Enes Ayhan Benli

202313171037

1. Öğretim Grubu

1. AŞAMA - SAYI ALGORİTMALARI

1. Mükemmel Sayılar: Pozitif bir sayının, kendisi hariç pozitif bölenlerinin toplamı kendisine eşitse mükemmel sayıdır. Ben de kodumda istenen 2 parçayı 1.si girilen sayıya kadar olan mükemmel sayıları yazdıracak 2.si ise girilen sayının mükemmel olup olmadığını kontrol edecek şekilde yazdım.

\* Örnek: 6'nın bölenleri 1, 2 ve 3'tür. 1 + 2 + 3 = 6 olduğu için 6 mükemmel sayıdır.

2. Fibonacci Sayıları: Her sayının kendisinden önce gelen iki sayının toplamı olduğu dizidir, algoritma olarak hem normal şekilde çalışan halini hem de recursive olarak çalışan halini yazdım.

\* Recursive: Fonksiyonun kendi kendini çağırarak sonuca ulaşması.

\* Standart: Döngü kullanarak hesaplama.

\* Örnek: 0, 1, 1, 2, 3, 5, 8, 13, 21...

3. Armstrong Sayısı: Bir sayının Armstrong sayısı olabilmesi için, o sayının her bir basamağının sayının basamak sayısı kadar üssü alınıp toplandığında, toplamın sayının kendisine eşit olması gerekir. Ben de burada sırayla basamak sayısını alıp sonrasında ise tek tek her hanedeki rakamın küpümü alıp topladım eşit olduğu durumda da sayı Armstrong Sayısı olmuş oldu.

153 sayısı 3 basamaklıdır.

\* Örnek = 1³ + 5³ + 3³ = 1 + 125 + 27 = 153

4. Tribonacci Sayıları: Dizideki her sayının kendisinden önce gelen üç sayının toplamına eşit olduğu bir sayı dizisidir. Fibonacci serisindeki mantığın 3lü halini burada kullanarak her sayının 3 öncesi ile toplanması ile oluşmasını sağladım,

\* Örnek: 0, 0, 1, 1, 2, 4, 7, 13, 24...

5. Palindrom Sayılar: Ters çevrildiğinde de aynı olan sayılar.

\* Örnek: 121, 454, 1001

6. Cullen Sayıları: n \* 2^n + 1 formülüyle elde edilen sayılar.

\* Örnek: n = 1 için 3, n = 2 için 9...

7. Lasa Sayısı: Herhangi bir sayının rakamlarının küpleri toplamının tekrar tekrar hesaplanmasıyla elde edilen sayı.

\* Örnek: 153 sayısı için: 1³ + 5³ + 3³ = 153, sonuç tekrar 153 olduğu için Lasa sayısıdır.

8. Fermat sayıları: 2^(2^n) + 1 formülüyle elde edilen sayılar.

\* Örnek: n = 0 için 3, n = 1 için 5...

9. Dost Sayılar: İki sayının, kendileri hariç bölenlerinin toplamı birbirine eşitse dost sayılardır.

\* Örnek: 220 ve 284

10. Zengin Sayılar: Bir sayının, kendisi hariç pozitif bölenlerinin toplamı, sayının kendisinden büyükse zengin sayıdır.

\* Örnek: 12'nin bölenleri 1, 2, 3, 4, 6'dır ve toplamları 16'dır. 16 > 12 olduğu için 12 zengin sayıdır.

11. Lucas Serisi: Fibonacci dizisine benzer, ancak başlangıç değerleri 2 ve 1'dir.

\* Örnek: 2, 1, 3, 4, 7, 11, 18...

12. Tetranacci Sayıları: dizideki her sayının kendisinden önce gelen dört sayının toplamına eşit olduğu bir sayı dizisidir.

\* Örnek: 0, 0, 0, 1, 1, 2, 4, 8, 15...

13. İkiz Sayılar: Aralarındaki fark 2 olan asal sayılar.

\* Örnek: (3, 5), (5, 7), (11, 13)...

14. Woodall Sayıları: n \* 2^n - 1 formülüyle elde edilen sayılar.

\* Örnek: n = 2 için 7, n = 3 için 23...

