BLM314 Algoritma Analizi Dersi Uygulama Proje Raporu

Programın Çalışma Şekli

Program, iki tanesi arama algoritması ve 7 tanesi sıralama algoritması olmak üzere toplam 9 adet algoritma içermektedir. Program istediğiniz aralıkta rastgele sayılar üreterek istediğiniz algoritmayı/algoritmaları çalıştırmanızı sağlar. Arama sonucu olarak aranan değerin olup olmadığını, bulunduğu indeks, arama için yapılan işlem sayısı ve arama için geçen süre gösterilmektedir. Sıralama sonucu olarak, sıralama sonucu oluşan veri, sıralama için yapılan işlem sayısı ve sıralama için geçen süre gösterilmektedir. Uygulamada bulunan algoritmalar:

1. Doğrusal Arama Algoritması (Linear Search)

2. İkili Arama Algoritması (Binary Search)

3. Eklemeli Sıralama Algoritması (Insertion Sort)

4. Birleştirmeli Sıralama Algoritması (Merge Sort)

5. Yığın Sıralama Algoritması (Heap Sort)

6. Çabuk Sıralama Algoritması(Quick Sort)

7. Kova Sıralama Algoritması (Bucket Sort)

8. Sayarak Sıralama Algoritması (Counting Sort)

9. Taban Sıralama Algoritması (Radix Sort)

1) Linear Search

Bu arama algoritması en basit ve çalışma zamanı olarak en kötü algoritmalardan 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.

2) 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.

3) Inserion Sort

Insertion sort algoritması temel sıralama algoritmalarından bir tanesidir. Algoritmanın mantığına göre elimizdeki A dizisinin elemanları arasında sıralama yapılmak istenildiğinde veri setinin ikinci elemanından başlayarak kendisinden önceki elemanla kıyaslar, eğer kendisinden küçükse yer değiştirir ve bir sonraki elemana geçerek aynı işlemleri tekrarlar.

4) Merge Sort

Bu algoritmada diziyi ardışık olarak en küçük alt dizilerine kadar yarılayan sonra da onları sıraya koyarak bireştiren özyinelemli bir algoritmadır. Yarılama işlemi en büyük alt dizi en çok iki öğeli olana kadar sürer. Sonra Merge işlemiyle alt diziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide birleşir. Süreç sonunda en üstte sıralı diziye ulaşılır.

5) Heap Sort

Heap Sort algoritması, veri setinin ilk elemanını root olarak seçer ve kalan elemanları sağ ve solmak üzere çocukları haline getirir (ağaç yapısı gibi). Daha sonra bu ağaç yapısına benzer yapıdan karşılaştırma yaparak yer değiştirmeler yapar ve algoritmayı sıralı hale getirir.

6) Quick Sort

Bu algoritma, başlarken dizinin terimleri arasından bir terimi pivot olarak seçer. Sonra verilen diziyi üç alt diziye ayrıştırır. Pivot terimden küçük olan terimlerin hepsini (soldaki) birinci alt diziye taşır. İkinci alt dizi sadece pivot terimi tutan tek terimli bir dizidir. Pivot terimden büyük olan terimlerin hepsini (sağdaki) ikinci alt diziye taşır. Sonra sol ve sağ alt dizilere aynı ayrıştırma yöntemini, alt diziler tek terimli birer diziye indirgenene kadar uygular ve sıralama işlemi biter. Algoritma özyinelimeli olarak çalışır.

7) Bucket Sort

Bucket Sort Algoritması, sıralanacak bir diziyi parçalara ayırarak sınırlı sayıdaki kovalara atan bir sıralama algoritmasıdır. Ayrışma işleminin ardından her kova kendi içinde ya farklı bir algoritma kullanılarak ya da kova sıralamasını özyinelemeli olarak çağırarak sıralanır.

8) Counting Sort

Bu algoritmada, dizideki değerlerin aralık bilgilerini yeni bir dizi oluşturmak için kullanır. Oluşturulan yeni dizinin her bir satırı ana dizide o satır numarasının değerine sahip ögelerin sayısını gösterir. Yeni dizideki öge değeri sayıları daha sonra ana dizideki tüm değerlerin doğru konuma konulması için kullanılır.

9) Radix Sort

Verilen sayıları sağ-dan sola doğru ya da soldan sağa doğru basamak değerlerine göre ayırmaktadır. Her geçişte, aynı basamak değerini alan sayılar bir araya getirilmektedir. .Bu işlem dizinin en büyük elemanının basamak sayısı kadar tekrar edilir. Geçişler bitince, sayılar sıralı hale gelmektedir

KAYNAKÇA

- https://tugrulbayrak.medium.com/search-arama-algoritmalari-binary-linear-5260431ba9a3
- http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/sorting/RadixSort/RadixSort.pdf
- http://www.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/sorting/MergeSort/MergeSort.pdf
- Sayarak Sıralama (Counting Sort) Algoritması - YBSBLog
- Bucket Sort in C and C++ | LaptrinhX
- Quicksort Algoritması (itu.edu.tr)
- http://furkanalniak.com/siralama-algoritmalari-heap-sort-yigin-siralamasi/
- https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/insertion-sort-algoritmas%C4%B1