27 September,2023

Recursion is better than Iteration.

Recursion-Dynamic Programming, Iteration-Greedy Approach.

Program Efficiency: Efficiency is all about time and space complexity.

Greedy Approach: In Greedy Approach, whatever is the soln for the problem given at the one go is fixed as the final soln.

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

In Dynamic Programming, we will find out all the possible solns for the given problem, out of which the best will be picked.

Time and Space Complexity:

Asymptotic Notations:

1.Big O(n)

2.Omega

3.Theta

Q: Swajith is having 1 lakh in his bank unit. Rate of interest is 12% per annum. In the 5th month, Swajith is withdrawing 25000 rupee, in order to buy a gift to loved one. In 9th month, 10000 is been withdrawn by his second loved one. End of the financial year, how much is having in his account? Take Simple Interest.

Sol: P=100000, R=12%

5th month, withdrawn

9th month, added

I1=(100000\*(4/12)\*12)/100=4000 (first 4 months)

I2=((100000-25000)\*(4/12)\*12)/100=3000 (next 4 months)

I3=((75000+10000)\*(4/12)\*12)/100=3400 (last 4 months)

85000+4000+3000+3400=95400

Difference b/w structure and union:

In C,Structure have different datatypes, where as array have similar datatypes.

In C,Structure have individual memory allocation(many members accessed one time) where as union have single memory allocation(one member accessed one time).

Structure

{

Int 4 bytes

Double 8 bytes

}

Space=4+8=12bytes

Union

{

Int 4 bytes

Double 8 bytes

}

Space=8 bytes(highest)

All languages char memory size is 1 byte except java, char memory size is 2 bytes.

Structure

{

Int 4 bytes

char 1 bytes

}

Space= 8 bytes

4 bytes for the **int**

1 byte for the **char**

3 bytes of padding for alignment (4-1=3) (Space)

Union

{

Int 4 bytes

char 1 bytes

}

Space=4 bytes(highest)

Structure

{

Int 4 bytes

char 1 bytes

double 8 bytes

}

Space= 24 bytes (4+1+8+(8-4)+(8-1)=24) or (4+4+8+1+7=24)

Time Complexity of Statements:

How many times statement is getting executed Why? Efficiency of the Program

Type1:

for(i=0;i<n;i++) ----n+1

{

Statements; -----n+1

}

‘n’ times executed; stops at ‘n’

Polynomial theorem f(n) =n+1

Time Complexity are Big O(n), Omega, Theta.

#include<stdio.h>

int main()

{

int i, n=5;

for(i=0;i<n;i++)

{

printf("%d",i);

}

printf("%d",i);

}

f(n)=n(iterations)

Time Complexity is O(n).

For(i=1;i<n;i+2)

{

Statements

}

f(n)=n/2 (iterations)

Degree of polynomial is n

So, n/ anything is n

So here also O(n)

Note: Irrespective of iteration time is gonna be the same O(n).

Q: Get one num as input. find the sum of digits.