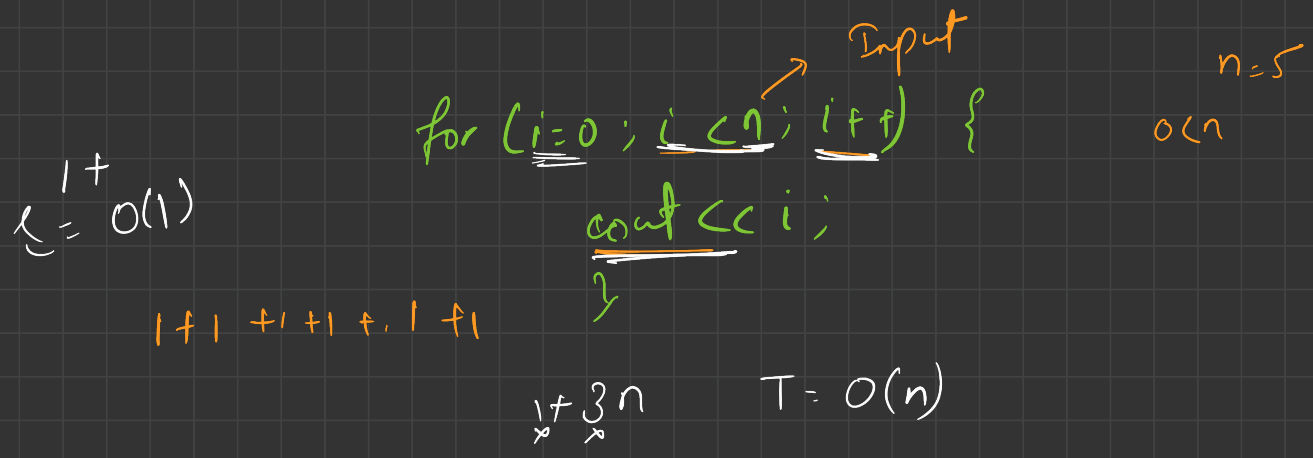
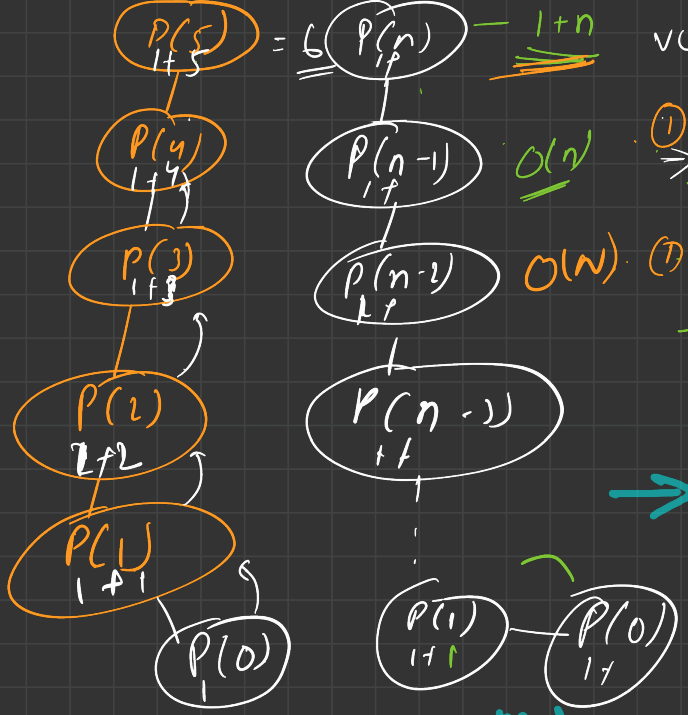
Time Complexity

* Total time taken by an algorithm to run as a function of its input size





1+n

O(n)

O(N)

5+T(0)

n+1

```
void print (int n) {
    if (n == 0)
        → return;
    cout << n;
    → print (n-1)
}
```

T(0) = 1

1+n

n n+1 n-2 n-3 1

P(n) = cout << n, print(n-1)

T(n) = 1 + T(n-1)

= 1+1 + T(n-2) = 2 + T(n-2)

T(n-1) = 1 + T(n-2)

= 2+1 + T(n-3)

= 3 + T(n-3)

= 4 + T(n-4) = 3+1 + T(n-4)

