Özyineleme

Suhap SAHIN
Onur GÖK

Özyineleme

Tanım:

- **Kendi kendini** çağıran fonksiyonlara özyineli (rekürsif-recursive) fonksiyon denilir. (TDK:Bir yordamın kendini çağırabilme özelliği)
- Problemleri **daha basit alt problemlere bölerek** çözme yöntemi.
- Özyinelemeli fonksiyon doğrudan veya dolaylı olarak kendisini çağıran fonksiyondur.

Yineleme: döngü:iteratif : Iterative algoritma döngü yapısını kullanır.
Özyineleme:özdöngü:recursif: Özyineleme algoritması dallanma (branching) algoritmasını kullanır

Özyineleme algoritma değildir. Algoritma tasarlanırken kullanılabilecek bir programlama tekniğidir.
Her özyinelemeli olarak tanımlanmış problemin iterative çözümüne geçiş yapılabilir.

Fonksiyon özyineli olarak her çağrılışında yerel değişkenler ve parametreler için bellekte yer ayrılır.

Böl Ve Yönet

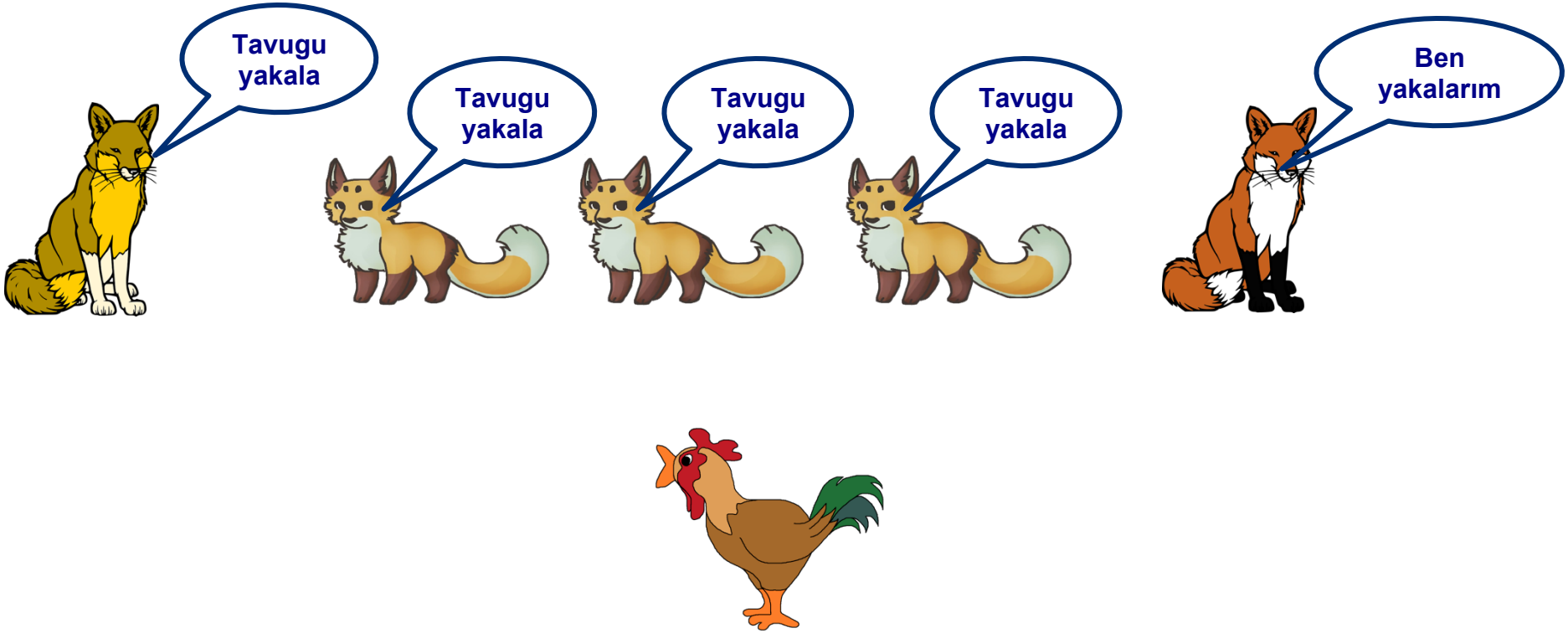
●Problem kendisine benzer küçük boyutlu alt problemlere bölünür. Alt problemler çözülür ve bulunan çözümler birleştirilir.

●Divide: Problem iki veya daha fazla alt probleme bölünür

● Conquer:Divide-and-conquerözyinelemeli (recursively) olarak kullanılarak alt problemleri çözer

● Combine: Alt problemlerin çözümleri alınır ve orijinal problemin çözümü olacak şekilde birleştirilir.

Özyineleme


Tavugu yakala
Tavugu yakala
Tavugu yakala
Tavugu yakala
Ben yakalarım

Özellikler

```
int function(int sayi) {
    if(sayi < 1)
        return 0;
printf("%d",sayi);
function(sayi - 1);
}
```

temel durum

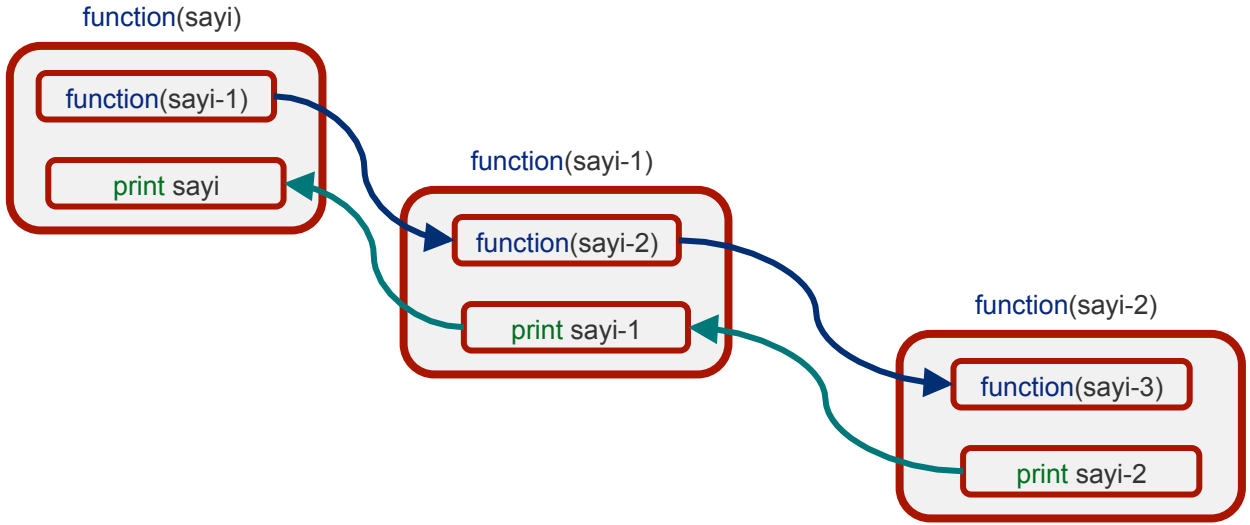
Bileşenler:

- Temel durum
- Rekürsif tanımlama
- Temel duruma indirme mekanizması

Rekürsif tanımlama ve ilerleme durumu

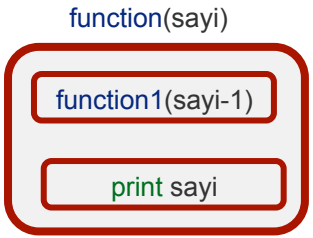
Dogrudan Özyineleme

```
int function(int sayi) {

   if(sayi < 0)
      return;

   function(sayi - 0);

   printf("%d",sayi);
}
```



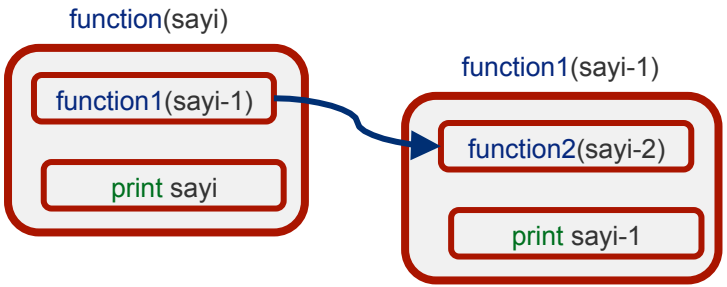
Dolaylı Özyineleme

```
int function(int sayi) {
   if(sayi < 0)
      return;
   function1(sayi - 1);
   printf("%d ",sayi);
}
```



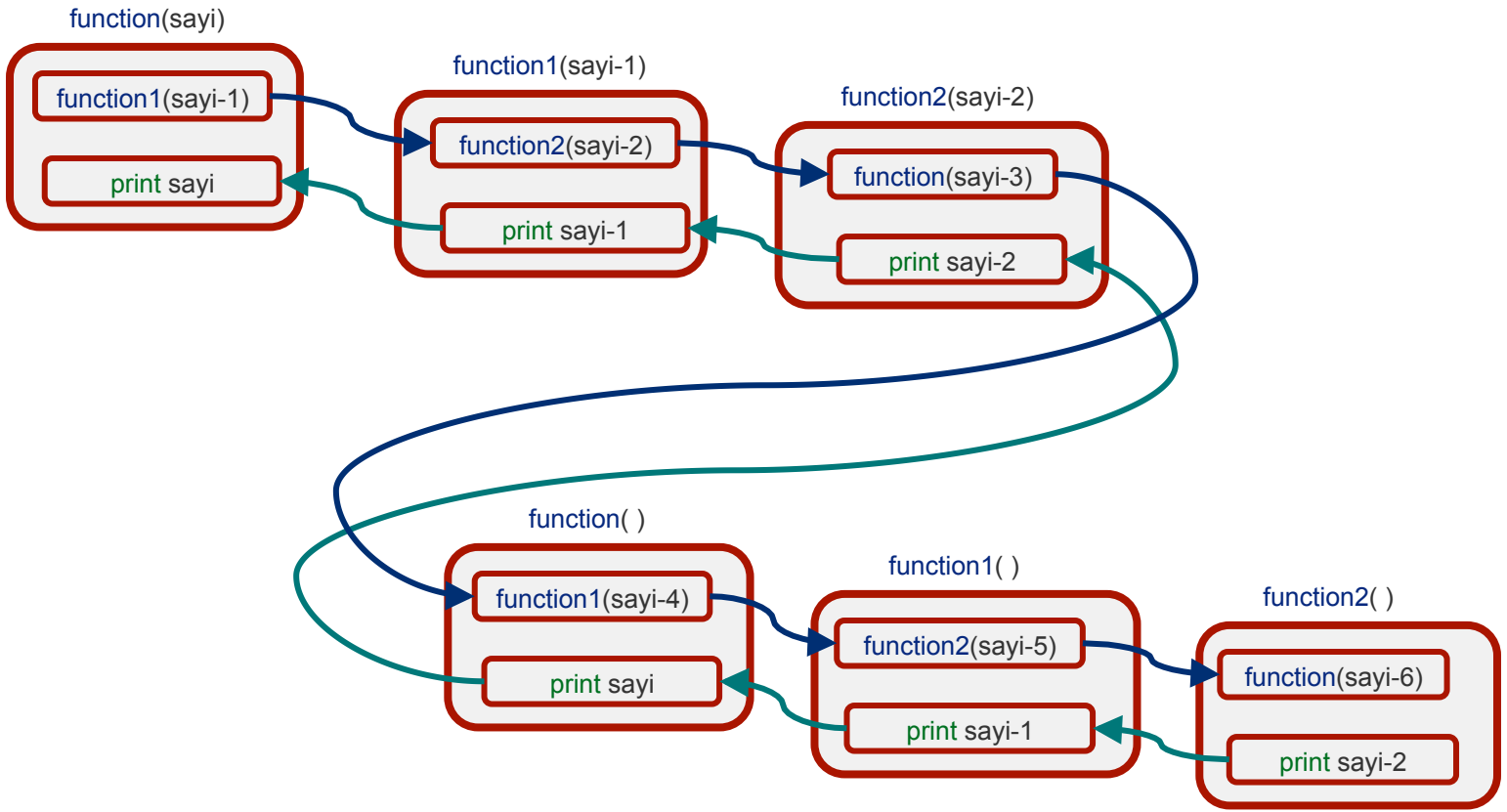
Dolaylı Özyineleme

```
int function(int sayi) {
   if(sayi < 0)
      return;
   function1(sayi - 1);
   printf("%d ",sayi);
}

int function1(int sayi) {
   if(sayi < 0)
      return;
   function2(sayi - 1);
   printf("%d ",sayi);
}
```



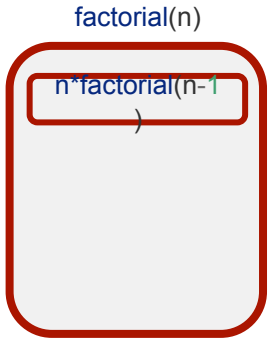
Dolaylı Özyineleme

```
int function(int sayi) {
   if(sayi < 0)
      return;
   function1(sayi - 1);
   printf("%d ",sayi);
}

int function1(int sayi) {
   if(sayi < 0)
      return;
   function2(sayi - 1);
   printf("%d ",sayi);
}

int function2(int sayi) {
   if(sayi < 0)
      return;
   function(sayi - 1);
   printf("%d ",sayi);
}
```



