1.Fibonacci Serisi

Bu seriyi yazmanın eminim onlarca yolu vardır. Fakat ben iki tanesini ele aldım. İlki sürekli yapılan toplama işlemleriyle serinin elemanını bulan kod, ikincisi yine rekürsif fonksiyonla sürekli kendini tekrar ederek istenen elemanı bulan kod. Bu kodları C# dilinde yazdım. Gelin şimdi ikisini de inceleyelim.

1.1.Sürekli Toplama Yöntemi

Öncelikle kullandığım metodun kod parçacığı :

```
public int fibonacciHesapla(int x)
   {
       int a = 0;
       int b = 1;
       if (x == 0 || x == 1) {
           degisken = 1;
       }
       else {
       for (int i = 1; i <= x; i++)
       {
         degisken = a + b;
           a = b;
           b = degisken;
       }
     }
       return degisken;
   }
```

Bu metod görüldüğü üzere istenen elemanın indisi int olarak alınıyor ve bu sayıya göre bir döngüyle ilk elemandan başlayıp verilen sayıya kadar olan elemanlar hesaplanıyor ve en son istenen eleman sonuç olarak döndürülüyor. Olumlu yönü sürekli toplama işlemi yapıldığı için herhangi bir hatayla karşılaşılmıyor. Zaten Unit Testi de kodun içindedir. Olumsuz yönü ise zaman ve satır sayısı. Bunlara performans karşılaştırmasında değineceğim.

1.2.Rekürsif Fonksiyon Yöntemi

Metodun kod parçacığı :

```
public int fibonacciHesapla(int x)
       {
           if (x == 1 || x == 0)
           {
               return 1;
           }

           return fibonacciHesapla(x - 1) + fibonacciHesapla(x - 2);
       }
```

Bu metod da istenen elemanın indisi alınıyor ve o elemanın değeri sürekli kendini çağıran bir kodla hesaplanıyor. Kendi kendini çağıran koda bakıldığında ve kodun tahlili yapıldığında bu fonksiyonun iç içe çağırma ağacı (recursion tree) kullanılarak bir hesaplama yaptığını görüyoruz. Olumlu yönü bu serinin rekürsif fonksiyonlar ile kolayca ve az sayıda satırla yazılabilmesidir. Olumsuz yönü ise verilen elemanı hesaplamak için o sayıya kadar olan her elemanı tekrar ve tekrar fibonacci ağacı oluşturularak serinin hesaplanmasıdır. Bu performansta bir dezavantajdır.

Sonuç olarak;

İki metodun da unit testleri yapılmıştır.

İki metodu karşılaştırdığımızda zaman açısından testlerde ilk önce serinin fazla büyük olmayan elemanları kullanıldı ve sadece birkaç saniyelik farklar meydana geldi. Fakat serinin elemanları büyüdükçe(örn 30.eleman hesaplanırken) rekürsif fonksiyonun test süresi açık ara daha fazla sürdü yani ağaç dallandıkça performans düştü. Bu verilere bakıldığında zamana göre ilk metodumuz daha iyi sonuçlar vermektedir.

Bir de ne kadar az satır o kadar az karmaşıklık mantığına göre bakarsak bunun sonucunda ikinci yani rekürsif fonksiyonun bu kod için daha uygun olduğunu söyleyebilirim.

Fakat ben yine de bu seri için ilk metodu tercih ederdim çünkü zaman çok önemli ve ağaç yapısı elemanlar büyüdükçe işlemleri daha da yavaşlatıyor.