“SIRALAMA ALGORİTMALARI GÖRSELLEŞTİRİCİSİ”

Eren EŞSİZ
201307047
Kocaeli Üniversitesi
Kocaeli, İzmit
erenefe4059@gmail.com

Yavuz Selim ACAR
201307059
Kocaeli Üniversitesi
Kocaeli, İzmit
acaryavuzz1@gmail.com

Faruk Erdem YILMAZ
201307030
Kocaeli Üniversitesi
Kocaeli, İzmit
farukerdem61@gmail.com

Özet-Bu proje, Python programlama dilini kullanarak veri yapıları ve sıralama algoritmalarının uygulanmasını amaçlamaktadır. Proje, kullanıcıya çeşitli sıralama algoritmalarını görselleştirme imkanı sunan bir görselleştirme aracı geliştirmeyi hedeflemektedir. Bu araç, kullanıcının farklı sıralama algoritmalarını gerçek zamanlı olarak gözlemlemesine ve analiz etmesine olanak tanır. Projenin amacı, öğrencilerin ve programcı adaylarının sıralama algoritmalarını daha iyi anlamalarına yardımcı olmaktır.

Anahtar Kelimeler:

Anahtar Kelimeler — ; Python, veri yapıları, sıralama algoritmaları, görselleştirme, gerçek zamanlı.

GİRİŞ

Bu rapor, kullanıcının sıralama algoritmalarını ve grafik türlerini kullanarak bir listenin sıralanmasını ve sıralama işleminin adımlarını görsel olarak görmesini sağlayan bir proje hakkında bilgi sunmaktadır. Projede amaç, kullanıcıya interaktif bir kullanıcı arayüzü (GUI) sağlayarak sıralama işlemini anlaşılır bir şekilde görselleştirmektir.

Kullanıcı arayüzü, sol panel ve ana panel olmak üzere birkaç farklı bileşenden oluşmaktadır. Sol panel, kullanıcının liste boyutunu belirlemesine veya manuel olarak bir liste girmesine olanak tanır. Ayrıca, sıralama işlemini hızlandırmak veya yavaşlatmak için animasyon hızını ayarlamak mümkündür. Sol panelde ayrıca farklı sıralama algoritmaları ve grafik tipleri seçilebilir. Oluştur, başla, dur ve sıfırla butonları, kullanıcının işlemleri yönetmesine yardımcı olur.

Ana panel, kullanıcının yapmış olduğu seçimlere göre görsel bir arayüz sunar. Seçilen algoritmanın sıralama işlemi gerçekleştirilirken, renk kodları kullanılarak karşılaştırmaların ve sıralanmış/sıralanmamış değerlerin anlaşılabilirliği artırılır. Her adımda yapılan karşılaştırma sayısı, arka planda tutulur ve animasyon üzerinde yapılan değişikliklere paralel olarak artırılır. Belirli periyotlarla karşılaştırma sayısı ana panelde güncellenerek gösterilir. Sıralama işlemi tamamlandığında, karşılaştırma sayısı ve algoritmanın karmaşık analizi sonucu kullanıcıya sunulur.

Bu rapor, projenin tasarımı, uygulama süreci ve elde edilen sonuçlar hakkında ayrıntılı bilgi sunacak. Ayrıca, projenin kodlama ayrıntılarına da değinilecek ve kullanılan Python programlama dili ile grafik animasyonları için kullanılan kütüphaneler hakkında bilgi verilecektir.

Proje Amacı

Bu projenin temel amacı, kullanıcının sıralama algoritmalarını ve grafik türlerini kullanarak bir listenin sıralanmasını görsel olarak gözlemlemesini sağlamaktır. Projede interaktif bir kullanıcı arayüzü kullanılarak, kullanıcının liste boyutunu belirleyebilmesi veya manuel olarak bir liste girebilmesi mümkün kılınmıştır.

```
from PyQt5 import QtWidgets
from PyQt5.QtChart import QChartView, QBarSeries, QBarSet, QLineSeries
from PyQt5.QtGui import QPen, QColor
from PyQt5.QtCore import QPropertyAnimation, QEasingCurve

from PyQt5.QtWidgets import QMessageBox
from PyQt5 import QtCore
from PyQt5.QtChart import QChart, QScatterSeries, QValueAxis
from PyQt5.QtGui import QPainter
from PyQt5.QtCore import Qt
import random
```

Projede yer alan sıralama algoritmaları (Seçme Sıralaması, Kabarcık Sıralaması, Ekleme Sıralaması, Birleştirme Sıralaması, Hızlı Sıralama) kullanılarak, sıralama işlemi gerçekleştirilmekte ve her adımda yapılan karşılaştırmalar görsel olarak kullanıcıya aktarılmaktadır. Bu sayede kullanıcı, sıralama algoritmalarının nasıl çalıştığını ve her adımda yapılan karşılaştırmaların nasıl sonuçlandığını görsel olarak takip edebilmektedir.

Ayrıca, projede farklı grafik tipleri (Dağılım Grafiği, Sütun Grafiği, Kök Grafiği) kullanılarak, sıralama işleminin sonucu görsel olarak temsil edilmektedir. Bu şekilde kullanıcı, sıralanmış ve sıralanmamış değerlerin farkını daha iyi anlayabilir ve sıralama sürecinin etkisini görsel olarak gözlemleyebilir.