Tekil Özyineleme

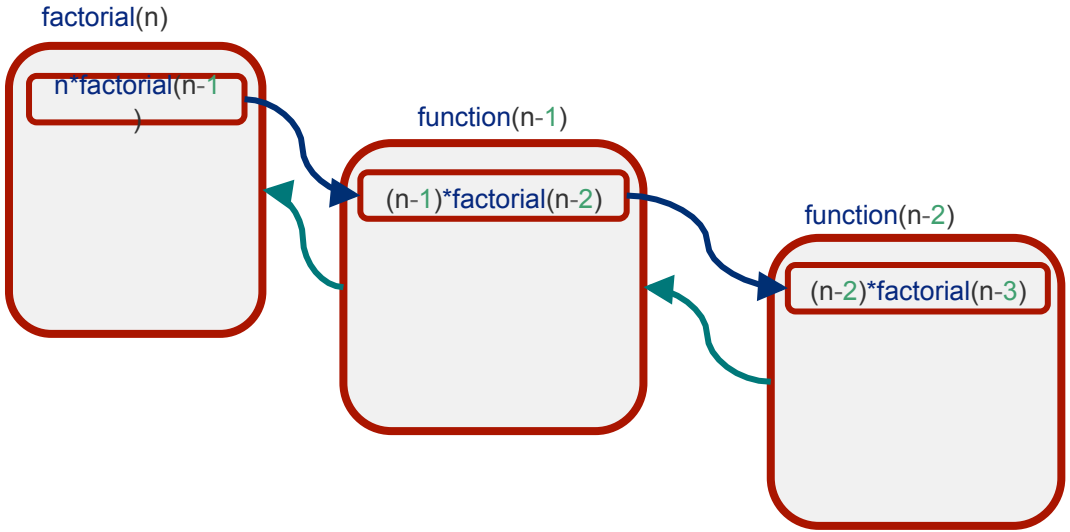
```
#include<stdio.h>
```



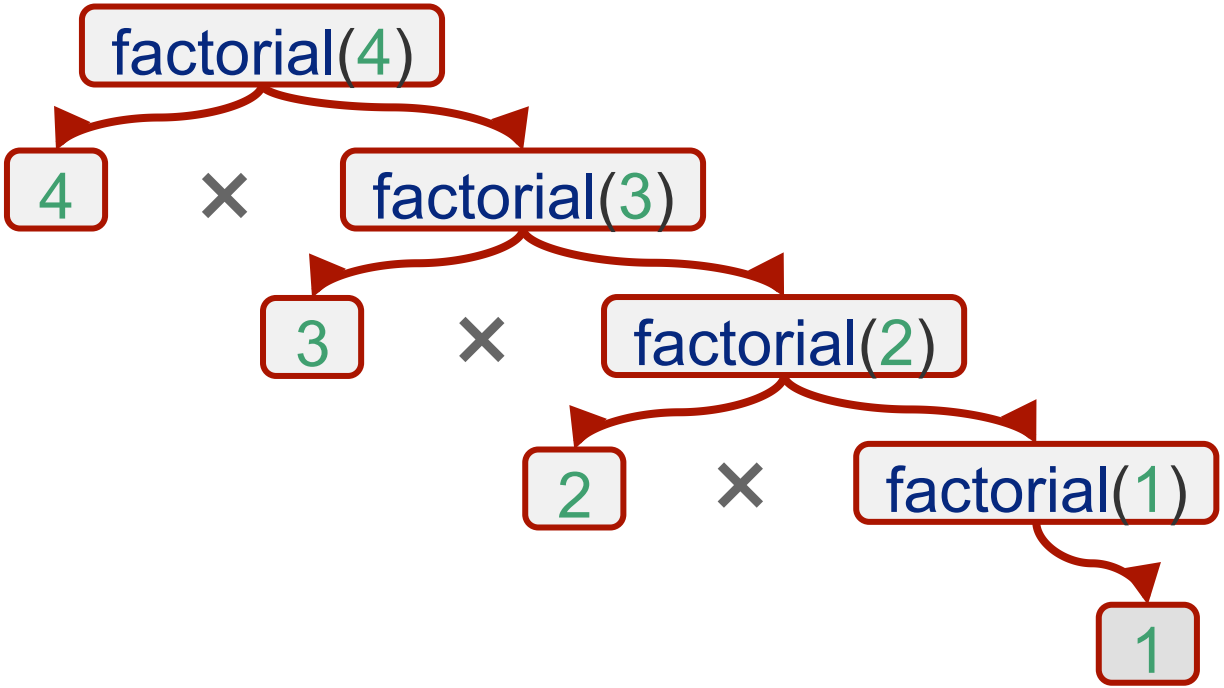

```
int main(){
    printf("%d",factorial(4));
}
```

Tekil Özyineleme

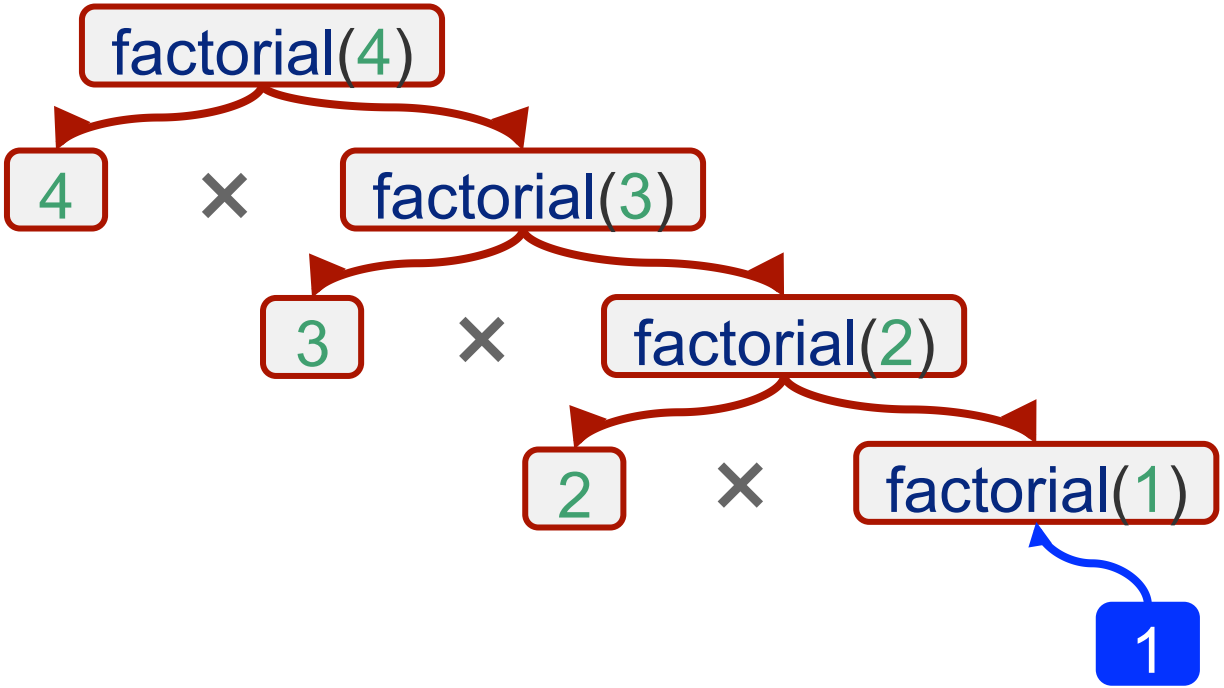
```
#include<stdio.h>
int factorial( int n ){
        if ( n <= 1 ){
                return 1;
        }else{
                return n*factorial(n-1);
        }
}

int main(){
    printf("%d",factorial(4));
}
```



