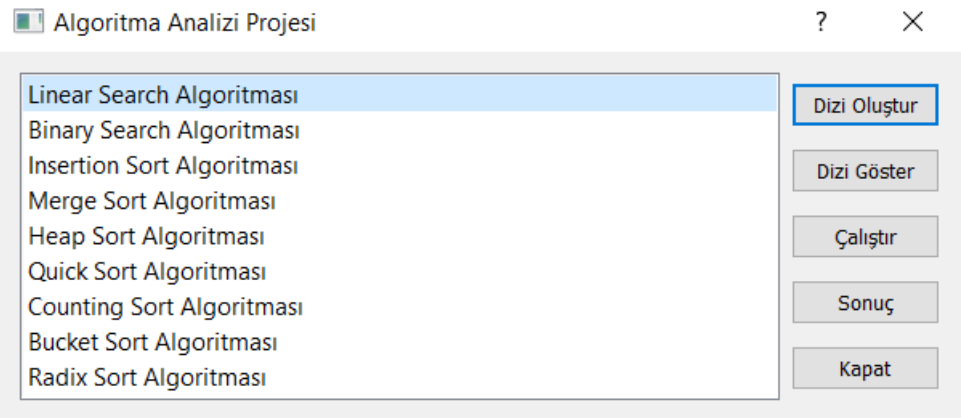
# BLM314 Algoritma Analizi Dersi Uygulama Projesi Raporu

*Mehmetcan DALMAZGİL*

Bursa Teknik Üniversitesi, 18360859001@ogrenci.btu.edu.tr

## 1. Giriş

Proje kapsamında Python programlama dili kullanılarak, 2'si arama 7'si sıralama algoritması olmak üzere bu algoritmaların program içerisinde oluşturulan sayı kümeleri(dizi) üzerinde kullanılabildiği ve kullanılan algoritmanın sonuçlarının (algoritma çalışma süresi, algoritma çalışırken gerçekleşen işlem sayısı, çalıştırılan algoritma arama algoritması ise aranan sayının dizide bulunup bulunmadığı, çalıştırılan algoritma sıralama algoritması ise dizinin son halinin) kullanıcıya sunulduğu GUI(Grafiksel Kullanıcı Arayüzü) destekli bir uygulama geliştirilmiştir. Uygulama arayüzü geliştirilirken Python içerisinde bulunan PyQt5 GUI araç seti kullanılmıştır. Uygulamaya ait ekran görüntüsü Şekil 1'de verilmiştir.




*Şekil 1.* Uygulama ekran görüntüsü

## 2. Kullanılan Algoritmalar

Bu bölümde uygulama içerisinde kullanılan algoritmalara değinilecektir. Giriş bölümünde uygulamada 2 arama ve 7 sıralama algoritması kullanıldığına değinilmişti. Bu algoritmalar şu şekildedir;

- Arama Algoritmaları
  - Linear Search Algoritması
  - Binary Search Algoritması

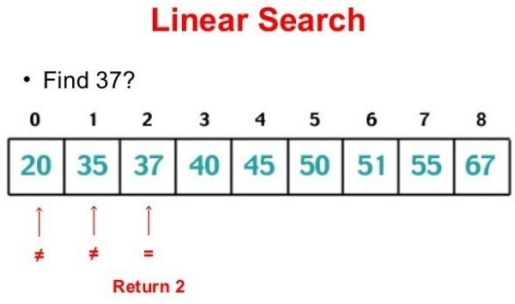
- Sıralama Algoritmaları
  - Insertion Sort Algoritması
  - Merge Sort Algoritması
  - Heap Sort Algoritması
  - Quick Sort Algoritması
  - Counting Sort Algoritması
  - Bucket Sort Algoritması
  - Radix Sort Algoritması

## 2.1. Kullanılan Arama Algoritmaları

Arama algoritması, çeşitli veri yapılarının (data structures) üzerinde bir bilginin aranması sırasına kullanılan algoritmaların genel ismidir. Örneğin bir dosyada bir kelimenin aranması, bir bilgisayar yapısında (tree) bir düğümün (node) aranması veya bir dizi(array) üzerinde bir verinin aranması gibi durumlar bu algoritmaların çalışma alanlarına girmektedir [1]. Uygulamada arama algoritmaları bir dizi üzerinde kullanıcının belirlemiş olduğu bir verinin aranması için kullanılmıştır.

