



Aslıhan Türkdönmez

## **İÇİNDEKİLER**

- A. Ara Sıra nedir?**
- B. Sitenin Kullanımı**
- C. Çalışma Prensibi**
- D. Kullanılan Algoritmalar**

## A. Ara Sıra Nedir?

Ara Sıra çeşitli algoritmik işlemler yapan bir web sitesidir. Kullanıcıların arama ve sıralama algoritmalarını karşılaştırabilmeleri amaçlanan bu sitede kullanıcılar; rastgele olarak sayılar oluşturabilir, bu sayıların maximum - minimum değerlerini belirleyebilir, aynı anda kaç sayı ile işlem yapacağını seçebilmektedir. Bunların yanı sıra kullanıcılar seçtikleri özelliklere sahip sayılara;

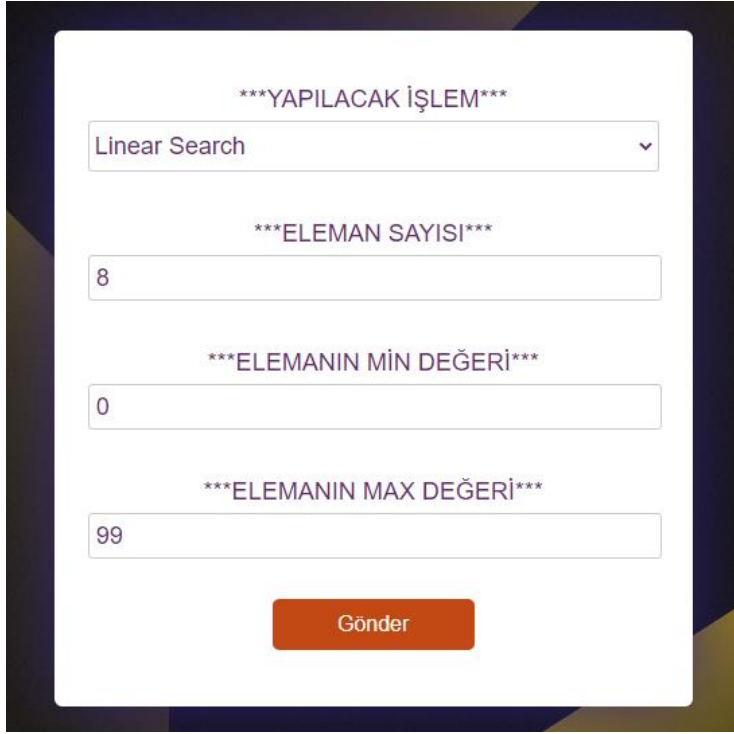
- Linear Search
- Insertion Sort
- Binary Search
- Merge Sort
- Heap Sort
- Quick Sort
- Counting Sort
- Bucket Sort
- Radix Sort

gibi arama veya sıralama algoritmaları uygulayabilmektedir. Seçilen algoritmaya göre gelen sayfa içerisinde (detaylar *B. Sitenin Kullanımı*'nda gösterilecektir.) kullanılan algoritma hakkında çeşitli bilgilendirmeler, ve ayrıca yapılan işlemlerin detayları bulunmaktadır.

## B. Sitenin Kullanımı

Ara Sıra web sitesinin kullanım şekli burada detaylıca açıklanacaktır.

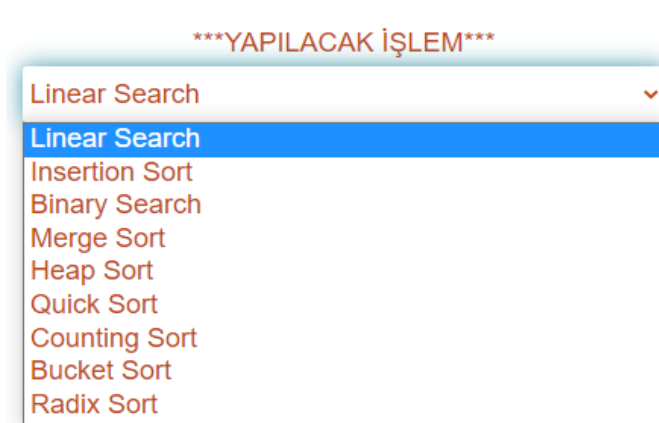
### B.1. Giriş Sayfası

A screenshot of a web form titled '\*\*\*YAPILACAK İŞLEM\*\*\*'. It contains four input fields: a dropdown menu for 'Linear Search', a text box for '8' under the label '\*\*\*ELEMAN SAYISI\*\*\*', a text box for '0' under the label '\*\*\*ELEMANIN MİN DEĞERİ\*\*\*', and a text box for '99' under the label '\*\*\*ELEMANIN MAX DEĞERİ\*\*\*'. At the bottom is an orange button labeled 'Gönder'.

