- DAY - 1

27/9/2023

* Program Efficiency

Efficiency is all about time and space complexity

* Dynamic Programming

In Greedy approach whatever is the solution of the problem given in the first go is fixed as the final solution.

Note : This is not the best approach for all the scenarios. however, it also works for some cases.

* In dynamic Programming we will be finding out all the possible solutions for the given Problem out of which the best will be Sdected.

* Time and Space complexity

* Asymptotic Notations : there are 3

1. Big O
2. Omega [ $\Omega$ ]
3. Theta [ $\Theta$ ]

* Sooji is having 1 Lakh his in bank account in that. Rate of interest is 12% per Annum in the 5th month. He withdraws 25k rupees in order to buy gift for his loved ones. In 9th month 10k rupees is deposited for his 2nd loved ones. End of the financial year how much his having in his amount. [S:I]

8th may — 25k deposited ⇒

4th Sep — 10k deposited withdraw

1 year = 12k Increase

total A = 1,000,00

T = 1 y

R = 12% ⇒ 12000

SI ⇒ 121000

5th A = 750000

T = 7mo ⇒ 1/7

R =

→ 1lakh 4 thousand

→ 75k → 750 I per months

→ 4A 750 ⇒ 3000.

→ 75+10 ⇒ 850000 + 12%

⇒ 850 I x4

~~4000~~
3000
3400
6400
4000
10400

$\frac{850}{4}$ / 3400 = 3400

* Space Complexity

Structure

{
int — 4 bytes
int — 4 bytes
}

Space = 8 bytes

{
int — 4
double — 8 }

Space ⇒ 12 bytes

Struct
{
int - 4
char - 1 }

Space = 8 bytes

{
double — 8
char — 1

⇒ Space - 8 + 1 + 7
= 16.

union

```
union
{
  int - 4
  char
  int - 4
}
```

space = 4

```
union
{
  int - 4
  double - 8
}
```

space = 8

```
union
{
  int - 4
  char - 1
}
```

space = 4.

```
struct
{
  int - 4
  char - 1
  double - 8
}
```

space = 24

OR

4 + 4 + 8 + 1 + 7

* Time complexity of Statements.

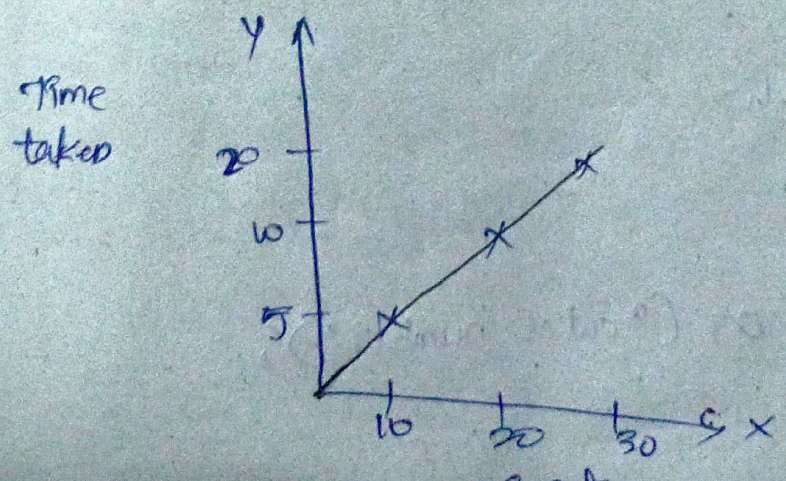
Type 1

```
for (i=0; i<5; i++)
{
  statements --
}
```

⇒ n+1 times the loop is executed.

⇒ Big O(n) — for polynomial theorem ⇒ f(n) = n+1

Big O(n)



* #include <stdio.h>
int main()
{
int i, n = 5;
for (i=0; i<n; i++)
Printf ("final %d \n", i);
Printf ("final i: %d", i);
return 0;
}

o/p: 5

⇒ Time complexity : O(n).

* for (i=1 ; i<5 ; i+2)
{
statements
}

o/p: 1, 3.

---

⇒ If $f(n) = \frac{n}{2}$ degree of polynomial is "n".

$f(n) = n$

So the time complexity is also O(n).

⇒ So irrespective of iteration time is gonna be the same.

* Get one number as input and find the sum of the digits of the given input

#include <stdio.h>
int main()
{
int number = 0; (" enter number");
while (n > 0)
{

using while loop.

```
#include <stdio.h>
int main()
{
  int rem,n; Printf("Enter number");
  Scanf("%d", &n);
  int Sum = 0;
  while (n>0)
  {
     rem = n%10;
     n=n/10;
     Sum = Sum+rem;
  }
  Printf ("%d \n",Sum);
}
```

Output: Enter a number: 153

9.

* using for loop

```
#include <stdio.h>
int main ()
{
  int rem, n, i;
  Printf("enter the number:");
  scanf("%d", &n);
  int sum=0;
  for (i=0; i<5; i++)
  {
     rem = n%10;
     n = n/10;
     Sum = Sum + rem;
  }
  Printf("%d \n", Sum);
  return 0;
}
```