### 2.1.1 Linear Search Algoritması (Lineer Arama)

Linear Search Algoritması aranan veriyi, dizinin ilk öğesinden başlayarak son öğesine doğru, her terimle tek tek karşılaştırır. Bu algoritma, dizi içinde aranana eşit olan bir terim bulursa onun indisini verir. Aranan veriye eşit olan terim bulamazsa -1 değerini verir. –1 değeri verdiğinde, aranan öğe dizi içinde değildir. Dizinin terimlerini, sırayla aranan öğe ile karşılaştırdığı için, bu algoritmaya dizisel arama algoritması adı da verilir [2].Bu arama algoritması en basit ve çalışma zamanı olarak en kötü algoritmalardan biridir. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.1'de verilmiştir.

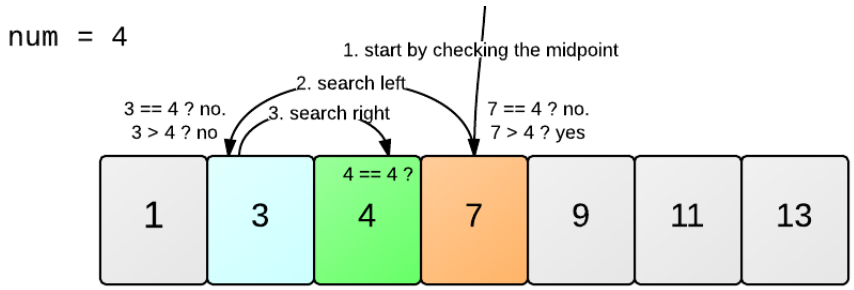



*Şekil 2.1.* Linear Search Algoritması mantığı [3]

### 2.1.2 Binary Search Algoritması (İkili Arama)

Binary Search, sıralı(sorted) bir veri yapısı için kullanılır. Yani algoritmaya aranan veri ve sıralı bir veri yapısı verilmektedir. 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 algoritma çalışmaktadır [4]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.2'de verilmiştir.



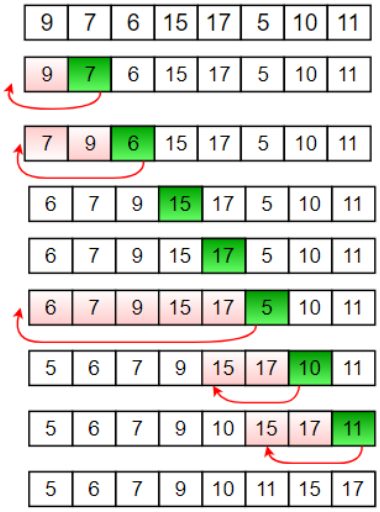

*Şekil 2.2.* Binary Search Algoritması mantığı [5]

## 2.2. Kullanılan Sıralama Algoritmaları

Sıralama algoritması, bilgisayar bilimlerinde ya da matematikte kullanılan, verilen bir listenin elemanlarını belirli bir sıraya sokan algoritmaların genel ismidir. En çok kullanılan sıralama türleri, sayı büyüklüğüne göre sıralama ve alfabetik sıralamadır. Sıralama işleminin verimli yapılması, arama ve birleştirme algoritmaları gibi çalışması için sıralanmış dizilere gereksinim duyan algoritmaların başarımının yüksek olması için önemlidir [6]. Uygulamada sıralama algoritmaları, kullanıcının belirlemiş olduğu değerlere göre(dizinin uzunluğu, dizinin alabileceği minimum değer ve maximum değer) random(rastgele) sayılardan oluşturulmuş dizi elemanlarını sıralamak için kullanılmıştır.

### 2.2.1 Insertion Sort Algoritması (Eklemeli Sıralama)

Insertion Sort Algoritması, düzensiz dizi elemanlarını tek tek ele alarak her birini dizinin sıralanmış kısmındaki uygun yerine yerleştirme esasına dayanır. Küçük boyutlu dizileri sıralamada hızlı olsa da çok sayıda veriyi sıralarken Insertion Sort diğer sıralama yöntemlerine göre çok yavaş kalmaktadır. Algoritmada, ikinci elemandan başlayarak elemanın kendinden önceki elemanlarla karşılaştırılması suretiyle büyük elemanlar dizide sağa doğru kaydırılır. Böylelikle açılan uygun pozisyona o anda sıralanmakta olan eleman yerleştirilir. Yani, algoritmanın küçükten büyüğe sıralama yaptığı düşünülürse, sayı dizisinin ikinci elemanını kendisine anahtar eleman olarak seçer. Bu anahtar eleman bir önceki elemandan başlayıp, kendinden önceki tüm sayılarla, anahtar olarak seçilen sayıyı kıyaslar. Kendinden büyük olan her sayıyla yerleri değiştirir. Kendinden küçük sayıyla karşılaştığında yer değiştirme işlemi biter. Ardından dizinin son elemanına kadar anahtar sayı seçimi ve devamındaki tüm işlemler devam eder [7]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.3'te verilmiştir.



