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

**Ders süresince işlenmiş tüm algoritmaların tek bir uygulamada toplanması**

**ALGORİTMA ANALİZİ VE TASARIMI PROJE**

**İhsan Oğuz Akpınar**

**18360859004**

**UYGULAMA**

Projemi Windows Form Application olarak geliştirdim. Açılışta karşımıza Hakkında sekmesi çıkıyor ve burada bilgilerim yer alıyor. Sekmelere baktığımızda isimleri “Arama algoritmaları” ve “Sıralama algoritmaları” olan sekme daha görüyoruz. Sekme ismine tıklayarak açıyoruz ve istenilen boşlukları doldurarak arama ve sıralama işlemlerimizi gerçekleştiriyoruz.

**Sıralama Algoritmalarımız 2 adet ve aşağıdaki gibidir**

* Linear Search
* Binary Search

**Sıralama Algoritmaları ise 7 adettir ve aşağıda sıralanmışlardır**

* Insertion Sort
* Merge Sort
* Heap Sort
* Quick Sort
* Counting Sort
* Bucket Sort
* Radix Sort

Uygulamamda girilen sayı kadar rastgele sayı üretilerek arama ve sıralama işlemi yapılabilmektedir. Girdi alanlarının yanında gerekli açıklamalar zaten yazılmış durumda. Gerekli boşluklar doldurulduktan sonra tek yapmanız gereken kullanmak istediğiniz algoritmayı butona basarak seçmeniz. Daha sonra istenilen tüm değerler hesaplanarak ilgili boşluklara yazılacak.

**Algoritmalar Hakkında Bilgiler**

**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. Yani en kötü durumda tüm veri setini gezmesi gerekir.

**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.

**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.

**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.

**Heap Sort:** Verinin hafızada sıralı tutulması için geliştirilen sıralama algoritmalarından (sorting algorithms) bir tanesidir. Yığınlama sıralaması, arka planda bir yığın ağacı(heap)  oluşturur ve bu ağacın en üstündeki sayıyı alarak sıralama işlemi yapar. Bu verilerin bir oluşumun  belirleyici alanları olduğunu düşünebiliriz. Yani örneğin vatandaşlık numarası veya öğrenci numarası gibi. Dolayısıyla örneğin öğrencilerin numaralarına göre sıralanması durumunda kullanılabilir.

Algoritmanın Çalışması

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.

**Quick Sort:**

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.

**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 ögelerin sayısını gösterir. Yeni dizideki öge değeri sayıları daha sonra ana dizideki tüm değerlerin doğru konuma konulması için kullanılır.

**Bucket Sort:**

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.

**Radix Sort:**  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.