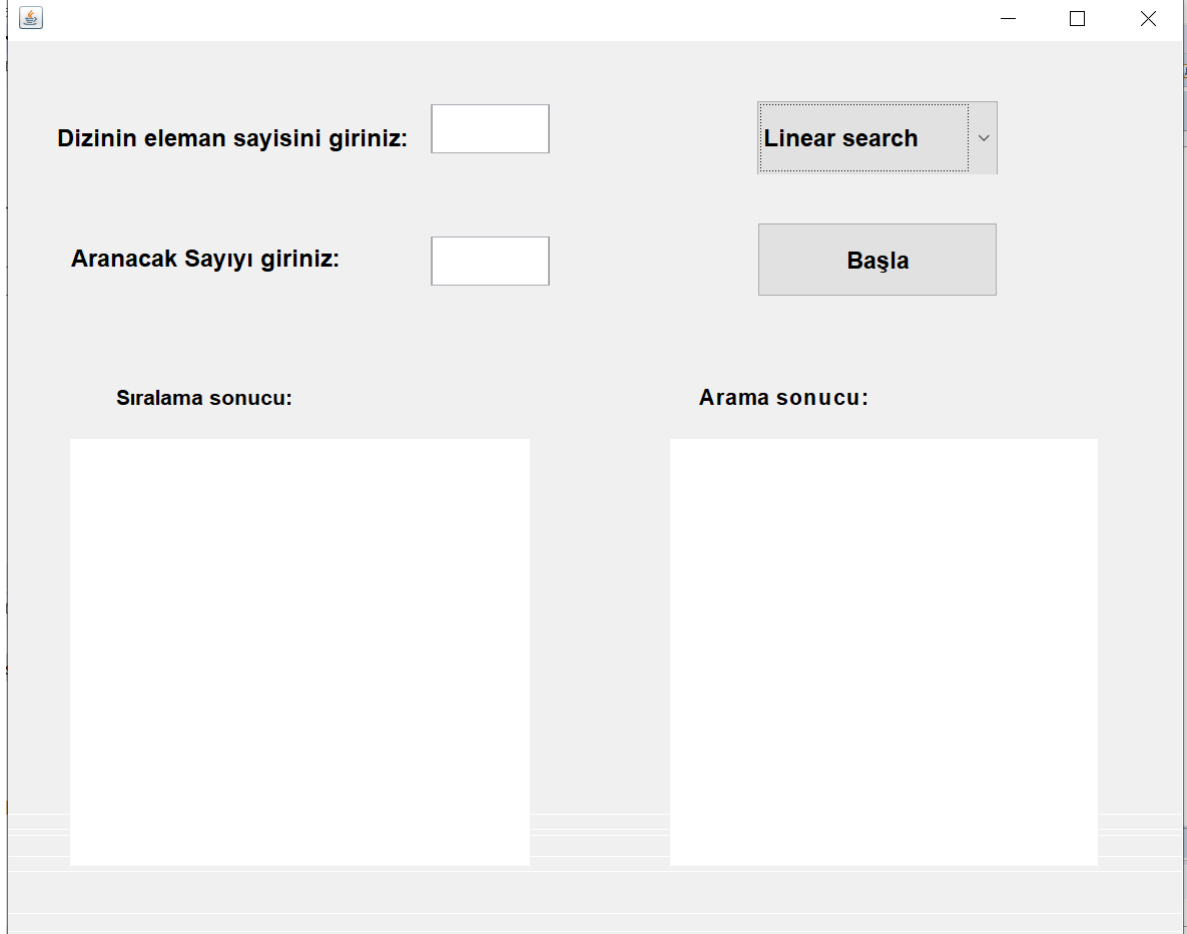


Algoritma Analizi Proje

Sıralama ve Arama algoritmalarının bulunduğu bir Java Projesidir.

Program açıldığında şu şekilde gözükmektedir:



The screenshot shows a Java application window with a light gray background. At the top left, there is a small icon. The main area contains the following elements:

- Dizinin eleman sayısını giriniz:** A text label followed by a white input field.
- Aranacak Sayıyı giriniz:** A text label followed by a white input field.
- Linear search**: A dropdown menu with a downward arrow.
- Başla**: A gray button.
- Sıralama sonucu:** A text label above a large white rectangular box.
- Arama sonucu:** A text label above a large white rectangular box.

Öncelikle dizinin kaç elemanlı olduğunu belirliyorsunuz. Belirlediğiniz eleman sayısı kadar rastgele bir dizi üretilecektir.

Sonrasında Sıralama algoritmalarından birini seçtiyseniz :

- Insertion sort algoritması
- Merge sort algoritması
- Heap sort algoritması
- Quick sort algoritması
- Counting Sort
- Radix Sort

Başlaya tıklayıp sıralanmış diziye ve işlem süresini göreceksiniz.

Arama algoritmalarından birini seçtiyseniz ise:

- Linear search algoritması
- Binary search algoritması

Aranacak sayıyı da girmeniz gerekiyor.