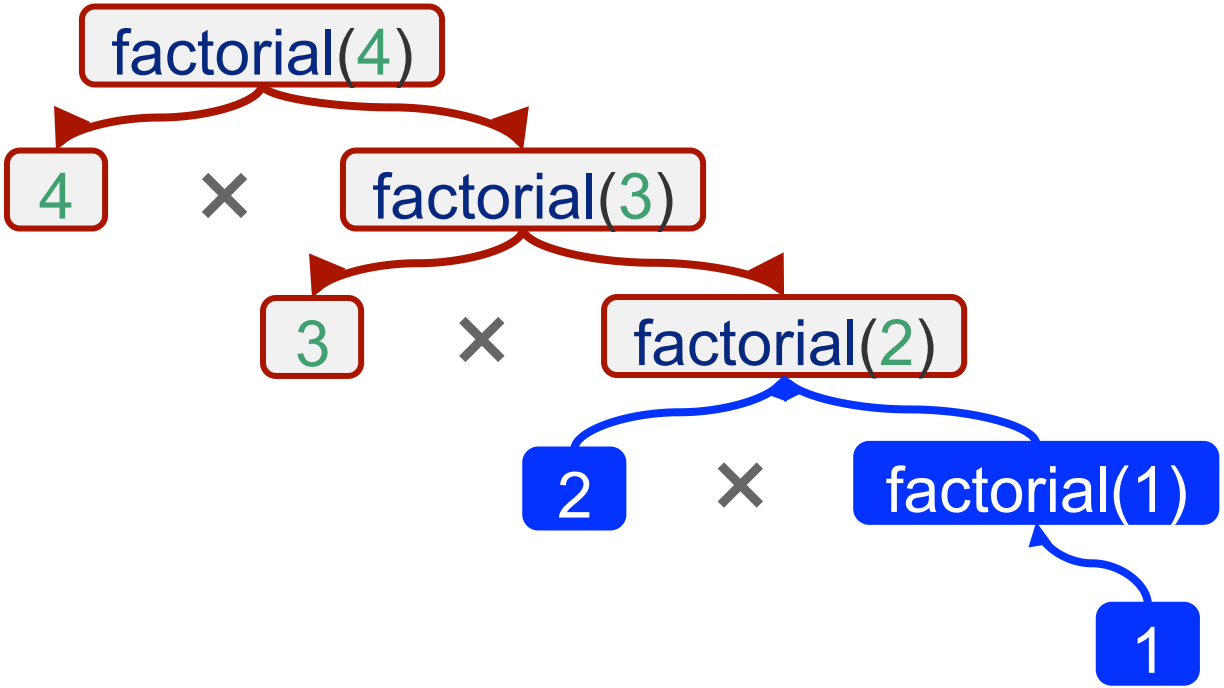
Faktöriyel



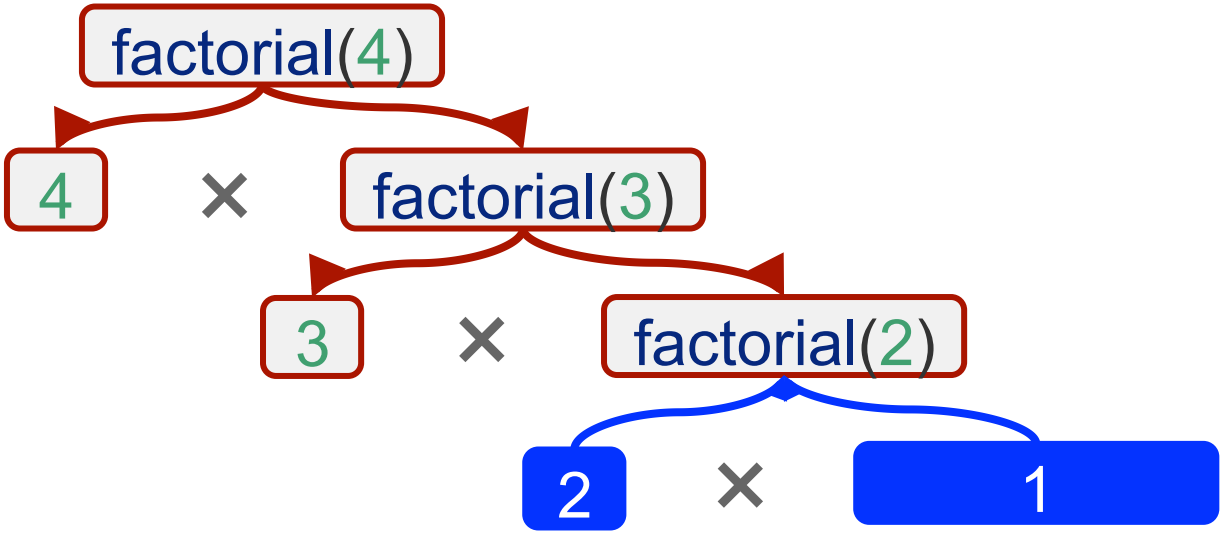
Faktöriyel



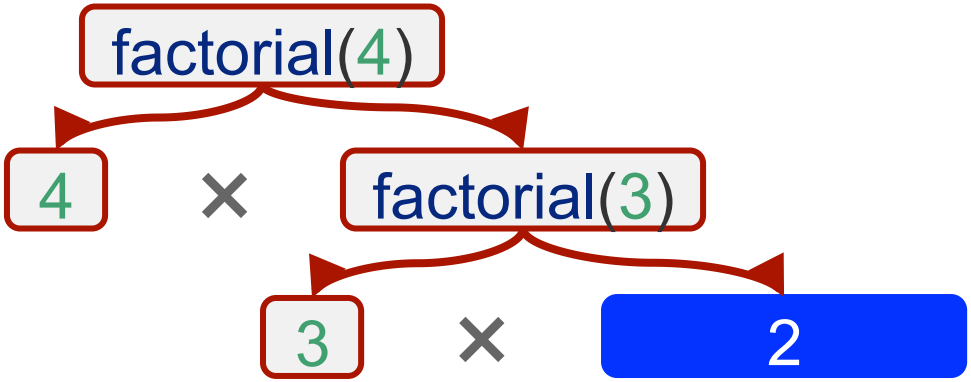
Faktöriyel



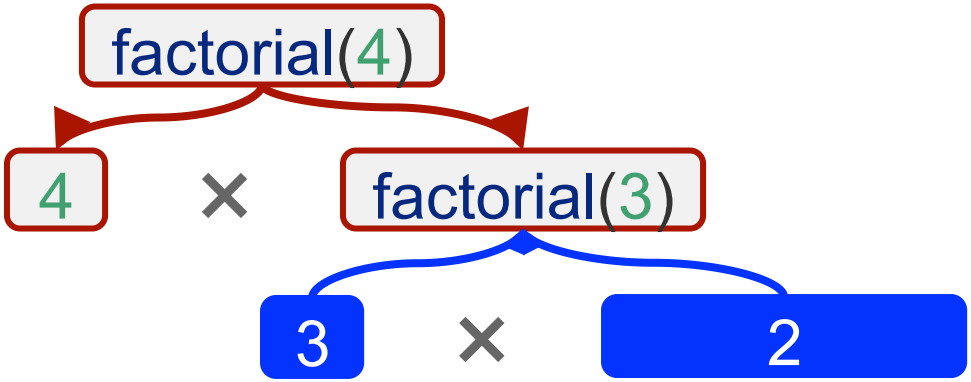
Faktöriyel



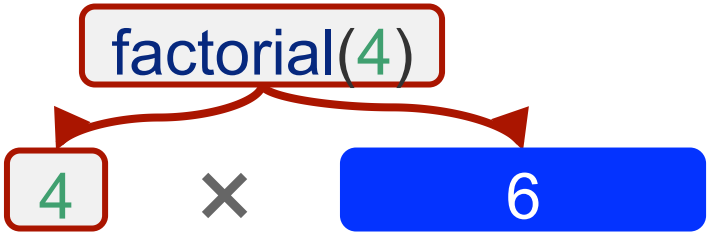
Faktöriyel



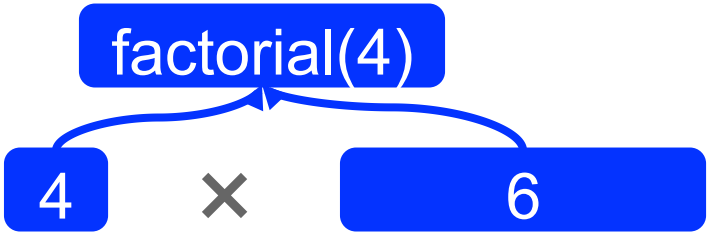
Faktöriyel



Faktöriyel



Faktöriyel



Faktöriyel

24

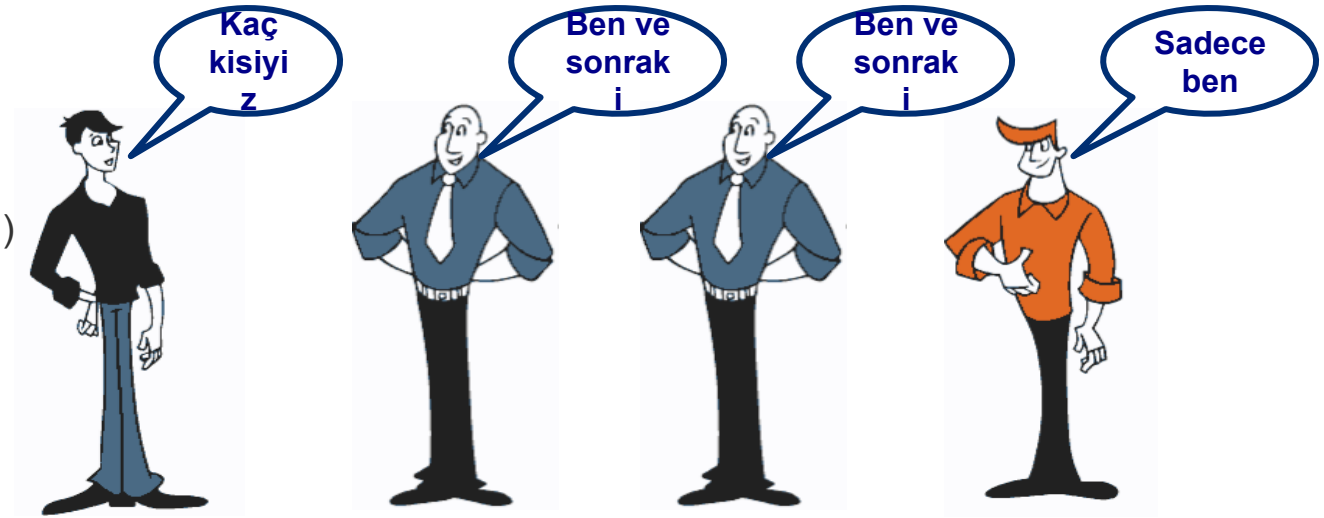
Kaç Kisiyiz

```
#include<stdio.h>
```


```
int main()
{
    kackisi(3);
}
```

Kaç Kisiyiz

```
#include<stdio.h>
int kackisi( int n ){
        if ( n <= 1 ){
    printf("Sadece ben");
                return 1;
        }else{
    printf("Ben ve sonraki, ");
                kackisi(n-1)
                return 1;
        }
}

int main()
{
   kackisi(3);
}
```



ikili Özyineleme

Fibonacci Sayıları

1 1 2 3 5 8 13 21

ikili Özyineleme

```
#include<stdio.h>

int main (){
  int n = 4;
  printf("%d", fib(n));
  getchar();
  return 0;
}
```

Fibonacci Dizisi 0, 1, 1, 2, 3, 5, 8, 13, 21,...

{

eger n < = 1

n;

f(n)

f(n-1) +

f(n-2); aksi halde

ikili Özyineleme

```
#include<stdio.h>
int fib(int n){
   if (n <= 1)
      return n;
   return fib(n-1) + fib(n-2);
}

int main (){
  int n = 4;
  printf("%d", fib(n));
  getchar();
  return 0;
}
```

Fibonacci Dizisi **0, 1, 1, 2, 3, 5, 8, 13, 21,...**

{

eger n < = 1

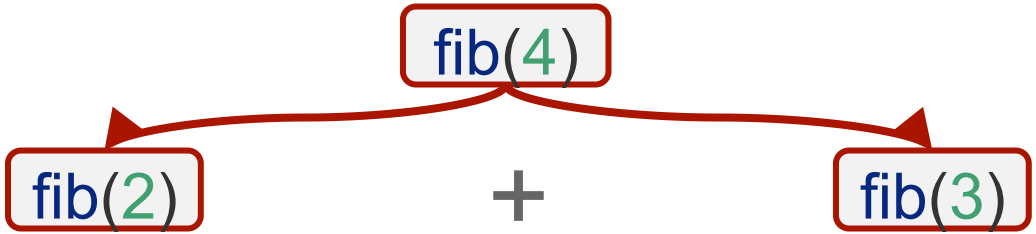
n;

f(n)

