ALGORİTMA ANALİZİ-ÖDEV 1

Kaba Kod

1. Başla
2. Bir dizi oluştur ve içine verileri ekle (örneğin: [15, 20, 35, 150, 450, 30, 600])
3. "buyuk" adında bir değişken oluştur ve dizinin ilk elemanını atayın
4. Diziyi tarayarak en büyük sayıyı bul:
   4.1. Dizi elemanlarını sırayla kontrol et
   4.2. Eğer eleman "buyuk"ten büyükse, "buyuk"ü güncelle
5. "buyuk"ü ekrana yazdır
6. Bitir

ALGORİTMANIN ANALİZİ

1. Özyinelemeli (Recursive) Algoritma

```
static int enbuyuk(int[] arr, int n)
        {
            if (n == 1)                                          //1
                return arr[0];                                   //1

            return Math.Max(arr[n - 1], enbuyuk(arr, n - 1));    //n
        }                                                        //1+1+n = n+2
```

- Bu algoritma, her bir adımda bir önceki elemanla karşılaştırma yaparak en büyük sayıyı bulur.
- Her adımda, dizinin boyutu bir azalır ve bu boyut n ulaşana kadar devam eder.
- Ancak, bu yaklaşım daha az tercih edilir, çünkü gereksiz yere işlem maliyetini artırır ve daha karmaşık bir kod yapısı gerektirir. Zaman karmaşıklığı yine O(n) olur, ancak işlem maliyeti açısından daha yüksektir.

2. Yinelemeli (Iteratif) Algoritma

```
            int numara = arr[0];                    //1

            for (int i = 1; i < arr.Length; i++)    // 1  n-1  n
            {
                if (arr[i] > numara)                //n
                {
                    numara = arr[i];                //n
                }
            }

            return numara;                          //1
        }                                           //1+1+n-1+n+n+n+1 = 4n+2
```

- Bu algoritma, bir döngü kullanarak diziyi tarar ve en büyük sayıyı bulur.
- Döngü, dizi elemanlarının tamamını tek tek kontrol eder.
- Bu nedenle, döngünün çalışma zamanı dizinin boyutuna bağlıdır, yani zaman karmaşıklığı O(n) olur.

ÖZET

- İki yaklaşım da doğru sonucu verecektir, ancak performans ve kod karmaşıklığı açısından iteratif yaklaşım tercih edilir.