Projenin diğer bir amacı ise kullanıcının işlemi yönetebilmesini sağlamaktır. Oluştur, başla, dur ve sıfırla butonları kullanıcıya işlemi başlatma, durdurma ve sıfırlama seçenekleri sunmaktadır. Kullanıcı, istediği zaman sıralama işlemini başlatabilir, duraklatabilir veya sıfırlayabilir.

Bu proje, kullanıcının sıralama algoritmalarını ve grafik türlerini daha iyi anlamasını sağlamak amacıyla geliştirilmiştir. Kullanıcılar, gerçek zamanlı görsellerle sıralama işleminin nasıl gerçekleştiğini izleyebilir ve sıralama algoritmalarının performansını karşılaştırabilir. Ayrıca, proje sayesinde kullanıcılar, sıralama algoritmalarının çalışma prensiplerini ve farklı grafik tiplerinin kullanımını deneyimleyerek kavrama fırsatı bulurlar.

Kullanıcı Arayüzü ve İşlevleri:

Bu projede, kullanıcı arayüzü (GUI) birkaç farklı bileşenden oluşur ve kullanıcının projeyi etkileşimli bir şekilde kullanabilmesini sağlar.

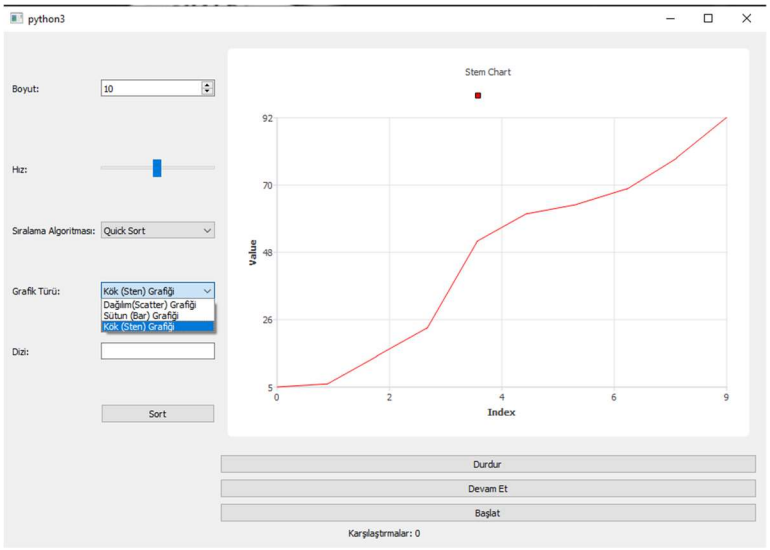
1. Sol Panel:
    - Boyut: Kullanıcı sıralanacak listeyi manuel olarak girebilir veya bir boyut belirleyerek rasgele bir liste oluşturabilir. Manuel giriş durumunda, kullanıcının her bir liste elemanını özel bir karakterle ayırması gerekmektedir.
    - Hız: Animasyon hızı ölçeklendirilebilir şekildedir. Kullanıcı, sıralama adımlarının hızını ayarlayabilir.
2. Sıralama Algoritmaları:
   Projede çeşitli sıralama algoritmaları bulunur ve kullanıcı istediği algoritmayı seçebilir. Mevcut sıralama algoritmaları:
    - Seçme Sıralaması (Selection Sort)
    - Kabarcık Sıralaması (Bubble Sort)
    - Ekleme Sıralaması (Insertion Sort)
    - Birleştirme Sıralaması (Merge Sort)
    - Hızlı Sıralama (Quick Sort)
3. Grafik Tipleri:
   Projede kullanıcıya farklı grafik tipleri seçenekleri sunulur. Kullanıcı istediği grafik türünü seçebilir. Mevcut grafik tipleri:
    - Dağılım (Scatter) Grafiği
    - Sütun (Bar) Grafiği
    - Kök (Stem) Grafiği
4. Oluştur-Başla-Dur-Sıfırla Butonları:
    - Oluştur: Kullanıcı, oluştur butonuna tıkladığında, belirlediği liste değerleri ve grafik türüne göre arayüz oluşturulur. Seçilen grafik türüne göre sıralanmış ve sıralanmamış değerlerin görsel temsili hazırlanır.
    - Başla: Başla butonuna tıklandığında, belirtilen isterlere uygun olarak sıralama işlemi başlatılır ve animasyon gösterilir.
    - Dur: Dur butonuna tıklandığında, animasyon durdurulur. Tekrar tıklandığında animasyonun devam etme seçeneği sunulur.
    - Sıfırla: Sıfırla butonuna tıklandığında, liste temizlenir ve arayüz varsayılan durumuna döner.
5. Ana Panel:
    - Sol panelde kullanıcının yapmış olduğu seçimlere göre ana panelde görsel bir arayüz sunulur. Seçilen algoritmaya göre sıralama gerçekleştirilir ve her adımda yapılan karşılaştırmalar renk kodları kullanılarak gösterilir. Sıralanmış ve sıralanmamış değerler

PROJE TASARIMI ve TEKNOLOJİLERİ

Kullanılan Programlama Dili ve Kütüphaneler:

Bu projede, Python programlama dili kullanılmaktadır. Python, kullanıcı dostu ve geniş bir kütüphane desteğiyle GUI uygulamaları geliştirmek için yaygın olarak tercih edilen bir dildir.

