**Program efficiency**

Efficiency is all about time and space complexity.

**Dynamic programming**

**Greedy approach:**

In greedy approach whatever is the solution for the problem give at the first go is fixed as the final solution

**Note:** This is not the best approach for all the scenarios however is also work for some cases.

In dynamic programming we will be finding or we will find out all the possible solution for the given problem out of which the best will be selected or picked.

**Time and space complexity:**

**Aymptotic notation:**

* Big O
* Omega
* Theta

**Program-1:**

Swajith is having 1 lakh in his bank Account; rate of interest is 12% per annum. In the 5th month Swajith is withdrawing 25000 rupees in order to buy gift for his loved one. In the 9th month 10000 is being deposited in his account by his second loved one. End of the financial year how much Swajith is having in his account .

**Space complexity**

**Structure:**

it allocates space for each individual member

**Union:**

it allocates space of the largest member.

**Time complexity:**

A for loop executes n+1 times



10-



5-



10 20

For (i=0;i<n;i+2){

Statements;

}

n/2

f(n)=n/2

Degree of polynomial is n

So n/ anything is n

So here also

O(n)

So irrespective of iteration time is gonna be same o(n)

**Nested loops**

For(i=0;i<n; i++){ …n+1 times

For(j=0;j<n; j++){ n\*(n+1)

Statements; n\*n---- n square

}

}

Time complexity = o(n square)

**Implement a 2d array rotate the array 90 degrees.**

**What is stack and heap memory. Which languages are using heap.**

**Nested loops continue**

For(i=0;i<n;i++){

For(j=0;j<n;j++){

Statements;

}

}

I=0 then 0<0 no I j

1. nothing
2. o will execure

will stop

1. will stop

For (i=1;i<n;i\*2){

Statements;

}

**Analyze**

I=1 1 time

I=2 2 times (1\*2)

I=3 4 times (1\*2)\*2=2 power 2

I=4 8 times (1\*2)\*2)\*2=2 power 3

So when stopes i>=n

I=2 power k

2 power k>=n

K=log n base 2

So when complexity O(log n base 2)

**Create an array (1d), it should contain number b/w 10 to 30, in this array extract and print**

1. **Even numbers**
2. **power values**

* **Constant Time complexity:** O(1)…
* **Linear Time complexity** :O(n)…
* **Logarithmic Time complexity**: O(log n)…
* **Quadratic Time complexity** :O(n2)…
* **Exponential Time complexity:** O(2n)…

**FINAL SUMMARY:**

I++ i- - i+2 O(n)

I\*2 i/2 log(n) base 2

P=0

For(i=0;P<n; i++) O(sqrt(n))

P+=i

def generate\_lists(n):

    table\_list=[]

    for num in range(n):

        row=[]

        for i in range(n):

            row.append(i)

        table\_list.append(row)

    return table\_list

print(generate\_lists(10))

it is called log-linear complexity

**Polynomial complexity:**

Space complexity grows proportionally square of input O(n2).