Sıralama Algoritmaları Görselleştiricisi Proje Raporu

Zeynep Serva Yıldız
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
201307052
Kocaeli, Türkiye
201307052@kocaeli.edu.tr

Fatma Nur Azbaş
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
201307038
İstanbul, Türkiye
201307038@kocaeli.edu.tr

Zehra Betül Taşkın
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
201307054
Ankara, Türkiye
201307054@kocaeli.edu.tr

***Özet*—Bu proje, kullanıcının sıralama algoritmalarını ve grafik türlerini kullanarak bir listenin sıralanmasını ve sıralama işleminin adımlarını görsel bir şekilde görmesini sağlamaktadır.**

Anahtar Kelimeler—sıralama algoritmaları, grafik türleri

I. GİRİŞ

Projenin amacı; kullanıcının girdiği verilerin, seçtiği grafik türü ve sıralamada kullanmak için seçtiği sıralama algoritmasına göre grafik olarak çizdirilmesidir. Çizilen grafikteki veriler, seçilen algoritmanın sıralamasına göre animasyonlu olarak sıralanmaktadır. Seçilecek sıralama algoritmalarının her biri farklı şekilde karşılaştırmalar yaparak sonuca ulaşmaktadır, dolayısıyla karmaşıklık analizi ve karşılaştırma sayısı sonuçları da veri sayısı ve grafik türüne göre değişkenlik göstermektedir. Kullanıcının girdiği veriler, seçilen grafik türü ve sıralama algoritması seçildikten sonra sıralama hızı ayarlanabilmekte olup grafik çizimi ve sıralama bittiğinde, grafiğin sıralanmasına ait karmaşıklık analizi ve karşılaştırma sayısı ekranda gösterilmektedir.

II. KULLANILAN TEKNOLOJİLER

Visual Studio Code

Hata ayıklama, Pyhon'da gerçek zamanlı çalışma gibi birçok desteği ve kullanışlı özelliği barındıran Microsoft tabanlı ücretsiz kaynak kod düzenleyicisi ve uygulama geliştirme idesi olan Visual Studio Code, bu proje için geliştirme ortamı olarak tercih edilmiştir.

Python

Çeşitli kütüphaneleri ile başta grafik ve tasarımsal özellikleri destekleyerek kolay yazım işlevi ve verimlilik sağlayan Python, projede temel olarak kullanılan programlama dilidir. Bu projede Python'ın 3.11.3 versiyonu kullanılmıştır.

Tkinter

Python'ın kurulması ile birlikte otomatik olarak kurulan, masaüstü programı için pencere oluşturmayı sağlayan, Python'ın kullanıcı arayüzü geliştirme kütüphanesidir. Projede ana pencerenin oluşturulması için tkinter kütüphanesi kullanılmıştır.

Matplotlib

Matplotlib, içerisinde birçok alt kütüphanesi bulunan ve temel olarak grafik çizimi sağlayan Python kütüphanesidir. Matplotlib kütüphanesi ile Matlab tarzındaki grafiklerin türetilmesini sağlayan komutları içeren alt kütüphanesi olan matplotlib.pyplot modülü projede grafik görselleştirmede kullanılmıştır.

Celluloid

Matplotlib kütüphanesi kullanılarak animasyon kullanımını kolaylaştıran celluloid, bir alt kütüphanedir. Celluloid ile her bir iterasyonda oluşturulan görseller bir çerçevede birleştirilir ve tek bir animasyon halinde çağrılmış olur. Projede celluloid kütüphanesi, oluşturulacak grafikler için yer ayrılması ve görüntülenmesi için kullanılmıştır.

III. GELİŞME

İlk olarak sıralama algoritması görselleştiricisi için yazacağımız algoritmanın taslağını oluşturduk. Daha sonra python dilinde yazacağımız programımızın yazım ve kullanım kolaylığı açısından Visual Studio Code'da yazmaya karar verdik. Tasarım tarafında arayüz olarak "Tkinter" kütüphanesini, grafikler için "canvas()" fonksiyonunu, grafiklerin animasyonu için ise "Celluloid" kütüphanesinden yararlandık.

