VERİ YAPILARI VE ALGORİTMALAR

Arş. Gör. FURKAN ATBAN

1.ödev

ARİF MUHAMMED

23080103348

Hanoi Kuleleri Problemini Çözme (Kod Açıklaması ve Detayları):

Bu raporda, Hanoi Kuleleri problemini çözmek için yazılan C++ kodunun işleyişi, kullanılan algoritmaların detayları ve kullanıcı ile etkileşim mekanizması açıklanacaktır. Ayrıca, rekürsif (özyinelemeli) ve iteratif (döngüsel) yöntemlerin performans farkları ve hesaplanma şekilleri ele alınacaktır.

Genel Bakış:

Bu program, Hanoi Kuleleri problemini çözmek için hem rekürsif hem de iteratif yöntemleri kullanır. Kullanıcı, program çalışırken disk sayısını ve çözüm yöntemini seçebilir. Ayrıca, rekürsif ve iteratif çözümlerin performanslarını ayrı ayrı test edebilir.

Fonksiyonlar ve İşlevleri:

1. printTowers(vector<vector<int>>& towers):

Bu fonksiyon, kulelerin mevcut durumunu ekrana yazdırır. Her bir disk, bulunduğu kulede gösterilir ve görsel olarak takip edilebilir.

2. moveDisk(vector<vector<int>>& towers, char from, char to, int& steps):

Bu fonksiyon, bir diski bir kuleden diğerine taşır. Taşıma işlemi sırasında diskler arasındaki hareketler ekrana yazdırılır ve toplam adım sayısı güncellenir.

3. recursiveHanoi(int n, char kaynak, char ara, char hedef, vector<vector<int>>& towers, int& steps):

Rekürsif yöntemi kullanarak Hanoi Kuleleri problemini çözer. Bir diski kaynak kuleden hedef kuleye taşır ve ara kuleyi yardımcı olarak kullanır.

4. iterativeHanoi(int n, char kaynak, char ara, char hedef, vector<vector<int>>& towers, int& steps):

Bu fonksiyon, yığın (stack) kullanarak iteratif yöntemle problemi çözer. Disklerin taşınması için gerekli hareketler yığına eklenir ve özyinelemeli çağrılar yapılmadan diskler taşınır.

5. solveHanoi(int diskSayisi, bool isRecursive):

Bu fonksiyon, Hanoi Kuleleri problemini çözmek için kullanılacak yöntemi belirler. Çözüm süresi ve adım sayısını hesaplar ve sonuçları ekrana yazdırır.

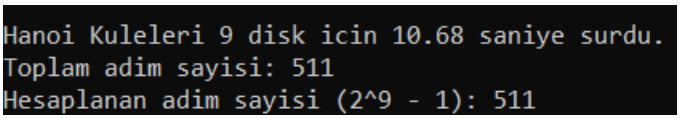
Kullanıcı İle Etkileşim:

Program, kullanıcıya dört ana seçenek sunar:

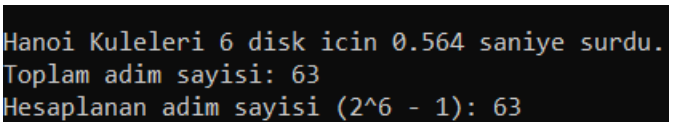
1. Klasik Hanoi Kuleleri Problemi: Kullanıcıdan 4 veya daha az disk sayısı girmesini ister ve problemi çözer.

```
Hanoi Kuleleri 4 disk icin 0.123 saniye surdu.
Toplam adim sayisi: 15
Hesaplanan adim sayisi (2^4 - 1): 15
```

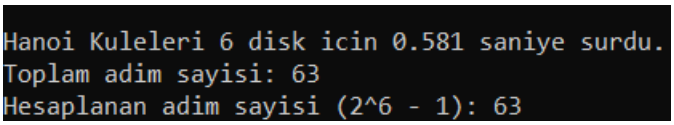
2. İleri Düzey Hanoi Kuleleri Problemi: Kullanıcıdan 5 veya daha fazla disk sayısı girmesini ister ve problemi çözer.



3. Rekürsif Çözüm: Kullanıcıdan disk sayısını alır ve rekürsif yöntemle problemi çözer.



4. İteratif Çözüm: Kullanıcıdan disk sayısını alır ve iteratif yöntemle problemi çözer.



5. Çıkış: Programdan çıkış yapar.

Performans Karşılaştırması:

Program, rekürsif ve iteratif yöntemlerin performansını çözüm süresi ve adım sayısı açısından karşılaştırır.

Rekürsif Yöntem:

- Zaman Karmaşıklığı: O(2^n)

- Adım Sayısı: 2^n - 1

Rekürsif yöntem, her disk hareketi için alt problemler oluşturur ve bu alt problemleri çözerek sonuca ulaşır. Bu yöntem, özellikle küçük disk sayılarında daha anlaşılır ve basit bir yaklaşımdır, ancak büyük disk sayılarında performans sorunu yaşayabilir.

İteratif Yöntem:

- Zaman Karmaşıklığı: O(2^n)

- Adım Sayısı: 2^n - 1

İteratif yöntem, yığın (stack) kullanarak tüm disk hareketlerini belirler ve bu hareketleri bir döngü içinde gerçekleştirir. Bu yöntem, rekürsif yöntemin aksine, büyük disk sayılarında daha yönetilebilir ve stack overflow hatası riski olmadan çalışır.

Zaman Ölçümü:

Her iki yöntemin performansını ölçmek için `clock()` fonksiyonu kullanılır. Bu fonksiyon, çözümün başlangıç ve bitiş zamanlarını kaydederek toplam geçen süreyi hesaplar. Örneğin:

```cpp
clock_t start = clock();

// Yöntem çağrısı burada

clock_t end = clock();

double time_taken = double(end - start) / double(CLOCKS_PER_SEC);
```

Bu hesaplama, her iki yöntemin de belirli bir disk sayısı için ne kadar sürede sonuç verdiğini karşılaştırmamızı sağlar.

Sonuç:

Bu program, Hanoi Kuleleri problemini hem rekürsif hem de iteratif yöntemlerle çözen kapsamlı bir çözümdür. Kullanıcıya çeşitli seçenekler sunarak problemi farklı disk sayıları ve yöntemlerle çözme imkanı tanır. Ayrıca, yöntemlerin performans farklarını karşılaştırarak hangi yöntemin daha verimli olduğunu gösterir. Rekürsif yöntem, küçük disk sayıları için daha uygunken, iteratif yöntem büyük disk sayılarında daha iyi performans gösterir. Bu özellikler, programı eğitim amaçlı ve problem çözme yeteneklerini geliştirmek için kullanışlı hale getirir.