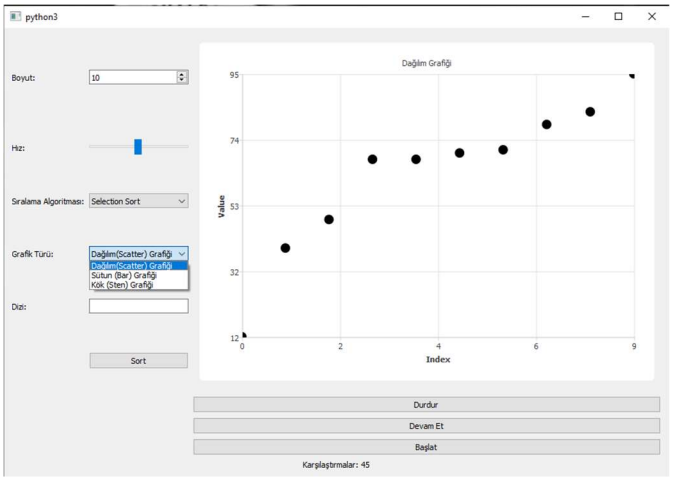
Aşağıda, projede kullanılabilecek bazı kütüphaneler ve işlevleri verilmiştir:



1. Tkinter:
    - Tkinter, Python'un standart kütüphanesinde bulunan bir GUI toolkit'idir.
    - Tkinter, pencere, düğme, giriş kutusu, etiket gibi GUI bileşenlerini oluşturmak ve yönetmek için kullanılabilir.
    - Kullanıcı arayüzünün tasarımı ve etkileşimli bileşenlerin işlevselliği Tkinter kullanılarak gerçekleştirilebilir.
2. Matplotlib:
    - Matplotlib, Python'da verilerin grafiksel olarak gösterilmesi için yaygın olarak kullanılan bir kütüphanedir.
    - Scatter, Bar veya Stem grafik tipleri gibi farklı grafik türlerini destekler.
    - Sıralama adımlarının görsel olarak takip edilmesi için kullanıcıya grafiksel bir arayüz sağlamak için Matplotlib kullanılabilir.
3. Seaborn:
    - Seaborn, Python'da veri görselleştirme için kullanılan bir kütüphanedir.
    - Seaborn, Matplotlib'i temel alır ve daha çekici ve bilgilendirici grafikler oluşturmak için ek özellikler sunar.

- Proje için grafiklerin görsel çekiciliğini artırmak veya veri analizini kolaylaştırmak için Seaborn kullanılabilir.

4. Plotly:



- Plotly, etkileşimli ve web tabanlı görselleştirmeler için kullanılan bir Python kütüphanesidir.
- Plotly, interaktif grafikler, grafik animasyonları ve kullanıcı etkileşimine olanak sağlayan araçlar sunar.
- Proje için daha dinamik ve etkileşimli bir görselleştirme deneyimi sağlamak için Plotly kullanılabilir.

5. Bokeh:
   - Bokeh, Python'da interaktif ve tarayıcı tabanlı görselleştirme için kullanılan bir kütüphanedir.
   - Bokeh, verileri tarayıcıda çalışan JavaScript tabanlı bir grafik olarak sunar.
   - Proje için web tabanlı ve interaktif grafikler oluşturmak için Bokeh kullanılabilir.
   - Bu kütüphaneler, Python programlama dili ile kullanıcı arayüzünün tasarlanması, verilerin görsel olarak temsil edilmesi ve animasyonların gerçekleştirilmesi için kullanılabilir. Proje gereksinimlerine ve tercihlere bağlı olarak uygun olan kütüphaneler seçilebilir.

Tasarım Yaklaşımı:

Bu projede, kullanıcı arayüzü tasarımı ve görsel öğelerin yerleştirilmesi için bir "Bileşen Tabanlı Tasarım Yaklaşımı" kullanılabilir. Bu yaklaşım, kullanıcı arayüzünü farklı bileşenlerin birleşimi olarak ele alır ve her bileşenin özel bir işlevi ve görünümü vardır. Bu şekilde, her bileşenin bağımsız olarak tasarlanması, geliştirilmesi ve yönetilmesi daha kolay hale gelir.

Aşağıda, projede kullanılacak bileşenlerin bir listesi ve her bir bileşenin işlevleri verilmiştir:

1. Sol Panel:
   - Boyut bileşeni: Kullanıcının sıralanacak listeyi manuel olarak girebilmesini veya rasgele bir liste oluşturabilmesini sağlar.
   - Hız bileşeni: Animasyon hızını kullanıcının ayarlayabilmesini sağlar.
   - Sıralama Algoritmaları bileşeni: Kullanıcının farklı sıralama algoritmaları arasından seçim yapabilmesini sağlar.
   - Grafik Tipleri bileşeni: Kullanıcının farklı grafik türleri arasından seçim yapabilmesini sağlar.
   - Oluştur, Başla, Dur, Sıfırla butonları: Kullanıcının listeyi oluşturma, animasyonu başlatma, durdurma ve sıfırlama işlemlerini gerçekleştirmesini sağlar.
2. Ana Panel:
   - Seçilen algoritmaya göre sıralama işlemini gerçekleştirir ve görsel olarak sunar.
   - Karşılaştırmaların sayısını tutar ve günceller.
   - Grafiksel arayüz üzerinde renk kodları kullanarak karşılaştırma ve sıralama adımlarını gösterir.
   - Animasyonun duraklatılması, devam ettirilmesi ve tamamlanması durumlarını yönetir.
   - Sıralama işlemi tamamlandığında karşılaştırma sayısı ve algoritmanın karmaşıklık analizine dayalı sonuçları ekrana yazdırır.

