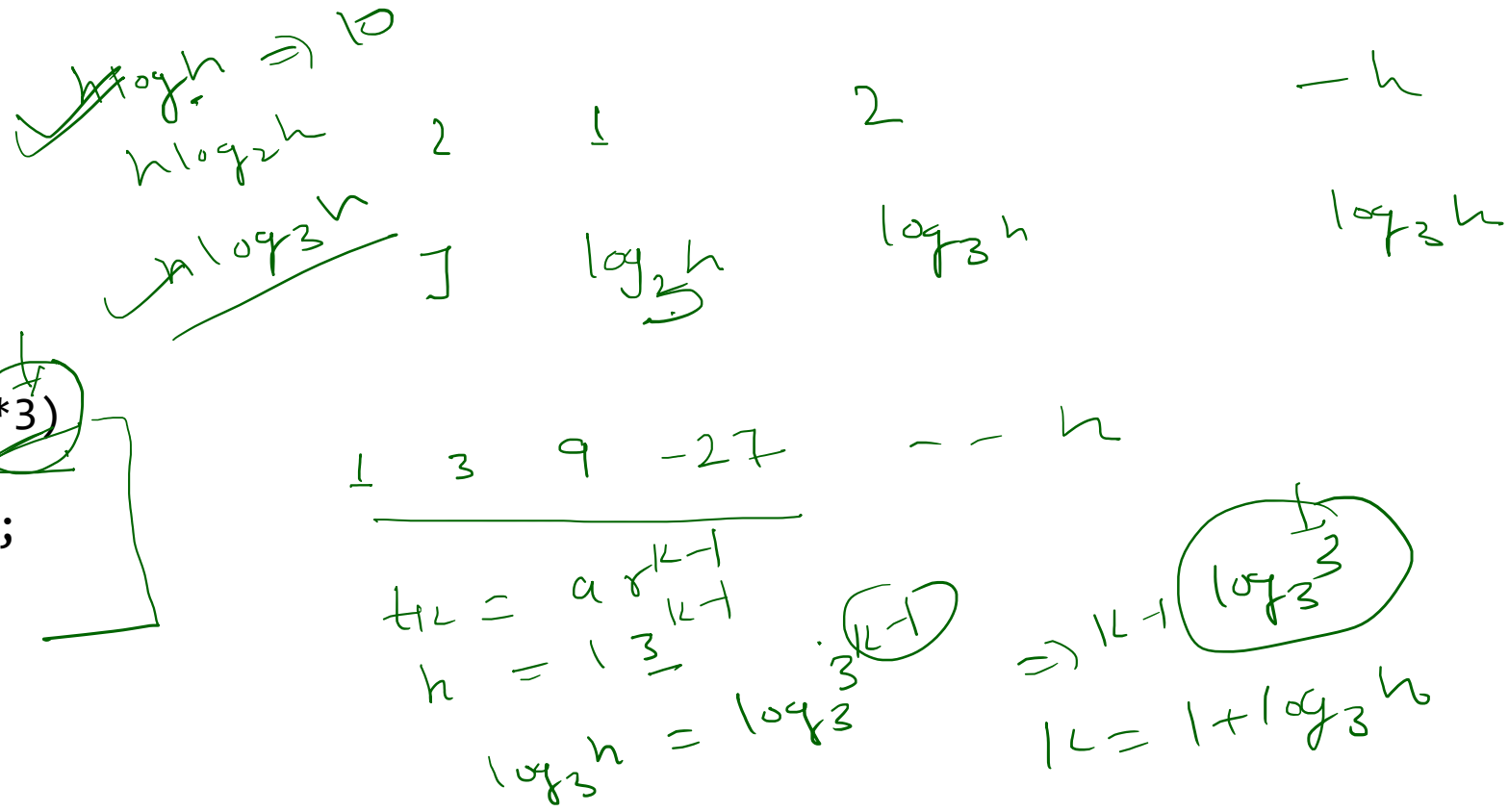


Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
	int sum=0,i,j;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j=j*3)
		{
			sum=sum+j;
		}
	}
	return 0;
}
```

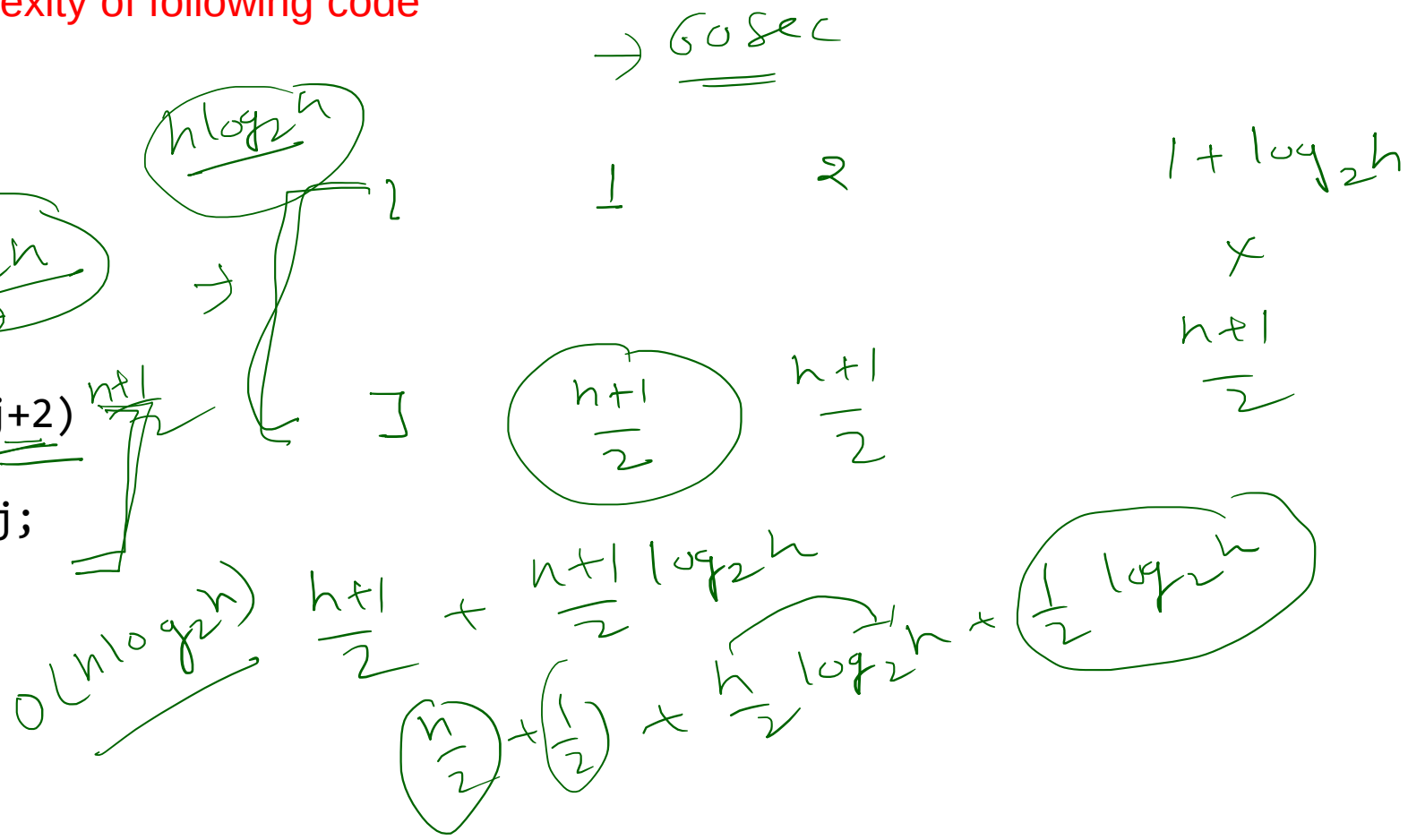


Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i=i*2)
        {
                for(j=1;j<=n;j=j+2)
                {
                        sum=sum+j;
                }
        }
        return 0;
}
```

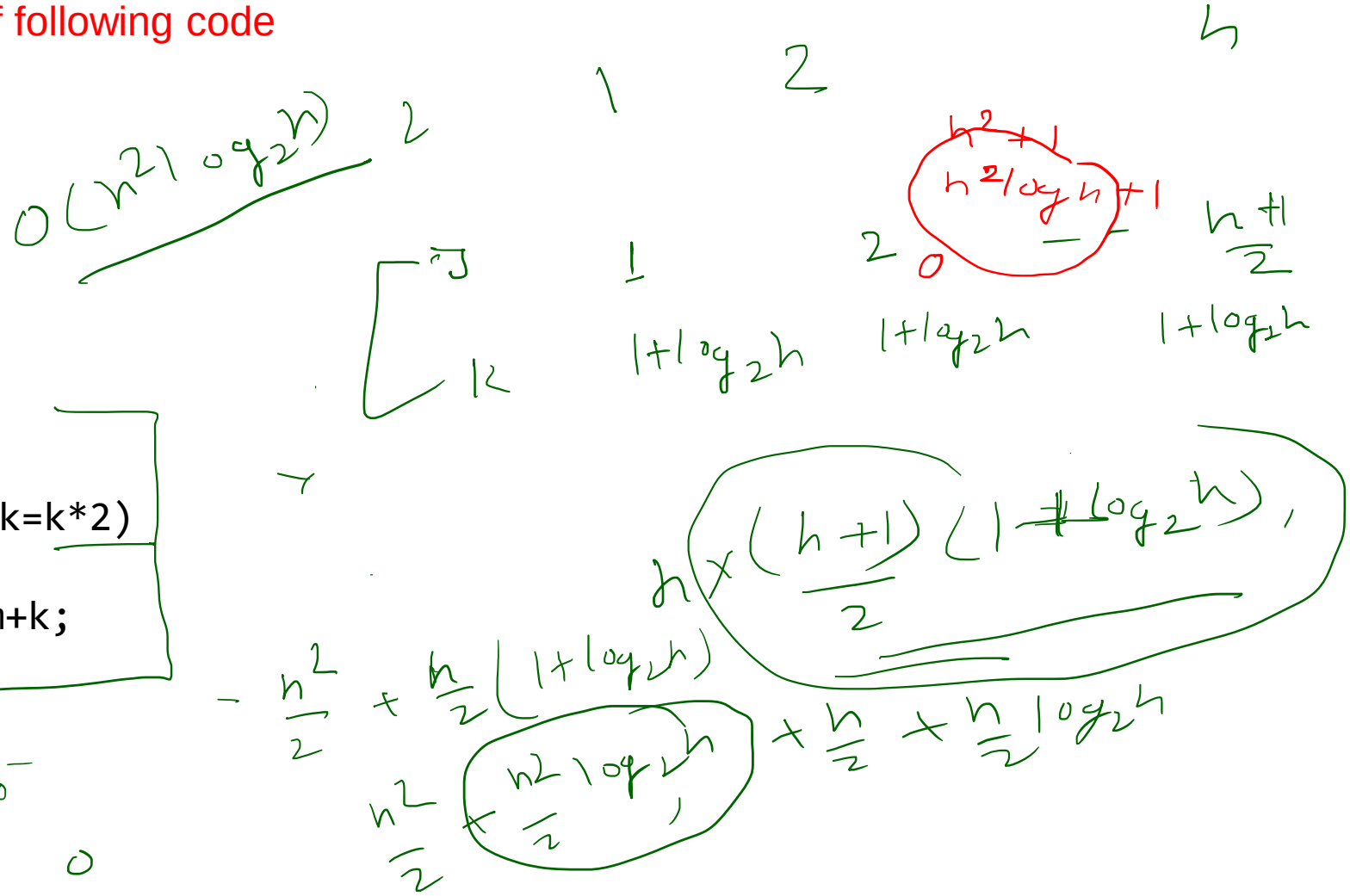


Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```c
#include <stdio.h>
int main(void)
{