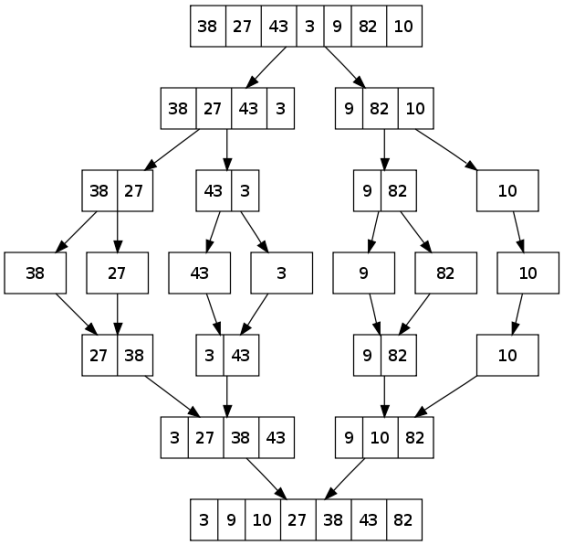

*Şekil 2.3.* Insertion Sort Algoritması mantığı [8]

### 2.2.2 Merge Sort Algoritması (Birleştirmeli Sıralama)

Merge Sort Algoritması, 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 (Birleştirme)" işlemiyle alt diziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide birleşir. Süreç sonunda en üstte sıralı diziye ulaşılır [9]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.4'te verilmiştir.

Algoritmanın çalışması kavramsal olarak şöyledir:

1. Sıralı olmayan diziyi ortadan eşit olarak iki alt diziye 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.



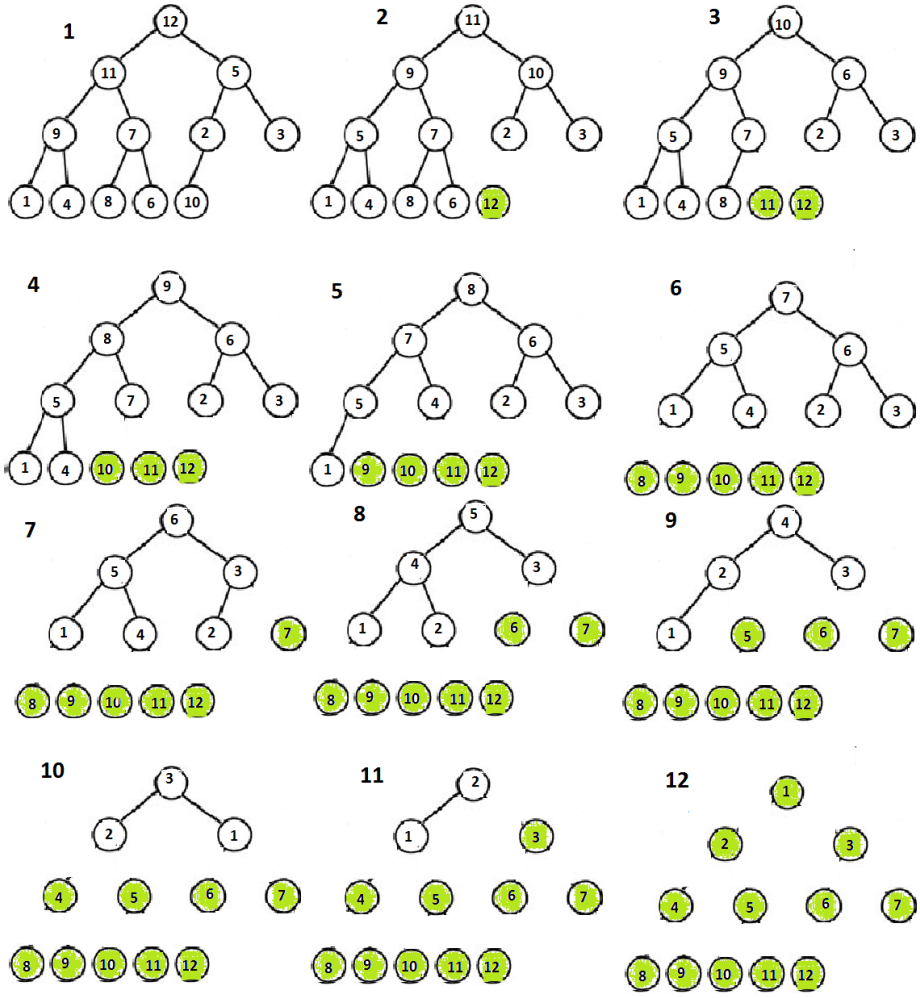

*Şekil 2.4.* Merge Sort Algoritması mantığı [10]

### 2.2.3 Heap Sort Algoritması (Yığın Sıralama)

Heap Sort Algoritması, verinin hafızada sıralı tutulması için geliştirilen sıralama algoritmalarından bir tanesidir. Bu algoritma arka planda bir yığın ağacı(heap) oluşturmakta ve bu ağacın en üstündeki sayıyı alarak sıralama işlemini gerçekleştirmektedir[11]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.5'te verilmiştir.

