Sıralama Algoritmaları Görselleştircisi

Deniz Beltan
Kocaeli Üniversitesi Teknoloji Fakültesi
Bilişim Sistemleri Mühendisliği
İstanbul, Türkiye
201307016@kocaeli.edu.tr

Oğuzhan Öztürk
Kocaeli Üniversitesi Teknoloji Fakültesi
Bilişim Sistemleri Mühendisliği
Kocaeli, Türkiye
201307007@kocaeli.edu.tr

Kaan Yıldırım Eser
Kocaeli Üniversitesi Teknoloji Fakültesi
Bilişim Sistemleri Mühendisliği
Kocaeli, Türkiye
201307083@kocaeli.edu.tr

***Özet*— Sıralama algoritmaları görselleştiricisi, kullanıcının elemanları kendisinin girdiği veya sistem tarafından rastgele oluşturulan dizileri, istediği sıralama algoritması ile istediği tabloyu seçerek ve görselleştirme hızını ayarlayarak görselleştirebileceği, görselleştirme sonunda karşılaştırma sayısı ve karmaşıklık analizini görebildiği bir programdır.**

Anahtar Kelimeler—Seçerek Sıralama, Birleştirme Sıralaması, Kabarcık Sıralaması, Hızlı Sıralaması, Ekleme Sıralaması, Görselleştirme

I. GİRİŞ

Günümüzde sıralama algoritmaları, bilgisayar biliminde önemli bir rol oynamaktadır. Veri setlerinin sıralanması, çeşitli uygulamaların temel bir gereksinimidir ve performansı doğrudan etkiler. Sıralama algoritmaları, bilgisayar programcıları ve öğrencileri tarafından öğrenilmesi ve anlaşılması gereken kritik bir konudur.

Ancak, sıralama algoritmalarını anlamak bazen karmaşık ve soyut olabilir. Kavramların anlaşılması, algoritmaların çalışma mantığının görsel olarak takip edilmesi ve deneyimlenmesi gerekebilir. İşte tam burada "Sıralama Algoritmları Görselleştiricisi" projesi devreye giriyor.

"Sıralama Algoritmları Görselleştiricisi" projesi, eğitici bir arayüz sunacak ve kullanıcıların farklı sıralama algoritmalarını keşfetmelerine, adım adım ilerlemelerini gözlemlemelerine ve sonuçları karşılaştırmalarına olanak sağlayacaktır. Proje, çeşitli sıralama algoritmalarını destekleyerek, kullanıcılara geniş bir perspektif sunacak ve algoritmalar arasındaki farkları ve performans farklarını görsel olarak analiz etme imkanı sağlayacaktır.

Projemiz arka yüz kısmında Python, ön yüz kısmında PyQt kullanarak oluşturulmuştur. Projemiz masaüstü tabanlıdır..

II. AMAÇ VE HEDEFLER

Proje amacı, sıralama algoritmalarını daha anlaşılır ve görsel bir şekilde sunarak, bu algoritmaları öğrenmek isteyenlerin ve bilgi sahibi olanların daha kolay anlamalarını sağlamaktır. Projemiz, sıralama algoritmalarının çalışma mantığını göstermek ve kullanıcıların bu algoritmaları etkileşimli bir şekilde deneyimlemelerini sağlamak üzerine odaklanmaktadır.

Hedefler:

1- Eğitici ve görsel bir arayüz sağlamak: Proje, sıralama algoritmalarını öğrenmek isteyen herkesin kolayca erişebileceği bir eğitim aracı olarak tasarlanmıştır. Kullanıcılar, sıralama algoritmalarının çalışma mantığını görsel olarak anlamak ve algoritmaları interaktif bir şekilde deneyimlemek için kullanıcı dostu bir arayüz üzerinden etkileşime geçebilirler.