        int sum=0,i,j;
        for(i=1;i<=n;i=i+1)
        {
                for(j=1;j<=i;j=j+2)
                {
                        for(k=1;k<=n;k=k*2)
                        {
                                sum=sum+k;
                        }
                }
        }
        return 0;
}
```



Data Structure Training(Algo&Complexity)

Nested for loop

Find total comparison and time complexity of following code

```
#include <stdio.h>
int main(void)
{
	int sum=0,i,j;
	for(i=0;i<=n;i=i+1)
	{
		for(j=1;j<=i;j=j+1)
		{
			if(j%i==0)
			{
				for(k=1;k<=n;k=k+1)
				{
					sum=sum+k;
				}
			}
		}
	}
	return 0;
```

Data Structure Training(Algo&Complexity)

Recursive Algorithm-

Find total comparison and time complexity of following code

```
int fact(int n)
{
        if(n==0 || n==1)
                return 1;
        else
                return n*fact(n-1);
}
```

Data Structure Training(Algo&Complexity)

Recursive Algorithm-

Find total comparison and time complexity of following code

```
int fib(int n)
{
        if(n==0)
                return 0;
        if(n==1)
                return 1;
        else
                return fib(n-1) +fib(n-2);
}
```

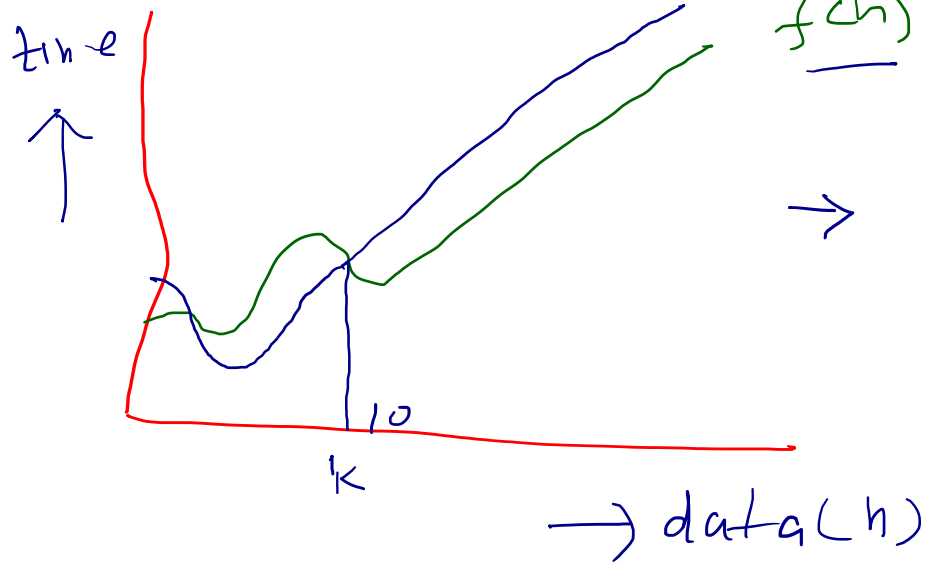