Arama sonucu olarak aranan değerin olup olmadığı, bulunduğu indeks, arama için geçen süre bilgilerini verecektir.

Algoritmaların Açıklamaları:

• Insertion sort algoritması:

-
- Türkçe ismi Sokma Sıralaması olan **Insertion Sort** algoritması diğer sıralama algoritmalarına nazaran kodlaması oldukça kolaydır. Kodlanması kolay olduğu gibi uygulanması da kolaydır. Genelde kart oyunlarında insanlar elindeki kartları sıralamak için farkında olmadan bu algoritmayı uygulamaktadır. Ancak karmaşıklığı N^2 olduğu için verimli bir algoritma sayılmaz.

• Insertion Sort Nasıl Çalışır?

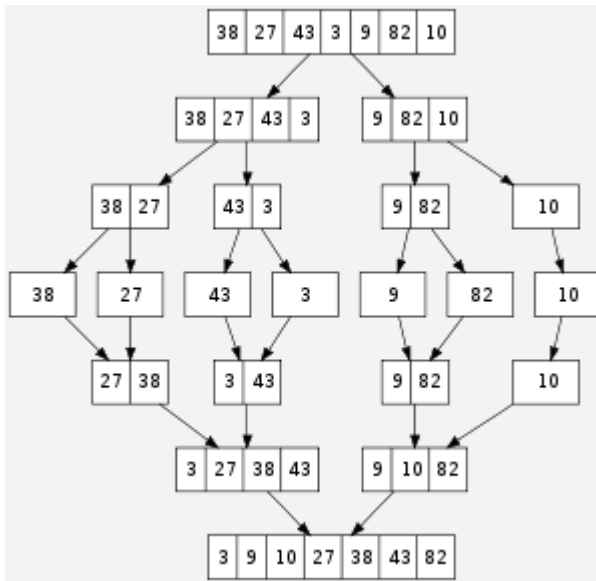
- Algoritmada döngümüz her bir tur döndüğünde sıradaki elemanı sondan başa doğru karşılaştırarak yerine yerleştirme esası çalışmaktadır

Merge Sort

Merge Sort (Birleştirme Sıralaması), diziyi ardışık olarak en küçük alt dizilerine kadar yarılayan sonra da onları sıraya koyarak bireştiren özyineli bir algoritmadır. Yarılama işlemi en büyük alt dizi en çok iki öğeli olana kadar sürer. Sonra "Merge (Birleştirme)" işlemiyle alt diziler ikiye ikiye bölünüş sırasıyla sıralı olarak bir üst dizide birleşir. Süreç sonunda en üstte sıralı diziyeye ulaşılır.

Algoritmanın çalışması kavramsal olarak şöyledir:

1. Sıralı olmayan diziyi ortadan eşit olarak iki alt diziyeye ayırır.
2. Bu ayırma işlemi, alt diziler en çok iki elemanlı olana kadar devam eder.
3. Alt dizileri kendi içinde sıralar.
4. Sıralı iki alt diziyi tek bir sıralı dizi olacak şekilde birleştirir.



Heap Sort:

Verileri zaman ve yer karmaşıklığı açısından etkili bir şekilde sıralamak bilgisayar bilimlerinin bir problemidir. Literatürde bu problemi farklı şekillerde ele alan çeşitli sıralama algoritmaları mevcuttur (Merge Sort, Insertion Sort gibi). Yığın sıralaması (Heap Sort) bu algoritmalarından bir tanesidir, sıralama işlemini yaparken heap veri yapısından faydalanarak karmaşıklığı azaltmayı amaçlamaktadır.

Heap bir ikili ağaç yapısıdır, fakat ikili arama ağacından farkları olduğunu unutmamalıyız. Maksimum ve minimum olmak üzere iki türü bulunur. Peki maksimum ve minimum olması neyi değiştirir? Maksimum heap türünde veriler yukarıdan aşağıya azalan şekilde yerleştirilir, diğer bir deyişle root (ana) düğümdeki eleman en büyüktür, minimum heap' te ise veriler yukarıdan aşağıya artan şekilde yerleştirilir yani root' taki eleman en küçüktür. Bu şekilde verileri sıralarken artan veya azalan sırada sıralama imkanı sağlar. Maksimum heap kullanılarak silme işlemi yapıldığında root düğüm silinerek dizinin sonuna yerleştirilir ve küçükten büyüğe doğru sıralama yapılabilir, minimum heap kullanarak bunun tersini yapabiliriz.

• Quick sort algoritması:

Quick Sort, bilgisayarda tuttuğumuz verileri belli bir düzene göre sıralamamızı sağlayan bir algoritmadır. Belli bir düzen diyorum çünkü her zaman küçükten büyüğe veya büyükten küçüğe sıralamak bizim istediğimiz sonuca ulaşmaz. Ancak basit ve anlaşılması kolay olsun diye bu örneğimizde küçükten büyüğe doğru sıralayacağız.