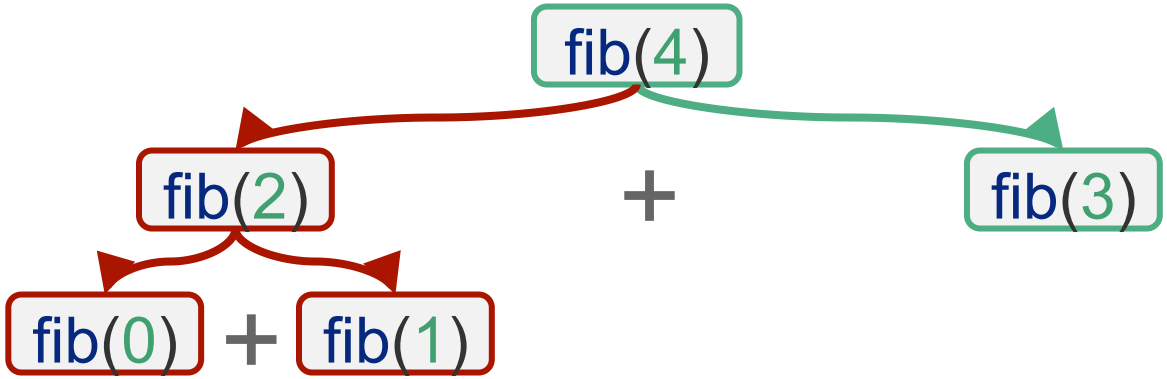
f(n-1) +

f(n-2); aksi halde

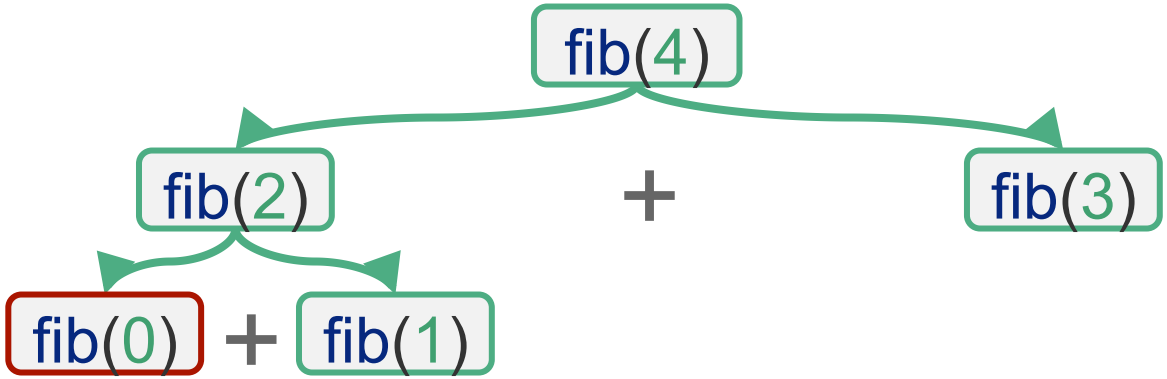
ikili Özyineleme



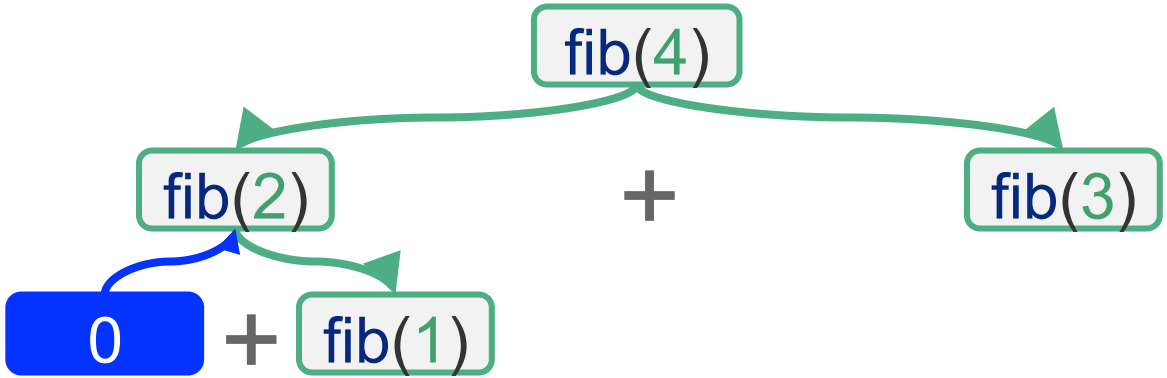
ikili Özyineleme



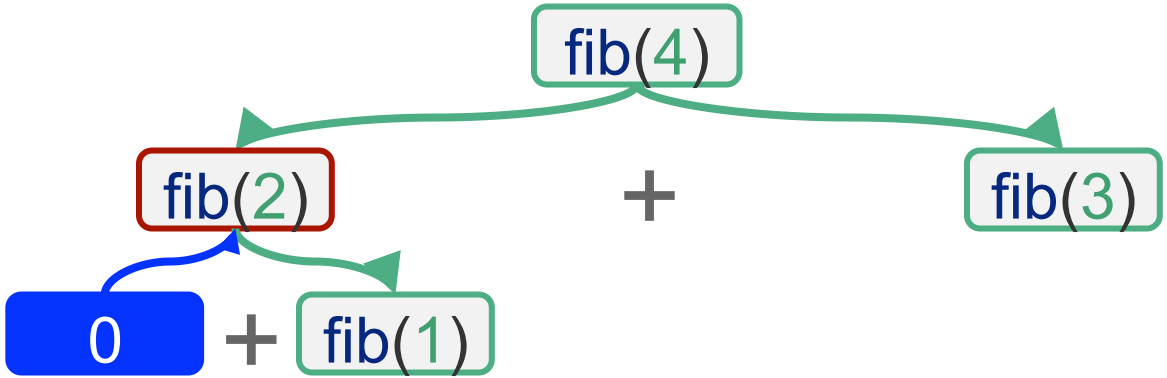
ikili Özyineleme



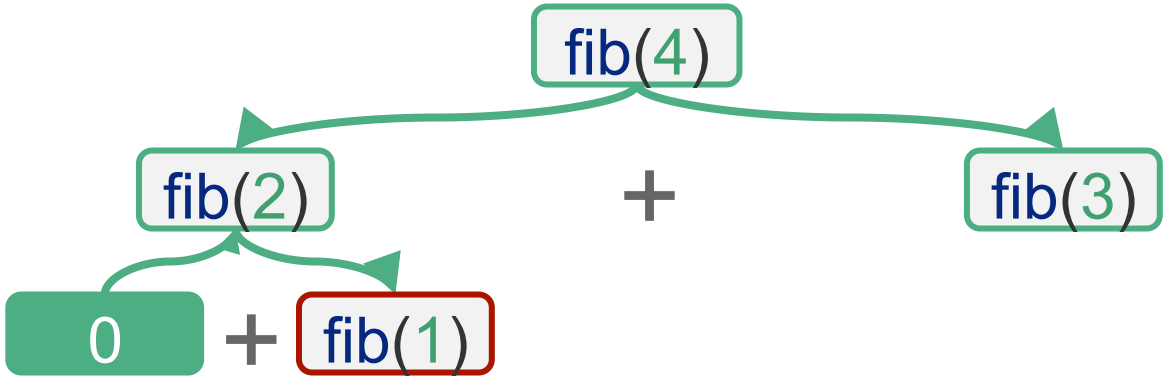
ikili Özyineleme



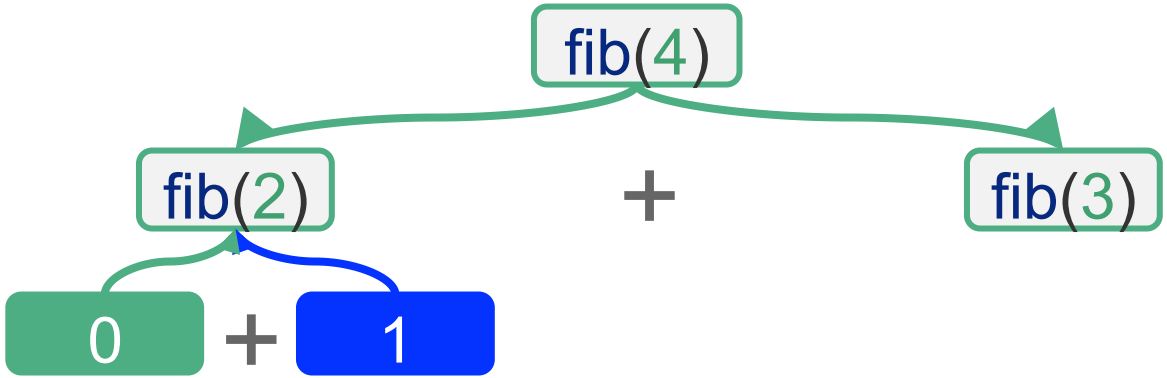
ikili Özyineleme



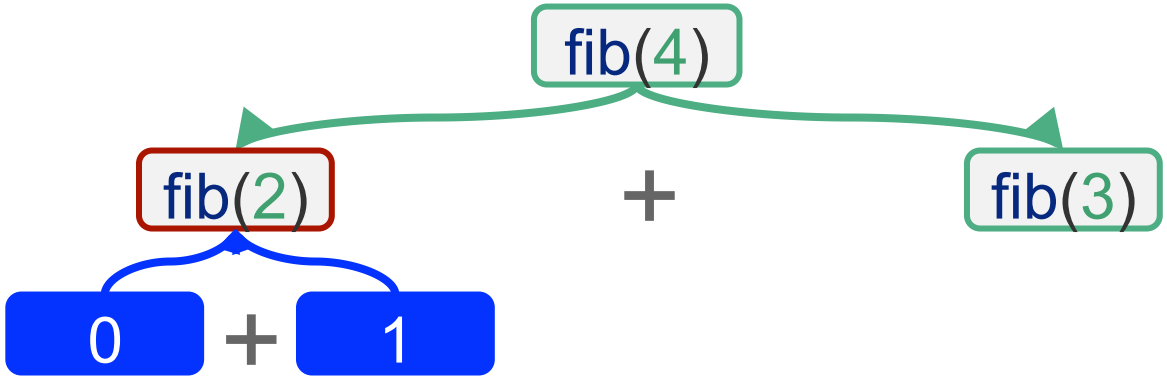
ikili Özyineleme



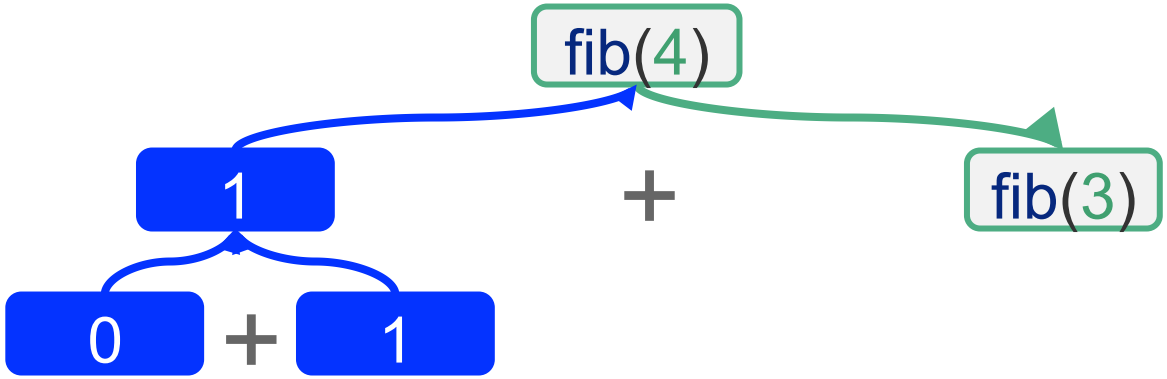
ikili Özyineleme



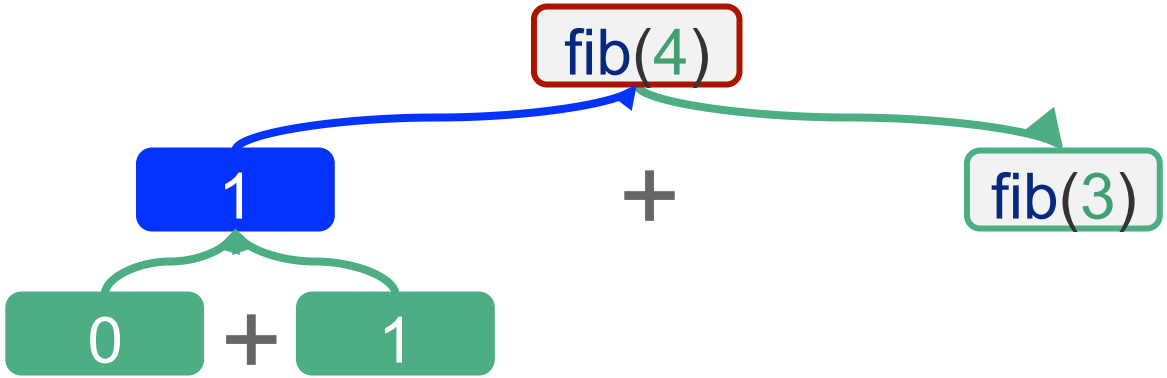
ikili Özyineleme



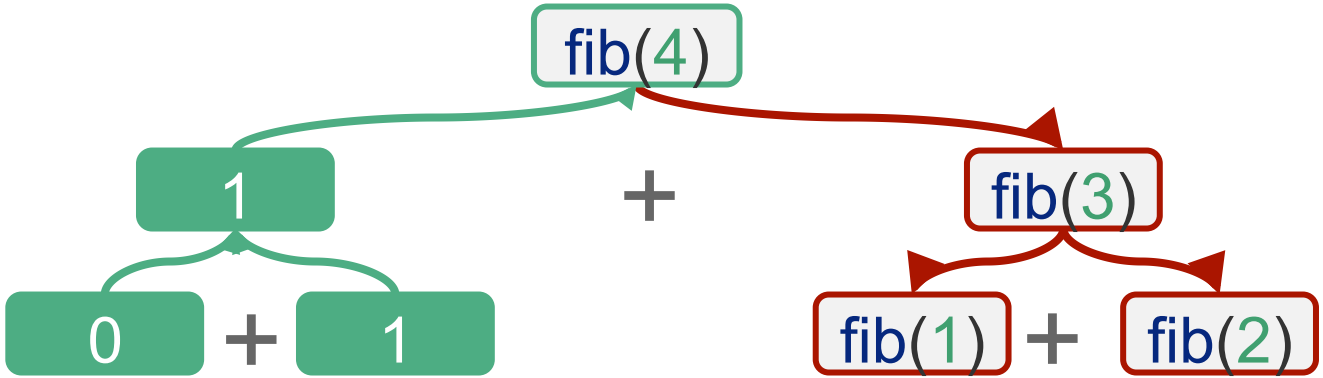
ikili Özyineleme



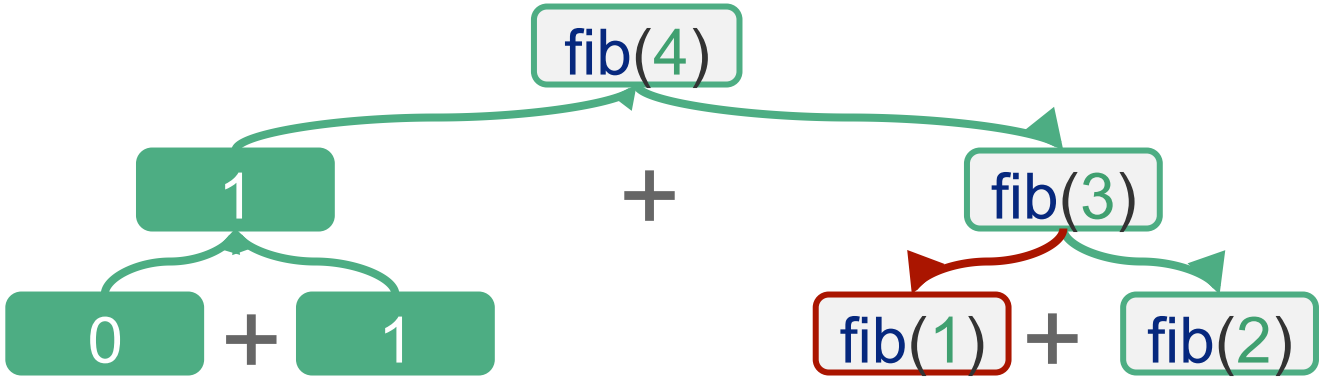
ikili Özyineleme



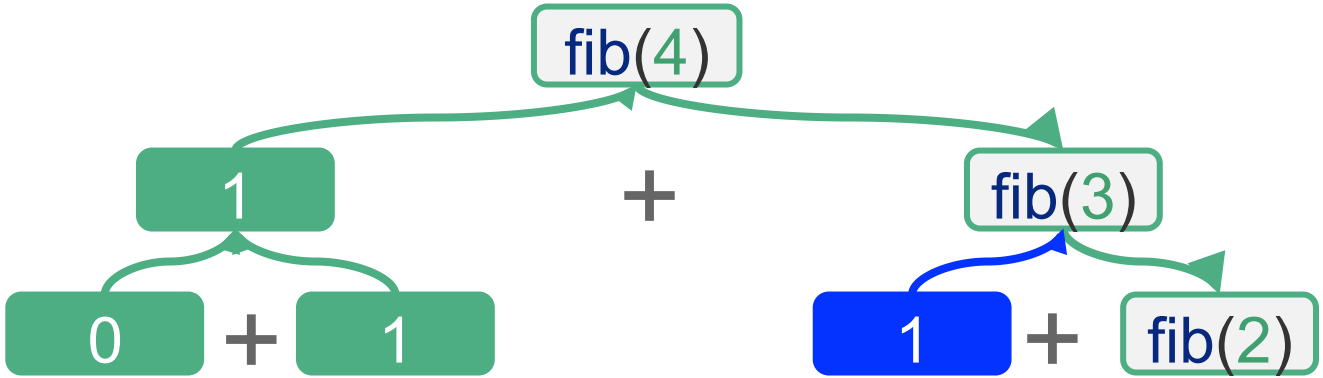
