Proje Raporu:

- **Proje Adı:** Hanoi Kuleleri (Tower of Hanoi) Probleminin Özyinelemeli Çözümü
- **Ders Adı: VERİ YAPILARI**
- **Hazırlayan: ESMER ALTAŞ**
- Öğrenci Numarası: 24080410208
- **Teslim Tarihi:** 27 Ekim 2025

2. Giriş

2.1. Problemin Tanımı

Hanoi Kuleleri, 1883 yılında Fransız matematikçi Édouard Lucas tarafından ortaya atılmış klasik bir matematiksel bulmaca ve bilgisayar bilimlerinde özyinelemeyi (recursion) açıklamak için kullanılan temel bir problemdir. Problem, üç dikey çubuk (iğne) ve farklı büyüklükteki N adet diskten oluşur. Başlangıçta tüm diskler, en büyük disk en altta olmak üzere, küçükten büyüğe doğru sıralanmış olarak ilk çubukta bulunur.

2.2. Projenin Amacı ve Kısıtları

Bu projenin temel amacı, Hanoi Kuleleri problemini C# programlama dilini kullanarak en az hamle sayısıyla çözecek özyinelemeli bir algoritma geliştirmektir. Çözüm, problemle ilişkili aşağıdaki katı kısıtlamalara uymak zorundadır:

- **Kısıt 1 (Hareket Kısıtı):** Aynı anda sadece tek bir disk taşınabilir.
- **Kısıt 2 (Üst Kısıtı):** Bir disk, her zaman kendisinden daha büyük bir diskin üzerine konulabilir veya boş bir çubuğa taşınabilir; ancak büyük bir disk asla küçük bir diskin üzerine konulamaz.
- **Amaç:** Disklerin tamamını en soldaki (Kaynak) iğneden en sağdaki (Hedef) iğneye taşımaktır.

2.3. Kullanılan Teknoloji ve Yöntem

- **Programlama Dili:** C#

Proje Raporu:

- **Çözüm Yöntemi:** Problem, doğal yapısı gereği **Özyineleme (Recursion)** kullanılarak çözülmüştür.

3. Teorik Arka Plan ve Algoritma

3.1. Özyineleme (Recursion) Kavramı

Özyineleme, bir fonksiyonun kendisini çağırma yeteneğidir. Hanoi Kuleleri probleminde özyineleme, N diskli problemi, $N-1$ diskli daha basit alt problemlere bölerek çalışır. Bir N diskli problemi çözmek için, sadece N diskini hareket ettirmeyi bilmek ve bu hareketin iki yanında $N-1$ diskli problemi çözmek yeterlidir.

3.2. Çözüm Algoritması (N Disk İçin)

N diski Kaynak iğneden Hedef iğneye taşımak için izlenen temel özyinelemeli algoritma adımları şunlardır:

1. **Adım 1:** En üstteki $N-1$ diski, Hedef iğneyi geçici olarak kullanarak Kaynak iğneden **Yardımcı** iğneye taşı.
2. **Adım 2 (Temel Hareket):** En alttaki ve en büyük olan N. diski Kaynak iğneden **Hedef** iğneye taşı.
3. **Adım 3:** $N-1$ diski, Kaynak iğneyi geçici olarak kullanarak Yardımcı iğneden **Hedef** iğneye taşı.

Bu süreç, $N=1$ disk durumuna (Temel Durum) ulaşılana kadar devam eder.

3.3. Minimum Hamle Sayısı

Hanoi Kuleleri probleminin minimum hamle sayısı, aşağıdaki formülle hesaplanır:

$$H(N) = 2^N - 1$$
$N=3$ disk için minimum hamle sayısı: $2^3 - 1 = 7$ hamle.

Proje Raporu:

4. Uygulama Detayları ve Kod

4.1. Kodun Yapısı

C# kodu, statik bir `HanoiCoz` metodu etrafında yapılandırılmıştır. `hamleSayisi` adında statik bir sayaç, her disk hareketi sırasında artırılarak toplam hamle sayısını takip eder. Fonksiyonun parametreleri (disk sayısı `n`, iğneler `kaynak`, `hedef`, `yardimci`) disklerin yerini ve hedefini belirler.