15. Mersenne Sayıları: 2^n - 1 formülüyle elde edilen sayılar.

\* Örnek: n = 2 için 3, n = 3 için 7...

16. Harshad Sayıları: Basamaklarının toplamına tam bölünebilen sayılar.

\* Örnek: 18 (1 + 8 = 9 ve 18, 9'a tam bölünür)

17. Cyclic (Döngüsel) Sayılar: Belirli bir işlem uygulandığında, basamakları yer değiştirerek aynı sayıyı veren sayılar.

\* Örnek: 142857 sayısı 2 ile çarpıldığında 285714 olur, basamakları yer değiştirmiş ama aynı sayı kalmıştır.

18. Tav Sayıları: Kendisini bölen sayıların sayısı, kendisiyle aynı olan sayılar.

\* Örnek: 6'nın bölenleri 1, 2, 3, 6'dır (4 adet) ve 4'ün bölenleri 1, 2, 4'tür (3 adet). 6, 4'ü böldüğü için 6 bir Tav sayısıdır.

19. Bağdaşık Sayılar (Amicable): İki sayının, kendileri hariç bölenlerinin toplamı birbirine eşitse bağdaşık sayılardır.

\* Örnek: 220 ve 284

20. 6174 sayısı: Kaprekar sayısı olarak da bilinir. Dört basamaklı herhangi bir sayının rakamlarını büyükten küçüğe ve küçükten büyüğe sıralayarak elde edilen sayıların farkı alınır. Bu işleme devam edildiğinde sonunda 6174 sayısına ulaşılır.

2. AŞAMA - SIRALAMA ALGORİTMALARI

Bu kısımda, her bir sıralama algoritması için kısa açıklamalar ekleyerek daha anlaşılır hale getirelim:

1. Insertion Sort (Eklemeli Sıralama): Dizideki her elemanı sırayla alıp, kendisinden önceki elemanlarla karşılaştırarak doğru yere yerleştiren algoritma.

2. Selection Sort (Seçmeli Sıralama): Dizideki en küçük elemanı bulup başa koyarak sıralama yapan algoritma.

3. Bubble Sort (Kabarcık/Baloncuk Sıralama): Yan yana elemanları karşılaştırarak yer değiştirme yöntemiyle sıralama yapan algoritma.

4. Divide and Conquer Sort (Böl ve Ayıkla Algoritması): Problemi alt problemlere bölerek çözen ve sonra birleştiren algoritma türü. Merge Sort ve Quick Sort bu yöntemi kullanır.

5. Shell Sort: Belirli aralıklarla elemanları karşılaştırarak sıralama yapan algoritma.

6. Merge Sort: Diziyi sürekli ikiye bölerek ve sıralı alt dizileri birleştirerek sıralama yapan algoritma.

7. Quick Sort: Bir pivot eleman seçerek diziyi küçük ve büyük elemanlar olarak ikiye bölen ve bu işlemi tekrarlayan algoritma.

8. Quick Sort 3: Quick Sort'un üç yönlü bölümleme kullanan bir çeşidi.

9. Heap Sort (Yığınlama): Diziyi bir heap veri yapısına dönüştürerek sıralama yapan algoritma.

10. Taban Sıralama (Radix): Elemanların basamak değerlerine göre sıralayan algoritma.

11. Shaker Sort (Sallama Sıralama): Bubble Sort'un geliştirilmiş bir versiyonu, iki yönlü tarama yapar.

12. Rastgele Sort (Random Sort): Elemanları rastgele sıralayarak ve sıralı olup olmadığını kontrol eden algoritma. (Verimsiz bir yöntemdir)

13. Lucky Sort (Şanslı Sıralama): Rastgele Sort'a benzer, ancak daha az karşılaştırma yapar. (Verimsiz bir yöntemdir)

14. Serseri Sort (Stooge Sort): Rekürsif olarak çalışan ve verimsiz bir sıralama algoritması.

15. Şimşek Sort (Flash Sort): Diziyi önceden gruplandırarak sıralama yapan algoritma.