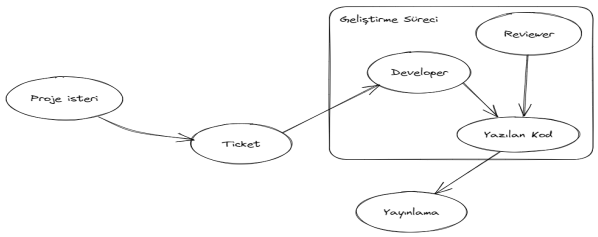
2- Farklı sıralama algoritmalarını desteklemek: Proje, çeşitli sıralama algoritmalarını içerecektir. Kullanıcılar, farklı algoritmaların nasıl çalıştığını görsel olarak takip edebilecek, adım adım ilerlemelerini gözlemleyebilecek ve sonuçları karşılaştırabileceklerdir. Örnek olarak, kabarcık sıralama, seçim sıralaması, merge sort gibi algoritmaları içerebilir.

3- Performans analizi sunmak: Projemiz, sıralama algoritmalarının performansını karşılaştırmak için kullanıcılara olanak sağlayacaktır. Kullanıcılar, farklı algoritmaların zaman karmaşıklığını, karşılaştırma sayısını ve yer değiştirme sayısını görsel olarak analiz edebileceklerdir. Bu sayede, algoritmaların farklı senaryolarda nasıl performans gösterdiğini daha iyi anlayabilecekler.

4- Öğrenme deneyimini geliştirmek: Proje, kullanıcıların sıralama algoritmalarını öğrenirken aktif bir şekilde katılımını teşvik etmeyi hedeflemektedir. Kullanıcılar, algoritmaları interaktif bir şekilde değiştirebilecek, verilerin nasıl etkilendiğini gözlemleyebilecek ve sonuçları keşfedebilecektir. Bu şekilde, kullanıcıların öğrenme deneyimi daha etkili ve keyifli olacaktır.

III. YÖNTEM

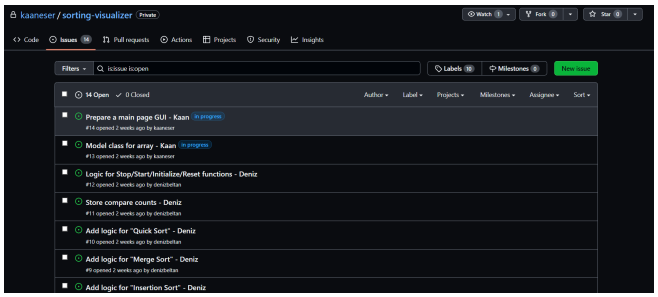
Projenin geliştirme sürecine başlamadan önce bir iş akış diyagramı oluşturulmuş ve bu iş akış diyagramına uygun şekilde geliştirme ortamları sağlanmıştır. Bu diyagramdan hareketle proje altta belirtilen başlıklar üzerinde değerlendirilebilmektedir:



Görsel 1. İş Akış Diyagramı

A. Ticket ve Proje İsterleri

Projede gerçekleştirilmesi beklenen isterler parçalara bölünmüş ve her biri belirli numaralarda ticket olarak oluşturulmuştur. Oluşturulan ticketlar yazılımcılara atanmıştır ve belirtilen kısmı, yine belirtilen sürelerde tamamlamaları hedeflenmiştir. Bu süreç GitHub'ın içerisinde yer alan Issue bölümünden takip edilmiştir. Proje takımları, yaptıkları ya da yapacakları görevleri listeye ekleyip takım arkadaşlarına ne durumda olduklarını gösterebilmektedirler. GitHub üzerinde yapılan iş dağılımları neticesinde Deniz Beltan *Scrum Master*, Oğuzhan Öztürk *Project Quality Manager* ve Kaan Yıldırım Eser *Project Manager* unvanlarını edinmiştir.



Görsel 2. GitHıb Issue Platformu

IV. Arka Yüz Tasarımı

A. Kullanılan Algortimalar

1- Seçme Sıralaması (Selection Sort): Seçme Sıralaması, bir veri dizisindeki elemanları sıralamak için kullanılan basit bir sıralama algoritmasıdır. Adından da anlaşılacağı gibi, her adımda en küçük (veya en büyük) elemanı bulup onu doğru konumuna yerleştirir.

Algoritmanın çalışma prensibi şu şekildedir: İlk olarak, veri dizisindeki en küçük elemanı bulmak için tüm elemanlar taranır. Bulunan en küçük eleman, dizinin başındaki elemanla yer değiştirilir. Ardından, ikinci en küçük elemanı bulmak için kalan elemanlar taranır ve bu elemanlar arasından en küçük olanı, dizinin ikinci sırasına yerleştirilir. Bu işlem, dizideki tüm elemanlar sıralanana kadar devam eder.