Yazılım Mimarisi

Yazılım tarafında Python dili kullanmamız istenmiştir. Python dilinde grafik tasarımlarında genelde "Matplotlib" kullanılır. Algoritmanın ilk tasarım aşamalarında biz de tercih etmiş olsakta, daha sonra tasarım kaygılarımızı karşılayamadığından dolayı "canvas" fonksiyonu ile grafikleri kullandık. Burada "Matplotlib" kütüphanesinin genelde tercih edilmesinin sebebi sadece bir fonksiyon çağırarak bir grafik türünün oluşturulabilmesi ve ayrıca bellekten tasarruf etmenizi sağlayan birçok fonksiyonunun bulunması olsa da bu kütüphane tasarım açısıdan "Tkinter" kütüphanesiyle çalışmaya uygun değildir ve arayüz tasarımının oldukça zayıf kalmasına sebep olacaktır.

Mimarisinde ise proje sıralama algoritmalarının fonksiyonlarını, grafik çeşitlerinin fonksiyonlarını, veri girişi, veriye ve sıralama algoritmasına göre grafik oluşabilmesi için oluşturulan "generate()" fonksiyonunu ve kullanılan tasarım unsurlarının birtakım fonksiyonlarını içeren tek bir ".py" dosyasından oluşmaktadır. Burada mimarinin tasarım tarafında ise Github'ta bir geliştirici tarafından "Tkinter" için oluşturulan "Forest-ttk-theme" temasını kullanılmıştır (kaynakça olarak belirtilmiştir). Bu tema sayesinde oldukça eski ve kullaışlı olmayan orijinal ttk arayüzü çok daha modern bir çizgiye getirilmiştir. Temanın ana ".py" dosyasında kullanımı ise oldukça kolaydır. "root.tk.call" işlevi ile "forest-light.tcl" tema dosyası çağırılır ve "ttk.Style().theme_use" işlevi ile tema dosyaya uygulanır.

Paketlerin Yüklenmesi

İlk olarak matplotlib kütüphanesini yükledik. Bu yükleme işlemi için terminal ekranımızda pip install matplotlib komutunu çalıştırdık. Yükleme tamamlandıktan sonra da kendi projemize dahil etmek istediğimiz metotları proje dosyamıza import ettik.

İkinci olarak tkinter kütüphanesini terminalden pip install tk komutunu çalıştırarak proje dosyamıza import etmeye hazır hale getirdik.

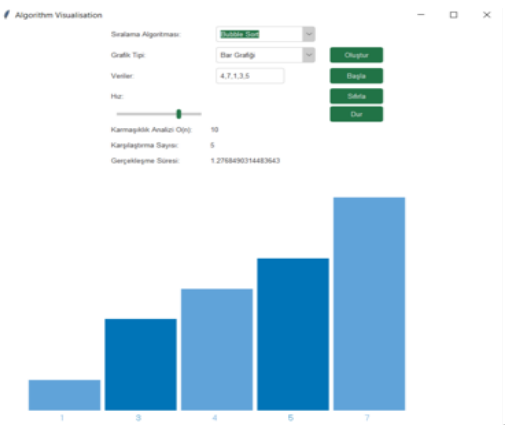
Son olarak animasyonlarımızı yaparken yararlandığımız celluloid kütüphanesini pip install celluloid komutunu terminalimizde çalıştırarak yüklemesini yaptık. Daha sonra da gerekli metotlarını proje dosyamızda import ettik.

Sıralama Algoritmalarının Gerçekleştirilmesi

Sıralama algoritmaları için her birine ayrı fonksiyon yazılmıştır. Bütün fonksiyonlar aynı değişkenlerle yazılmış, böylece genel olarak kodun bütünlüğü sağlanmıştır. Data, draw, timeTick ve sayac değişkenleri bütün sort fonksiyonları için ortak işlevdedir.