Algoritma adımları şu şekilde özetlenebilir:

1. Sayı grubundan bir ağaç oluşturulur.
2. Bu ağaç yaprak olmayan en son elemandan ilk(kök) elemana doğru heapify metoduyla yığınlaştırılır.
3. En üstte(kökte) duran yani en büyük olan değer alınarak sonuç dizisinin son elemanı yapılır.
4. Sonra geriye kalan sayılar tekrar yığınlaştırılır (heapify) ve bu işlem eleman kalmayana kadar yapılırsa sonuç dizisindeki veriler sıralanmış olarak elde edilir.
5. Bu sayı dizisi ilk başta verilen sayı dizisinin küçükten büyüğe sıralanmış halidir. Şayet büyükten küçüğe sıralanmak istenirse algoritmanın biraz değiştirilmesi gerekir.



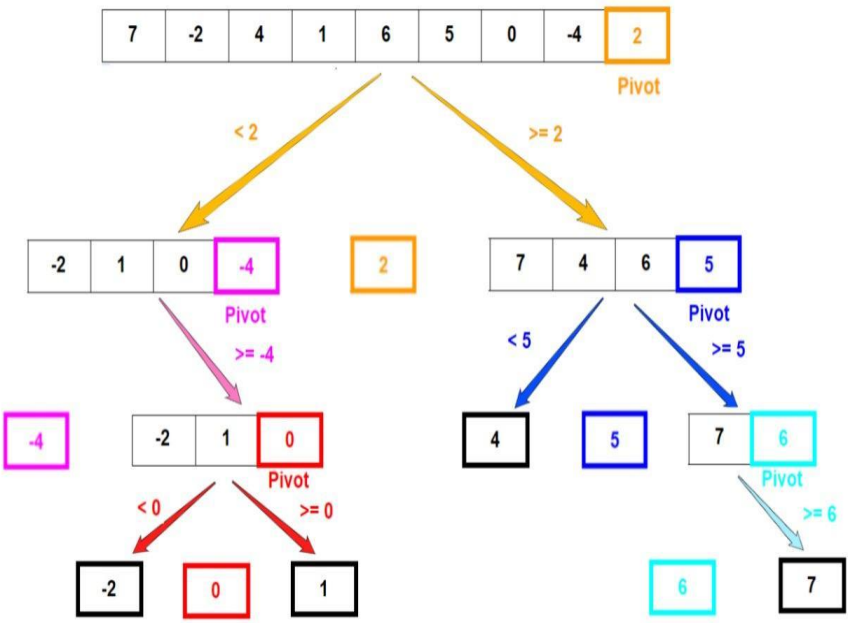

*Şekil 2.5.* Heap Sort Algoritması mantığı [12]

### 2.2.4 Quick Sort Algoritması (Hızlı Sıralama)

Quick Sort Algoritması, sıralanacak bir diziyi daha küçük iki parçaya ayırıp oluşan bu küçük parçaların kendi içinde sıralanması mantığıyla çalışır[13]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.6'da verilmiştir.

Algoritma adımları şu şekilde özetlenebilir:

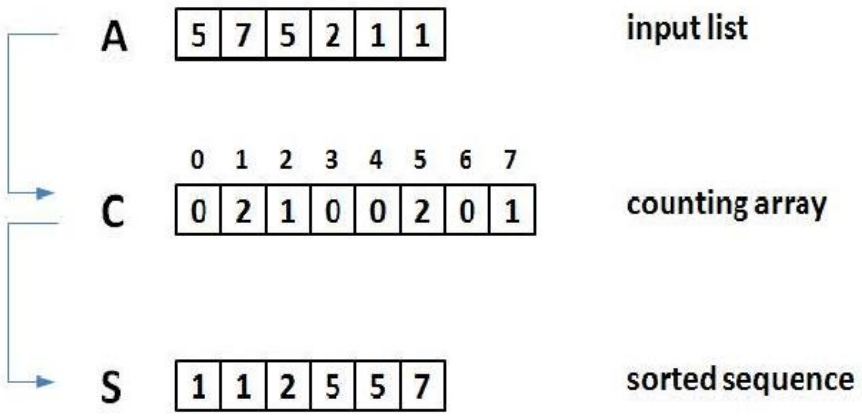
1. Diziden herhangi bir elemanı pivot(kilit) eleman olarak seçer.
2. Diziyi, pivot elemandan küçük olan bütün elemanlar pivot elemanın önüne, pivot elemandan büyük olan bütün elemanlar pivot elemanın arkasına gelecek biçimde düzenler. Pivot elemana eşit olan sayılar sıralamanın küçükten büyüğe ya da büyükten küçüğe olmasına bağlı olarak pivot elemanın her iki tarafına da geçebilir.
3. Quick Sort Algoritması özyinelemeli çağrılarak, oluşan küçük diziler tekrar sıralanır.
4. Algoritma eleman sayısı sıfır olan bir alt diziye ulaşana kadar bu işlem devam eder.
5. Eleman sayısı sıfır olan bir alt diziye ulaşıldığında algoritma bu dizinin sıralanmış olduğunu varsayar ve sıralama işlemi tamamlanmış olur.