Siteyi açıldığında kullanıcıyı sade ve anlaşılır arayüz karşılamaktadır. Bir adet form kutucuğu bulunmakta olup kullanıcı bu form kutucuğunda ihtiyacına göre düzenlemeler yaptıktan sonra gönder tuşuna basması gerekmektedir.

(Görsel 1.1)

Form kutucuğunu detaylıca açıklamak gerekirse:

A screenshot of a dropdown menu titled '\*\*\*YAPILACAK İŞLEM\*\*\*'. The menu is open, showing a list of sorting algorithms: 'Linear Search', 'Insertion Sort', 'Binary Search', 'Merge Sort', 'Heap Sort', 'Quick Sort', 'Counting Sort', 'Bucket Sort', and 'Radix Sort'. The 'Linear Search' option is currently selected and highlighted in blue.

(Görsel 1.2)

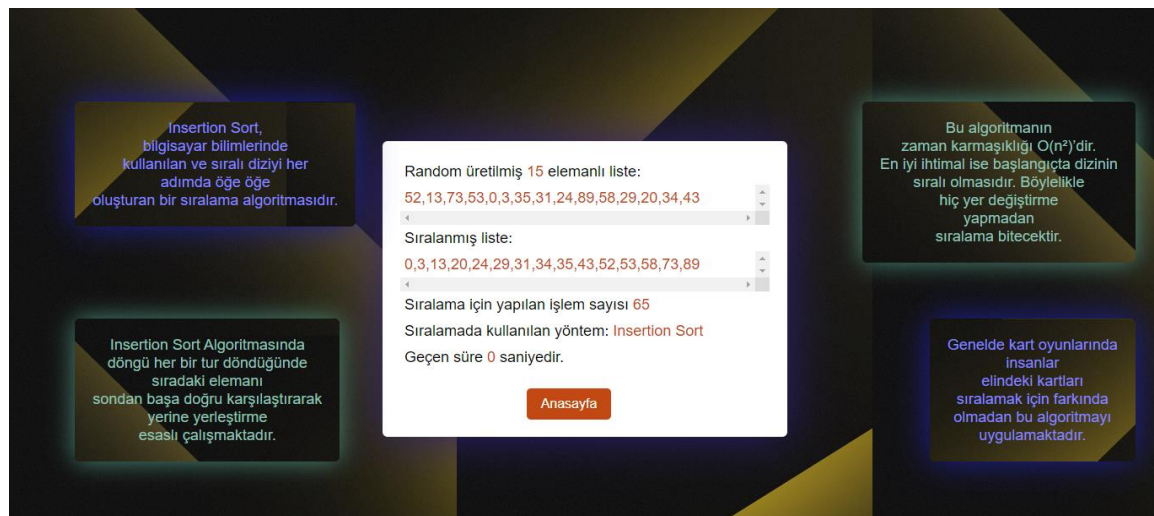
Yapılacak işlem kutusuna tıklandığında işlem yapılabilecek algoritmalar ekrana gelmektedir. Kullanıcı bu algoritmalar arasından ihtiyacına göre bir seçim yapabilir.

İlk seçimi gerçekleştirdikten sonra Görsel 1.1'den de görüldüğü üzere;

- **Eleman Sayısı:** Rastgele oluşturulacak sayıların kaç elemanlı olacağını belirler.
- **Elemanın Min Değeri:** Rastgele oluşturulacak sayıların en küçük değerinin kaç olacağını belirler.
- **Elemanın Max Değeri:** Rastgele oluşturulacak sayıların en büyük değerinin kaç olacağını belirler.

Tüm değerler girildikten sonra sonuçları elde etmek için “Gönder” butonuna basılmalıdır.