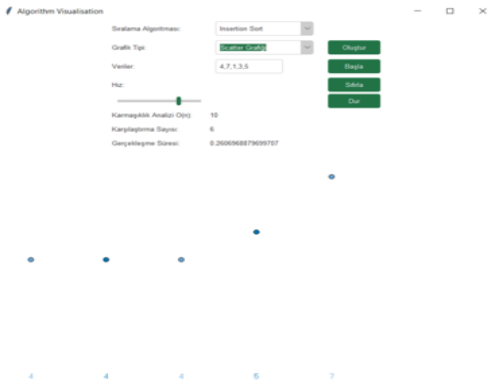
1. Bubble Sort

Bubble_Sort() fonksiyonu data,draw ve TimeTick değişkenlerini almaktadır. Bu fonksiyon iç içe for döngüsü içermektedir böylece ikinci for döngüsünde değerler dönerken bubble sort mantığına göre yanlış olan değer doğru olan değer ile yer değiştirmektedir. Colors adında listeye datalar yerleştirilir. Bu , görselleştirilen grafiğin renklerini ayırt etmek için kullanılır. Ayrıca her sort fonksiyonunda "sayac" değişkeni bulunur. Bu değişken karşılaştırma sayısı için gereklidir. TimeTick değişkeni ise her adımda geçen süreyi kontrol eder. Buna göre hız değişkeni ayarlanabilir hale gelir.



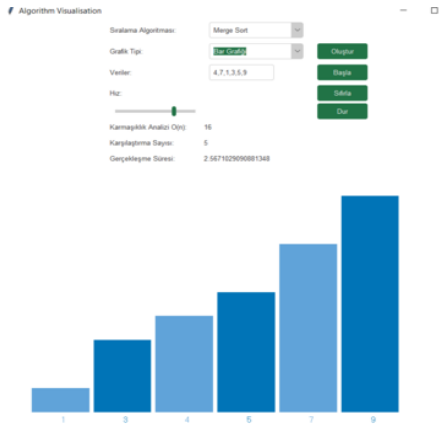
(Görsel: Bubble_Sort() ile bar grafiği çizimi)

Insertion_Sort() fonksiyonu ise bir döngü kullanarak sıralama işlemi gerçekleştirilir. Döngü, i değişkeni ile 1'den len(data)'ya kadar döner. Key adında bir değişken oluşturulur ve bu değişkene data[i] ataması yapılır. Bu, sıralanacak olan elemanı temsil eder. J değişkeni, i - 1 olarak atanır. Bu değişken, elemanları geriye doğru karşılaştırmak için kullanılır. Bir while döngüsü, j değeri sıfırdan büyük veya eşit ve data[j] değeri key değerinden büyük olduğu sürece çalışır. Data[j] değeri, data[j + 1]'e atanır ve j değeri bir azaltılır. Bu, elemanların kaydırılmasını sağlar. Döngü tamamlandığında, j + 1 indeksine key değeri atanır. Bu, doğru konumda olduğu anlamına gelir.



(Görsel: Insertion_Sort() ile scatter grafiği çizimi)

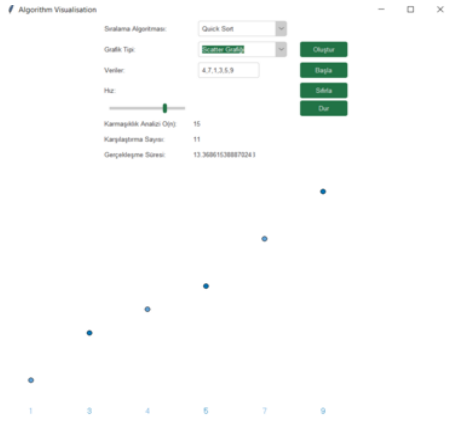
Merge_Sort() fonksiyonu merge_sort_alg adlı başka bir fonksiyonu çağırmak için kullanılır. "merge_sort_alg" adlı yardımcı bir fonksiyon tanımlanmıştır ve left, right aralığındaki veriyi sıralamak için merge sort algoritmasını uygular. Left değeri right değerinden küçük olduğu sürece (yani bir veya daha fazla eleman olduğu sürece) bazı adımlar gerçekleştirilir. Öncelikle middle değişkeni, left ve right arasındaki orta noktayı temsil eder. Merge_sort_alg fonksiyonu kendini iki ayrı aralık için tekrar çağırır: left ile middle arasını sıralamak için ve middle+1 ile right arasını sıralamak için. Ardından, merge fonksiyonu çağrılır ve bu fonksiyon, left, middle ve right aralıklarını birleştirir. Eğer data, sıralanmış bir diziye eşitse (yani sıralama tamamlanmışsa), drawData fonksiyonu kullanılarak sonuç görselleştirilir. Merge fonksiyonu, birleştirme adımını gerçekleştirir. Bu adımda, left ile middle aralığındaki elemanlar ve middle+1 ile right aralığındaki elemanlar ayrı ayrı sıralanır ve birleştirilir. İlk olarak, drawData fonksiyonu kullanılarak aralıkların renkleri görselleştirilir. Ardından, leftPart ve rightPart adında iki parça oluşturulur ve gerekli elemanlar bu parçalara atanır. LeftIdx ve rightIdx değişkenleri, leftPart ve rightPart parçalarındaki indeksleri takip eder. Bir döngü kullanarak, left ile right arasındaki tüm indeksler üzerinde dolaşılır. İç içe if ve elif ifadeleri kullanılarak, leftPart ve rightPart parçalarının elemanları karşılaştırılır ve sıralı bir şekilde birleştirilir.



