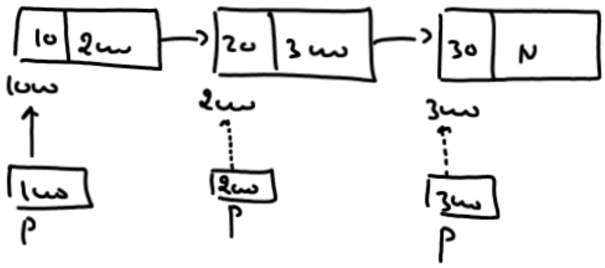
Iterative Code

```
void  display(struct node *p)
{
   if(p==NULL)
      return;
 while(p!=NULL)
{
   printf("\n%d",p->data);
   p=p->next;
}
}
```

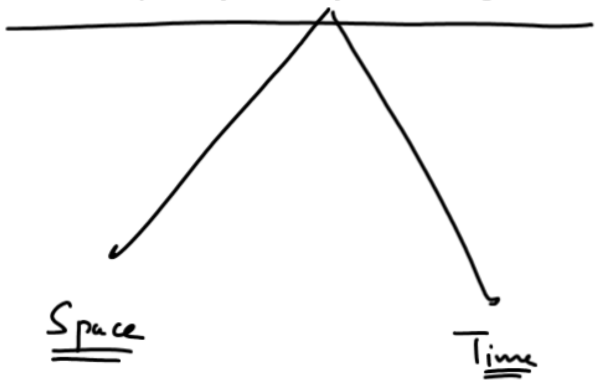
Recursive Code

```
void  display(struct node *p)
{
   if(p==NULL)
      return;
  printf("\n%d",p->data);
  display(p->next);
}
```

10
20
30



Time Complexity / Analysis Of Algorithm



How To Calculate Time Complexity ?

1. Analyze the code
2. assignment, calc, fn - call → 1 unit
3. calculate total time
4.

Linear Search
===========

```
int linear_search(int arr[],int n,int x)
{

    for(int i=0;i<n;i++)
   {
        if(arr[i]==x)
            return i;
   }
return -1;
}
```

① <u>Best Case</u> : $C(n) = O(1)$

② <u>Worst Case</u> : $C(n) = O(n)$

③ <u>Avg Case</u>

a. $C(n) = 1 \times \frac{1}{n} + 2 \times \frac{1}{n} + 3 \times \frac{1}{n} \ldots n \times \frac{1}{n}$

b. $C(n) = \frac{1}{n} \times (1 + 2 + \ldots n)$

c. $C(n) = \frac{1}{n} \times \frac{\cancel{n}(n+1)}{2}$

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

e. $C(n) = O(n)$

$x = 29$

<u>Binary Search</u>

0	1	2	3	4	5	6	7	8	9	10
15	30	42	51	62	69	74	81	86	93	98
L	M									H

① $L = 0,\ H = 10,\ M = 5$

M

② $L = 0,\ H = 4,\ M = 2$

③ $L = 0,\ H = 1,\ M = 0$

④ $L = 1,\ H = 1,\ M = 1$

$\frac{n}{2^k} = 1$

$n = 11 \qquad n = 2^k$

$\log_2 11 = \underline{3-4} \qquad k = \log_2 n$