Time complexity analysis of iterative programs

Algorithms	
Iterative	Recursive
Looping	Calls itself

```
A() {
	I=1, s=1, n=99;
	while(s≤n) {
		i++;
		s=s+i;
		pf(“print”);
	}
}
```

S: 1, 3, 6, 10, 15, 21 …. >n
I : 1, 2, 3, 4, 5 ……

$k(k+1)/2 > n \Rightarrow k^2 = n \Rightarrow k = O(\sqrt{n})$

```
A(){
	if(n>1)
	return A((n-1))
}
```

=>T(n) = 1+T(n-1)
By ***Back substitution***
T(n-1) = 1+T(n-2)
T(n-2) = 1+T(n-3)
T(n-3) = 1+T(n-4)
	. .
	. .
	. .
Substitute T(n-1) in T(n)
T(n) = 1+1+T(n-2)
	= 2+T(n-2)
Substitute T(n-2) in T(n)
	= 2+1+T(n-3)
	= 3+T(n-3)
	.
	.
	.
Hence, T(n) = k+T(n-k)

Alpha condition is n>1; n-k =1 => k = n-1
T(n) = (n-1)+T(n-(n-1))
= (n-1)+T(1)
= (n-1)+1
= n
Therefore, O(n)