(Görsel: Merge_Sort() ile bar grafiği çizimi)

Quick_Sort() fonksiyonuna gelecek olursak "swaps" ve "border" değişkenleri tanımlanır. swaps, eleman değişimlerinin sayısını tutar. Pivot, data dizisindeki tail indeksine karşılık gelen elemandır. "draw" fonksiyonu kullanılarak, veri ve renklerin görselleştirilmesi yapılır. Bir döngü kullanarak, head ile tail arasındaki elemanlar üzerinde dolaşılır. Eğer data[j] pivot elemandan küçükse, bu durumda eleman değişimi yapılması gerekmektedir. "draw" fonksiyonu kullanılarak, değişim yapılacak elemanın ve sınırların renkleri güncellenir, swaps değişkeni bir artırılır ve eleman değişimi gerçekleştirilir. Ayrıca, kodun içinde quick_sort adlı bir fonksiyon daha tanımlanmıştır. Bu fonksiyon, Quick Sort algoritmasını başlatmak için kullanılır.

Başlangıçta, head ve tail değerleri kontrol edilir. Eğer head değeri tail değerinden küçükse (yani en az 2 eleman varsa) Partition fonksiyonu kullanılarak, veri parçalanır ve pivot elemanı belirlenir. Quick_sort fonksiyonu, head ile pi-1 aralığını sıralamak ve pi+1 ile tail aralığını sıralamak için iki ayrı aralık için kendini tekrar çağırır. Eğer data, sıralanmış bir diziye eşitse, draw fonksiyonu kullanılarak sonuç görselleştirilir ve swaps değeri sıfırlanır.



(Görsel: Quick_Sort() ile scatter grafiği çizimi)

Selection_Sort() fonksiyonuna son olarak gelecek olursak, Bir döngü kullanarak, her bir elemanın doğru konuma yerleştirilmesi sağlanır. Dıştaki döngü, i değişkeni ile data dizisinin her bir elemanını dolaşır. Her bir döngü adımında, min_index değişkeni i olarak başlatılır. Bu değişken, henüz sıralanmamış bölgedeki en küçük elemanın dizindeki indeksini tutacaktır.

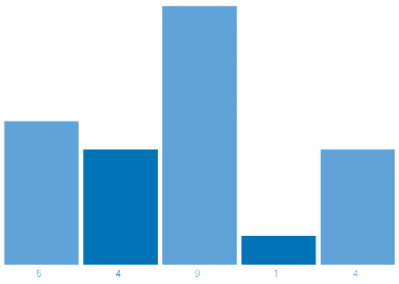
İçteki döngü, j değişkeni ile i + 1 ile len(data) arasındaki elemanları dolaşır. Bu döngü, sıralanmamış bölgedeki en küçük elemanın indeksini bulmak için kullanılır. Eğer data[j] değeri, data[min_index] değerinden küçükse, min_index değeri j olarak güncellenir. İçteki döngü tamamlandığında, min_index değeri en küçük elemanın indeksini tutar. Eğer min_index değeri i değerine eşit değilse, bu durumda eleman değişimi yapılması gerekmektedir. Swap işlemi gerçekleştirilir.

Grafik Türlerinin Entegre Edilmesi

Arayüzde görüntülenecek ve kullanıcı tarafından seçim işlemi yapılacak üç tane grafik türü vardır. Biz projemizdeki grafik türlerini Tkinter kütüphanesinden olan canvas üzerinde grafik için şekiller tanımlayarak oluşturduk.

