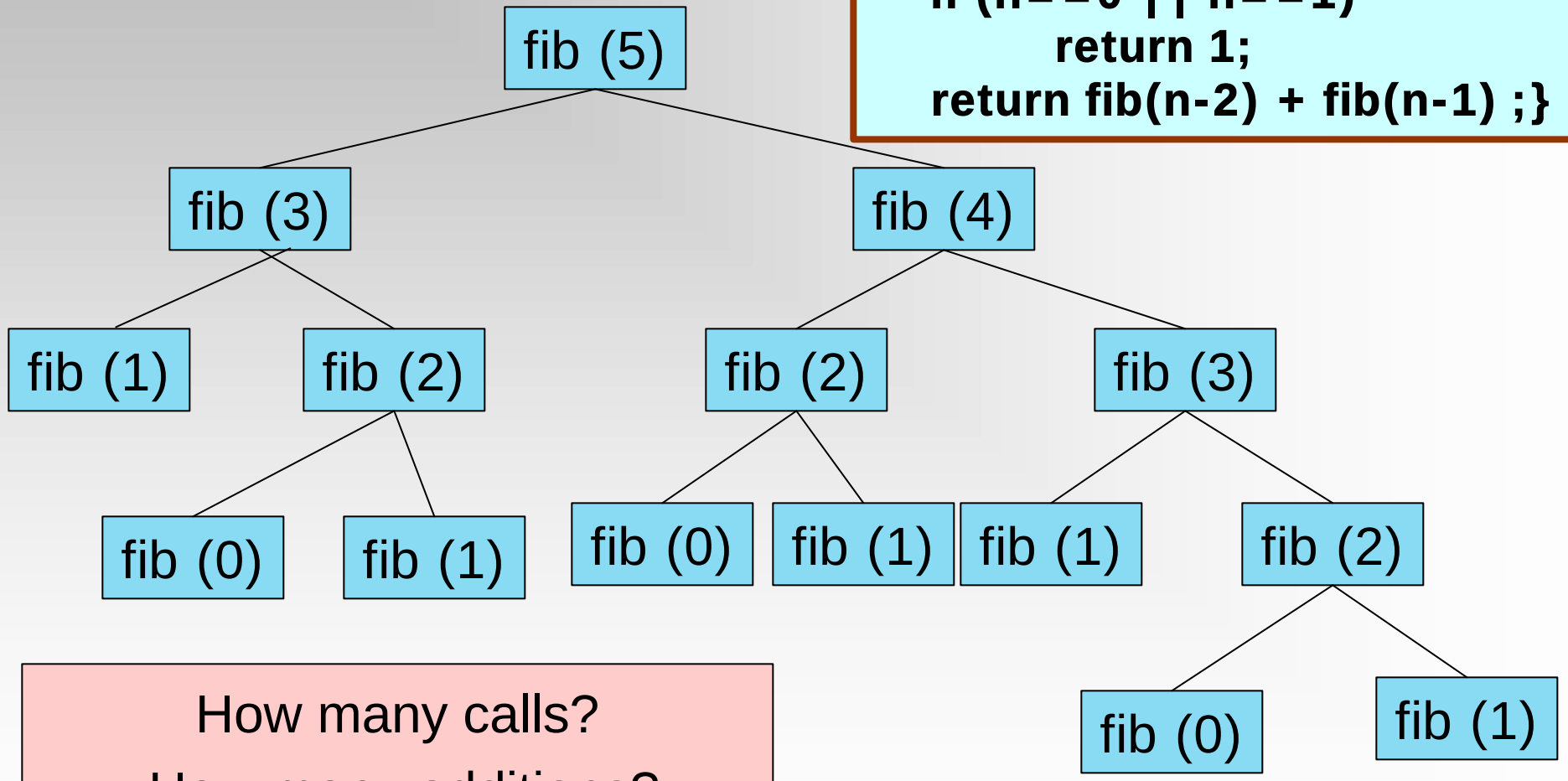
Recursion & Efficiency

P. P. Chakrabarti

Relook at recursive Fibonacci:

Not efficient !! Same sub-problem solved many times.

```
int fib (int n)    {
   if (n==0 || n==1)
         return 1;
   return fib(n-2) + fib(n-1) ;}
```



How many calls?

How many additions?