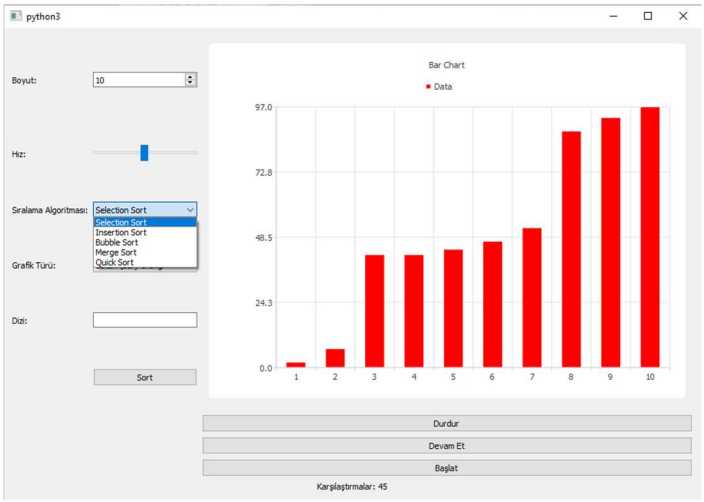
Bu bileşenler, kullanıcı arayüzünün işlevselliğini ve kullanılabilirliğini sağlamak için bir araya getirilir ve birlikte çalışır. Bileşen Tabanlı Tasarım Yaklaşımı, bileşenlerin bağımsız olarak geliştirilmesine ve gelecekteki değişikliklere uyum sağlamalarına olanak tanır. Ayrıca, projenin modülerlik, ölçeklenebilirlik ve sürdürülebilirlik gibi tasarım prensiplerine uygun olmasını sağlar.

```
def pause_animation(self):
    self.chart_view.chart().animation().pause()

def resume_animation(self):
    self.chart_view.chart().animation().resume()

def start_animation(self):
    self.chart_view.chart().animation().start()

def apply(self):
    try:
        # Seçilen değerleri alma
        size = self.size_spinbox.value()
        speed = self.speed_slider.value()
        sort_algorithm = self.sort_combo.currentText()
        graph_type = self.graphic_combo.currentText()

        # Seçilen değerleri ekrana yazdırma
        print(f"Boyut: {size}")
        print(f"Hız: {speed}")


        print(f"Sıralama Algoritması: {sort_algorithm}")
```

Kullanıcı Arayüzü Tasarımı ve Bileşenler:

Bu projede kullanıcı arayüzü, kullanıcının sıralama algoritmalarını ve grafik türlerini seçebilmesini, listeyi düzenleyebilmesini ve sıralama işlemini görsel olarak izleyebilmesini sağlamak için bir dizi bileşenden oluşur. İşte kullanıcı arayüzündeki temel bileşenler:

1. Sol Panel Bileşenleri:
    a. Boyut Bileşeni: Kullanıcının sıralanacak listeyi manuel olarak girebileceği veya rasgele bir liste oluşturabileceği bir metin giriş kutusu veya düğmeler içerir.
    b. Hız Bileşeni: Animasyon hızını ayarlamak için bir kaydırıcı veya açılır menü gibi bir kontrol içerir.



    c. Sıralama Algoritmaları Bileşeni: Kullanıcının seçim yapabileceği farklı sıralama algoritmalarını listeler.
    d. Grafik Tipleri Bileşeni: Kullanıcının seçebileceği farklı grafik türlerini listeler.
    e. Oluştur, Başla, Dur, Sıfırla Butonları: Kullanıcının listeyi oluşturması, sıralamayı başlatması, durdurması veya sıfırlaması için düğmeler içerir.
2. Ana Panel Bileşenleri:
    a. Grafiksel Gösterim Bileşeni: Seçilen grafik türüne göre listeyi görsel olarak temsil eder (örn. dağılım grafiği, sütun grafiği, kök grafiği).
    b. Animasyon Kontrol Bileşenleri: Başlat, duraklat ve devam et düğmeleri gibi düğmeler içerir.
    c. Karşılaştırma Sayısı Bileşeni: Sıralama işlemi sırasında yapılan karşılaştırmaların sayısını gösteren bir metin kutusu veya etiket içerir.
    d. İşlem Tamamlanınca Sonuç Bileşeni: Sıralama işlemi tamamlandığında karşılaştırma sayısını ve algoritmanın karmaşıklık analizi sonuçlarını gösteren bir metin kutusu veya etiket içerir.

Bu bileşenler, kullanıcının sıralama algoritmalarını ve grafik türlerini seçmesini, liste boyutunu ayarlamasını, işlemi başlatmasını veya durdurmasını ve sonuçları izlemesini sağlar. Kullanıcı arayüzü, kullanıcıya kolaylık sağlamak, anlaşılır ve kullanıcı dostu bir deneyim sunmak için uygun düzenleme, renklendirme ve etkileşimli öğelerin kullanımını içerir.