Quick Sort Nasıl Çalışır?

Amacımız belli. Ancak bu algoritmayı diğer sıralama algoritmalarından farklı kılan durum probleme **yaklaşma biçimidir**. Biraz bahsetmek gerekirse, biz her bir adımda rastgele bir eleman seçiyoruz. Daha sonra

diziyi baştan sona gezerek bu elemandan küçük olanları sola, büyük olanları sağa atıyoruz. Böylelikle seçtiğimiz elemandan küçük bütün sayılar solda, büyük sayılar ise sağda kalmış oluyor.

Bir örnek dizi vererek adım adım ne yaptığımızı inceleyelim:

Başlangıç: 7 2 3 6 10 1 5

(5) elemanını seçelim.

1. Adım: 2 3 1 (5) 10 7 6

Gördüğümüz gibi 5'ten küçükler solda, büyükler sağda kalmış oldu. Bu adımda bir çıkarım yapalım. Biz artık biliyoruz ki soldaki hiçbir eleman 5'in sağına geçemez çünkü hepsi 5'ten küçük olduğundan 5'in sağ tarafında yer alması mümkün değildir. Aynı sebepten ötürü 5'ten büyük elemanların 5'in soluna geçemeyeceğini de rahatlıkla söyleyebiliriz.

Böylelikle artık diziyi ikiye bölebiliriz. {2,3,1} olan diziyi ayrıca sıralayıp yanına (5) ekleyip onun yanına da {10,7,6} dizisini sıralarsak cevaba ulaşmış oluruz.

Counting Sort:

Sayarak sıralama algoritması dizideki değerlerin aralık bilgilerini yeni bir dizi oluşturmak için kullanır. Oluşturulan yeni dizinin her bir satırı ana dizide o satır numarasının değerine sahip öğelerin sayısını gösterir. Yeni dizideki öğe değeri sayıları daha sonra ana dizideki tüm değerlerin doğru konuma konulması için kullanılır.

Radix Sort:

Sirasız verilen dizileri küçükten büyüğe sıralamanın bir yolu da radix sort yani taban sıralamasını kullanmaktır. Gerçek hayatta pek de kullanımı olmamaktadır.

Radix sort yani taban sıralaması isminden de anlaşılacağı üzere sayıları Digitlerine yani basamaklarına göre sıralar. İlk geçişte sadece 1'ler basamağına bakarak bir sıralama yapar, ikinci geçişte 10'lar basamağına bakarak sıralama yapar, 3. geçişte 100ler... bu şekilde gider.

Linear search algoritması

Bu arama algoritması en basit ve çalışma zamanı olarak en kötü algoritmalarından biridir. Çünkü en kötü ihtimal ile veri yapımız üzerinde tüm elemanları gezmesi gerekir. Burada en kötü ihtimalden kastımız algoritmamızın zaman karmaşıklığı(time complexity). Bunun içinde bilgisayar bilimlerinde algoritmaların analizi için çeşitli notasyonlar(asymptotic notations) kullanılır. Bunlar Big O(O), Omega(Ω) ve Theta(θ) notasyonlarıdır.

Big O notasyonu en kötü durum(worst case) için, Omega en iyi durum(best case), Theta ise ortalama durum(average case) için kullanılır. Birazdan bu iki algoritmanın zaman karmaşıklıklarını inceleyeceğiz. Önce mantıklarına bakalım.

Linear Search, verilen veri seti üzerinde her bir eleman ile aranan değeri karşılaştırarak arar. Eğer aranan veri dizide bulunursa dizinin indeksini döner. Bulamazsa -1 gibi bir değer döner.

Binary Search:

Binary Search, sıralı(sorted) bir veri yapısı için kullanılır. Yani algoritmaya aranan veri ve sıralı bir veri yapısı verirsiniz. Algoritma da size önceki örnekteki gibi eğer bulunursa aranan verinin indeksini döner. Bunun için önce elimizdeki verinin sıralanması gerekir. Bunun bir sorting algoritması kullanırsınız.

Binary Search çalışma zamanı olarak Linear Search'den daha iyidir. Her iterasyonda arama uzayını yarıya indirmek üzere tasarlanmıştır. Öncelikle dizinin ortasındaki değeri aranan değer ile karşılaştırır. Eğer aranan değer ortanca değerden küçükse dizinin ikinci yarısını görmezden gelerek ilk yarısında aramaya devam eder. Daha sonra tekrar ilk yarının ortanca değeri ile karşılaştırır. Eğer aranan değer ortanca değerden küçükse sol yarı, büyükse sağ yarı ile devam eder. Bu şekilde aranan değeri bulana kadar sürer. Aranan değer ilk iterasyonda da bulunabilir son iterasyonda da. Ancak Linear Search'den farklı olarak her bir elemanı gezmediği için aranan değeri daha hızlı bulacaktır. İki algoritmanın karşılaştırmasını aşağıda görebilirsiniz. Aralarında ciddi bir hız farkı bulunuyor.