**BURSA TEKNİK ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ ALGORİTMA ANALİZİ VE TASARIMI**

**WINDOWS FORMS İLE MASAÜSTÜ UYGULAMASI GELİŞTİRME**

**ALGORİTMA ANALİZİ VE TASARIMI PROJE**

**BERK SOĞUKPINAR**

**18360859054**

**OCAK 2021**

**Özgünlük Bildirisi**

1.Bu çalışmada ,başka kaynaklardan yapılan alıntıların ilgili kaynaklar referans gösterilerek açıkca belirtildiğini,

2. Alıntılar dışındaki bölümlerin,teorik çalışmaların ve yazılım/donanım kısmının benim tarafımdan yapıldığını bildiririm.

**Tarih:07.01.2021**

**Ad Soyad:Berk Soğukpınar**

**İÇİNDEKİLER**

**1-Uygulama Hakkında Bilgi……………………………………………………….**

**2-Uygulama’nın Kullanılışı………………………………………………………….**

2.1 Giriş Ekranı…………………………………………………………………………..

2.2 Ana Ekran…………………………………………………………………………..

2.3 Arama Algoritmaları……………………………………………………………

2.4 Linear Search ve Binary Search……………………………………………

2.5 Sıralama Algoritmaları Ekranı…………………………………………….

2.6 Tüm Sıralama Algoritmaları………………………………………………….

**3-Algoritmalar Hakkında Bilgiler………………………………………………**

3.1 Linear Search……………………………………………………………………...

3.2 Binary Search……………………………………………………………………..

3.3 Insertion Sort……………………………………………………………………..

3.4 Merge Sort…………………………………………………………………………

3.5 Heap Sort………………………………………………………………………….

3.6 Quick Sort…………………………………………………………………………

3.7 Counting Sort…………………………………………………………………..

3.8 Bucket Sort………………………………………………………………………

3.9 Radix Sort………………………………………………………………………..

**1-UYGULAMA HAKKINDA BİLGİ**

Verilen proje ödevinde istenilen Windows Forms masaüstü uygulaması C# dilinde Microsoft Visual Studio programı aracılığıyla geliştirilmiştir.Uygulamanın içerisinde Arama ve Sıralama Algoritmaları yer almaktadır. Arama Algoritmaları;

* **Linear Search**
* **Binary Search** olmak üzere ikiye ayrılır.

Sıralama Algoritmaları;

* **Insertion Sort**
* **Merge Sort**
* **Heap Sort**
* **Quick Sort**
* **Counting Sort**
* **Bucket Sort**
* **Radix Sort** olmak üzere 7’ye ayrılırlar.

Geliştirilen uygulama ile kullanıcının istediği kadar sayı üretip o üretilen sayıların uygulamamızda olan 2 tane Arama algoritması ile istenilen sayının oluşturulan dizide aratılması, 7 tane Sıralama algoritması ile üretilen sayıların sıralanması sağlanmıştır.

Uygulamada Formlar arası geçiş olduğundan istediğiniz herhangi bir Arama veya Sıralama algoritmasına istediğiniz zaman erişebilirsiniz.

Eğer Kullanıcı isterse tüm sayfalarda Çıkış butonuna tıklayarak uygulamayı kapatabilir.

**2-UYGULAMA’NIN KULLANILIŞI**

**2.1 GİRİŞ EKRANI**

Uygulama çalıştırıldığında ilk sayfa olarak karşımıza kullanıcının Kullanıcı Adı ve Şifresini girerek uygulamaya giriş yapması beklenir.Kullanıcı eğer Kullanıcı Adı veya Şifreden bir tanesini hatalı girerse karşısına bir uyarı ekranı çıkacaktır.Kullanıcının 3 kere hatalı giriş yapması sonucunda karşımıza bir uyarı ekranı çıkar ve uygulama kendini kapatır. Kullanıcı Adı ve Şifre doğru girilirse karşımıza Sıralama ve Arama algoritmalarından hangisini seçmemizi istediğimiz ekran çıkar.

**DİPNOT:**

Kullanıcı adı ve Şifre aşağıda ve paylaşılan dosyanın ReadMe.txt bölümünde verilmiştir. Lütfen bu kısmı okumadan uygulamayı çalıştırmayınız.

**Kullanıcı Adı:Berk44**

**Şifre:1234**

**2.2 ANA EKRAN**

Bu ekrana geldiğimizde Kullanıcı artık uygulamaya giriş yapmış demektir. Bu ekranda Kullanıcı ne tür bir algoritma istediğini seçecektir.Arama algoritmaları butonuna tıklarsa karşısına Linear Search ve Binary Search’i seçebileceği bir ekran açılır. Sıralama algoritmaları butonuna tıklarsa Insertion Sort, Merge Sort, Heap Sort, Quick Sort, Counting Sort, Bucket Sort ve Radix Sort’u seçebileceği bir ekran açılır.

**2.3 ARAMA ALGORİTMALARI**

Kullanıcı bu sayfaya geldiğinde Linear Search veya Binary Search butonuna tıklayarak istediği algoritmayı seçebilir. Eğer Kullanıcı isterse bu sayfadayken de Çıkış butonuna tıklayarak uygulamayı kapatabilir.Eğer Kullanıcı isterse Seçim Ekranına Dönüş butonuna tıklayarak ANA EKRAN sayfasına dönebilir.