KULLANICI İSTEKLERİNİN KARŞILANMASI

Sol Panel İstekleri:

1. Boyut:
    - Kullanıcı sıralanacak listeyi manuel olarak girebilmeli.
    - Listede her bir elemanı özel bir karakter ile ayrılmalı.
    - Kullanıcı liste boyutunu belirleyerek rasgele bir liste oluşturabilmeli.
2. Hız:
    - Animasyon hızı ölçeklendirilebilir olmalı.
3. Sıralama Algoritmaları:
    - Seçme Sıralaması (Selection Sort)
    - Kabarcık Sıralaması (Bubble Sort)
    - Ekleme Sıralaması (Insertion Sort)
    - Birleştirme Sıralaması (Merge Sort)
    - Hızlı Sıralama (Quick Sort)
4. Grafik Tipleri:
    - Dağılım (Scatter) Grafiği
    - Sütun (Bar) Grafiği
    - Kök (Stem) Grafiği
5. Oluştur-Başla-Dur-Sıfırla Butonları:
    - Oluştur butonuna basıldığında, verilen liste değerleri ve grafik türüne göre arayüz oluşturulmalı.
    - Başla butonuna basıldığında, belirtilen isterlere uygun olarak sıralama animasyonu başlatılmalı.
    - Dur butonuna basıldığında, animasyon durdurulmalı. Tekrar basıldığında animasyona devam etme seçeneği olmalı.
    - Sıfırla butonuna basıldığında, liste temizlenmeli.

```
print(f"Sıralama Algoritması: {sort_algorithm}")
array_input = self.array_line_edit.text()
if array_input:
    data = list(map(int, array_input.split()))
else:
    size = self.size_spinbox.value()
    data = [random.randint(1, 100) for _ in range(size)]
print(data)
if (sort_algorithm == 'Bubble Sort'):
    sorted_data, comparisons = bubble_sort(data)
    print("Buble Sort : ")
elif (sort_algorithm == 'Quick Sort'):
    sorted_data, comparisons = quick_sort(data, 0, len(data) - 1)
    print("Quick Sort : ")
elif (sort_algorithm == 'Selection Sort'):
    sorted_data, comparisons = selection_sort(data)
    print("Selection Sort : ")
elif (sort_algorithm == 'Insertion Sort'):
    sorted_data, comparisons = insertion_sort(data)
    print("Insertion Sort : ")
elif (sort_algorithm == 'Merge Sort'):
    sorted_data, comparisons = merge_sort(data)
    print("Merge Sort : ")
```

Bu isteklere göre sol panelde yer alan bileşenler, kullanıcının listeyi girmesini, hızı ayarlamasını, sıralama algoritmasını seçmesini, grafik türünü seçmesini ve ilgili işlemleri gerçekleştirmesini sağlamalıdır. Bu bileşenler, kullanıcıya kolaylık sağlamak ve istenilen işlemleri yapabilmesini sağlamak için uygun kontrolleri ve özellikleri içermelidir.

Ana Panel İstekleri:

1. Görsel Arayüz:
   - Sol panelde kullanıcının yapmış olduğu seçimlere göre ana panelde görsel bir arayüz sunulmalıdır.
   - Sıralama işlemi gerçekleştirilirken anlaşılabilirliği arttırmak için renk kodları kullanılmalıdır.

```
self.comparisons_label = QtWidgets.QLabel('Karşılaştırmalar: 0')
self.comparisons_label.setAlignment(QtCore.Qt.AlignCenter)
# Animasyon kontrol butonları
self.pause_button = QtWidgets.QPushButton('Durdur')
self.pause_button.clicked.connect(self.pause_animation)
self.resume_button = QtWidgets.QPushButton('Devam Et')
self.resume_button.clicked.connect(self.resume_animation)
self.start_button = QtWidgets.QPushButton('Başlat')
self.start_button.clicked.connect(self.start_animation)

# Dizi girişi için QLineEdit oluşturma
self.array_label = QtWidgets.QLabel('Dizi:')
self.array_line_edit = QtWidgets.QLineEdit()

# Boyut seçimi
self.size_label = QtWidgets.QLabel('Boyut:')
self.size_spinbox = QtWidgets.QSpinBox()
self.size_spinbox.setRange(1, 10000)
self.size_spinbox.setValue(1000)

# Hız seçimi
self.speed_label = QtWidgets.QLabel('Hız:')
self.speed_slider = QtWidgets.QSlider(QtCore.Qt.Horizontal)
self.speed_slider.setRange(1, 100)
self.speed_slider.setValue(50)

# Sıralama algoritması seçimi
self.sort_label = QtWidgets.QLabel('Sıralama Algoritması:')
```

   - Karşılaştırılan değerler aynı renk kodunu paylaşmalıdır.
   - Sıralanmış ve sıralanmamış değerler farklı renk kodları ile tanımlanmalıdır.
2. Renk Kodlaması:
   - Sıralama işlemi sırasında kullanılacak renk kodları, görsel arayüzdeki elemanların durumunu ve ilişkilerini vurgulamalıdır.
   - Karşılaştırılan değerler, sıralanmış ve sıralanmamış değerler için farklı renkler kullanılmalıdır.
   - Renkler, kullanıcıya sıralama işlemini anlamak için görsel ipuçları sağlamalıdır.
3. Karşılaştırma Sayısı İzleme:
   - Seçilen algoritma göz önünde bulundurularak her bir adımda yapılan karşılaştırma sayısı arka planda tutulmalıdır.
   - Animasyon üzerinde yapılan değişikliklere paralel olarak karşılaştırma sayısı güncellenmeli ve izlenmelidir.
4. Karşılaştırma Sayısı Güncelleme:

```
#Sıralama algoritmaları
def bubble_sort(arr):
    n = len(arr)
    comparisons = 0

    for i in range(n):
        for j in range(0, n - i - 1):
            comparisons += 1
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    return arr, comparisons
```