*Şekil 2.6.* Quick Sort Algoritması mantığı [14]

### 2.2.5 Counting Sort Algoritması (Sayarak Sıralama)

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 ögelerin sayısını gösterir. Yeni dizideki öge değeri sayıları daha sonra ana dizideki tüm değerlerin doğru konuma konulması için kullanılır [15]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.7'de verilmiştir.



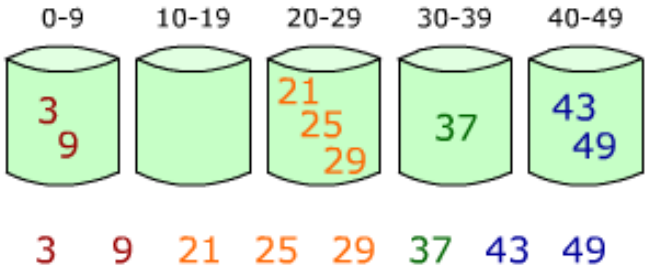

*Şekil 2.7.* Counting Sort Algoritması mantığı [16]

### 2.2.6 Bucket Sort Algoritması (Kova Sıralama)

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 [17]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.8'de verilmiştir.

Kova sıralaması aşağıdaki biçimde çalışır:

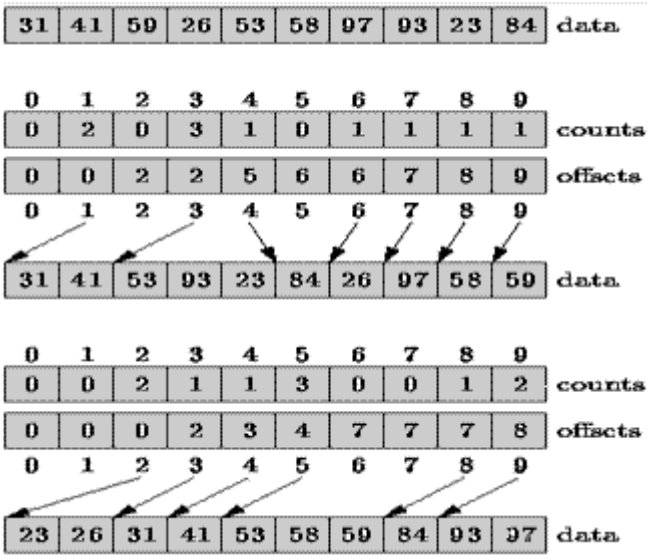
1. Başlangıçta boş olan bir "kovalar" dizisi oluştur.
2. Asıl dizinin üzerinden geçerek her öğeyi ilgili aralığa denk gelen kovaya at.
3. Boş olmayan bütün kovaları sırala.
4. Boş olmayan kovalardaki bütün öğeleri yeniden diziye al.




*Şekil 2.8.* Bucket Sort Algoritması mantığı [18]

### 2.2.7 Radix Sort Algoritması (Taban Sıralama)

Radix Sort Algoritması, şimdiye dek açıklanan sıralama algoritmalarının aksine dizinin terimlerini birbirleriyle mukayese etmemektedir. Onun yerine, sayıları basamak (hane) değerlerine göre gruplandırmaktadır. 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. Geçişler bitince, sayılar sıralı hale gelmektedir [19]. Algoritmanın çalışma mantığıyla ilgili görsel Şekil 2.9'da verilmiştir.