K

n=K

n-n = 0

T(0) = 1

T(0) = 1

n + T(n-n)

n + T(0)

n+1

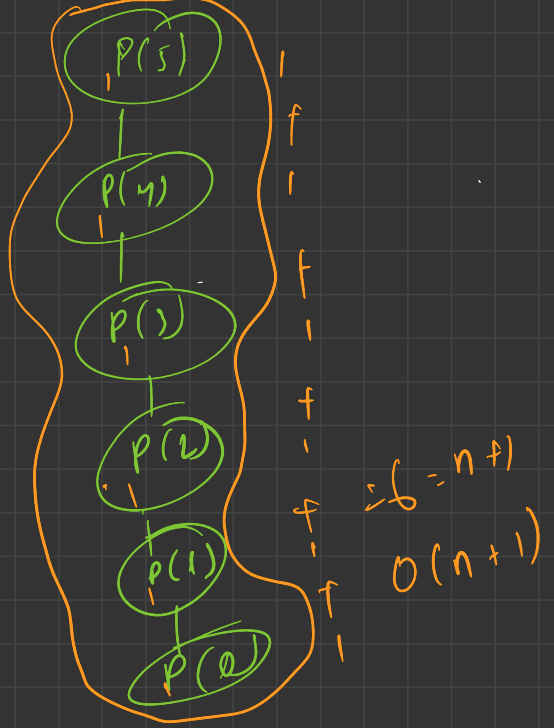
T(n-K) = T(0)

n - K = 0

K = n

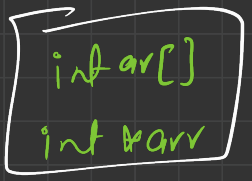
: K + T(n-K)

: n + T(n-n)




```
void P(int n)
{
  if(n == 0)
    return
  cout<<n;
  P(n-1);
}

int main(){
  P(5)
}
```



4 byte
8 byte

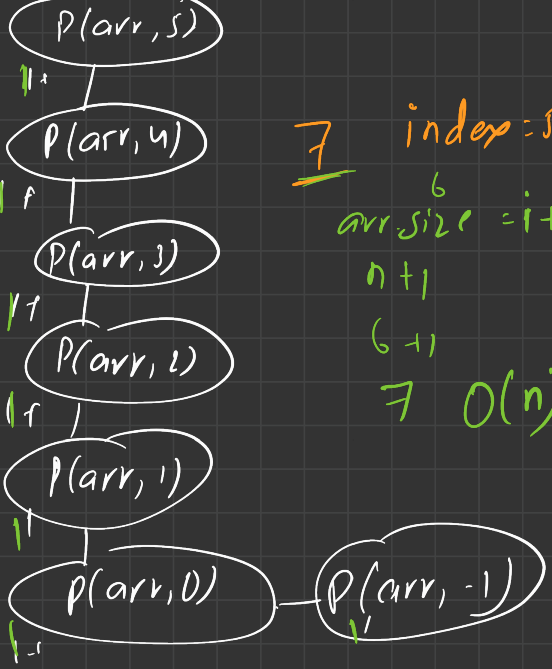
n+1

i = 6

i = arr.size - 1

5 + 1 = arr.size

6 = n



7 index = 5

arr.size = i+1

n+1

6+1

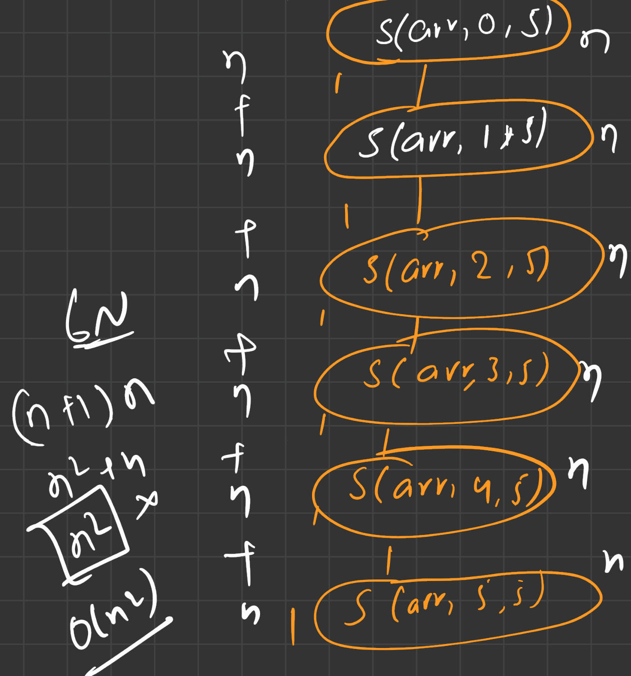
≈ O(n)