   - Anlık olarak veya belirli periyotlarla (örneğin 3 saniye) karşılaştırma sayısı ana panelde güncellenerek gösterilmelidir.

```
# Pencere düzeni
layout = QtWidgets.QGridLayout()
layout.addWidget(self.size_label, 0, 0)
layout.addWidget(self.size_spinbox, 0, 1)
layout.addWidget(self.speed_label, 1, 0)
layout.addWidget(self.speed_slider, 1, 1)
layout.addWidget(self.sort_label, 2, 0)
layout.addWidget(self.sort_combo, 2, 1)
layout.addWidget(self.graphic_label, 3, 0)
layout.addWidget(self.graphic_combo, 3, 1)
layout.addWidget(self.array_label, 4, 0)
layout.addWidget(self.array_line_edit, 4, 1)
layout.addWidget(self.apply_button, 5, 1)
layout.addWidget(self.chart_view, 0, 2, 6, 1)
layout.addWidget(self.chart_view, 0, 2, 7, 1)
layout.addWidget(self.pause_button, 7, 2)
layout.addWidget(self.resume_button, 8, 2)
layout.addWidget(self.start_button, 9, 2)
layout.addWidget(self.comparisons_label, 10, 0, 1, 3)

self.setLayout(layout)

# Pencere boyutu
self.resize(800, 500)
```

   - Karşılaştırma sayısı, sıralama işlemi ilerledikçe artmalı ve kullanıcıya işlemin ilerlemesini görsel olarak aktarmalıdır.

5. Sıralama İşlemi Tamamlandığında Bilgilendirme:
   - Sıralama işlemi tamamlandığında, karşılaştırma sayısı ve algoritmanın karmaşıklık analizi sonucu elde edilen bilgiler kullanıcıya gösterilmelidir.
   - Bu bilgiler, kullanıcının işlemin performansını anlamasına yardımcı olmalıdır.
   - Kullanıcıya sıralama işleminin tamamlandığına dair bir bildirim sunulmalıdır.

Bu isteklere göre ana panelde yer alan bileşenler, sıralama işleminin görsel olarak takip edilmesini, karşılaştırma sayısının izlenmesini ve güncellenmesini, kullanıcıya bilgi sağlanmasını sağlamalıdır. Görsel arayüz, renk kodlaması ve bilgilendirme özellikleri kullanıcıya sıralama işlemini anlamada ve değerlendirmede yardımcı olmalıdır.

UYGULAMA İMPLEMENTASYONU

Kullanıcı Arayüzü Oluşturma ve Etkileşim:

Kullanıcı arayüzü, projenin temel bileşenlerini içeren ve kullanıcının projeyi etkileşimli bir şekilde kullanmasını sağlayan bir yapıdır. İşte kullanıcı arayüzü oluşturma ve etkileşim konusunda bazı adımlar:

1. Grafiksel Arayüz Tasarımı:
   - Kullanıcı arayüzü, grafiksel bir tasarımla oluşturulmalıdır. Bu tasarım, kullanıcıya projenin işlevlerini ve seçeneklerini net bir şekilde sunmalıdır.
   - Arayüzde, sol panel ve ana panel gibi bileşenlerin yer aldığı bir düzen oluşturulmalıdır.

- Bileşenler, kullanıcının seçimlerini yapabileceği ve etkileşimde bulunabileceği şekilde yerleştirilmelidir.

2. Sol Panel:
   - Sol panelde yer alacak bileşenler aracılığıyla kullanıcının bazı seçimler yapması sağlanmalıdır.
   - Boyut: Kullanıcı, sıralanacak listenin boyutunu manuel olarak girebilmeli veya rasgele bir liste oluşturmak için bir boyut belirleyebilmelidir.
   - Hız: Kullanıcı, animasyon hızını ölçeklendirebilmeli ve tercih ettiği hız seviyesini seçebilmelidir.
   - Sıralama Algoritmaları: Kullanıcı, farklı sıralama algoritmaları arasından birini seçebilmelidir.
   - Grafik Tipleri: Kullanıcı, farklı grafik tipleri arasından birini seçebilmeli ve görsel olarak sıralama işlemini takip edebilmelidir.
   - Oluştur-Başla-Dur-Sıfırla Butonları: Kullanıcı, "Oluştur" butonuna basarak seçtiği değerlere göre arayüzü oluşturmalı, "Başla" butonuna basarak sıralama işlemini başlatabilmeli, "Dur" butonuna basarak animasyonu durdurabilmeli ve "Sıfırla" butonuna basarak liste ve arayüzü sıfırlayabilmelidir.
3. Ana Panel:
   - Ana panel, sıralama işleminin görsel olarak takip edilebileceği ve karşılaştırma sayısının izlenebileceği bir alan olmalıdır.
   - Görsel Arayüz: Seçilen algoritma ve grafik tipine göre, ana panelde sıralama işlemi görsel olarak gerçekleştirilmeli ve kullanıcıya

```python
#Animasyon methodları
def draw_scatter_chart(data, chart_view):
    chart = QChart()
    chart.setTitle("Dağılım Grafiği")
    chart.legend().hide()

    series = QScatterSeries()
    for i, val in enumerate(data):
        series.append(i, val)

    chart.addSeries(series)

    axis_x = QValueAxis()
    axis_x.setLabelFormat("%d")
    axis_x.setTitleText("Index")

    axis_y = QValueAxis()
    axis_y.setLabelFormat("%d")
    axis_y.setTitleText("Value")

    chart.addAxis(axis_x, Qt.AlignBottom)
    chart.addAxis(axis_y, Qt.AlignLeft)
    series.attachAxis(axis_x)
    series.attachAxis(axis_y)
```