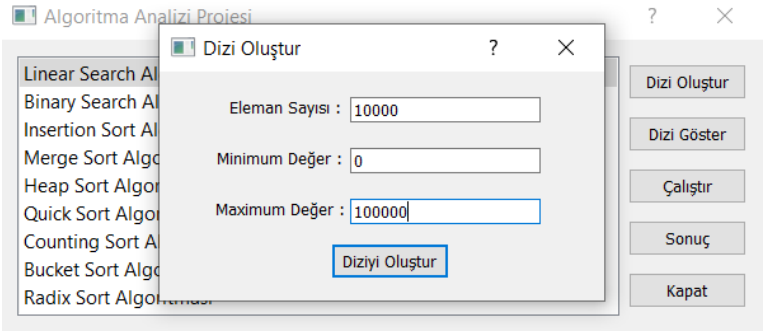



*Şekil 2.9.* Radix Sort Algoritması mantığı [20]

## 3. Uygulamanın Kullanılması

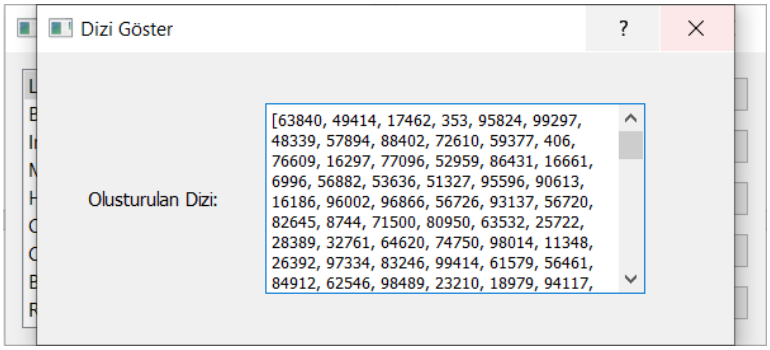
Uygulamanın kullanımını göstermek adına Counting Sort ve Linear Search algoritmaları için uygulama çalıştırılmıştır. Uygulama çalıştırılırken izlenen adımlar şu şekildedir.

1) Algoritmaların çalıştırılabilmesi için öncelikli olarak kullanıcı uygulama içerisinde bir dizi oluşturmalıdır. Dizi oluşturmak için ana ekranda bulunan 'Dizi Oluştur' butonuna tıklıyoruz. Tıkladığımızda uygulama random sayılardan oluşturacağı dizi için kullanıcıdan dizinin sahip olacağı eleman sayısı, dizi için oluşturulacak random sayıların minimum ve maximum sınır değerlerini istemektedir. Bölüm ile ilgili görsel Şekil 3.1'de verilmiştir.



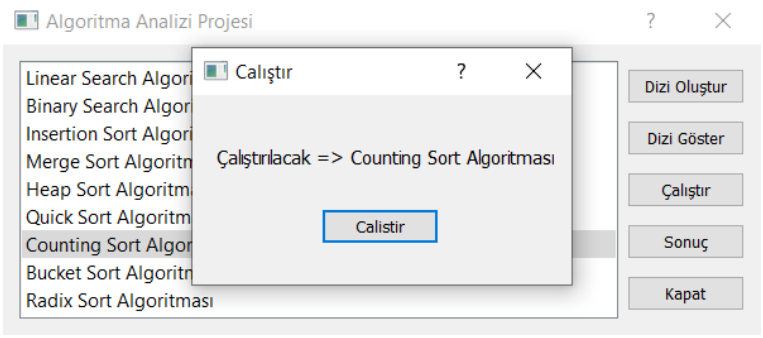

*Şekil 3.1.* Uygulamada dizi oluşturma

0 – 100000 aralığından seçilmiş rastgele sayılarla 10000 elemanlık bir dizi oluşturduk. Oluşturulan diziye 'Dizi Göster' bölümünden ulaşabiliyoruz.




*Şekil 3.2.* Oluşturulan dizi

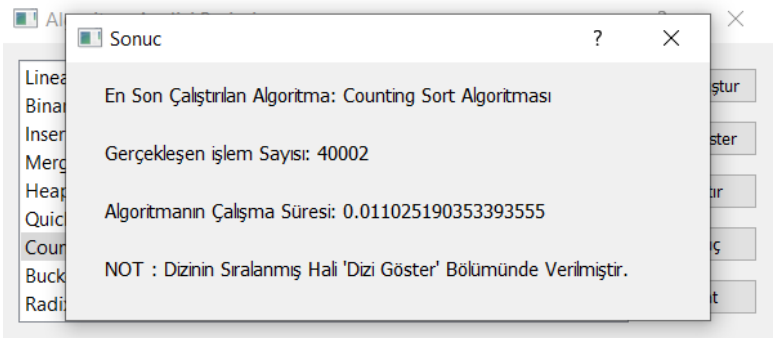
2) Dizimizi oluşturduk. Uygulamamızda Counting Sort algoritmasını çalıştırmak için algoritmaların verilmiş olduğu bölümden algoritmamızı seçerek 'Çalıştır' butonuna tıklıyoruz. Butona tıkladığımızda Şekil 3.3'te verilen ifadeyle karşılaşıyoruz.




