Types of Recursion
1. Tail
2. Head
3. Tree
4. Indirect
5. Nested.

1. Tail

```
{...
 ...
fun(n-1)  ← last statement.
}
```

no operation at returning time

compare with loop.

```
void fun(int n)
{
  while (n>0)
  {
    printf("%d", n);
    n--;
  }
}
—
fun(3)
```

```
void fun(int n)
{
  if (n>0)
  {
    printf("%d", n)
    fun(n-1);
  }
}
—
fun(3);
```

amount of time same.

O(n) O(n)

space.

O(1) 4 → O(n)

2. Head Recursion.

```
void fun(int n).
 {
    if(n>0)
    {
     fun(n-1);          ←  no lines before the call
     printf("%d", n);      only do things returning time.
    }
 }
```

operation "

fun(3)

↓

loop.

```
void fun(int n)
 {
   while (n>0)
   {         ↓.
            void fun(int n)
            {
              int i=1;
              while (i<=n)
              {
              printf("%d", i);
              i++;
              }

   }
```

head recursion

Cannot be easily converted into a loop.

3. Tree Recursion.

(psedw code)

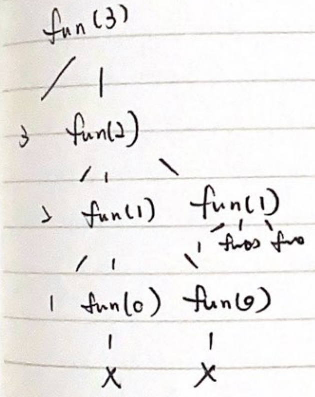
Linear Recursion

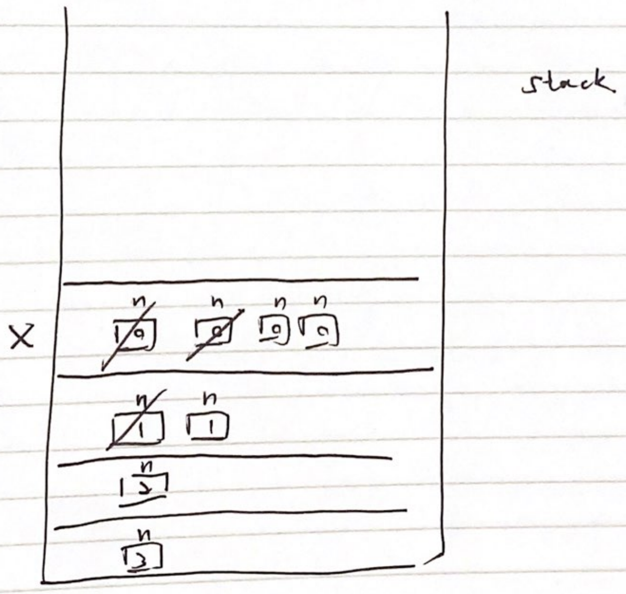
```
fun(n)
{
  if(n > 0)
  {
    —
    —
  fun(n-1)
    =
  }
}
```

Tree Recursion.

```
fun(n)
{
  if (n>0)
  {
    print("...)
    —
    fun(n-1)
    =
    fun(n-1)
    =
  }
}
```

fun(3).





o/p: 3, 2, 1, 1 - - - - -

time complexity : $O(2^n)$

space: $O(n)$ (same space is reused)

4. Indirect recursion.

$$\begin{tikzcd} & A \arrow[dr, bend left] & \\ C \arrow[ur, bend left] & & B \arrow[ll, bend left] \end{tikzcd}$$

```
void A(int n)
{
   if (<->)
   {
     -
     -
     B(n-1);
   }
}

 void B(int n)
{
   if (      )
     -
     -
     A
   }
}
```

5. Nested Recursion.

```
void fun(int n) {
    if  —
    {
        :

        fun(fun(n-1));
    }
```

```
int (fun (int n))
{
  if (n>100)
     return n-10;
  else
     return fun (fun(n+11));
}

fun (95)
```

```
      fun(95)
        |
   fun(fun(95+11))
              106
       fun(106)
   fun(106)
        |
   fun(96)
```

Sum of