## B.2. Sonuç Sayfası (Sıralama İçin)



(Görsel 2.1)

Seçilen algoritmaya göre şekil alan bu sayfada kullanıcılar;

- ✓ Rastgele üretilen listenin tamamına
- ✓ Sıralanmış listenin tamamına
- ✓ Listeyi sıralamak için yapılan işlem sayısına
- ✓ Hangi yöntem ile sıralama yapıldığına
- ✓ Sıralama yapılırken geçen sürenin saniye cinsinden miktarına ulaşabilmektedir.

Aynı zamanda solda ve sağda olmak üzere toplam 4 tane olan kutucuklar içerisinde seçilen algoritma hakkında bilgiler yer almaktadır.

## B.2. Sonuç Sayfası (Arama İçin)

Bu sayfanın mesajı

Arama algoritması seçtiniz...

Lütfen aranacak sayıyı giriniz.

Tamam İptal

(Görsel 2.2)

Arama algoritmalarından herhangi bir tanesi seçilip, “Gönder” butonuna basıldığında ekranın üst kısmında Görsel 2.2’de görülen mesaj ile karşılaşılır. Kullanıcı boş olan kutucuğa rastgele oluşturulacak listede aramak istediği elemanı girmelidir. Aramak istediği sayıyı girdikten sonra “Tamam” butonuna bastığında sonuç sayfasına yönlendirilecektir.

**Not1:** Boş kutucuk içerisine sayı dışında (harf,sembol,işaret vb) girilecek hiçbir değer kabul edilmeyecektir. Herhangi bir değer girilmemesi de bu durumla eşdeğerdir.

Sayı dışında bir değer girilmesi durumunda aşağıda gösterilen (Görsel 2.3) uyarı ile karşılaşılacaktır.

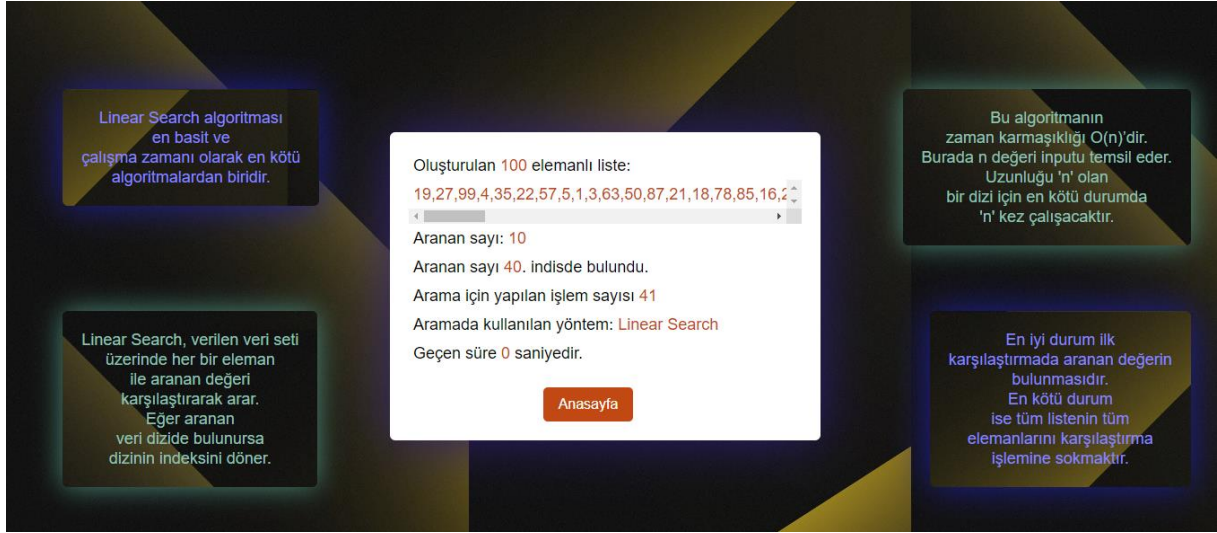
Bu sayfanın mesajı

Lütfen sayı giriniz!

Tamam

(Görsel 2.3)

Görsel 2.3’de “Tamam” butonuna tıklandıktan sonra tekrar Görsel2.2’de gösterilmiş olan kutucuk ekrana gelecektir. Kullanıcı “İptal” butonuna tıklayarak seçimlerini tekrar düzenleyebilir veya aramak istediği sayıyı girerek sonuç ekranına ulaşabilir.