2- Kabarcık Sıralama (Bubble Sort): Kabarcık Sıralama, bir veri dizisindeki elemanları sıralamak için kullanılan basit bir sıralama algoritmasıdır. Algoritma adını, elemanların sıralanırken yukarı doğru kabarcık gibi yükselmesinden alır.

Algoritmanın çalışma prensibi şu şekildedir: İki eleman karşılaştırılır ve eğer sıralama kriterine uygun değilse (örneğin, küçükten büyüğe sıralamada bir önceki eleman bir sonrakinden büyükse), bu elemanlar yer değiştirilir. Bu işlem dizinin başından sonuna kadar tekrarlanır, böylece en büyük eleman sona doğru "kabarcık" gibi ilerler.

Daha sonra, son elemanın doğru konumda olduğunu kabul ederek, işlem yeniden baştan başlar ve bir sonraki büyük elemanın doğru konuma yerleşmesi sağlanır. Bu işlem, dizinin tüm elemanlarının sıralanana kadar devam eder.

3- Ekleme Sıralaması (Insertion Sort): Ekleme Sıralaması, bir veri dizisindeki elemanları sıralamak için kullanılan basit bir sıralama algoritmasıdır. Adından da anlaşılacağı gibi, her adımda sıralanmış bölgeye yeni bir eleman eklenir ve doğru konumuna yerleştirilir.

Algoritmanın çalışma prensibi şu şekildedir: İlk olarak, dizinin ilk elemanı sıralanmış kabul edilir. Ardından, ikinci elemanı sıralanmış bölgeye eklenir ve bu elemanın doğru konumuna yerleştirilir. Daha sonra, üçüncü eleman sıralanmış bölgeye eklenir ve doğru konuma yerleştirilir. Bu işlem, dizideki tüm elemanlar sıralanana kadar devam eder.

Elemanların doğru konumlarına yerleştirilmesi, sıralanmış bölgede elemanların sağa doğru kaydırılmasıyla gerçekleştirilir. Karşılaştırma yapılarak doğru konum bulunduğunda, eleman yer değiştirilir ve sıralanmış bölge genişler.

4- Ekleme Sıralaması (Insertion Sort): Birleştirme Sıralaması, bir veri dizisindeki elemanları sıralamak için kullanılan etkili bir sıralama algoritmasıdır. Algoritma, "böl ve fethet" prensibine dayanır ve rekürsif bir yaklaşım kullanır.

Algoritmanın çalışma prensibi şu şekildedir: İlk olarak, veri dizisi ortadan ikiye bölünür ve her iki yarı dizi de ayrı ayrı sıralanır. Ardından, iki sıralı yarı diziyi birleştirirken elemanları karşılaştırarak sıralı bir dizi oluşturulur.

Birleştirme aşamasında, iki ayrı sıralı yarı dizideki elemanlar karşılaştırılır ve doğru sıralama sırasına göre birleştirilir. Bu işlem, tüm elemanlar birleştirilene kadar tekrarlanır. Sonuç olarak, sıralanmış bir dizi elde edilir.

5- Hızlı Sıralama (Quick Sort): Hızlı Sıralama, bir veri dizisindeki elemanları sıralamak için kullanılan etkili bir sıralama algoritmasıdır. Algoritma, "böl ve fethet" prensibine dayanır ve rekürsif bir yaklaşım kullanır.

Algoritmanın çalışma prensibi şu şekildedir: İlk olarak, bir "pivot" eleman seçilir. Pivot eleman, dizinin herhangi bir elemanı olabilir, genellikle dizinin orta elemanı tercih edilir. Dizi, pivot elemanın solunda ve sağında iki alt diziye bölünür.

Bölme aşamasında, pivot elemandan küçük olan elemanlar sol tarafta, pivot elemandan büyük olan elemanlar ise sağ tarafta olacak şekilde yerleştirilir. Bu işlem, dizi içindeki elemanların pivot elemanı etrafında doğru konumlandırılmasıyla gerçekleşir.