1. Sütun Grafiği

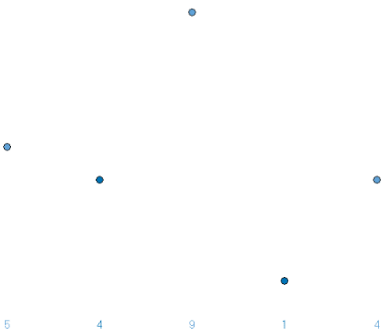
Sütun grafiğini oluştururken Tkinter kütüphanesinden canvas'ın create_regtangle metodunu tanımlayarak ve onun görselliğini, rengini, yüksekliğini ve genişliğini gelen veriler doğrultusunda düzenleyerek elde ettik.



(Görsel: Sütun grafiği)

2. Dağılım Grafiği

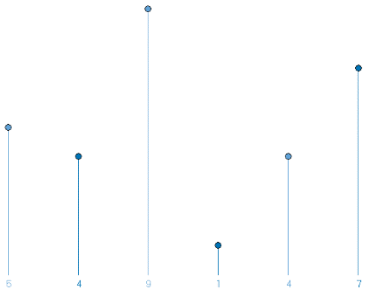
Dağılım grafiğini oluştururken Tkinter kütüphanesinden olan canvas'ın create_oval metodunu kullanarak bu grafiğin görselliğini, rengini ve boyutunu ayarladık. Ayrıca gelen veriler doğrultusunda düzenleyerek animasyonlara uyumluluğunu sağladık.



(Görsel: Dağılım grafiği)

3. Kök Grafiği

Kök grafiğini oluştururken Tkinter kütüphanesinden canvas'ın hem create_oval hem de create_line metotlarını kullandık. Bu iki metodun birbirlerine uyum sağlamaları için ikisi içinde özenle görselliği, renkleri, yükseklik ve genişliklerini düzenledik.



(Görsel: Kök grafiği)

Son olarak bu bütün grafik türlerini arayüz için Combobox oluşturup seçimi yapılacak olan grafik türlerini yerleştirdik. Ayrıca bu seçimleri her bir sıralama algoritmasında doğru bir şekilde çalışabilmesi için tanımlamalarını yaptık.

Butonların Oluşturulması

Proje arayüzünde bulunması gereken dört buton vardır.

Oluştur butonu, girilen değerleri seçilen grafik türüne göre oluşturup arayüz ekranında görüntülemeyi sağlamaktadır. Butona verilen generate işlevi ile, grafikte çizilmesi için girilen değerlerin her biri data adlı diziye eleman olarak atanır ve seçilen grafik türüne parametre olarak gönderilerek elemanların grafik elemanı olarak çizdirilmesi sağlanır. Elemanlar, değerlerine göre grafikte rastgele sıralanarak oluşturulur.

Başla butonu, oluşturulmuş olan grafiğin hangi sıralama algoritması ile sıralanacağı seçildikten sonra o algoritmaya göre grafik elemanlarını, değerleri küçükten büyüğe olacak şekilde sıralamaya başlar. Elemanların sıralanması ve sıralanırken karşılaştırılmaları, seçilen algoritma sıralamasının mantığına göre değişiklik göstermektedir. Elemanların sıralanması bittiğinde, karmaşıklık analizi ve karşılaştırma sayısı arayüzde gösterilmektedir.

Dur butonu, oluşturulan ve sıralaması başlatılan grafiğin, herhangi bir anda basıldığında durmasını ve tekrar basıldığı takdirde kaldığı yerden sıralamaya devam etmesini sağlamaktadır.

Sıfırla butonu, oluşturulan ve daha sonrasında sıralaması bitmiş olan grafiğin görseli, grafiğe ait karmaşıklık analizi ve karşılaştırma sayısı hesaplamaları kaldırarak yeniden grafik oluşturulmak üzere temiz bir arayüz ekranı sağlar. Sıfırla butonunun işlevlendirildiği reset_chart fonksiyonu, ekranı temizlemek için canvas nesnesini kullanmaktadır.

```
def reset_chart():
    label2 = tk.Label(UI_frame, text=str(0))
    label2.grid(row=5, column=1, padx=0, pady=5, sticky=W)
     # Verileri sıfırla
    data = []
     # Grafik nesnesini sıfırla
    fig.clear()
     # Canvas'ı temizle
    canvas.delete("all")
     # Canvas'ı güncelle
    canvas.draw()
```