ikili Özyineleme



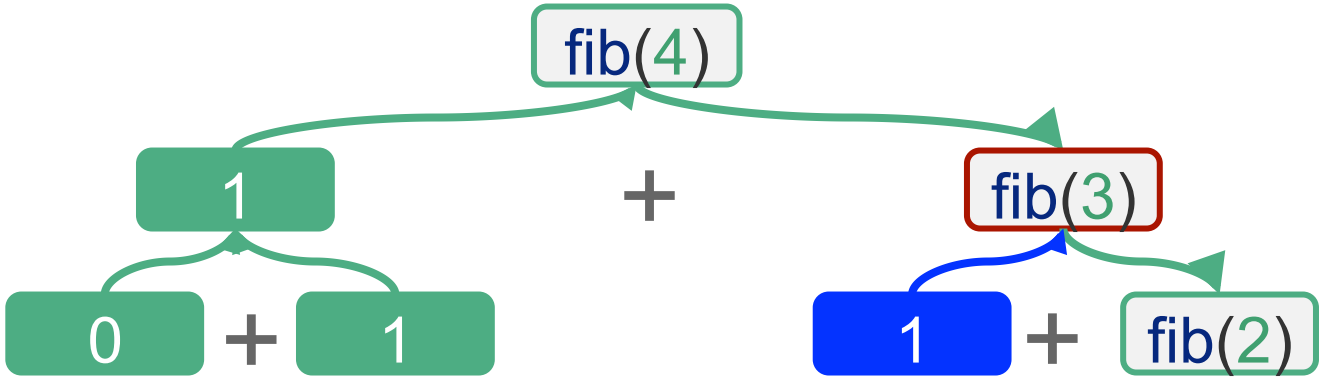
ikili Özyineleme



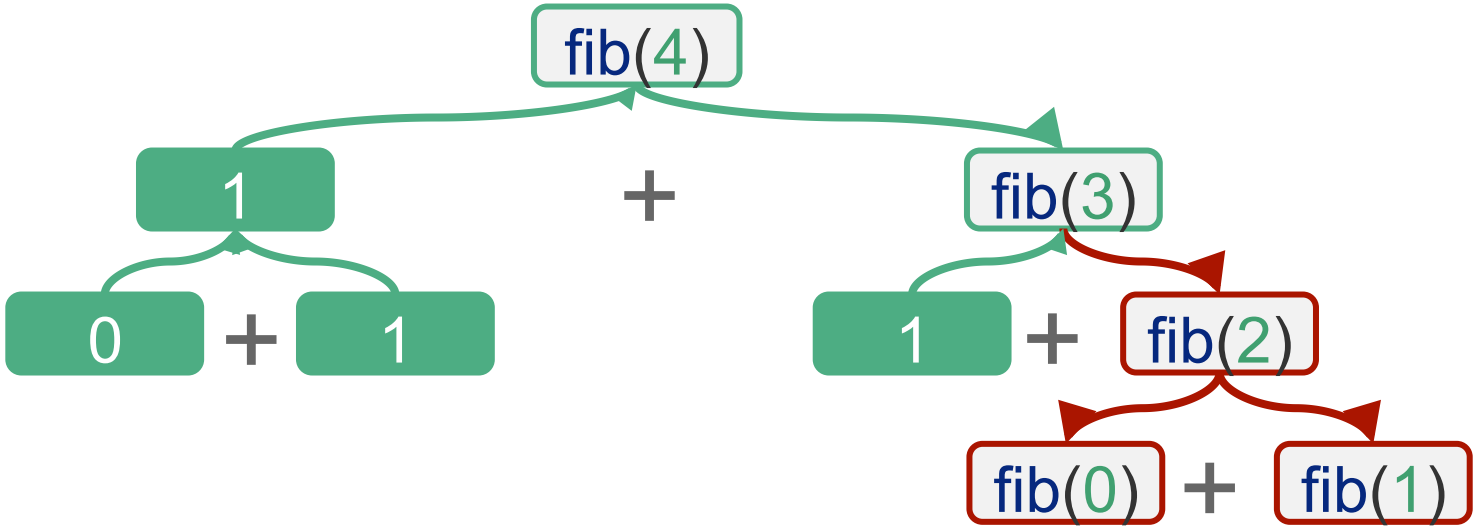
ikili Özyineleme



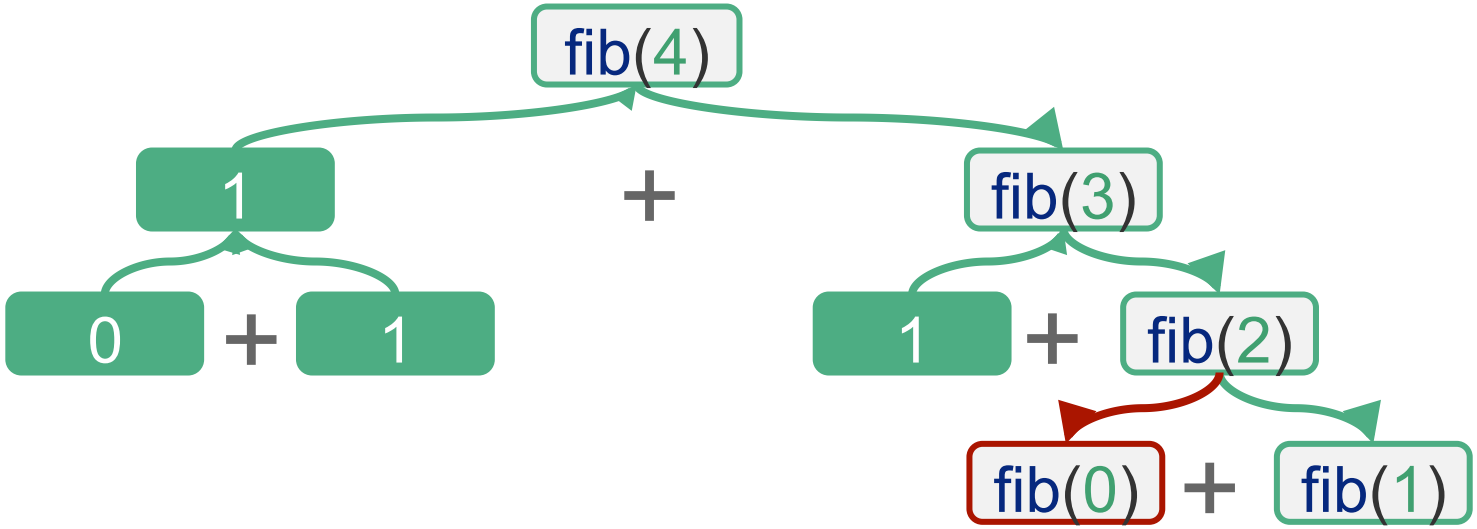
ikili Özyineleme



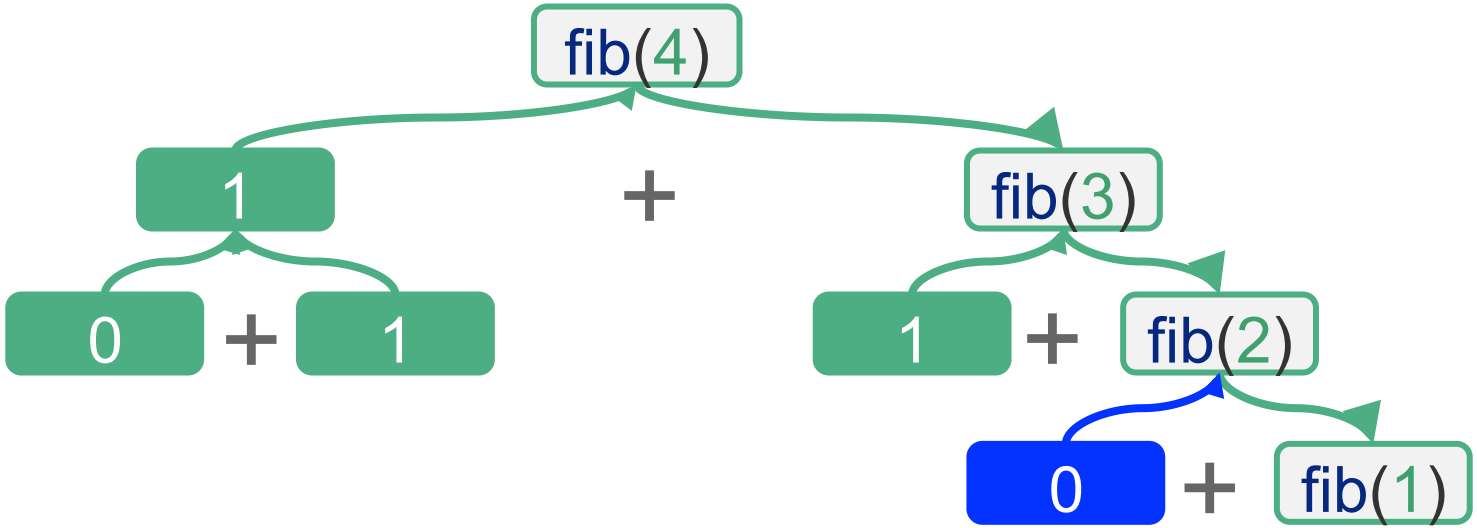
ikili Özyineleme



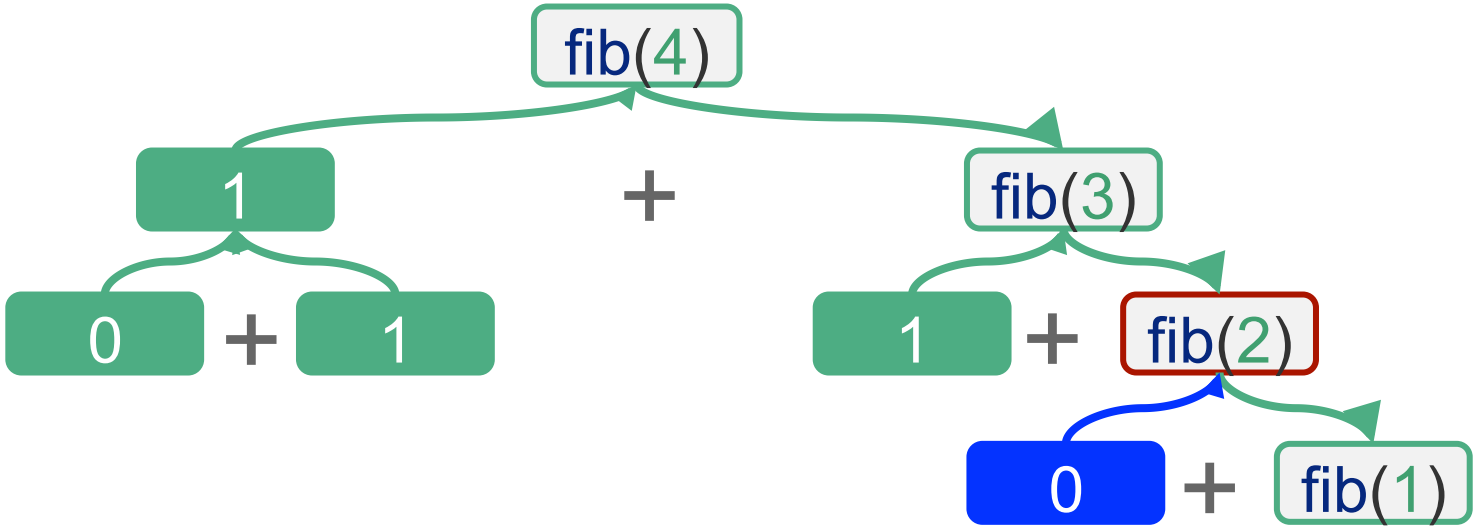
ikili Özyineleme



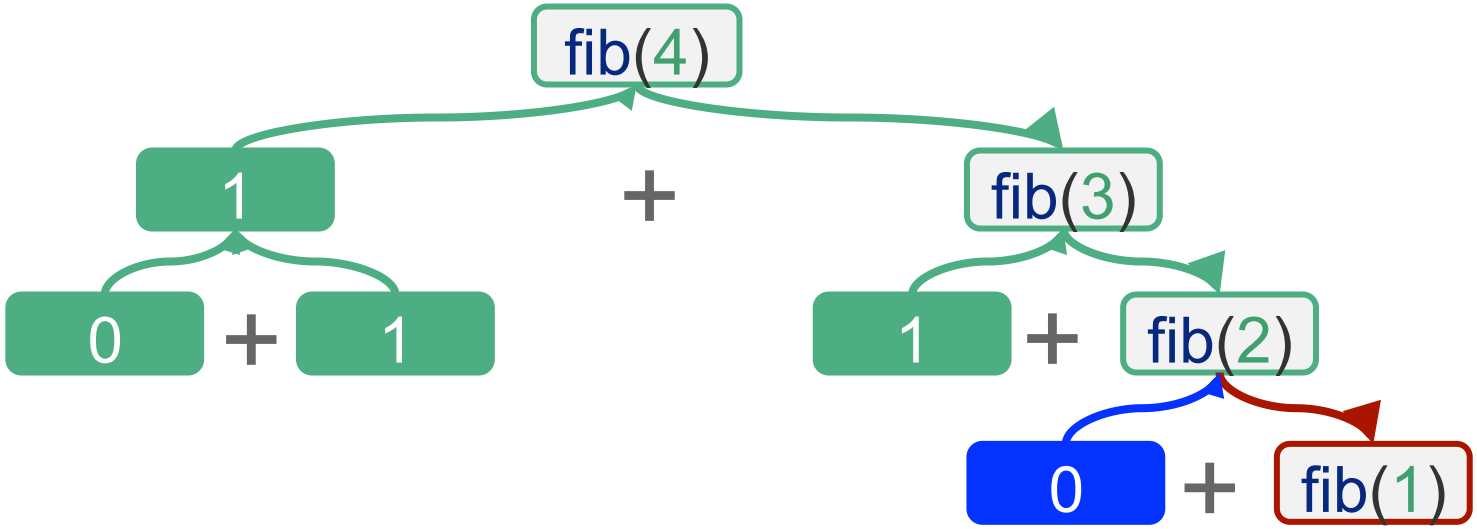
ikili Özyineleme



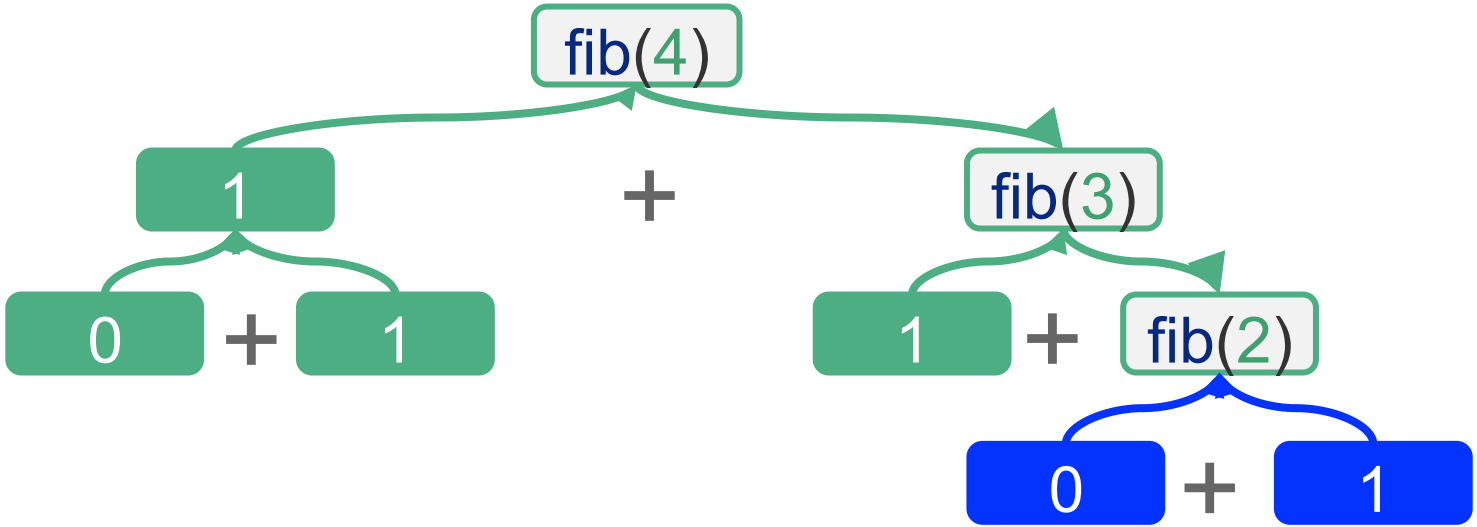
ikili Özyineleme



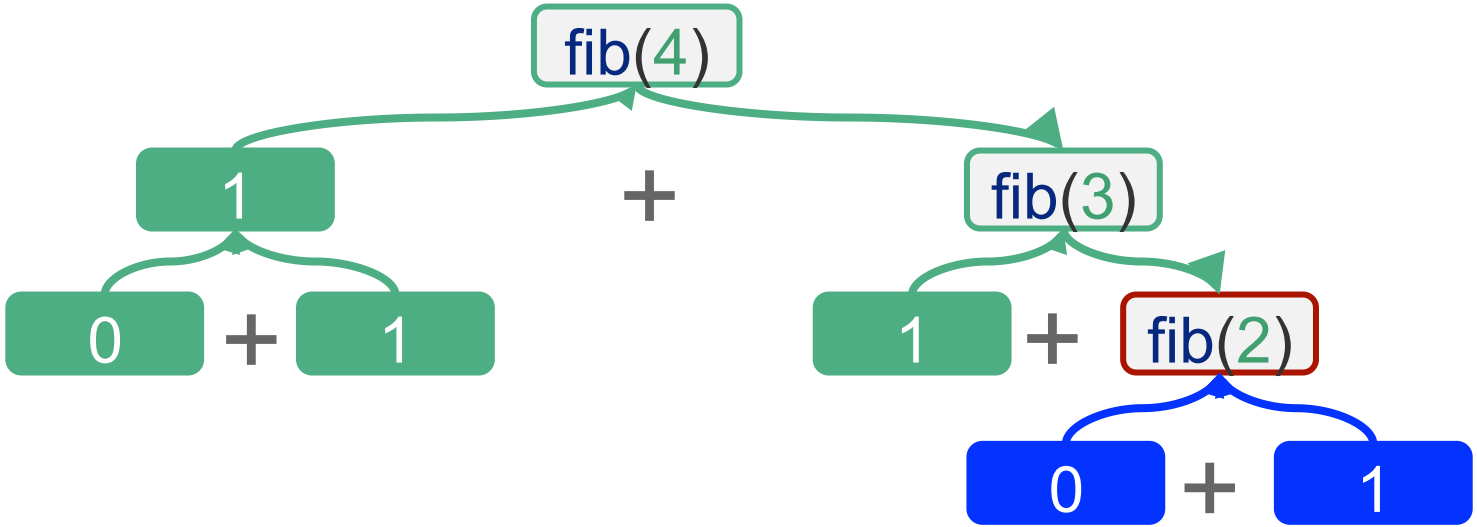
ikili Özyineleme



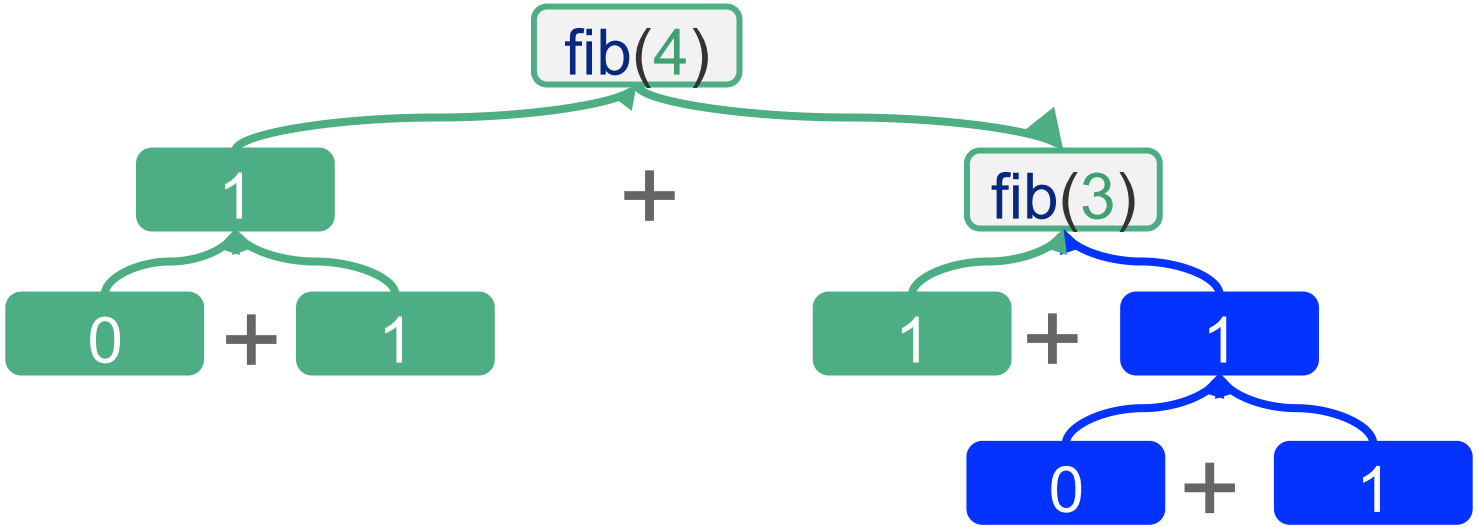
ikili Özyineleme



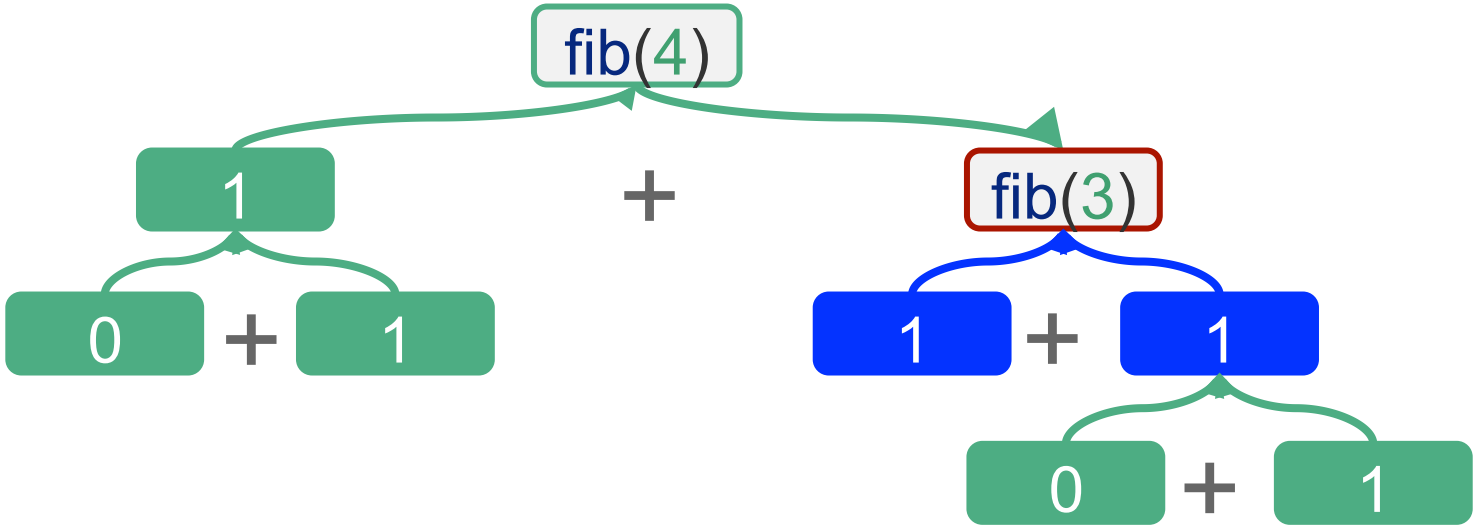
ikili Özyineleme



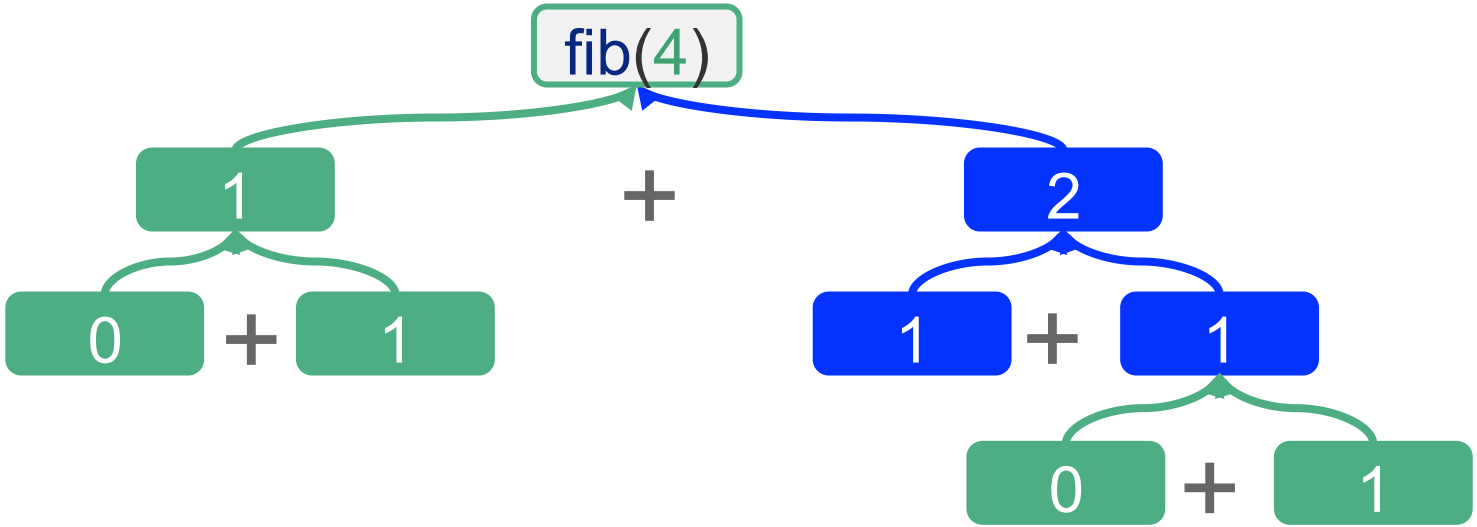
ikili Özyineleme



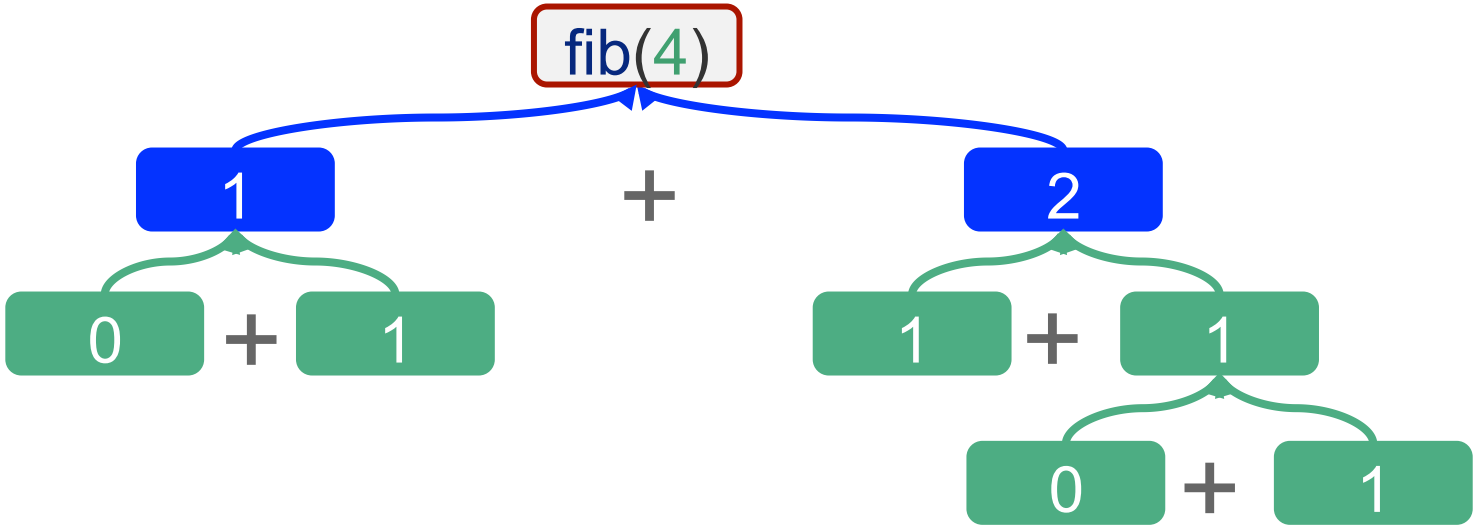