*Şekil 3.3.* Algoritmanın çalıştırılması

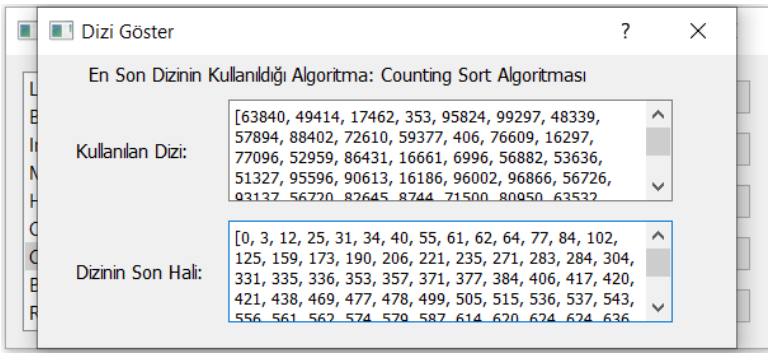
Son olarak çıkan ekranda da 'Çalıştır' butonuna tıklayarak oluşturduğumuz dizi için Counting Sort algoritmasını çalıştırmış oluyoruz.

3) Counting Sort algoritması çalıştırıldı. Algoritmanın çalıştırılması esnasında gerçekleşen işlem sayısı ve algoritmanın çalışma süresi değerlerine 'Sonuç' bölümünden ulaşıyoruz.



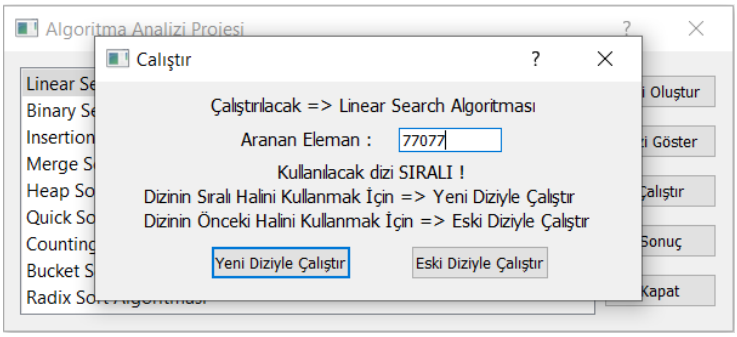

*Şekil 3.4.* Elde edilen sonuçlar

Algoritmanın çalıştırılması sonucunda elde edilen dizinin ilk ve son haline 'Dizi Göster' bölümünden ulaşıyoruz.



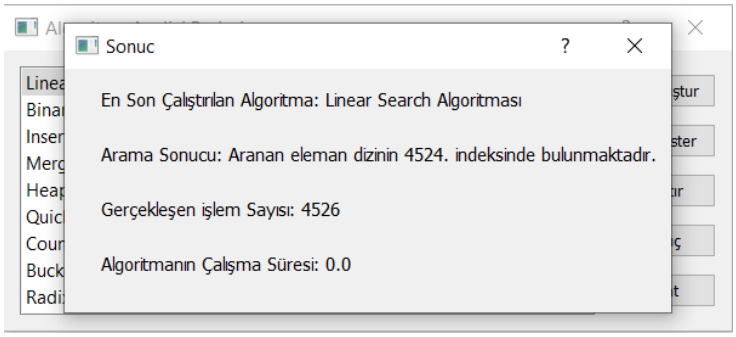

*Şekil 3.5.* Dizinin ilk ve son hali

4) Counting Sort algoritması için işlemleri tamamladık. Şimdi Linear Search algoritmasını çalıştırmak istiyoruz. Uygulamamız içerisinde önceden dizi tanımlamıştık. Linear Search algoritmasını seçip 'Çalıştır' butonuna tıkladığımızda Şekil 3.6'da verilen bir ifadeyle karşılaşıyoruz.