(Görsel 2.4)

Kullanıcı “Tamam” butonuna tıkladıktan sonra Görsel 2.4’de gösterilen ekran ile karşılaşacaktır. Sıralama algoritmalarından farklı olarak ;

- ✓ Aranan sayının kaç olduğu
- ✓ Aranan sayının bulunduğu/bulunamadığı
- ✓ Aranan sayı bulundu ise kaçınıcı indiste bulunduğu

bilgilerini bize vermektedir.

Kullanıcı her iki algoritma çeşidinin sonuç sayfasında da “Anasayfa” butonuna tıklayarak giriş sayfasına dönebilmekte ve tekrar tekrar seçim yapabilmektedir.

## C. Çalışma Prensibi

Ara Sıra sitesinin çalışma prensibi şu şekildedir;

Öncelikle giriş sayfasındaki “Gönder” butonuna tıklandığında *onclick()* metodu devreye girer ve javascript dosyası içinde bulunan *gonder()* fonksiyonu çalışır.

*gonder()* fonksiyonu Yapılacak İşlem kutusundan gelen veriyi alır ve bu veriye göre seçilen algoritma fonksiyonunu çalıştırır.

```
function gonder() {  
    var sayi=(document.getElementById("kacsayi").value);  
    if(sayi>10000000){  
        alert("Eleman sayısı '10000000' sayısından büyük olamaz!");  
        Location(index.html);  
    }  
    islem = (document.getElementById("islem").value);  
    if (islem == "Linear Search")  
        LinearSearch();  
    else if (islem == "Insertion Sort")  
        InsertionSort();  
    else if (islem == "Binary Search")  
        BinarySearch();  
    else if (islem == "Merge Sort")  
        MergeSort();  
    else if (islem == "Heap Sort")  
        HeapSort();  
    else if (islem == "Quick Sort")  
        QuickSort();  
    else if (islem == "Counting Sort")  
        CountingSort();  
    else if (islem == "Bucket Sort")  
        BucketSort();  
    else if (islem == "Radix Sort")  
        RadixSort();  
    else  
        document.writeln("seçim yap!");  
}
```

Yanda  
*gonder()*  
fonksiyonu  
içinde yer alan  
kod bloğunu  
görüntü-  
leyebilirsiniz.

Aynı zamanda veri setinin alabileceği maximum boyut kontrolünü yapar.



Seçilen algoritmaya göre değişiklik gösterilecek işlemlerin yanı sıra her algoritma fonksiyonu içinde bulunan durumlardan bahsetmek gerekirse;

Algoritma fonksiyonlarında kullanılan dizilerin oluşturulduğu, ayrı bir *generatorMax()* fonksiyonu bulunuyor. Bu fonksiyon girilen min-max değerlerini de göz önüne alarak istenilen boyutta rastgele sayılarla bir dizi üretiyor.

```
function generatorMax(){
    elemanSayısı = parseInt(document.getElementById("kacsayi").value);
    max=parseInt(document.getElementById("maxsayi").value);
    var min=parseInt(document.getElementById("minsayi").value);
    if(min>max){
        alert("Elemanın min değeri max değerinden büyük olamaz. Lütfen düzeltiniz!");
        Location(index.html);
    }
    for (var i = 0; i < elemanSayısı; i++) {
        olustur = Math.floor(Math.random() * (max - min + 1)) + min;
        sayıListesi[i] = olustur;
    }

    return sayıListesi;
}
```

Yukarıda kodları gösterilen bu fonksiyon aynı zamanda giriş sayfasında seçilen min-max eleman değerlerini karşılaştırıyor. Ve büyüklük/küçüklük kontrolü yapmamızı sağlıyor. Minimum değer, maximum değerden büyük olması durumunda ise karşımıza bir uyarı kutucuğu çıkarıyor. Kutucukta çıkan “Tamam” butonuna basıldığında ise işlem iptal edilmiş oluyor.

Bu fonksiyondan gelen değer her algoritmada değişkene atanıyor ve Sıralama/Arama kodlarını çağırırken parametre olarak veriliyor.

## D. Kullanılan Algoritmalar

### 1. Linear Search

Linear Search algoritması; veri yapısı üzerinde tek tek gezer ve aracak değer ile karşılaştırma yapar. Kodlaması ve algılaması basit olan bu algoritma, çalışma zamanı olarak oldukça kötüdür. Çünkü en kötü durumda tüm veri setini gezmesi gerekmektedir.