(Görsel: reset_chart fonksiyonu)

Veri Girişi ve Hız Ayarı

Kullanıcının girdiği veriyi alabilmek için ilk olarak Tkinter kütüphanesinden bir giriş birimi özelliği taşıyan Entry ile bir textbox oluşturduk. Bu sayede kullanıcı verileri tek bir alandan ve istediği kadar girebilmektedir. Daha sonra verileri alabilecek ve grafik türlerine verilerin atamasını yapabilecek bir metot oluşturduk. Bu metot ile kullanıcının girdiği verileri elde ettik. Ayrıca kullanıcının ve programın daha doğru işlem gerçekleştirebilmesi için verileri, aralarındaki virgüller aracılığıyla karışıklık olmadan çekebildik.

(Görsel: Kullanıcı veri girişi)

Hız ayarı bilgilerini kullanıcı arayüzünde Tkinter kütüphanesinin Scale aracı ile çektik. Ayrıca 1 saniyeden 0.000001 saniyeye kadar kullanıcının seçim yapabilmesi için bir aralık verdik. Ayrıca kullanıcı bir seçim yapmasa bile sıkıntı çıkarmaması için varsayılan bir değer atadık. Daha sonrada hız bilgisini, animasyon hızına entegrede doğru bir şekilde kullanabilmek için metotlardan çağırdık.

(Görsel: Hız ayarı)

Karşılaştırma Sayısının Hesaplanması

Sıralama algoritmalarının; bir grafik için her birinin ortaya çıkardığı sonuç aynı olsa da, sonucu ortaya çıkarma sırasında elemanları karşılaştırma ve buna göre geçici olarak yerleştirme durumları farklıdır. Her sıralama algoritmasının kendine özgü sıralama mantığı vardır ve buna göre karşılaştırma sayıları da değişmektedir. Örneğin 5 elemanlı bir bar grafiğini sıralamada merge sort ile bubble sort'un elemanları karşılaştırma sayısı farklıdır. Oluşturulan bir grafik, her bir sıralama algoritması ile sıralanarak karşılaştırma sayıları kıyaslanabilmekte ve hangisinin en avantajlı sıralama algoritması olduğu da bu şekilde belirlenebilmektedir.

Global bir sayaç değişkeni, main fonksiyonunda tanımlanarak başlangıç değeri 0 olarak atanmıştır. Ardından her bir sıralama algoritmasının fonksiyonu içerisinde, ilgili kısımlarda değeri 1 artırılarak her bir iterasyon için sayacın artırılması ve sonuç olarak grafik sıralaması tamamlandığında sayaç değerinin arayüz tasarımında gösterimi sağlanmıştır.

Karmaşıklık Analizinin Hesaplanması

Kullanıcının girdiği veri değerlerine göre oluşturulan ve seçtiği algoritmaya göre sıralaması yapılan grafiğin karmaşıklık analizi; grafiğin çizim süresi ve kaynak(bellek, işlemci, veri boyutu, veri yapısı vb.) kullanımı değerlendirilerek hesaplanmaktadır. Tıpkı karşılaştırma sayısının hesaplanmasında olduğu gibi, karmaşıklık analizinin hesaplanması da algoritmaların sıralama mantığına göre değişiklik göstermektedir.

Merge sort dışında kullanılan tüm sıralama algoritmalarının karmaşıklık analizi hesaplaması aynı metotla gerçekleştirilmektedir çünkü bu dört sıralama algoritmasının karmaşıklık analizi hesaplama yöntemi, genellikle karşılaştırma sayısı ve yer değiştirme (swap) sayısı üzerine kuruludur. Bu algoritmalarda genellikle döngüler aracılığıyla elemanlar karşılaştırılıp yer değiştirme işlemleri gerçekleştirilir. Karşılaştırma sayısı ve yer değiştirme sayısı, bu algoritmalardaki temel işlemler olup genellikle karmaşıklık analizi için temel bir ölçü olarak kullanılır.