Data Structure Training(Algo&Complexity)

Asymptotic Notation-

to compare two algorithms rate of growth with respect to time and space, need asymptotic notation.

Big-Oh (O)

f(n) is O(g(n)) iff there exist some constants c>0 and k>=0 such that f(n) <= c.g(n) for every n>=k



→ f(n) = low

→ g(n) = high

→ low = O(high)

f(n) = O(g(n))

Data Structure Training(Algo&Complexity)

① → $f(n) = a_0 + a_1 n + a_2 n^2 + \cdots \rightarrow a_m n^m$

$g(n)$ x

$O(n^m)$

$f(n) = O(g(n))$

$= O(n^2)$

② $low = O(high)$ →

$f(n) = n^2 + n + 1$

$O(g(n))$?

$n^2 \leq n^2$

$n^2 + 1 \leq n^2 + n^2$

$n^2 + n + 1 \leq n^2 + n^2 + n^2$

$n^2 + n + 1 \leq 3n^2$

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

Data Structure Training(Algo&Complexity)

$n^2 \leq n^3$

$n^2+n+1 \leq n^3+n^3+n^3$

$f(n) \leq 3n^3$

$\downarrow \quad \downarrow$

$\leq \quad g(n)$

$O(n^3)$

$f(n) \leq 3n^4$

$\downarrow$

$O(n^4)$

$\vdots$

n^n

$n^2 \quad n^3 \quad n^4 \quad \text{---} \quad n^3$

$\Rightarrow$ lUb = least upper bound

Data Structure Training(Algo&Complexity)

$f(n) = n^2 \log n$ $\quad$ $g(n) = n(\log n)^{10}$

$\rightarrow$ $f(n) = O(g(n))$, $g(n) \neq O(f(n))$ ✓

$\rightarrow$ $f(n) = O(g(n))$, $g(n) = O(f(n))$ ✓