ikili Özyineleme



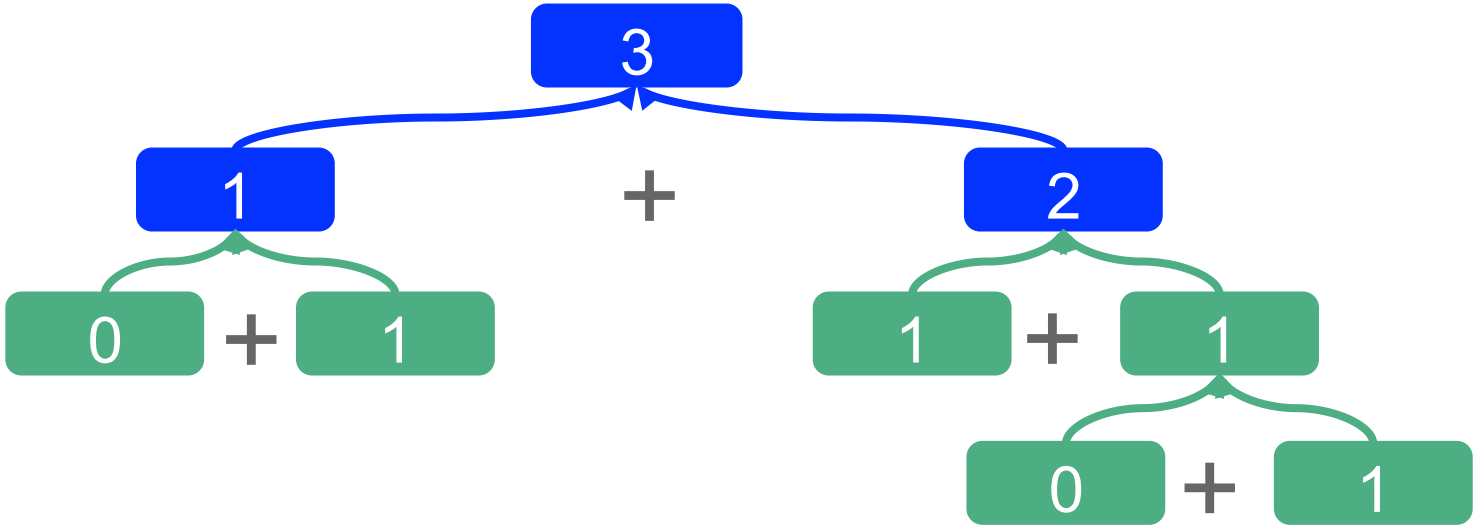
ikili Özyineleme



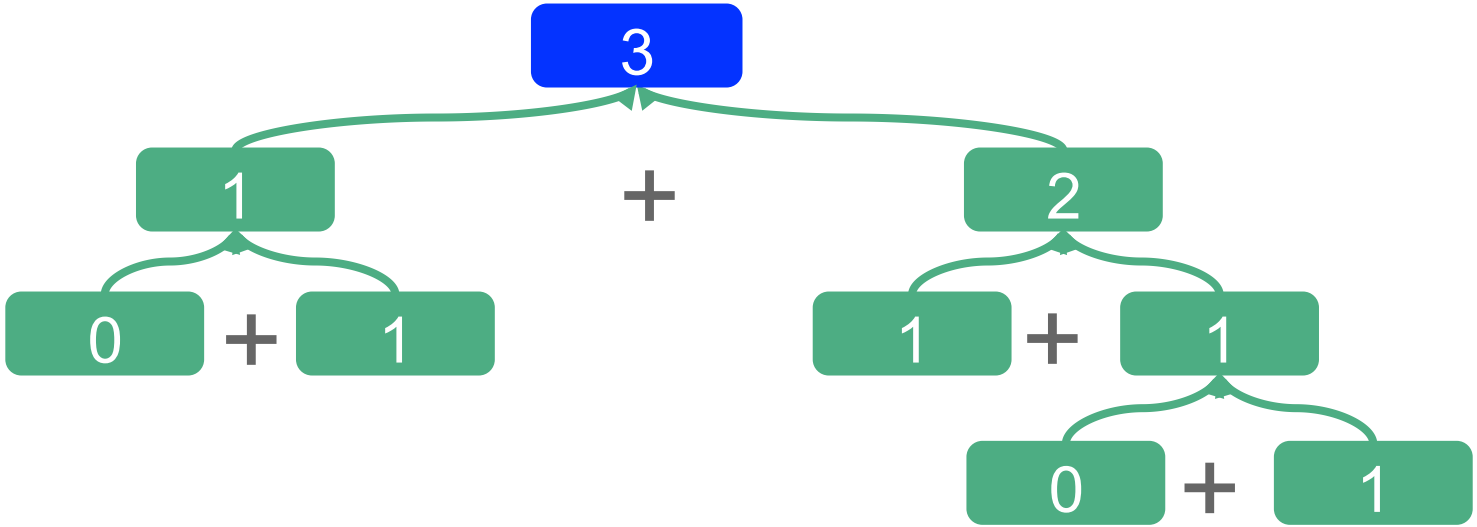
ikili Özyineleme



ikili Özyineleme

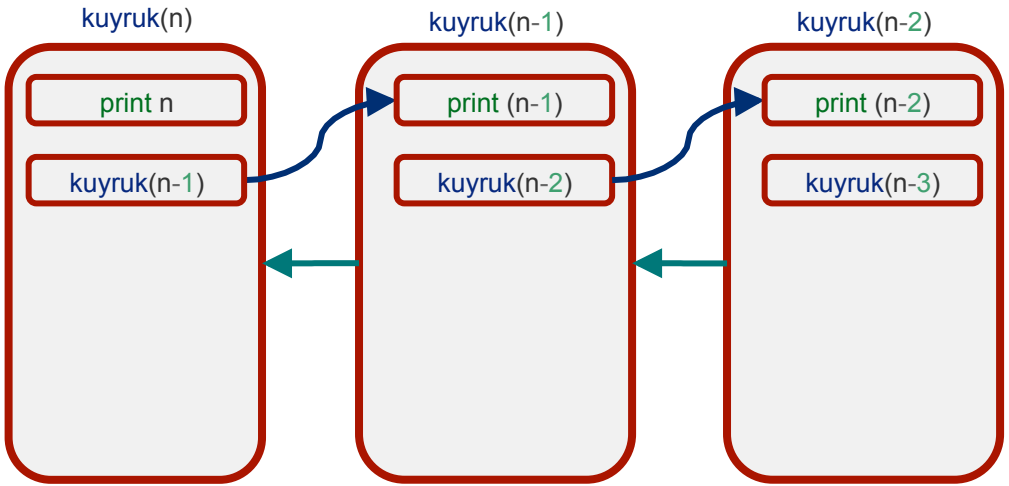


ikili Özyineleme



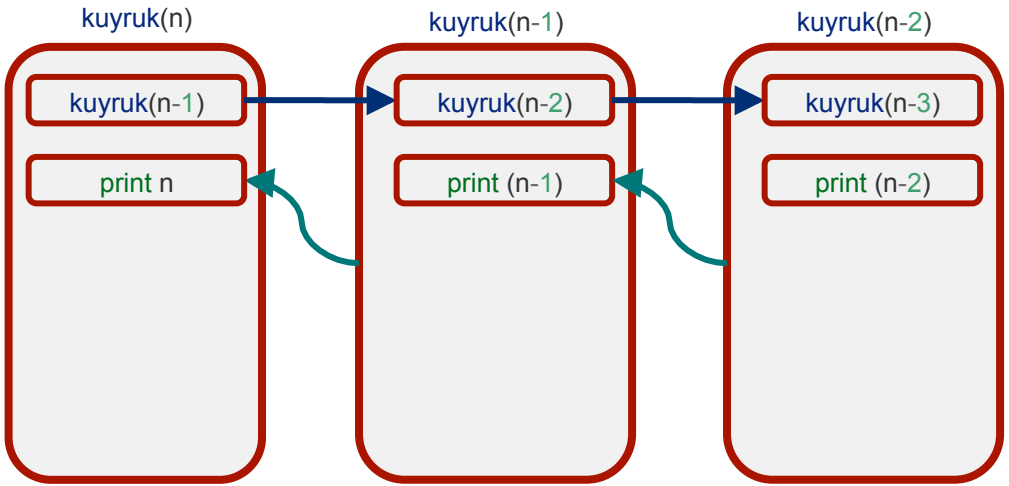
Kuyruk Özyineleme

```
#include<stdio.h>
void kuyruk(int n){
    if (n < 1)
        return;
    else{
        printf(" %d\n",n);
        kuyruk(n-1);
        return;
    }
}
int main(){
    int n = 2;
    kuyruk(n);
}
```



Kuyruk Olmadan Özyineleme

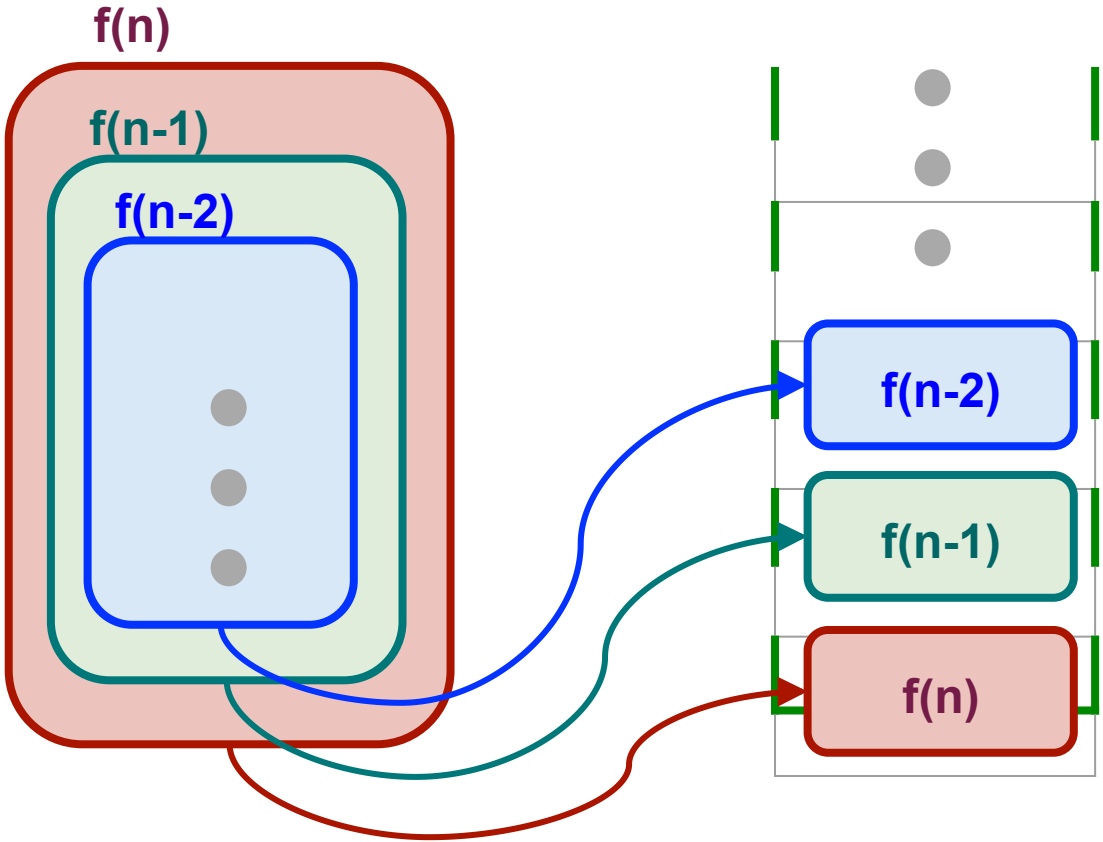
```c
#include<stdio.h>
void kuyruk(int n){
    if (n < 1)
        return;
    else{
        kuyruk(n-1);
        printf(" %d\n",n);
        return;
    }
}
int main(){