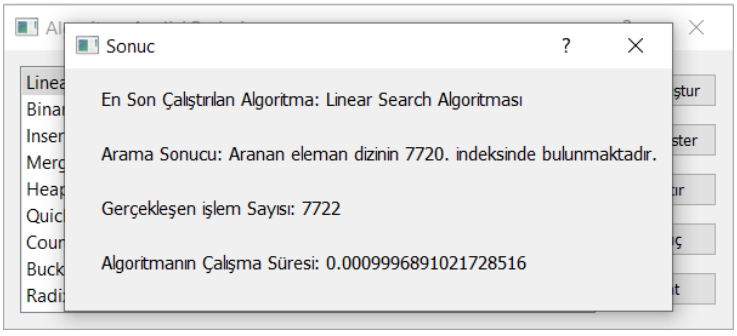



*Şekil 3.6.* Linear Search algoritması çalıştırma penceresi

Linear Search algoritması çalıştırılamadan önce bir sıralama algoritması olan Counting Sort algoritması çalıştırıldığı için en başta oluşturduğumuz dizinin elemanları sıralandı. Sıralı bir dizide yapılan arama işlemiyle sıralı olmayan bir dizide yapılan arama işlemini sonuçları farklı olacağından uygulamada bu seçim kullanıcıya bırakılmıştır. Kullanıcı ister dizinin sıralı halini(son halini), ister önceki halini kullanarak Linear Search algoritmasını çalıştırabilmektedir(Dizinin önceki hali ve son haline Şekil 3.5'te de görüldüğü gibi 'Dizi Göster' bölümünden ulaşılabilmektedir). Şekil 3.7'de dizinin eski hali kullanılarak elde edilen sonuçlar, Şekil 3.8'de dizinin yeni(sıralı) hali kullanılarak elde edilen sonuçlar verilmiştir.




*Şekil 3.7.* Dizinin eski hali kullanılarak elde edilen sonuçlar




*Şekil 3.8.* Dizinin yeni(sıralı) hali kullanılarak elde edilen sonuçlar

‘Kapat’ butonuyla uygulama istenilen zamanda kapatılabilmektedir. Uygulama içerisinde kullanıcı tarafından yapılabilecek hatalı işlemler (dizi oluşturuken minimum değeri maximum değerden büyük girmek, dizi oluşturmadan herhangi bir algoritma çalıştırmak vb.) için uyarı pencereleri de bulunmaktadır. Uygulama içerisinde bulunan diğer arama ve sıralama algoritmaları da raporda örneği verilen algoritmalar gibi başarılı bir şekilde çalıştırılabilmektedir.

## 4. Uygulamanın Çalışması için Gerekli Yazılımlar

Uygulamanın çalışabilmesi için bilgisayarınızda Python3 ile Python PyQt5 GUI araç seti eklentisinin kurulu olması gerekmektedir.

- Python3 kurulum linki => https://www.python.org/downloads/
- PyQt5 kurulum linki => https://pypi.org/project/PyQt5/

Kurulum işlemlerinden sonra ‘AlgoritmaAnaliziProjesi.py” dosyasını çalıştırarak uygulamayı açabilirsiniz.

## 5. Kaynakça


[1] https://medium.com/@enesates03/arama-algoritmalar%C4%B1-search-algorithms-nedir-7c8be09d541a
[2] http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/searching/LinearSearch.pdf
[3] https://www.slideshare.net/ReemAlattas/linear-search-binary-search
[4] https://tugrulbayrak.medium.com/search-arama-algoritmalari-binary-linear-5260431ba9a3
[5] https://www.learneroo.com/modules/71/nodes/399
[6] https://tr.wikipedia.org/wiki/S%C4%B1ralama_algoritmas%C4%B1
[7] https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/insertion-sort-algoritmas%C4%B1
[8] https://www.geeksforgeeks.org/recursive-insertion-sort/
[9] https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/merge-sort-(bile%C5%9Ftirme-s%C4%B1ralamas%C4%B1)-algoritmas%C4%B1
[10] https://barisariburnu.github.io/blog/2012/10/22/siralama-algoritmalari-5-merge-sort.html
[11] https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/heapsort-algoritmas%C4%B1
[12] https://formulafunction.wordpress.com/2017/07/18/heapsort/
[13] http://www.aybe.itu.edu.tr/seyir-defteri/blog/2013/09/08/quicksort-algoritmas%C4%B1
[14] https://stackabuse.s3.amazonaws.com/media/quicksort-in-javascript-1.jpg
[15] https://www.ybsblog.com/sayarak-siralama-counting-sort-algoritmasi/
[16] https://www.researchgate.net/figure/Example-of-counting-sort_fig4_220686480
[17] https://tr.wikipedia.org/wiki/Kova_s%C4%B1ralamas%C4%B1
[18] https://laptrinhx.com/bucket-sort-in-c-and-c-1328728244/
[19]http://mail.baskent.edu.tr/~tkaracay/etudio/ders/prg/dataStructures/sorting/RadixSort/RadixSort.pdf
[20] https://radixplym.wordpress.com/2014/06/19/radix-sort/
https://wmaraci.com/nedir/gui
https://birhankarahasan.com/pyqt-nedir-qt-designer-nedir-python-arayuz-olustur
https://en.wikipedia.org/wiki/PyQt