16. Comb Sort (Tarak Sort): Bubble Sort'un geliştirilmiş bir versiyonu, daha büyük aralıklarla elemanları karşılaştırır.

17. Gnome Sort: Yanlış sıralanmış bir eleman bulunana kadar diziyi ileri doğru tarayan ve ardından bu elemanı doğru konumuna "geri takas" ederek çalışan basit bir sıralama algoritmasıdır. Bubble Sort'a benzer, ancak elemanları sadece bir yönde hareket ettirir.

18. Permütasyon Sort (Permütasyon Sıralama): Bir dizinin tüm olası permütasyonlarını (sıralamalarını) üreten ve sıralı olanı bulan bir algoritmadır. Son derece verimsizdir ve pratik uygulamalarda kullanılmaz.

19. Strand Sort (Tel Sıralama): Diziyi sıralı alt dizilere ("teller") bölen ve ardından bu telleri birleştirerek sıralı bir dizi oluşturan bir algoritmadır. Insertion Sort'a benzer, ancak ek bellek kullanır.

20. Kova Sort (Bucket Sort): Verileri sonlu sayıda "kova"ya dağıtan ve ardından her kovayı ayrı ayrı sıralayan bir algoritmadır. Verilerin dağılımı hakkında bilgi varsa etkili olabilir.

## 3. AŞAMA - ARAMA ALGORİTMALARI

Bu kısımda, her bir arama algoritması için kısa açıklamalar ekleyerek daha anlaşılır hale getirelim:

1. Doğrusal Arama (Linear Search): Dizideki her elemanı sırayla kontrol ederek aranan elemanı bulmaya çalışan basit bir arama algoritmasıdır. Sıralı veya sırasız dizilerde kullanılabilir.

2. İkili Arama (Binary Search): Sıralı bir dizide, aranan elemanı bulmak için diziyi sürekli ikiye bölen verimli bir arama algoritmasıdır.

3. İnterpolasyon Araması (Ara Değer Araması, Interpolation Search): Sıralı bir dizide, aranan elemanın konumunu tahmin ederek arama yapan bir algoritmadır. Verilerin düzgün dağıldığı durumlarda İkili Arama'dan daha hızlı olabilir.

4. Şekiller (Graflarda) Üzerinde Çalışan Algoritmalar: Graflarda dolaşmak, yollar bulmak ve diğer işlemleri gerçekleştirmek için kullanılan algoritmalardır. Örnekler:

5. Sabit Maliyetli Arama (Uniform Cost Search): Her adımın maliyetinin aynı olduğu durumlarda en düşük maliyetli yolu bulmak için kullanılan bir arama algoritmasıdır.

6. Floyd Warshall Algoritması: Bir graftaki tüm düğüm çiftleri arasındaki en kısa yolları bulan bir algoritma.

7. Prim’s Algoritması: Ağırlıklı bir grafta minimum maliyetli yayılma ağacını (MST) bulan bir algoritma.

8. Kruskal Algoritması: Ağırlıklı bir grafta minimum maliyetli yayılma ağacını (MST) bulan bir diğer algoritma.

9. Dijkstra Algoritması: Ağırlıklı bir grafta tek bir kaynaktan tüm diğer düğümlere olan en kısa yolları bulan bir algoritma.

10. Bellman Ford Algoritması: Ağırlıklı bir grafta tek bir kaynaktan tüm diğer düğümlere olan en kısa yolları bulan bir diğer algoritma. Negatif ağırlıklı kenarları da işleyebilir.

11. İkili Arama Ağacı (Binary Search Tree): Her düğümün sol alt ağacındaki tüm düğümlerin değerinin kendisinden küçük, sağ alt ağacındaki tüm düğümlerin değerinin ise kendisinden büyük olduğu bir ağaç veri yapısıdır. Verimli arama, ekleme ve silme işlemlerine olanak tanır.

12. Prüfer Dizilimi: Etiketli bir ağacı temsil etmek için kullanılan bir dizi.

13. Metin Arama Algoritmaları: Bir metin içinde belirli bir kalıbı arayan algoritmalardır. Örnekler:

14. Horspool Arama Algoritması: Boyer-Moore algoritmasına benzeyen, ancak daha basit bir kaydırma tablosu kullanan bir algoritma.

15. Kaba Kuvvet Metin Arama Algoritması (Brute Force Text Search, Linear Text Search): Kalıbı metin içinde her konumda karşılaştıran basit bir algoritma.