Merge sort ise rekürsif bir algoritma olduğu için daha karmaşık bir analize ihtiyaç duyar. Merge sort algoritması, birleştirme adımlarında ekstra bellek kullanır ve genellikle rekürsif olarak uygulanır. Karmaşıklık analizinde, bölme işlemi ve birleştirme işlemi üzerinden ilerlenir ve daha karmaşık bir hesaplama yapılır.

Proje Gerçekleştirilirken Karşılaşılan Hatalar

- Grafik türlerini sıralama algoritmalarına entegre etmede sorun yaşadık. Bu hata yüzünden kullanıcı arayüzünde girdiğimiz sıralama algoritmasına sadece tek bir grafik tipi atanabiliyordu. Bu mantık hatasını çözebilmek amacıyla her sıralama algoritmaları için tüm grafik tiplerinin oluşturulabildiği bir fonksiyon oluşturduk. Tüm çağırma işlemlerini de bu fonksiyonu kullanarak gerçekleştirdik.
- Karşılaştırma sayısının hesaplanmasında kullanılan global sayaç değişkeninin değeri, klasör içindeki sıralama algoritmalarının bulunduğu dosyalar üzerinden main dosyaya çekilemediğinden, algoritma metotlarının tümü main dosyaya alınıp sayaç değerleri bu metotlar üzerinden çekilmiştir.
- AttributeError: 'Canvas' object has no attribute 'draw' hatası. “Dur” butonuna atanan stop_animation fonksiyonu, grafik sıralandığı esnada, içinde bulunan animasyonu durdurma komutlarını çalıştırmadığından sıralama algoritması, bu butona tıklandığında da verileri sıralamayı bitirmekte ve durmamaktadır. Grafik sıralaması bittiğinde belirtilen Attribute hatasını vermektedir.

IV. SONUÇ

Python günümüzde birçok alanda kullanılan bir dil. Masaüstü uygulamalarda, web site yapımlarında veya bizim projemizde olduğu gibi ver görselleştirme uygulamalarında ve birçok alanda kullanılan bir dil. Birçok yazılımcının en az bir kere deneyimlemiş olduğu ve diğer programlama dillerinin yazım hamallığının önüne geçmiş bir programlama dilidir.

Biz projemizi Python’ı ilk defa deneyimleyerek hazırladık. Bu sebeple başlarda birçok hatayla ve sıkıntıyla karşılaştık. Fakat Python’ın yazım kolaylığı ve internet aleminde hataların çözümü için ayrıca birçok kaynak bulunduğu için projemizin hatalarını gidermemiz fazla zamanımızı almadı.

Python’ın kütüphanelerinin bu projede oldukça kullandık. Birçok işlevi bize sağlayan kütüphaneler ve onların metotları sayesinde animasyonları ve grafik işlemlerini halletmek çok daha kolay oldu. Ayrıca projemizin konusu olan sıralama algoritmaları konusu bilgisayar ile haşır neşir olan mühendislik bölümleri için olmazsa olmazlardan olduğu için eski bilgilerimizi pekiştirmemizi ve yeni bilgiler edinmemizi de sağladı. Bu proje sıralama algoritmalarının mantığını anlamamız konusunda bize oldukça yardımcı oldu.

Hazırladığımız projenin belirttiğimiz gibi birçok açıdan bize katkısı oldu. Ek olarak grup arkadaşlarımızla ortak bir proje yapmak, organize olabilmek, o projeyi yönetebilmek, eşit dağılımlar yapabilmek, hataları birlikte çözebilmek, gerçekleştirilecek her ister için birlikte kafa yormak ve kendi yazılım bilgilerimizi birbirimizle paylaşabilmek gibi birçok unsur sayesinde, aslında ilerideki çalışma hayatımızda bir projenin yapım aşamalarını yönetebilmek amacıyla, sahip olmamız gereken sosyal ve teknik yeteneklerimizi de pekiştirmemizi sağladı.

KULLANILAN KAYNAKLAR

[1] Matplotlib Rehberi
[2] Pip Özel Versiyon İndirme Yöntemi
[3] Python Packages Tutorial
[4] Bir Dosya Üzerinden Diğer Dosyadaki Değişkene Erişim
[5] Hatalar İçin Çözüm Araştırması
[6] Paket Yüklemeleri
[7] Python 3.11.3 Versiyon İndirme
[8] Tema Dosyası