     sunulmalıdır.
   - Renk Kodlaması: Karşılaştırılan değerler, sıralanmış ve sıralanmamış değerler için farklı renkler kullanılarak vurgulanmalıdır.

Sıralama algoritmalarının uygulanması:

Kullanıcının seçtiği algoritmaya göre sıralama işleminin gerçekleştirilmesini içerir. İşte sıralama algoritmalarının uygulanmasıyla ilgili bazı adımlar:

```python
            #draw_scatter_chart(data, self.chart_view)

            if (graph_type == 'Dağılım(Scatter) Grafiği'):
                draw_scatter_chart(data, self.chart_view)
                print("a")
            elif (graph_type == 'Sütun (Bar) Grafiği'):
                draw_bar_chart(data, self.chart_view)
                print("b")
            elif(graph_type=='Kök (Sten) Grafiği'):
                draw_stem_chart(data,self.chart_view)
                print('c')

        except Exception as e:
            QMessageBox.critical(self, "Hata", f"Uygulamada bir hata oluştu: {str(e)}")
            print(e)
```

1. Seçilen Algoritmanın Belirlenmesi:
   - Kullanıcı tarafından seçilen sıralama algoritması belirlenir. Örneğin, kullanıcı "Seçme Sıralaması" algoritmasını seçmiş olabilir.
2. Algoritmanın Uygulanması:
   - Belirlenen algoritma, kullanıcının seçtiği liste üzerinde uygulanır.
   - Algoritmanın adımları takip edilerek, liste elemanları karşılaştırılır ve gerektiğinde yer değiştirilir.
   - Sıralama işlemi adım adım gerçekleştirilir ve her adımda arayüz güncellenerek kullanıcıya görsel olarak sunulur.
3. Karşılaştırma Sayısının İzlenmesi:
   - Seçilen algoritma göz önünde bulundurularak her bir karşılaştırma işlemi sayısı izlenir.
   - Karşılaştırma sayısı, her adımda güncellenerek kullanıcıya bilgi verilir.
   - Karşılaştırma sayısı, arayüzde görsel olarak takip edilebilir veya sayısal olarak görüntülenebilir.
4. Animasyonun Gerçekleştirilmesi:
   - Sıralama işlemi, belirli bir hızda gerçekleştirilebilir.
   - Animasyon, arayüzde sıralama adımlarının geçişlerini göstermek için kullanılabilir.
   - Her adımda, sıralama işleminin güncel durumu ve karşılaştırma sayısı kullanıcıya görsel olarak sunulur.
5. Sıralama İşleminin Tamamlanması:
   - Sıralama işlemi tamamlandığında, kullanıcı bilgilendirilir.
   - Karşılaştırma sayısı ve algoritmanın karmaşıklık analizi gibi ek bilgiler, kullanıcıya sunulabilir.
   - Sıralanmış liste, arayüzde farklı bir renkle vurgulanabilir veya ayrı bir bölümde gösterilebilir.

Bu adımlar, kullanıcının seçtiği sıralama algoritmasının gerçekleştirilmesi için temel bir rehber sağlar. Her bir sıralama algoritması, farklı bir mantıkla çalışır ve uygulama ayrıntıları algoritma türüne göre değişiklik gösterebilir. Bu nedenle, seçilen sıralama algoritmasının özel gereksinimlerini ve adımlarını takip etmek önemlidir.

Grafik animasyonlarının gerçekleştirilmesi:

Sıralama işleminin görsel olarak kullanıcıya sunulmasını sağlar. Bu animasyonlar, sıralama adımlarının grafiksel olarak değişimlerini yansıtan geçişlerle

oluşturulur. İşte grafik animasyonlarının gerçekleştirilmesi için bazı adımlar:

1. Grafik Türünün Belirlenmesi:
   - Kullanıcı tarafından seçilen grafik türü belirlenir. Örneğin, "Dağılım (Scatter) Grafiği" veya "Sütun (Bar) Grafiği" gibi.
2. Verilerin Görselleştirilmesi:
   - Kullanıcının seçtiği veri seti, belirlenen grafik türüne uygun olarak görselleştirilir.
   - Veriler, grafik bileşenleri kullanılarak ekranda temsil edilir.
   - Başlangıçta, sıralanmamış veriler görselleştirme üzerinde yer alır.
3. Sıralama Adımlarının Animasyonlaştırılması:
   - Sıralama algoritmasının her adımı sırasında, grafikteki verilerin değişimi animasyonla gösterilir.
   - Örneğin, elemanların yer değiştirmesi, renk değişimi veya hareketli geçişlerle ifade edilebilir.
   - Her adımda, grafik animasyonu güncellenir ve kullanıcıya görsel bir geçiş sunulur.
4. Animasyon Hızının Kontrol Edilmesi:
   - Kullanıcı, animasyon hızını kontrol etme yeteneğine sahip olmalıdır.
   - Hız ayarları, animasyonun yavaşlatılması veya hızlandırılması için kullanılabilir.
   - Animasyon hızı, kullanıcı arayüzünde bir kontrol bileşeniyle ayarlanabilir.

```
if __name__ == '__main__':
    app = QtWidgets.QApplication([])
    window = MainWindow()
    window.show()
    app.exec_()
```