Daha sonra, pivot elemanın sol ve sağ taraflarındaki alt diziler ayrı ayrı sıralanır. Bu sıralama işlemi de aynı rekürsif mantıkla yapılır. Her bir alt dizi tekrar pivot eleman seçilerek bölünür ve bu işlem alt diziler sıralanana kadar devam eder.

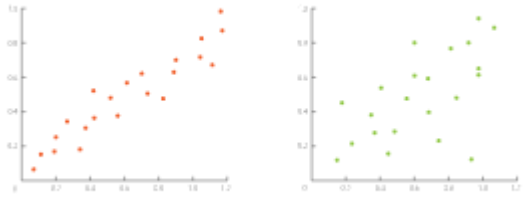
B. Grafik Tipleri

1- Dağılım (Scatter) Grafiği: Dağılım (Scatter) grafiği, veri noktalarının x ve y eksenlerinde konumlarını görselleştirmek için kullanılan bir grafik türüdür. Her veri noktası, iki sayısal değeri temsil eder ve grafikte nokta olarak işaretlenir.

Dağılım grafiği, veri noktalarının birbiriyle ilişkisini veya dağılımını göstermek için kullanılır. Her veri noktası, x ve y eksenlerinde belirli bir konumda yer alır. Bu sayede, veri setindeki eğilimleri, kümeleri, aykırı değerleri veya veriler arasındaki ilişkileri kolayca gözlemleyebiliriz.

Dağılım grafiği genellikle noktaların renk, boyut veya şekil gibi özelliklerini kullanarak ek bir boyut ekleyebilir. Böylece, veri noktaları arasında daha fazla farklılık veya kategorizasyon yapabiliriz.

Dağılım grafiği, istatistiksel analiz, veri keşfi, tahmin modelleri oluşturma gibi alanlarda sıkça kullanılır. Örneğin, iki değişken arasındaki ilişkiyi anlamak, aykırı değerleri tespit etmek veya veri setindeki grupları belirlemek için kullanılabilir.

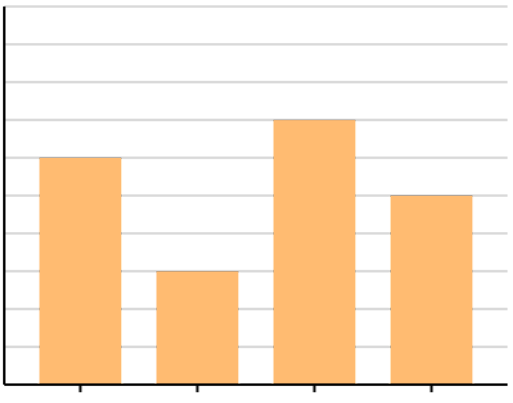


Görsel 3. Dağılım (Scatter) Grafiği

2-Sütun(Bar) Grafiği: Sütun (bar) grafiği, kategorik verilerin görselleştirilmesi için kullanılan bir grafik türüdür. Bu grafik, farklı kategorilerin veya grupların sayısal değerlerini göstermek amacıyla sütunlar veya çubuklar kullanır.

Sütun grafiği, x ekseni boyunca kategorilerin yer aldığı bir eksen ve y ekseni boyunca sayısal değerlerin yer aldığı bir eksen içerir. Her kategoriye ait sütun veya çubuk, kategoriye karşılık gelen değeri temsil eder ve yüksekliği veya uzunluğu ile gösterilir.
Sütun grafiği, veri setindeki farklı kategorilerin sayısal değerlerini karşılaştırmak, kategoriler arasındaki ilişkileri anlamak veya eğilimleri gözlemlemek için kullanılır.
Sütunlar veya çubuklar, görsel olarak farklı boyutlara veya renklere sahip olabilir, bu da veriler arasındaki farklılıkları daha iyi vurgulayabilir.

Sütun grafiği genellikle tek bir veri setini görselleştirmek için kullanılır, ancak birden fazla veri setini karşılaştırmak için gruplu sütun grafiği veya yığılmış sütun grafiği gibi varyasyonları da bulunmaktadır.