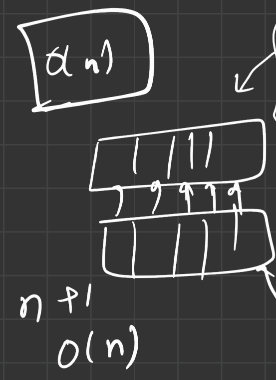
```
void P(int arr[], int i){
  if(i == -1)
    return;
  cout<< arr[i];
  P(arr, i-1);
}
```

index

1	2	3	4	5	6

```
int main(){
  int ar[] = {1,2,3,4,5,6}
  P(arr[], 5)
}
```

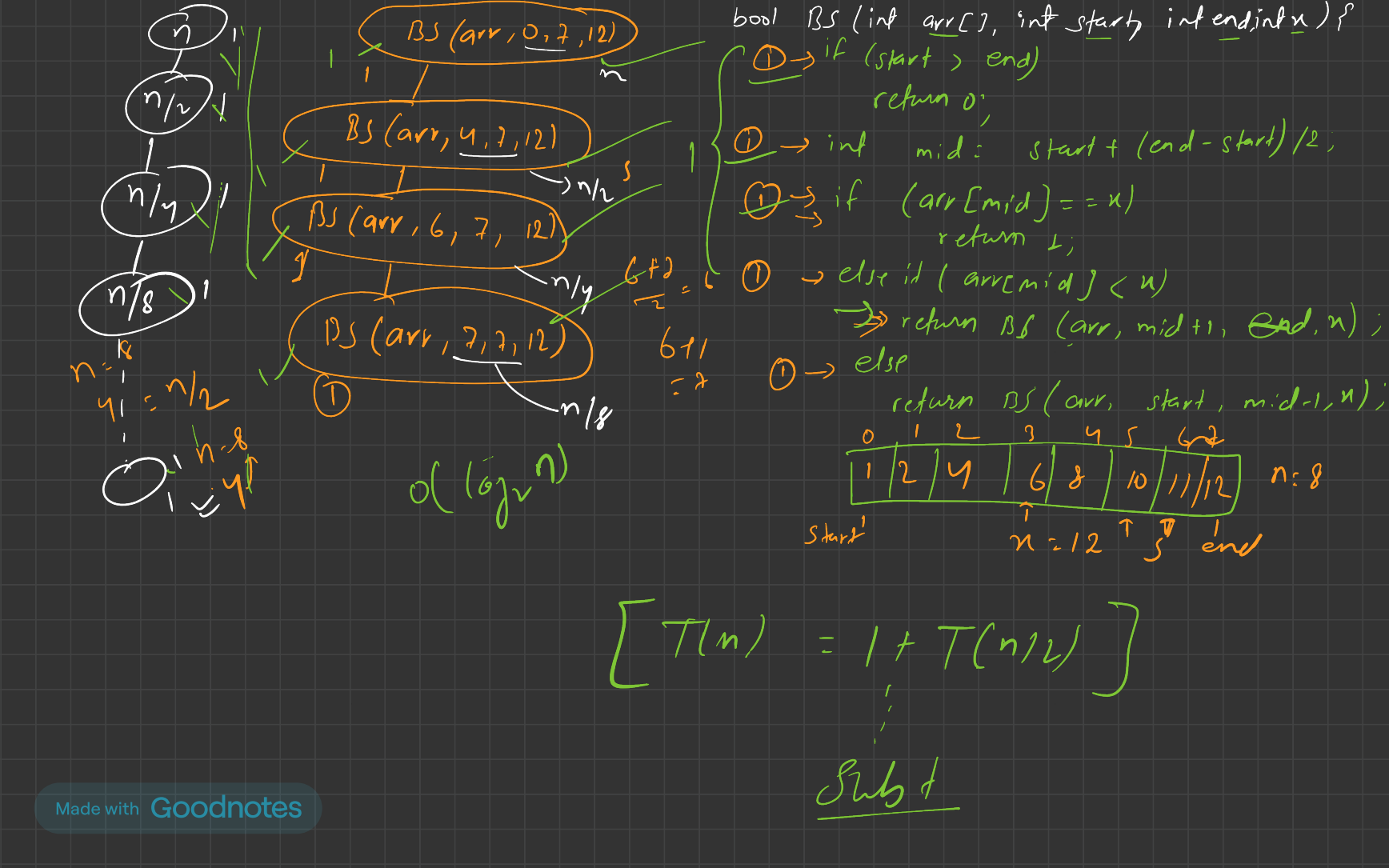


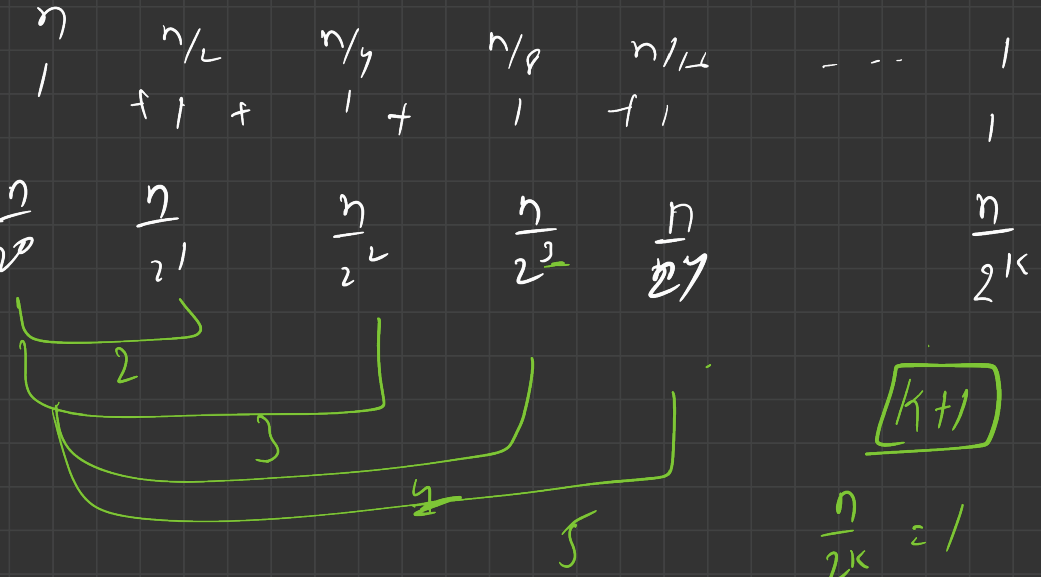