4.2. C# Kaynak Kodu:

```
using System;


public class HanoiKuleleri
{
    // Hamle sayısını takip etmek için sayaç
    private static int hamleSayisi = 0;

    /// <summary>
    /// Hanoi Kulesi problemini çözen özyinelemeli (recursive) metot.
    /// </summary>
    /// <param name="n">Taşınacak disk sayısı.</param>
    /// <param name="kaynak">Başlangıç çubuğu.</param>
    /// <param name="hedef">Bitiş çubuğu.</param>
    /// <param name="yardimci">Geçici çubuk.</param>
```

Proje Raporu:

```
  public static void HanoiCoz(int n, char kaynak, char hedef, char yardimci)
  {
    if (n == 1)
    {
      // Temel Durum
      Console.WriteLine($"Adım {++hamleSayisi}: Diski {kaynak}'dan {hedef}'a taşı.");
    }
    else
    {
      // 1. n-1 diski Kaynaktan Yardımcıya (Hedef kullanarak)
      HanoiCoz(n - 1, kaynak, yardimci, hedef);

      // 2. En büyük (n.) diski Kaynaktan Hedefe
      Console.WriteLine($"Adım {++hamleSayisi}: Diski {kaynak}'dan {hedef}'a taşı.");

      // 3. n-1 diski Yardımcıdan Hedefe (Kaynak kullanarak)
      HanoiCoz(n - 1, yardimci, hedef, kaynak);
    }
  }

  public static void Main(string[] args)
```

Proje Raporu:

```
  {
      int diskSayisi = 3;
      char baslangicCubugu = 'A';
      char hedefCubugu = 'C';
      char yardimciCubuk = 'B';

      hamleSayisi = 0;

      Console.WriteLine($"Hanoi Kuleleri Çözümü ({diskSayisi} disk ile):");
      Console.WriteLine(new string('-', 40));

      HanoiCoz(diskSayisi, baslangicCubugu, hedefCubugu,
yardimciCubuk);

      Console.WriteLine(new string('-', 40));
      int minimumHamle = (int)Math.Pow(2, diskSayisi) - 1;
      Console.WriteLine($"Toplam minimum hamle sayısı: {hamleSayisi}");
      Console.WriteLine($"Doğrulama (Formül 2^n - 1): {minimumHamle}");
  }
}
```

Proje Raporu:

5. Programın Çalışması ve Çıktısı

5.1. Çalıştırma Parametreleri

Problemde belirtilen görseldeki duruma uygun olarak, program 3 disk ile çalıştırılmıştır.

- Disk Sayısı (N): **3**
- Kaynak İğne: **'A'**
- Hedef İğne: **'C'**
- Yardımcı İğne: **'B'**

5.2. Konsol Çıktısı

Programın çalıştırılmasıyla elde edilen adım adım disk hareketleri aşağıdadır:

```
Hanoi Kuleleri Çözümü (3 disk ile):
----------------------------------------
Adım 1: Diski A'dan C'ye taşı.
Adım 2: Diski A'dan B'ye taşı.
Adım 3: Diski C'den B'ye taşı.
Adım 4: Diski A'dan C'ye taşı.
Adım 5: Diski B'den A'ya taşı.
Adım 6: Diski B'den C'ye taşı.
Adım 7: Diski A'dan C'ye taşı.
----------------------------------------
Toplam minimum hamle sayısı: 7
Doğrulama (Formül 2^n - 1): 7
```

5.3. Çalıştırma Ekran Görüntüsü

Programın konsolda çalıştırıldığını ve çıktısını gösteren ekran görüntüsü aşağıdadır.

[BU ALANA C# KODUNUZUN ÇALIŞMA EKRAN GÖRÜNTÜSÜ YERLEŞTİRİLECEKTİR]

Proje Raporu:

6. Sonuç ve Değerlendirme

6.1. Sonuçların Analizi

Geliştirilen özyinelemeli C# çözümü, 3 disk için toplam **7 hamle** ile problemi başarıyla tamamlamıştır. Bu sonuç, Hanoi Kuleleri problemi için teorik olarak hesaplanan minimum hamle sayısıyla ($2^3 - 1 = 7$) birebir örtüşmektedir. Bu, algoritmanın hem doğru çalıştığını hem de en uygun (optimal) çözümü sunduğunu göstermektedir.

6.2. Projenin Başarısı

Proje, Hanoi Kuleleri problemini verilen tüm kısıtlamalara uyarak (aynı anda tek disk, büyük disk küçük üzerine konulamaz) ve C# dilinin özyineleme özelliğini etkin bir şekilde kullanarak çözmüştür. Diskler, başarıyla 'A' çubuğundan 'C' çubuğuna minimum hamle sayısıyla taşınmıştır.