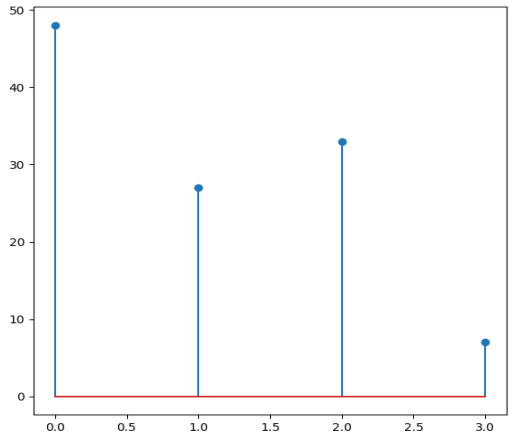
Görsel 3. Sütun (Bar) Grafiği

3-Kök (Stem) Grafiği: Kök (Stem) grafiği, sayısal verilerin dağılımını ve eğilimini görselleştirmek için kullanılan bir grafik türüdür. Her veri noktası, bir çizgi ile temsil edilir ve bu çizgiler, bir eksen boyunca yukarı doğru uzanır.

Kök grafiği, veri setindeki her bir sayının kökünü veya temsil edilen sayıyı alarak çizgiyi oluşturur. Bu çizgiler, bir eksen boyunca düzenli aralıklarla yerleştirilir ve genellikle dikey olarak yukarı doğru uzanırlar.

Kök grafiği, veri setindeki değerlerin dağılımını ve eğilimini göstermek için kullanılır. Her veri noktasının yüksekliği, temsil ettiği sayının büyüklüğünü gösterir. Ayrıca, çizgilerin yanında veri noktalarının değerleri de belirtilebilir.

Bu grafik türü, özellikle çok sayıda veri noktası olduğunda ve verilerin dağılımını görselleştirmek istediğimizde faydalı olabilir. Kök grafiği, istatistiksel analiz, veri keşfi, trendleri gözlemleme gibi alanlarda kullanılır.



Görsel 4. Kök (Stem) Grafiği

C.Kullanılan Kütüphaneler

Matplotlib: Python programlama dilinde güçlü bir veri görselleştirme kütüphanesidir. Bilimsel ve sayısal hesaplamalar yaparken verileri görselleştirmek için yaygın olarak kullanılır. Matplotlib, grafikler, histogramlar, çizgi grafikleri, dağılım grafikleri ve daha birçok görsel sunum türünü oluşturmak için kullanıcı dostu bir arayüz sağlar.

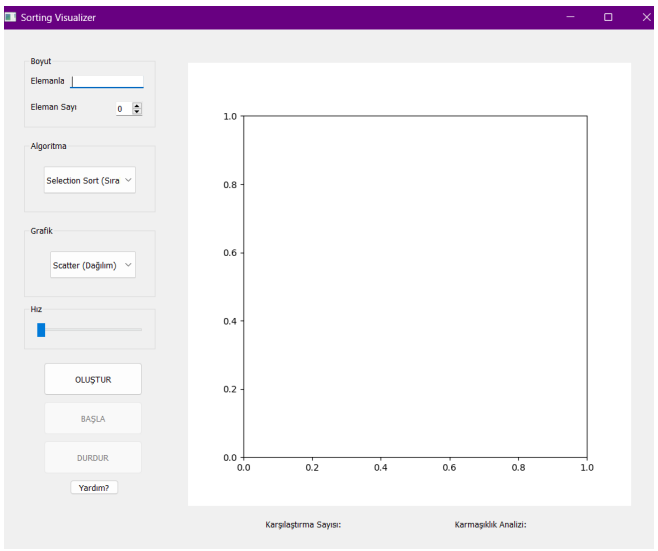
NumPy (Numerical Python): Python programlama dilinde kullanılan bir temel bilimsel hesaplama kütüphanesidir. NumPy, çok boyutlu dizilerin ve matrislerin işlenmesi ve veriler üzerinde yüksek performanslı matematiksel işlemler yapılması için tasarlanmıştır. NumPy, Python'a eklenen bu güçlü araç setiyle bilimsel ve sayısal hesaplamalar için ideal bir seçenek haline gelmiştir.

D.Oluşturulan Fonksiyonlar

Proje içinde insertion sort, bubble sort, merge sort, quick sort,selection sort, update, start, stop, visualize fonksiyonları bulunmaktadır.