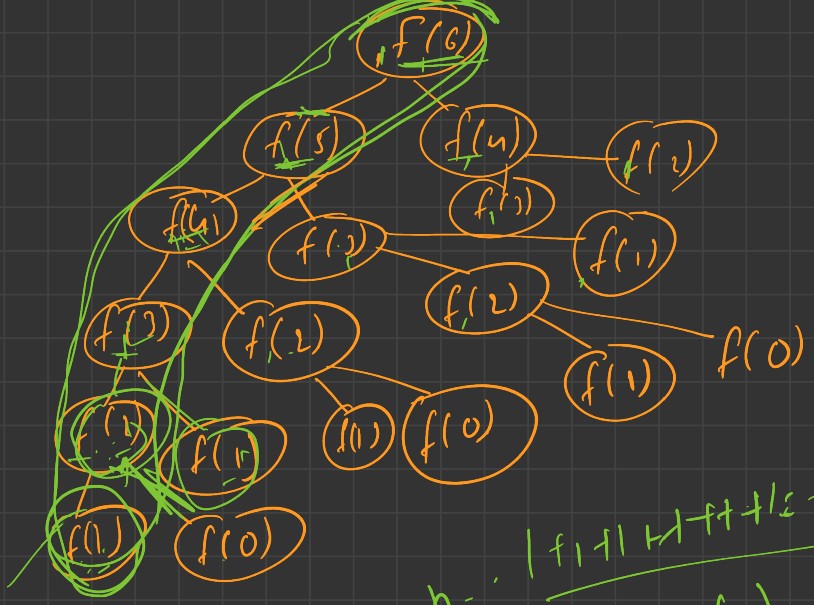

```
int sum (vector<int>& arr, int i, int n) {
    if (i == n)
        return 0;
    return arr[i] + sum(arr, i+1, n)
}
```

n = 5

```
int main() {
    int n;
    cin>>n;
    vector<int> v(n);
    cout << sum(arr, 0, n);
}
```







