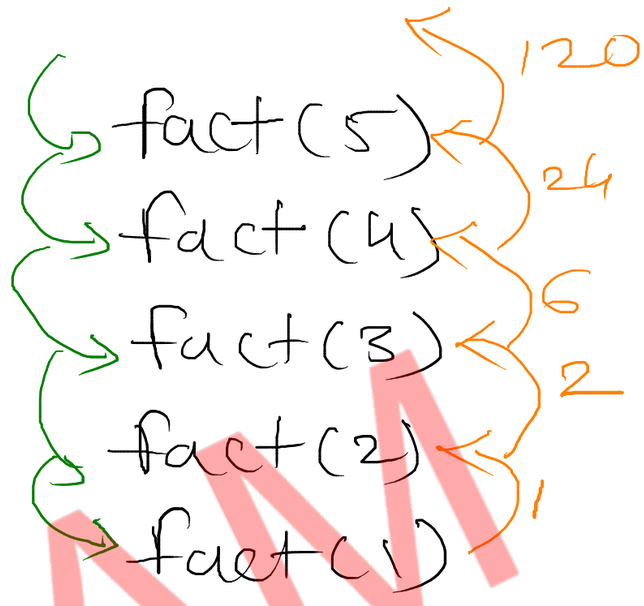
Analysis of Recursion

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



n = 5

recursive calls = 5

Time ∝ no. of recursive calls

Time ∝ n

$$T(n) = O(n)$$

Sorting Algorithms

- arrangement of data/values in either ascending or descending order of their values
- basic algorithms
    1. Selection sort
    2. Bubble sort
    3. Insertion sort
- advanced algorithms
    4. Merge sort
    5. Quick sort
    6. Heap sort

Selection sort

//1. select one position of the array (generally start from 0)
//2. compare selected position element with all other elements one by one
//3. if selected position element is greater than other position element,
then swap both elements
//4. repeat above three steps till array is not sorted

no. of elements = n

no. of passes = n-1

n=6

pass 1 → 5 → n-1
pass 2 → 4 → n-2
pass 3 → 3 → n-3
pass 4 → 2 → .
pass 5 → 1 → 1

15 → 21

Total comps = 1 + 2 + 3 + ... + (n-2) + (n-1)

$$= \frac{n(n+1)}{2}$$

$$= \frac{n^2+n}{2}$$

Time $\propto$ comps

Time $\propto \frac{n^2+n}{2}$

$$T(n) = O(n^2)$$

Best
Avg
Worst

while writing time complexity only consider degree term because it is highest growing term

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

↓

mathematical polynomial

↳ Degree of polynomial

↓

highest power of var

e.g. degree = 2

n	n^2
1	1
10	100
100	10000
1000	1000000

Auxiliary space $S(n) = O(1)$

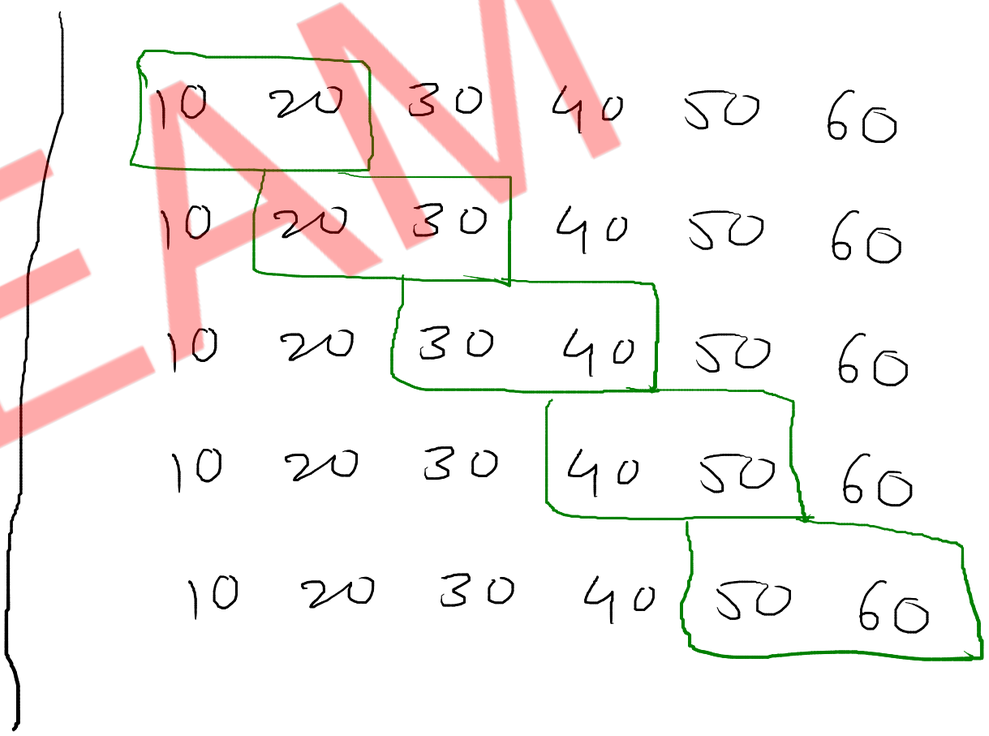
Bubble sort

//1. compare all pairs of consecutive elements
//2. if left element is greater than right element then swap both elements
//3. repeat above two steps till array is not sorted

$n=6$

$i<n$ $\quad j<n-1$ $\quad j<n-i$

i	j	j
1	0,1,2,3,4	0,1,2,3,4
2	0,1,2,3,4	0,1,2,3
3	0,1,2,3,4	0,1,2
4	0,1,2,3,4	0,1
5	0,1,2,3,4	0



$T(n)=O(n^2)$ { Avg, Worst

Auxiliary space $S(n)=O(1)$

$T(n)=O(n)$ - best

Insertion sort

```
//1. pick one element of array (start from 1 index)
//2. compare picked element with all its left neighbours one by one
//3. if left neighbour is greater than picked element
      //then move it by one position ahead
//4. insert picked element at its appropriate position
//5. repeat above steps till array is not sorted
```

```
public static void insertionSort(int arr[], int N) {
        for(int i = 1 ; i < N ; i++) {
            int temp = arr[i];
            int j = i - 1;
            while(j >= 0 && arr[j] > temp) {
                arr[j+1] = arr[j];
                j--;
            }
            arr[j+1] = temp;
        }
    }
```

N = 6

20 40 50 60 10 30

0 1 2 3 4 5

i	i<6	j	temp	j>=0 && arr[j]>temp
1	T	0,-1	40	T, F
2	T	1,0,-1	20	T, T, F
3	T	2	60	F
4	T			
5	T			

No. of element = n

No. of passes = n-1 (pick elements)

n = 6

pass 1 - 1 1

pass 2 - 2 2

pass 3 - 3 3

pass 4 - 4 :

pass 5 - 5 n-1

Total comps = 1+2+3+----+n-1 (↓ n)

$= \frac{n(n+1)}{2}$

Time $\propto \frac{n^2+n}{2}$

$T(n) = O(n^2)$ } Avg, Worst

$T(n) = O(n)$ } best

$S(n) = O(1)$

10, 20, 30, 40, 50, 60