**ALGORİTMA ANALİZİ ÖDEVİ -3**

**n\*n Matris Çarpımı sayıyı bulma probleminin kaba kodu:**

1-İki n x n matris alınır, bunlar A ve B olsun.

2-Çarpım sonucu oluşacak yeni bir n x n matris oluşturulur, bunu C olarak adlandıralım. C matrisi başlangıçta boş olacak.

3-C matrisinin her bir elemanı için çarpma işlemi yapılacaktır. Bu işlemi gerçekleştirmek için her bir C[i][j] elemanını hesaplamamız gerekir (i satır ve j sütunu temsil etmektedir).

4-C matrisinin i. satırı ve j. sütunundaki elemanı hesaplamak için, A matrisinin i. satırı ile B matrisinin j. sütununu ele alırız.

5-İki matrisin ilgili satırı ve sütununun elemanlarını çarparız ve sonuçları toplarız. Yani, C[i][j] elemanını hesaplamak için A matrisinin i. satırındaki elemanları sırayla B matrisinin j. sütunundaki elemanlarla çarparız ve bu sonuçları toplarız.

6-Bu işlemi tüm C matrisinin elemanları için tekrarlarız, yani her i ve j için C[i][j] elemanını hesaplarız.

7-Son olarak, C matrisi çarpım sonucunu içerir ve bu sonucu döndürürüz.

**İTERATİF KODUN ALGORİTMA ANALİZİ(KARMAŞIKLIĞI):**

Bu kod, öncelikle iki 3x3 matrisi tanımlar, ardından bu matrislerin çarpımını hesaplar ve sonucu ekrana yazdırır. Karmaşıklık analizi şu şekildedir:

İç içe üç adet döngü vardır: İlk iki döngü, matrisin satır ve sütunlarını gezme işlemlerini gerçekleştirirken, üçüncü iç içe döngü ise matrislerin elemanlarını çarparak toplamak için kullanılır.

Bu nedenle, bu iteratif çözümün zaman karmaşıklığı O(n^3) olacaktır. Çünkü her bir döngü n kadar dönüyor ve bu döngüler iç içe olduğundan dolayı, işlem sayısı n^3 olarak ifade edilir.

Bu karmaşıklık, büyük matrislerde hesaplama süresinin artmasına neden olabilir. Bu nedenle büyük matrislerde daha verimli algoritmalar tercih edilmelidir.

**RECURİSİVE KODUN ALGORİTMA ANALİZİ(KARMAŞIKLIĞI):**

Algoritmanın karmaşıklığı, n boyutunda bir matris çarpımı için O(n^3) olacaktır. Bunun nedeni, matrisin her bir parçasını işlemek için üç döngü kullanılmasıdır ve bu döngüler her bir parça için O(n^2) zaman alır. Buna ek olarak, matrislerin dört parçaya ayrılması ve tekrar birleştirilmesi gibi adımlar da O(n^2) zaman alır.

Özyinelemeli (recursive) algoritma, her bir adım için alt problemlerin çözümünü çağırarak ve bunları birleştirerek çalışır. Bu nedenle, alt problemler tarafından yapılan işlem miktarı ile ana problemin karmaşıklığı arasında bir bağlantı vardır.

**ZEYNEB BAYRAM**

**22080410005**