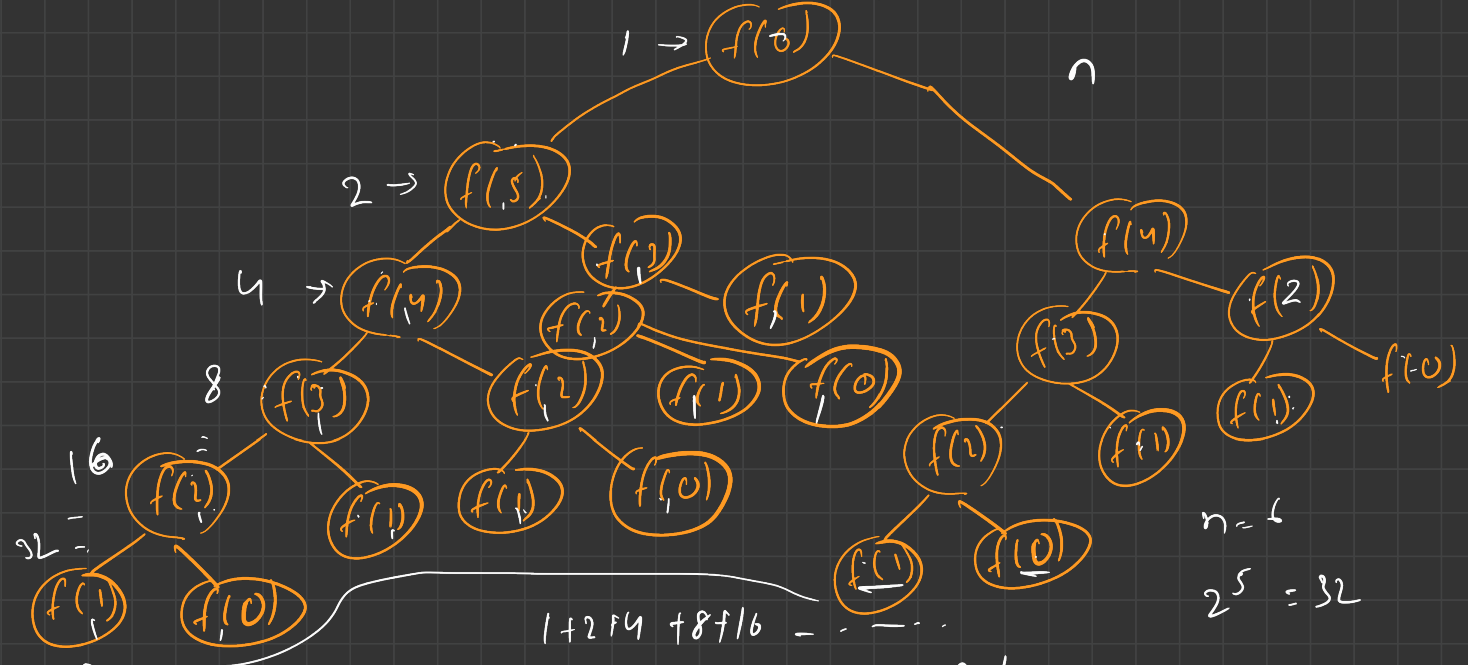
n = 6 : 1+1+1+1+1+1 = 6

O(n)

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

$T(n) = T(n-1) + T(n-2) + 1$

$2^n - 1$



$1+2+4+8+16 - - - -$

$2^0+2^1+2^2+2^3 - - - - 2^{n-1}$

$2^n - 1 \Rightarrow O(2^n)$

$2^0+2^1+2^2+2^3$

$1+2+4+8$

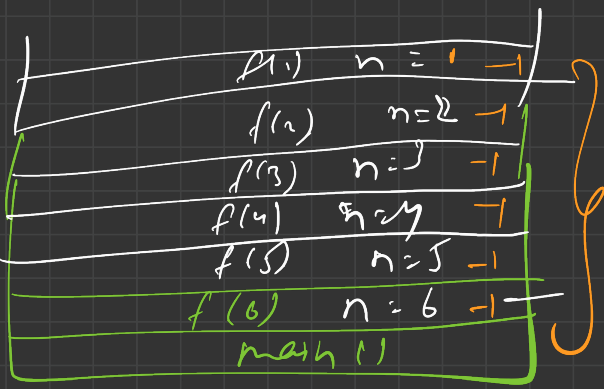
$= 15$

$3 \rightarrow 2^{n-1} \Rightarrow 2^2-1$

$\rightarrow 2^3-1 = 8-1$

$\rightarrow 2^{4-1} = 16-1$

$15 \rightarrow 2^{n-1}$



Stack memory

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

f(6)

6

O(n)

Space