$\Rightarrow$ $g(n) = O(f(n))$, $f(n) \neq O(g(n))$ ✓

$g(n) = O(f(n))$

$n > (\log n)^2$ ✓

$2 > (\log_2 n)^2 \rightarrow n$

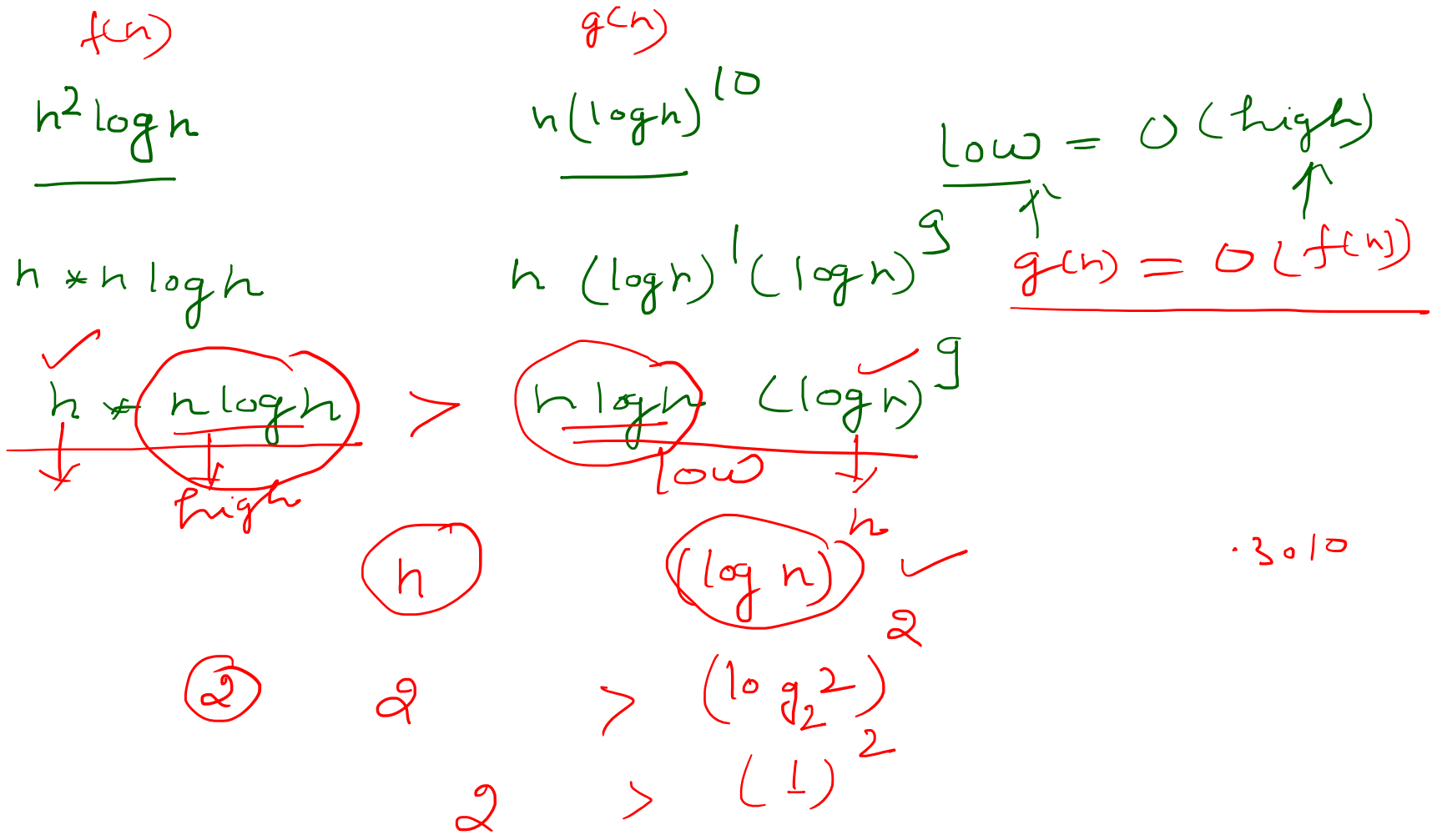
$2 > 1$

$n^2 \log n$ $\quad$ $n(\log n)^{10}$

$n * n\log n$ $\quad$ $n\log n \cdot (\log n)^9$

$f(n) > g(n)$

Data Structure Training(Algo&Complexity)



Data Structure Training(Algo&Complexity)

$f(n) = n!$ $\quad O(?)$

$= n(n-1)(n-2) \cdots 1$

$= n \quad n(1-\frac{1}{n}) \quad n(1-\frac{2}{n}) \cdots n(\frac{1}{n})$

$= n^n \left[1 \cdot (1-\frac{1}{n}) \cdot (1-\frac{2}{n}) \cdots \frac{1}{n}\right]$

$f(n) = a_0 + a_1 n + \cdots a_m n^m$

$= O(n^n)$

