Algoritma Analizi-Ödev 2

```
1. Başla
2. Onlu sayıyı kullanıcıdan al ve "onlu_sayi" olarak sakla
3. İkili sayıyı saklamak için boş bir dizi oluştur ve "ikili_sayi" olarak
adlandır
4. Sürekli olarak aşağıdaki adımları tekrarla:
   4.1. "onlu_sayi"nin 2'ye bölümünden kalanı hesapla ve "kalan" olarak
sakla
   4.2. "onlu_sayi"yı 2'ye böl ve sonucu "onlu_sayi" olarak güncelle
   4.3. "kalan"ı "ikili_sayi" dizisinin başına ekle
   4.4. Eğer "onlu_sayi" sıfıra eşitse, döngüden çık
5. "ikili_sayi" dizisini ters çevir (en düşük basamaktan en yüksek basamağa
doğru)
6. "ikili_sayi"yi ekrana yazdır
7. Bitir
```

Algoritmanın Analizi

1. **Özyinelemeli Algoritma:**
   - `onluktanikiliye` adlı özyinelemeli fonksiyon, her bölme işlemi için kendisini tekrar çağırır.
   - Her özyinelemeli çağrıda sabit bir işlem yapılır (mod işlemi ve bölme).
   - Özyinelemeli algoritmanın derinliği, girdi olarak verilen ondalık sayının logaritmasıdır (log2(n)).
   - Bu nedenle, özyinelemeli algoritmanın zaman karmaşıklığı **O(log n)**'dir.
   - Fonksiyon çağrıları nedeniyle özyinelemeli algoritmanın ekstra bir maliyeti vardır.

2. **İteratif Algoritma:**
   - `onluktanikiliye` adlı iteratif fonksiyon, bir while döngüsü kullanarak sayıyı sürekli 2'ye böler.
   - Döngü, sayı 0 olana kadar çalışır.
   - Döngü her iterasyonda sabit bir işlem yapar (mod işlemi ve bölme).
   - İterasyon sayısı, girdi ondalık sayısının ikili temsilinin bit sayısına eşittir.
   - Bit sayısı logaritma tabanında log2(n) olduğundan, iteratif algoritmanın zaman karmaşıklığı da **O(log n)**'dir.
   - İteratif algoritma, sabit boyutlu bir dizi (binaryArray) kullanır ve bu nedenle bellek karmaşıklığı **O(1)**'dir.

Genel olarak, her iki algoritma da ondalık sayıyı ikili sayıya dönüştürmek için verimlidir, ancak iteratif algoritma daha az bellek kullanır.