**2.4 LINEAR SEARCH ve BINARY SEARCH**

Kullanıcı bu sayfalardan herhangi birini tercih ettiğinde Linear Search ve Binary Search algoritmaları kullanılarak kaç tane sayı üretmek istediğini ve üretilen sayılarda aradığı elemanı textboxların içine girer.Ardından Ara butonuna tıklayarak uygulamayı çalıştırır. Dizideki elemanlar yazdırılır. Eğer üretilen bu sayılarda kullanıcının istediği eleman bulunursa ekrana “Aradığınız sayı dizinin içinde bulunuyor”,bulunmuyorsa” Ardığınız sayı dizinin içinde bulunamadı.” yazısını Sonuç ekranında görürüz.Aranan eleman dizide bulunursa kaçıncı indeksde bulunduğu ve algoritmanın gerçekleşmesinin kaç adımda olduğu yazıyor.Algoritmanın tamamının kaç saniyede gerçekleştiği de textboxların içinde yazdırılıyor.Kullanıcı isterse Arama Algoritmalarına Dönüş butonuna tıklayarak Arama Algoritmalara sayfasına geri dönebilir.

**2.5 SIRALAMA ALGORİTMALARI EKRANI**

Kullanıcı bu sayfaya geldiğinde herhangi bir Sıralama algoritmasını seçebilir.Seçim Ekranına Dönüş butonuna tıklayarak ANA EKRAN sayfasına dönebilir.

**2.6 TÜM SIRALAMA ALGORİTMALARI**

Önceki sayfada yaptığımız herhangi bir Sıralama algoritması seçimiyle karşımıza bir ekran açılır. Insertion Sort algoritmasını seçtiğimizi varsayarsak kaç tane sayı üretmek istediğimizi girmemiz gerekir.Ardından BAŞLAT butonuna tıklayarak uygulamayı çalıştırırız.Butona tıklandıktan kullanıcının istediği kadar sayı üretilir.Sayıların sıralanmamış hali ve sıralanmış hali gösterilir.Sıralama işleminin kaç saniyede ve kaç adımda gerçekleştiği gösterilir.Kullanıcı isterse Sıralama Algoritmalarına Dönüş butonuna tıklayarak önceki sayfaya dönüp başka bir algoritma seçebilir.

**3-Algoritmalar Hakkında Bilgiler**

**3.1 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. 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.

**3.2 Binary Search:** Binary Search, sıralı(sorted) bir veri yapısı için kullanılır. Yani algoritmaya aranan veri ve sıralı bir veri yapısı verirsiniz.  Algoritma da 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.

**3.3 Insertion Sort:** Insertion Sort, bilgisayar bilimlerinde kullanılan ve sıralı diziyi her adımda öge öge oluşturan bir sıralama algoritmasıdır. Insertion Sort Algoritması, düzensiz dizi elemanlarını tek tek ele alarak her birini dizinin sıralanmış kısmındaki uygun yerine yerleştirme esasına dayanır. 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.

**3.4 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 altdiziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide bireşir. Süreç sonunda en üstte sıralı diziye ulaşılır.

**3.5 Heap Sort:** Yığınlama sıralaması, arka planda bir yığın ağacı(heap)  oluşturur ve bu ağacın en üstündeki sayıyı alarak sıralama işlemi yapar. Algoritma adımları şu şekilde açıklanabilir:

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

**3.6 Quick Sort:** Basitçe sıralanacak olan dizideki orta noktada (mean) bulunan bir sayıyı seçerek diğer bütün sayıları bu orta sayıdan büyük veya küçük diye sınıflayarak sıralama yapmayı hedeflemektedir. Bu açıdan bir [parçala fethet (divide and conquere)](http://bilgisayarkavramlari.com/2008/08/09/2007/12/03/parcala-fethet-yontemi-divide-and-conquer/) yaklaşımıdır. Ayrıca bu seçilen orta noktaya eksen (pivot) adı da verilir çünkü bütün diğer sayılar bu sayının ekseninde sıralanacaktır.

**3.7 Counting Sort:** 1954 yılında Harold H. Seward tarafından yapılmıştır.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.

**3.8 Bucket Sort: Kova Sıralaması** (ya da **sepet sıralaması**), sıralanacak bir diziyi parçalara ayırarak sınırlı sayıdaki *kovalara* (ya da *sepetlere*) atan bir [sıralama algoritmasıdır](https://tr.wikipedia.org/wiki/S%C4%B1ralama_algoritmas%C4%B1). Ayrışma işleminin ardından her kova kendi içinde ya farklı bir algoritma kullanılarak ya da kova sıralamasını [özyinelemeli](https://tr.wikipedia.org/wiki/%C3%96zyineleme) olarak çağırarak sıralanır. Bucket Sort algoritması şu şekilde çalışır:

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

**3.9 Radix Sort:** Türkçe'de taban sıralaması , basamaklı sıralama , kök sıralaması veya hane sıralaması olarak geçen bu algoritmada sıralanacak olan veriler hanelerine göre sıralanır.En değersiz olan haneden en değerli haneye doğru sıralama işlemi yapılır.

Sıralanacak verilerin tamsayı olduğu durumlarda kullanılan bu 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.Gayet iyi bir çalışma zamanına sahip olmasının nedeni bu algoritmanın karşılaştırmalı bir sıralama algoritması olmamasıdır.En büyük dezavantajı ise her basamak işlemi için yeni bir bellek alanı gerektirmesidir.