Iterative Fib:

Much Less Computation here!
(How many additions?)

```
int fib( int n)
{ int i=2, res=1, m1=1, m2=1;
  if (n ==0 || n ==1) return res;
  for (  ; i<=n; i++)
  { res = m1 + m2;
    m2 = m1;
    m1 = res;
  }
  return res;
}
main()
{ int n;
  scanf("%d", &n);
  printf(" Fib(%d) = %d \n", n, fib(n));
}
```

```
[ppchak]$ ./a.out
0
 Fib(0) = 1
[ppchak]$ ./a.out
1
 Fib(1) = 1
[ppchak]$ ./a.out
2
 Fib(2) = 2
[ppchak]$ ./a.out
3
 Fib(3) = 3
[ppchak]$ ./a.out
4
 Fib(4) = 5
```

An efficient recursive Fib:

```
int Fib ( int, int, int, int);

main()
{
  int n;
  scanf("%d", &n);
  if (n == 0 || n ==1)
  printf("F(%d) = %d \n", n, 1);
  else
  printf("F(%d) = %d \n", n, Fib(1,1,n,2));
}
```

```
int Fib(m1, m2, n, i)
int m1, m2, n, i;
{
  int res;
  if (n == i)
     res = m1+ m2;
  else
     res = Fib(m1+m2, m1, n, i+1);
  return res;
}
```

Much Less Computation here!
(How many calls/additions?)

Run:

```
int Fib ( int, int, int, int);
main()
{ int n;
  scanf("%d", &n);
  if (n == 0 || n ==1)  printf("F(%d) = %d \n", n, 1);
  else  printf("F(%d) = %d \n", n, Fib(1,1,n,2));
}
int Fib(m1, m2, n, i)
int m1, m2, n, i;
{ int res;
  printf("F: m1=%d, m2=%d, n=%d, i=%d\n",
                                m1,m2,n,i);
 if (n == i)
    res = m1+ m2;
  else
    res = Fib(m1+m2, m1, n, i+1);
 return res; }
```

```
[ppchak]$ ./a.out
3
F: m1=1, m2=1, n=3, i=2
F: m1=2, m2=1, n=3, i=3
F(3) = 3
[ppchak]$ ./a.out
5
F: m1=1, m2=1, n=5, i=2
F: m1=2, m2=1, n=5, i=3
F: m1=3, m2=2, n=5, i=4
F: m1=5, m2=3, n=5, i=5
F(5) = 8
[ppchak]$
```

Static Variables:

```
int Fib (int, int);

main()
{
  int n;
  scanf("%d", &n);
  if (n == 0 || n ==1)
  printf("F(%d) = %d \n", n, 1);
  else
  printf("F(%d) = %d \n", n, Fib(n,2));
}
```

```
int Fib(n, i)
int n, i;
{
  static int m1, m2;
  int res, temp;
  if (i==2) {m1 =1; m2=1;}
   if (n == i)  res = m1+ m2;
  else
     {  temp = m1;
        m1 = m1+m2;
        m2 = temp;
        res = Fib(n, i+1);
     }
  return res;
}
```

Static variables remain in existence rather than coming and going each time a function is activated

Static Variables: See the addresses!

```
int Fib(n, i)
int n, i;
{ static int m1, m2;
  int res, temp;
  if (i==2) {m1 =1; m2=1;}
  printf("F: m1=%d, m2=%d, n=%d,
                  i=%d\n", m1,m2,n,i);
  printf("F: &m1=%u, &m2=%u\n",
                        &m1,&m2);
  printf("F: &res=%u, &temp=%u\n",
                    &res,&temp);
  if (n == i)  res = m1+ m2;
  else {  temp = m1;  m1 = m1+m2;
       m2 = temp;
       res = Fib(n, i+1);    }
  return res;
}
```

```
[ppchak@cse programs]$ ./a.out
5
F: m1=1, m2=1, n=5, i=2
F: &m1=134518656, &m2=134518660
F: &res=3221224516, &temp=3221224512
F: m1=2, m2=1, n=5, i=3
F: &m1=134518656, &m2=134518660
F: &res=3221224468, &temp=3221224464
F: m1=3, m2=2, n=5, i=4
F: &m1=134518656, &m2=134518660
F: &res=3221224420, &temp=3221224416
F: m1=5, m2=3, n=5, i=5
F: &m1=134518656, &m2=134518660
F: &res=3221224372, &temp=3221224368
F(5) = 8
```