    int n = 2;
    kuyruk(n);
}
```



içiçe Özyineleme

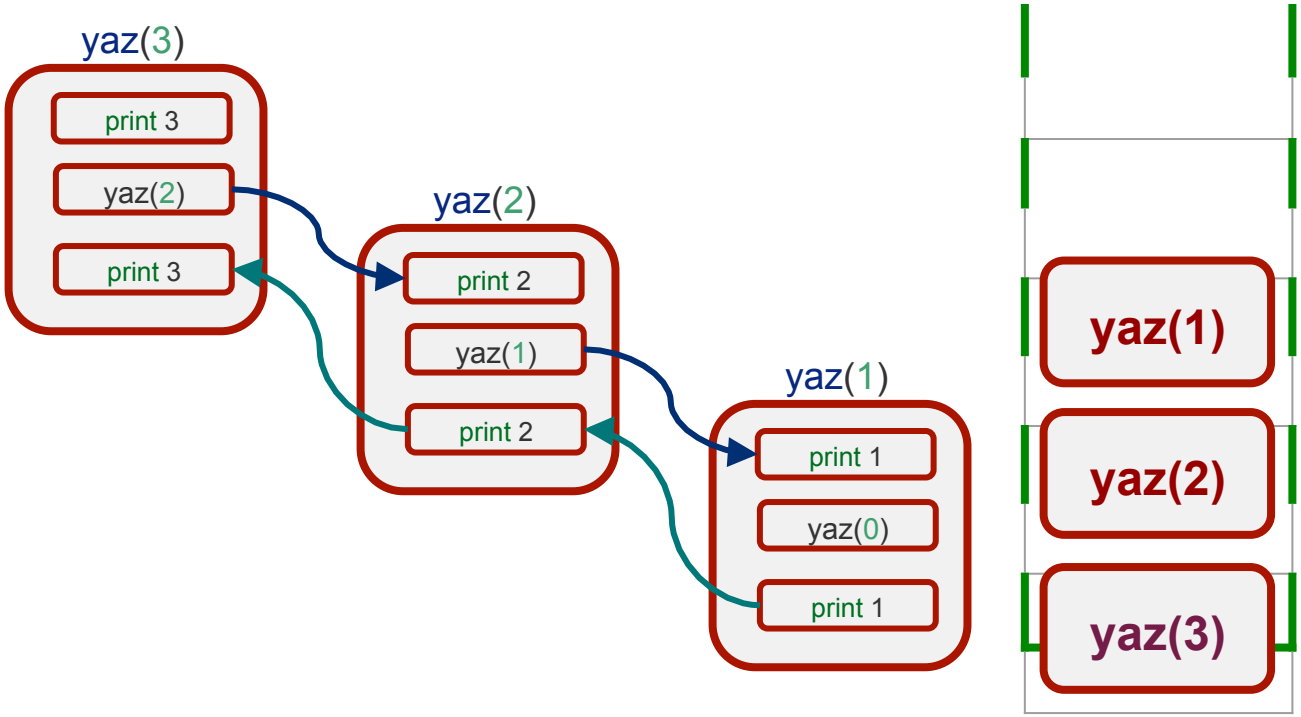
```
public int A(int n, int m)
{
        if (n <= 0) return 1;
        return A(n-1, A(n-1, m-1));
}
```

Bellek Kullanımı



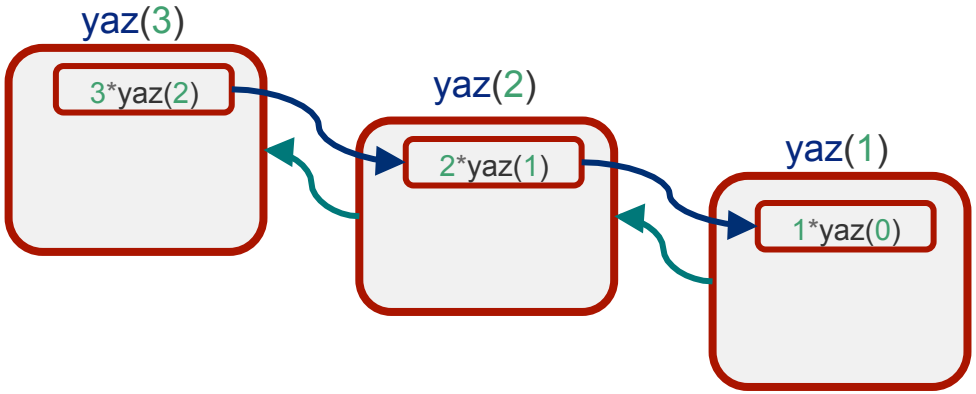
Bellek Kullanımı

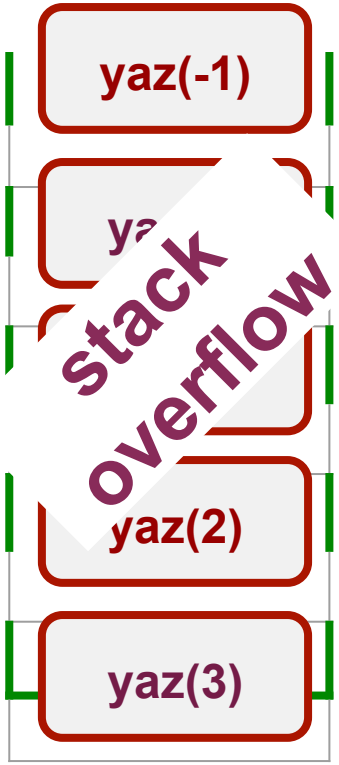
```
#include<stdio.h>
void yaz(int test)
{
    if (test < 1)
        return;
    else
    {
        printf("%d ",test);
        yaz(test-1);
        printf("%d ",test);
        return;
    }
}
int main()
{
    int test = 3;
    yaz(test);
}
```



Bellek Kullanımı

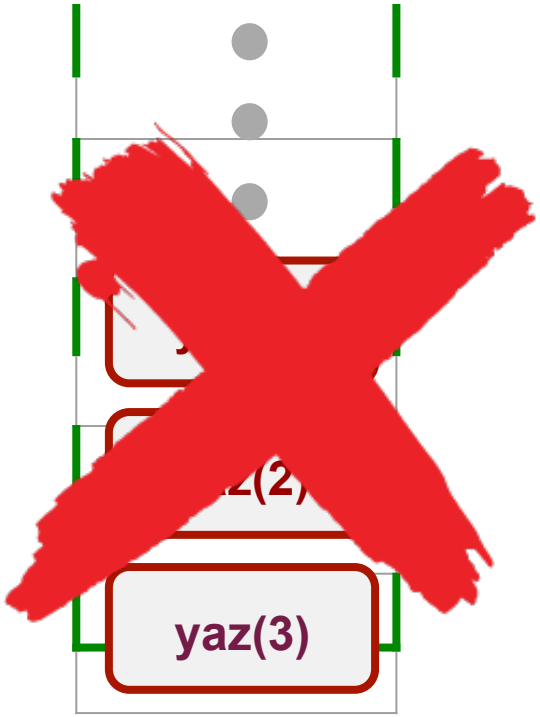
```
int fact(int n)
{
    if (n == 1)
        return 1;

    else
        return n*fact(n-1);
}
```