V. ÖN YÜZ TASARIMI

A. Giriş Ekranı

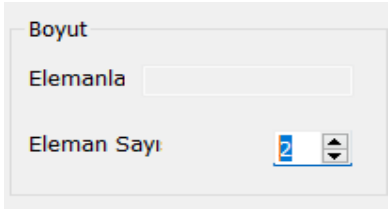


Görsel 5. Giriş Ekranı

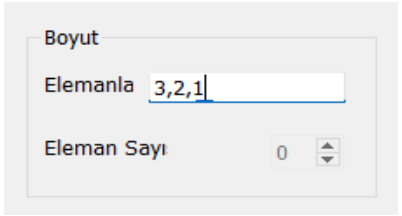
Program başlatıldığında çıkan ekran görsel 5'deki gibidir. Sol navigasyon menüsünde boyut, algoritma, grafik, hız panelleri ve oluştur ile yardım butonu bulunmaktadır.

Ölçeklenmiş beyaz alanda oluşturmak istediğimiz grafiği görülebilir, agrafiğin altında kalan alanda karşılaştırma sayısına ve karmaşıklık analizine ulaşabilir.

B. Boyut Seçme

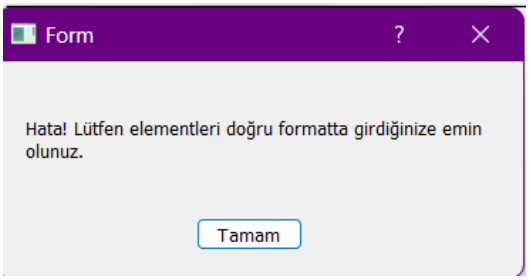


Görsel 6. Eleman Sayısını Girme



Görsel 7. Elemanları Manuel Girme

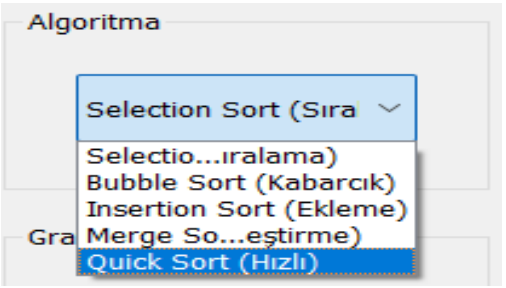
Kullanıcı elemanları manuel olarak elden girmek isterse eleman sayısının gireceği alan devre dışı kalır. Eğer eleman sayısını girerse eleman sayısını seçeceği alan devre dışı kalır.



Görsel 8. Boyut Seçme

Kullanıcı elden girdiği diziyi virgüllerle ayırmaz ise uygun formatta veri girişi yapmaları için gerekli uyarı çıkar.

C. Algoritma Seçme

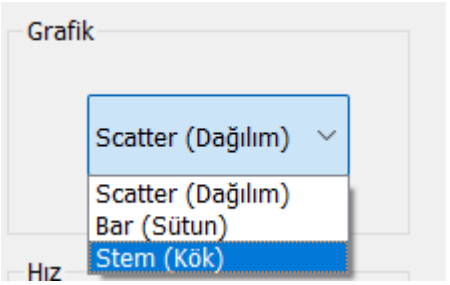


Görsel 9. Algoritma Seçme

Kullanıcı algoritma seçme kısmından istediği algoritmayı seçebilir. Listede var olan sıralama algoritmaları:

- Seçme Sıralaması (Selection Sort)
- Kabarcık Sıralaması (Bubble Sort)
- Ekleme Sıralaması (Insertion Sort)
- Birleştirme Sıralaması (Merge Sort)
- Hızlı Sıralama (Quick Sort)

D. Grafik Seçme



Görsel 10. Algoritma Seçme

Kullanıcı grafik seçme kısmından istediği grafiği seçebilir. Listede var olan grafikler:

