Onlu sayının ikli sayıya dönüşümünü

Kaba kodu:

1-Öncelikle, dönüştürmek istediğimiz ondalık sayıyı alırız.
2-Ondalık sayının sıfır olup olmadığını kontrol ederiz. Eğer sıfırsa, sonuç olarak "0" döndürülür.
3-Ondalık sayı sıfır değilse, ikili sayıyı saklamak için bir değişken oluştururuz (örneğin, "binary").
4-Ondalık sayı sıfır olana kadar döngü başlatırız.
5-Döngü içinde, ondalık sayının 2'ye bölümünden kalanı (yani ondalık sayının 2'ye bölünmesinden kalan) alırız.
6-Bu kalanı, ikili sayının başına ekleriz.
7-Ondalık sayıyı, 2'ye bölerek güncelleriz (tam bölme yaparız).
8-Döngü, ondalık sayı sıfır olana kadar devam eder.
9-Döngüden çıktıktan sonra, ikili sayıyı saklayan değişkeni döndürürüz.

Gerçek Kodu:

```
using System;

class Program
{
    static string OndaliktandanIkiliye(int onlukSayi)
    {
        // Özel durum: Eğer ondalık sayı 0 ise, ikili sayı da 0'dır.
        if (onlukSayi == 0)
            return "0";

        string ikiliSayi = ""; // Dönüştürülecek ikili sayıyı saklayacak değişken

        // Ondalık sayı sıfır olana kadar döngü çalısır işlem devam eder
        while (onlukSayi > 0)
        {
            // Ondalık sayının 2'ye bölümünden kalan alınır
            int kalan = onlukSayi % 2;

            // Kalan, ikili sayının başına eklenir
            ikiliSayi = kalan + ikiliSayi;

            // Ondalık sayı, 2'ye bölünür (tam bölme yapılır)
            onlukSayi /= 2;
        }

        return ikiliSayi;
    }

    static void Main(string[] args)
    {
        // Örnek bir ondalık sayı girdik
        int onlukSayi = 7;

        // Onluk sayıyı ikili sayıya dönüştürme işlemi yapılır
```

```
        string ikiliSayi = OndaliktanIkiliye(onlukSayi);

        // Sonuçları ekrana yazdır
        Console.WriteLine("Onluk sayı " + onlukSayi + " ikili sayıya dönüştürüldü: " +
ikiliSayi);
        Console.ReadLine();
    }
}
```

❇❇

Bu algoritmanın karmaşıklığı, ondalık sayının ikili sayıya dönüştürülmesi için gereken adımlarla doğru orantılıdır.
While döngüsü, ondalık sayının 2'ye bölünüp 0 olana kadar devam eder. Bu işlem, ondalık sayının bit sayısına ($\log_2(n)$) kadar devam eder. Dolayısıyla, ondalık sayının bit sayısına bağlıdır ve zaman karmaşıklığı O(log n) olarak ifade edilir.

İteratif Kodu:

```
using System;

class Program
{
  // İteratif bir fonksiyon ile ondalık sayıyı ikili sayıya dönüştüren metot
  static string DecimalToBinary (int decimalNumber)
  {
    // Özel durum: Eğer ondalık sayı 0 ise, ikili sayı da 0'dır.
    if (decimalNumber == 0)
      return "0";

    string binary = ""; // Dönüştürülecek ikili sayıyı saklayacak değişken

    // Ondalık sayı sıfır olana kadar döngü çalışır işlem devam eder
    while (decimalNumber > 0)
    {
      // Ondalık sayının 2'ye bölümünden kalan alınır
      int remainder = decimalNumber % 2;

      // Kalan, ikili sayının başına eklenir
      binary = remainder + binary;

      // Ondalık sayı, 2'ye bölünür (tam bölme yapılır)
      decimalNumber /= 2;
    }

    return binary; // İkili sayı döndürülür
  }

  static void Main(string[] args)
  {
    int decimalNumber = 25; // Örnek bir ondalık sayı
    // Ondalık sayıyı ikili sayıya dönüştürme işlemi yapılır
    string binaryNumber = DecimalToBinary(decimalNumber);
    // Sonuçları ekrana yazdır
    Console.WriteLine("Ondalık sayı " + decimalNumber + " ikili sayıya dönüştürüldü( iteratif): " +
binaryNumber);
```

```
    }
}
```

❇❇

Algoritma analizi:

Karmaşıkliği:

Bu iteratif kod çözümünde , her bir ondalık basamak için bir işlem gerçekleştirilir. Dolayısıyla, ondalık sayının bit sayısına ($\log_2(n)$) bağlıdır ve zaman karmaşıklığı O(log n) olarak ifade edilir.
Kullanılan hafıza miktarı(ramdeki kapladığı alan), binary adlı bir dize içindir ve bu dizenin boyutu, ondalık sayının ikili dönüşümündeki maksimum basamak sayısına bağlıdır. Dolayısıyla, hafıza karmaşıklığı O(log n) olarak ifade edilir.

Recursive Kodu:

```
using System;
class Program
{
    // Rekürsif bir fonksiyon ile ondalık sayıyı ikili sayıya dönüştüren metot
    static string DecimalToBinaryRecursive(int decimalNumber)
    {
        // Özel durum: Eğer ondalık sayı 0 ise, ikili sayı da 0'dır.
        if (decimalNumber == 0)
            return "0";
        else
            // Ondalık sayının ikili dönüşümü, ondalık sayının yarısı ile bu fonksiyonun çağrılması ve son basamağın
eklenmesiyle elde edilir
            return DecimalToBinaryRecursive(decimalNumber / 2) + (decimalNumber % 2);
    }

    static void Main(string[] args)
    {
        int decimalNumber = 25; // Örnek bir ondalık sayı
        // Ondalık sayıyı ikili sayıya dönüştürme işlemi yapılır
        string binaryNumber = DecimalToBinaryRecursive(decimalNumber);
        // Sonuçları ekrana yazdır
        Console.WriteLine("Ondalık sayı " + decimalNumber + " ikili sayıya dönüştürüldü: " + binaryNumber);

        Console.ReadLine();

    }
}
```

❇❇

Algoritma analizi:

Karmaşıklığı:

Her bir çağrıda, bir işlem gerçekleştirilir ve her bir çağrı ondalık sayının yarısını alır. Dolayısıyla, çağrı sayısı ondalık sayının bit sayısına ($\log_2(n)$) bağlıdır ve zaman karmaşıklığı O(log n) olarak ifade edilir.

Rekürsif çağrılar, çağrı yığını üzerinde bir maliyet oluşturur. Dolayısıyla, hafıza karmaşıklığı(bellek karmaşıklığı) O(1) olarak kabul edilebilir.yani bellek tüketimi daha fazladır.