Data Structure Training(Algo&Complexity)

$f(n) = \log(n!)$

$= \log(O(n^n))$

$\leq O(\log n^n)$

$= O(n \log n)$

$f(n) = O(?)$

23
32

Data Structure Training(Algo&Complexity)

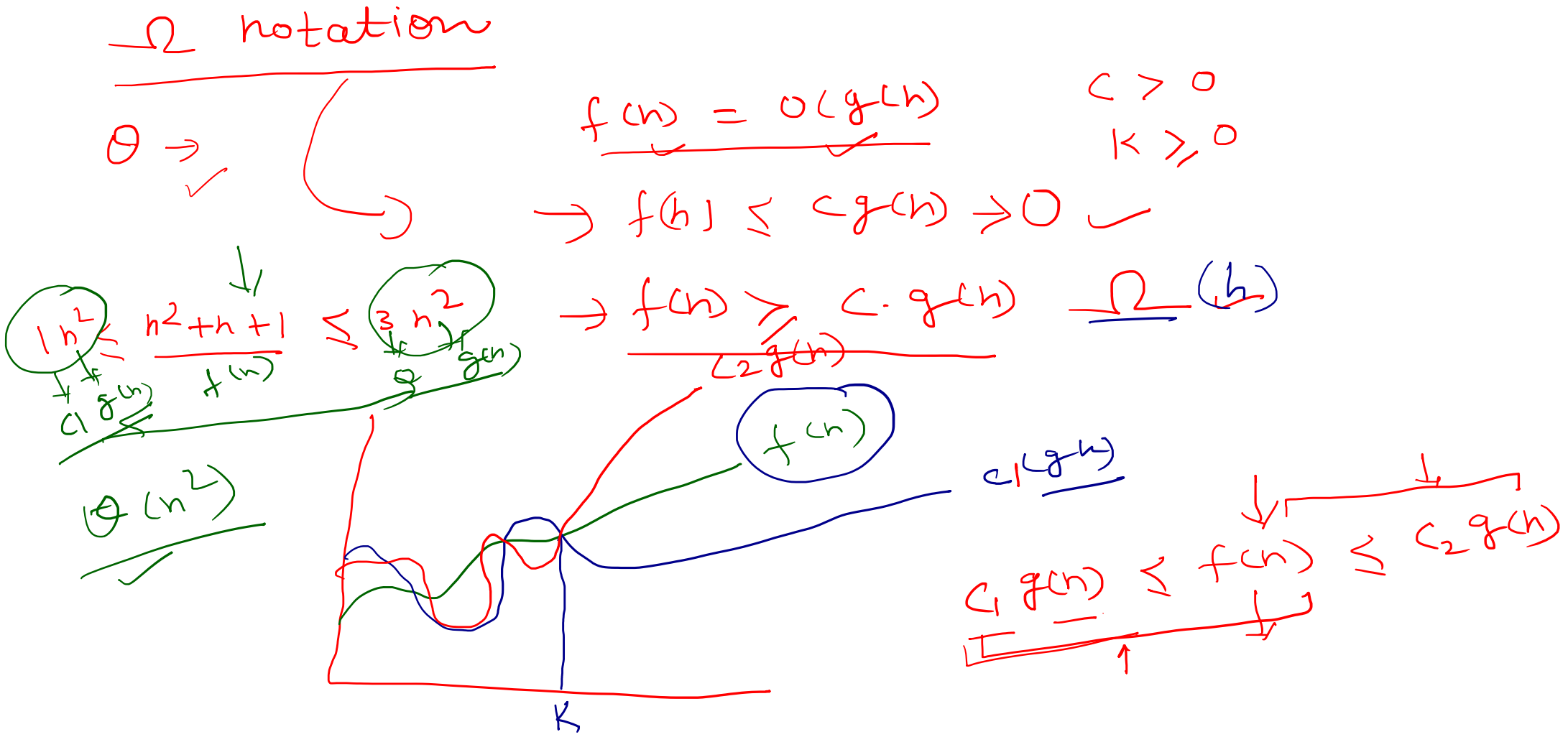
$f(n)$ $\qquad\qquad$ $g(n)$

$$2^{2^n} \geq n! \geq 4^n \geq 2^n \geq n^3 \geq n^2 \geq n\log n \geq n$$

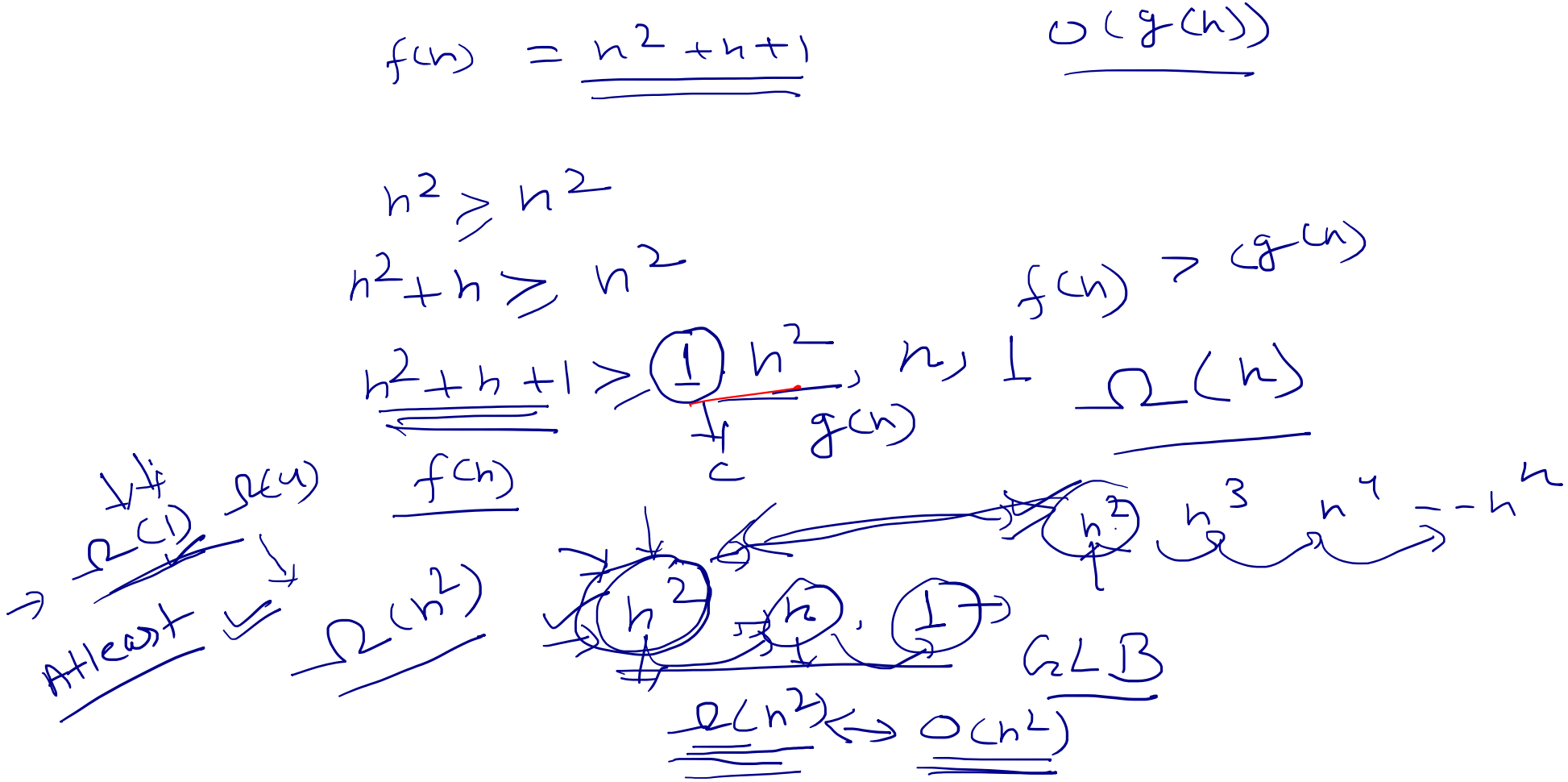
$$\rightarrow \geq \log\log n \geq \frac{\log n}{\log\log n} \geq 1$$

Dominance Ranking $\qquad$ $g(n) = O(f(n))$

Data Structure Training(Algo&Complexity)



Data Structure Training(Algo&Complexity)



Data Structure Training(Algo&Complexity)