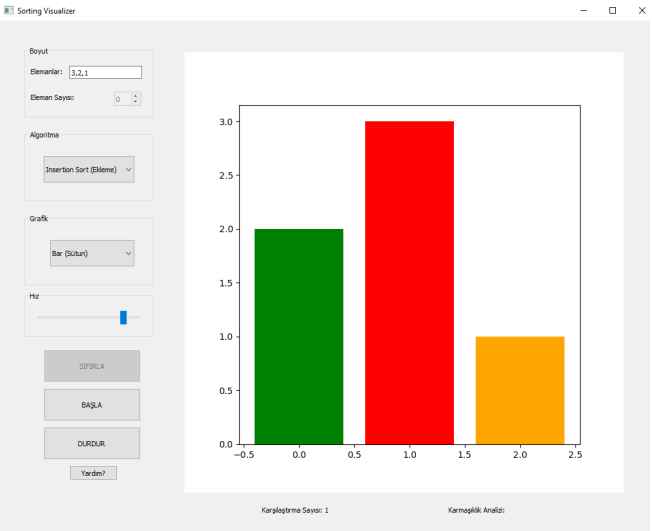
- Dağılım (Scatter) Grafiği
- Sütun (Bar) Grafiği
- Kök (Stem) Grafiği

E. Hız Ayarlama

Görsel 11. Algoritma Seçme

Kullanıcı hız gösterge kısmından grafiğin ne kadar hızlı şekilde görselleştirilmesini isterse ona göre ayarlayabilir. Süre maksimum 10 minimum 1 saniye olarak ayarlanabilmektedir.

Sıralama anında yapılan görselleştirmenin örnek görseline aşağıdadır.



Görsel 12. Algoritma Seçme

VI. Sonuç

Bu projenin sonucunda, kullanıcılar sıralama algoritmalarının çalışma prensiplerini daha iyi anlamış, algoritmaları görsel olarak takip ederek işleyişlerini kavramışlardır. Ayrıca, farklı algoritmalar arasındaki performans farklarını görsel olarak analiz ederek veri setlerini sıralarken en uygun algoritmayı seçme becerilerini geliştirmişlerdir. Proje, sıralama algoritmalarını öğrenmek isteyen herkesin kullanabileceği kullanıcı dostu bir araç olarak başarılı bir şekilde tamamlanmıştır.

Projenin gelecek sürümleri için eklenebilecek birkaç özellik listelenmiştir:

- Daha Fazla Sıralama Algoritması: Proje, şu anda popüler sıralama algoritmalarını desteklemektedir. Ancak, daha fazla sıralama algoritması eklenerek kullanıcılara daha geniş bir seçenek sunulabilir. Örneğin, radix sort gibi diğer sıralama algoritmalarının da görselleştirilmesi sağlanabilir..
- Performans Analizi: Projeye, sıralama algoritmalarının performansını analiz etme özelliği eklenerek kullanıcılara daha kapsamlı bir değerlendirme imkanı sunulabilir. Kullanıcılar, farklı veri setleri üzerinde algoritmaları çalıştırarak zaman ve bellek kullanımı gibi performans ölçütlerini karşılaştırabilirler.
- Animasyon ve İnteraktivite: Projeye animasyon ve interaktivite özellikleri eklenerek kullanıcı deneyimi daha da geliştirilebilir. Sıralama algoritmalarının adımlarının animasyonlu bir şekilde gösterilmesi veya kullanıcının adımları kendisi kontrol etme imkanı gibi özellikler, kullanıcıların algoritmaları daha etkileşimli bir şekilde deneyimlemesini sağlayabilir.

Kaynakça

[1] https://excalidraw.com

[2] https://edestek3.kocaeli.edu.tr/pluginfile.php/35913/mod_resource/content/1/git-cheat-sheet-education.pdf

[3] https://github.com/kaaneser/sorting-visualizer

[4] https://www.serkanseker.com/tr/algoritama-karmasiklik-analizi/

[5] https://bilgisayarkavramlari.com/2008/08/09/hizli-siralama-algoritmasi-quick-sort-algorithm/

[6] https://bilgisayarkavramlari.com/2008/12/12/sokma-siralamasi-ekleme-siralamasi-insertion-sorting/

[7] https://bilgisayarkavramlari.com/2008/08/09/secerek-siralama-selection-sort/

[8] https://bilgisayarkavramlari.com/2008/08/09/kabarcik-siralamasi-baloncuk-siralamasi-bubble-sort/

[9] https://bilgisayarkavramlari.com/2008/08/09/birlestirme-siralamasi-merge-sort/