### 2. Insertion Sort

Insertion Sort algoritması; veri setinin ikinci elemanından başlayarak kendisinden önceki elemanla kıyaslar, eğer kendisinden küçükse yer değiştirir ve bir sonraki elemana geçerek aynı işlemleri tekrarlar.

Oldukça basit olan bu algoritmanın karmaşıklığı için aynı şeyi söyleyemeyiz.  $N^2$  karmaşıklığa sahip Insertion Sort bu sebeple pek verimli sayılmaz.

### 3. Binary Search

Binary Search algoritması; sıralı veri seti üzerinde işlem yapar. Veri setini her seferinde ikiye bölerek aranacak sayı ile kıyaslar. Eğer aranacak sayı veri seti ikiye bölündüğünde elde edilen değerden küçük ise sol, büyük ise sağ taraf ile aynı işlemi aranan değere eşit olana kadar (bulana kadar) sürdürür. Zaman karmaşıklığı  $O(\log n)$ 'dir.

## 4. Merge Sort

Merge Sort algoritması; veri setini her seferinde ikiye bölerek ayrı ayrı sıralama yapar. Daha sonra bu sıralamaları birleştirme mantığına dayanan algoritma iki fonksiyondan oluşur. İlk fonksiyon veri setini ikiye bölüp ayrı ayrı sıralarken, diğer fonksiyon sıralanmış iki veri setini birleştirmeyi sağlıyor. Bu algoritmanın zaman karmaşıklığı ise  $O(n \log n)$ 'dir.

## 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. Bu algoritmanın zaman karmaşıklığı  $O(n \log n)$ 'dir.

## 6.Quick Sort

Quick Sort algoritması; her adımda veri setinden rastgele bir eleman seçer. Diziyi baştan sona gezerek seçilen elemandan küçük olanları elemanın soluna, büyük olanları ise sağına yerleştirir. Daha sonra böl ve fethet yönteminden faydalanılacaktır. Dizi iki parçaya bölünür ve özyinelemeli olarak tekrar işlemlerden geçer. Son olarak bu iki dizi birleştirilerek sıralanmış dizi elde edilir.

## 7.Counting Sort

Counting Sort algoritması; basitçe sıralanacak olan dizideki her sayının kaç tane olduğunu farklı bir dizide sayar. Daha sonra bu sayıların bulunduğu dizinin üzerinde bir işlemle sıralanmış olan diziyi elde eder. Zaman karmaşıklığı:  $O(n+k)$ dır. 'k' burada girdi aralığıdır.

## 8.Bucket Sort

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

*NOT: Bucket Sort algoritmasında hem ondalıklı sayılarla, hem de tam sayılarla işlem yapabilirsiniz. Gönder butonuna bastıktan sonra gelen kutucuktan seçim yapmanız yeterlidir.*

## 9.Radix Sort

Radix Sort algoritması; en değersiz olan haneden en değerli haneye doğru sıralama işlemi yapar. Algoritma işlenirken ilk olarak sıralanacak olan veri kümesindeki elemanların en büyük elemanının kaç basamaklı olduğu tespit edildikten sonra sayıların en değersiz olan basamağından itibaren incelenmeye başlanır ve yeni bir diziye yerleştirilir. Bu işlem dizinin en büyük elemanının basamak sayısı kadar tekrar edilir. Bu algoritmanın çalışma zamanı  $O(nk)$  ve yer karmaşıklığı  $O(n+k)$  olacaktır.

```
//Kullanılan Algoritmalar
function LinearSearch() { ...
}
function InsertionSort() { ...
}
function BinarySearch() { ...
}
function MergeSort() { ...
}
function HeapSort(){ ...
}
function QuickSort(){ ...
}
function CountingSort(){ ...
}
function BucketSort(){ ...
}
function RadixSort(){ ...
}
```

Görselden görüldüğü üzere projede kullanılan tüm algoritmalar bu şekildedir. Algoritmaların ve projenin detaylarını, kodlarını github hesabım üzerinden görüntüleyebilirsiniz.

**Github repo:**

[https://github.com/aslihanturkdonmez/ara\\_sira](https://github.com/aslihanturkdonmez/ara_sira)