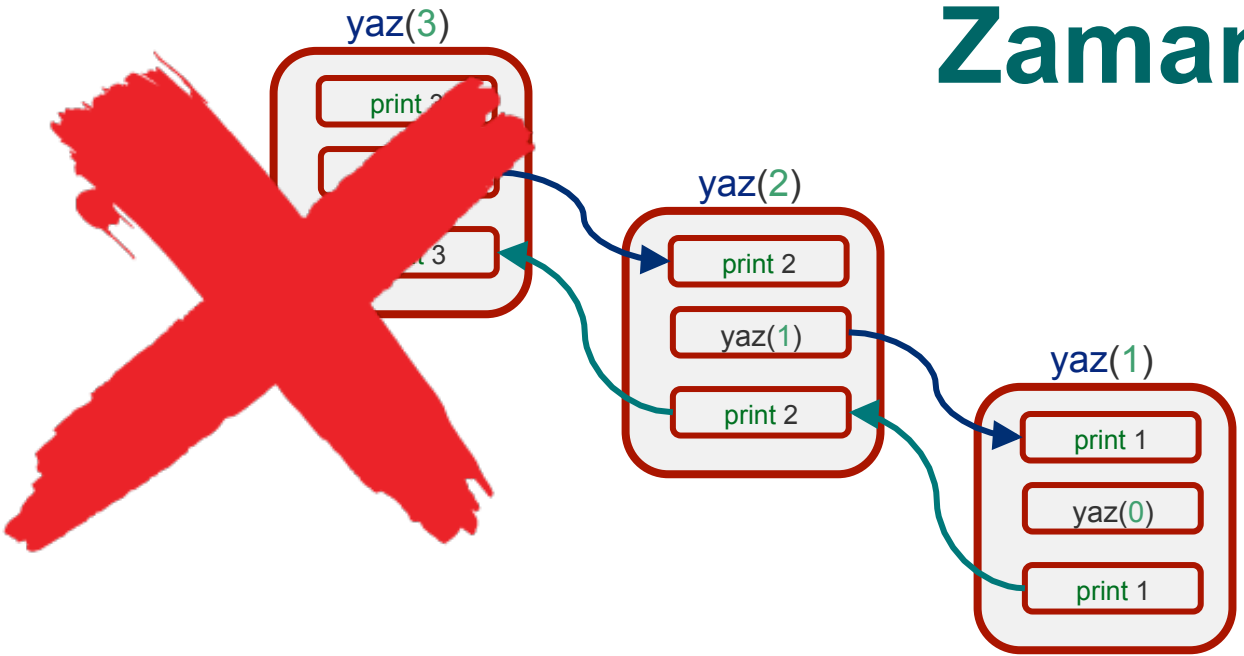


Recursive Vs iteratif

Yer

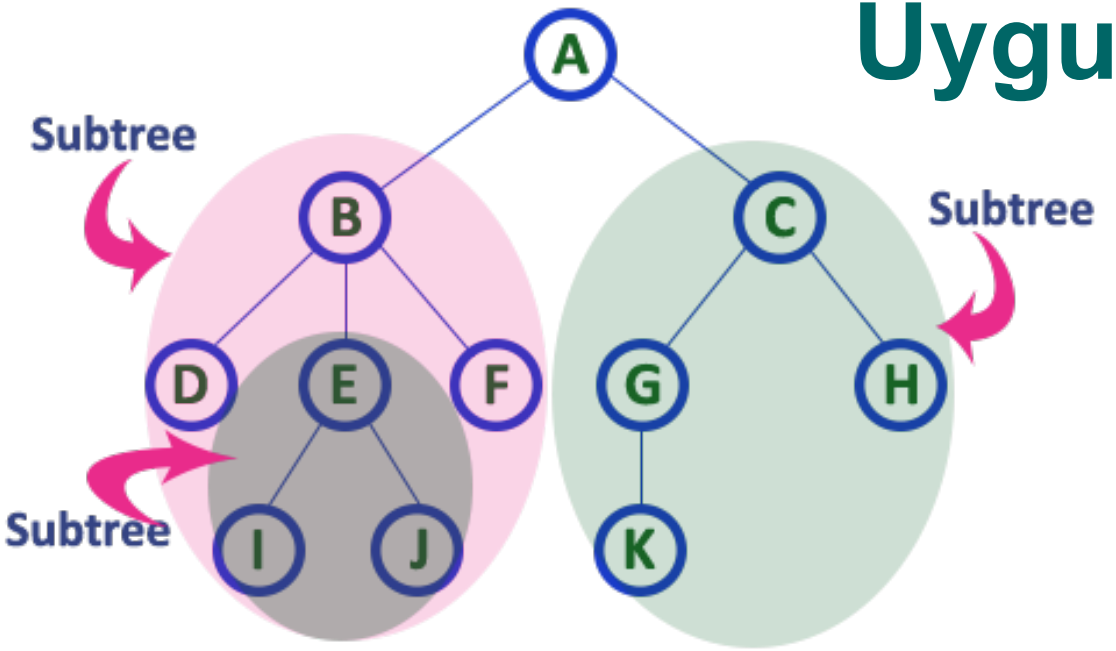


Recursive Vs iteratif



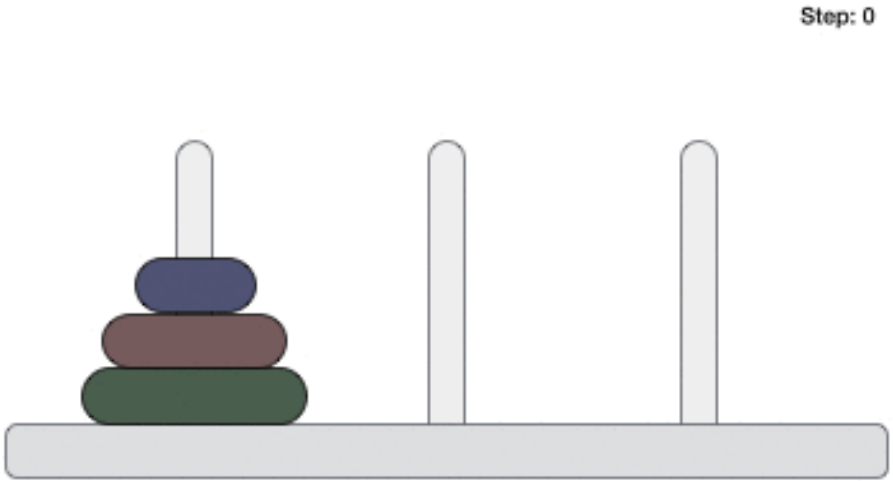
Recursive Vs iteratif

Uygulama



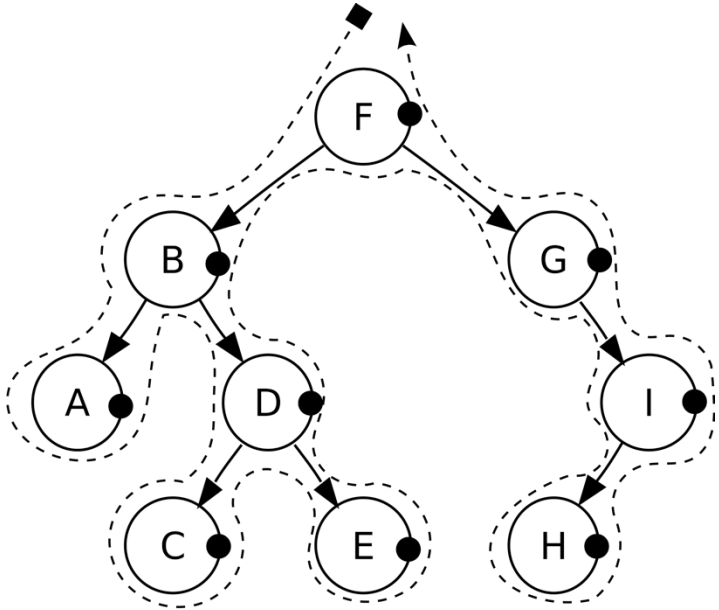
Uygulamalar

Tower of Hanoi



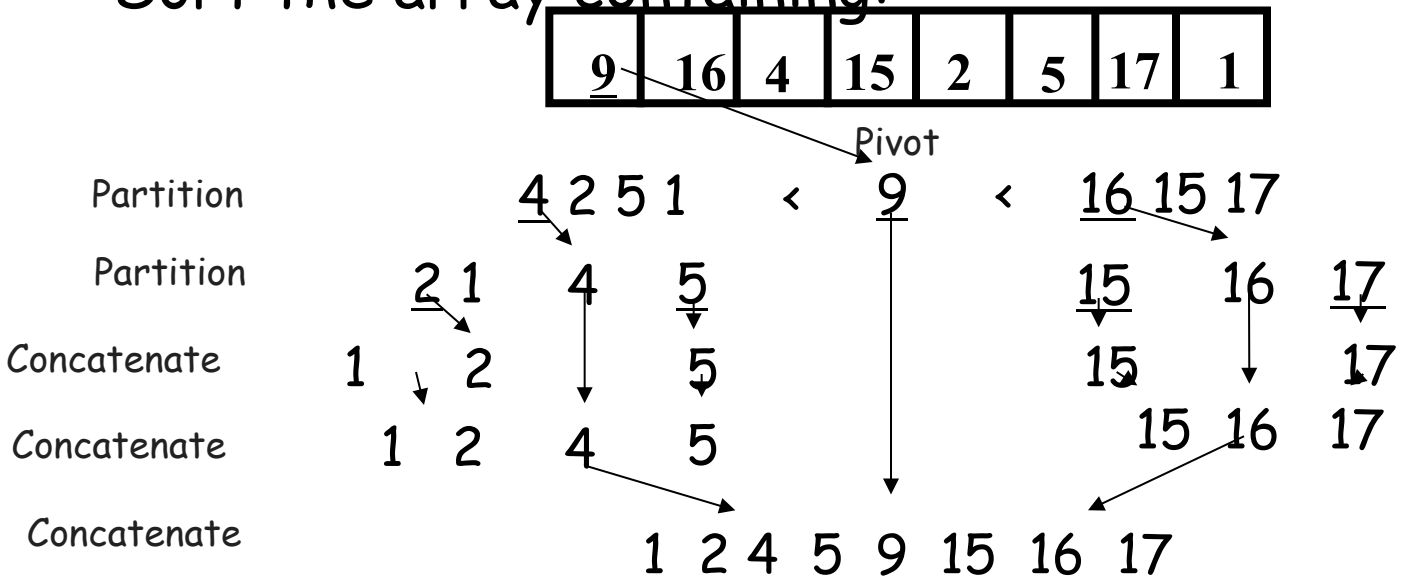
Uygulamalar

Tree traversals

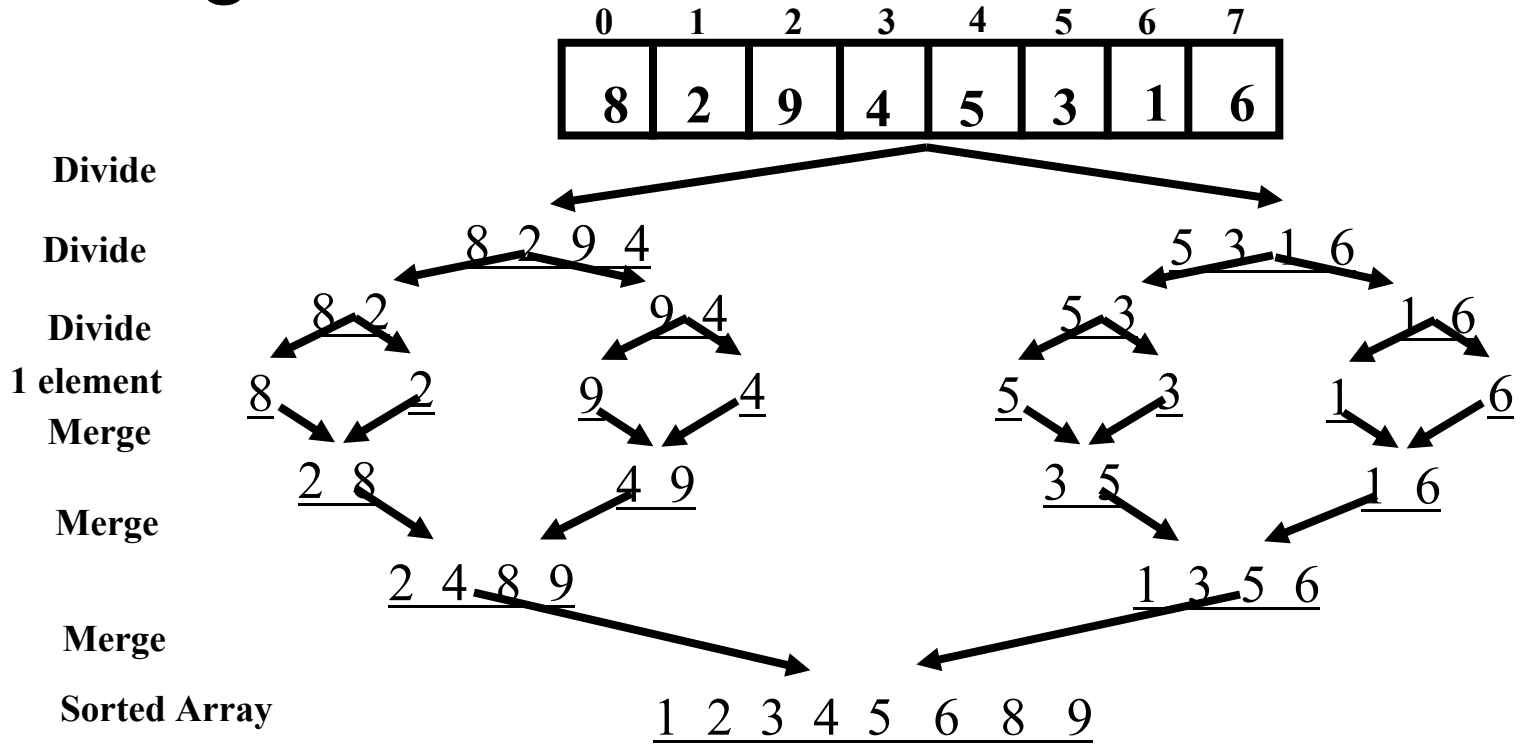


Quicksort örnek

- Sort the array containing:



MergeSort Örnek



Sorular