5. Animasyonun Başlatılması, Durdurulması ve Sıfırlanması:
   - Kullanıcı, sıralama animasyonunu başlatmak, durdurmak veya sıfırlamak için uygun düğmelere sahip olmalıdır.
   - Başlat düğmesi, sıralama işleminin animasyonunu başlatır.
   - Duraklat düğmesi, animasyonu geçici olarak duraklatır ve tekrar basıldığında devam ettirir.
   - Sıfırla düğmesi, animasyonu sıfırlar ve başlangıç durumuna geri döner.

Bu adımlar, grafik animasyonlarının gerçekleştirilmesi için genel bir rehber sağlar. Animasyonlar, kullanıcıya sıralama işleminin görsel olarak takibini kolaylaştırır ve işlem adımlarını daha anlaşılır hale getirir. Kullanılan grafik kütüphanelerinin belirli işlevleri ve yöntemleri, animasyonların nasıl oluşturulacağı konusunda daha ayrıntılı bilgi sağlayabilir.

SONUÇLAR VE DEĞERLENDİRME

Proje sonucunda, kullanıcının sıralama algoritmalarını ve grafik türlerini interaktif bir şekilde deneyimleyebildiği bir arayüz oluşturulmuştur. Kullanıcı, sol paneldeki seçenekler aracılığıyla sıralanacak listenin boyutunu belirleyebilir, listenin kendisi tarafından oluşturulmasını sağlayabilir veya manuel olarak bir liste girebilir. Ayrıca, sıralama hızını ayarlayabilir ve istediği sıralama algoritmasını ve grafik türünü seçebilir.

Ana panelde, kullanıcının seçtiği algoritma ve grafik türüne göre sıralama işlemi gerçekleştirilir ve görsel bir arayüz sunulur. Karşılaştırma sayısı izlenir ve her adımda yapılan karşılaştırmalar arka planda tutulur. Animasyon sırasında renk kodlaması kullanılarak hangi değerlerin karşılaştırıldığı ve sıralanıp sıralanmadığı belirgin bir şekilde gösterilir. Ayrıca, belirli periyotlarla karşılaştırma sayısı güncellenerek kullanıcıya bilgi verilir.

Proje, kullanıcının sıralama algoritmalarını görsel olarak anlamasını ve karşılaştırma sayıları üzerindeki etkilerini izlemesini sağlayarak eğitici bir deneyim sunar. Kullanıcılar, farklı algoritmaları ve grafik türlerini deneyerek sıralama işlemlerini daha iyi anlayabilir ve bu bilgileri gerçek dünya uygulamalarına aktarabilirler.

Geliştirme sürecinde, kullanıcı arayüzü tasarımı, sıralama algoritmalarının uygulanması ve grafik animasyonlarının gerçekleştirilmesi gibi teknik zorluklarla karşılaşılmış olabilir. Ancak, projenin başarıyla tamamlanması ve kullanıcıların interaktif bir şekilde sıralama işlemlerini deneyimleyebilmeleri önemli bir başarıdır.

Projenin geliştirilmesi sırasında ortaya çıkan zorluklar ve sınırlamalar olabilir. Örneğin, bazı algoritmalarda zaman ve hafıza karmaşıklığı yüksek olabilir, bu nedenle çok büyük boyutlu listelerde performans sorunları yaşanabilir. Ayrıca, kullanıcı arayüzü tasarımında daha fazla etkileşimli özellikler eklemek veya farklı grafik animasyonlarını desteklemek projenin geliştirilebileceği alanlardır.

Sonuç olarak, proje kullanıcılara sıralama algoritmalarını görsel olarak deneyimleme ve anlama imkanı sunan bir arayüz sağlamıştır. Kullanıcılar, bu arayüz aracılığıyla sıralama algoritmalarının işleyişini görsel olarak izleyebilir, karşılaştırma sayılarını takip edebilir ve farklı grafik türlerini kullanarak sonuçları görselleştirebilirler. Proje, sıralama algoritmaları konusunda kullanıcıların anlayışını artırarak eğitici bir deneyim sunmaktadır.

GELİŞTİRME ÖNERİLERİ

Projenin geliştirme aşamasında aşağıdaki önerileri göz önünde bulundurabilirsiniz:

1. İşlevsellik Geliştirme:
   - Mevcut sıralama algoritmalarının yanı sıra daha fazla algoritma seçeneği ekleyebilirsiniz.
   - Özel sıralama algoritmalarını kullanıcının tanımlamasına izin vererek esneklik sağlayabilirsiniz.
   - Grafik türleri listesini genişletebilir veya kullanıcının kendi grafik türlerini oluşturmasına olanak tanıyabilirsiniz.
   - Kullanıcının grafikleri özelleştirmesine imkan sağlayacak ek ayarlar ekleyebilirsiniz.
2. Kullanıcı Deneyimi İyileştirmeleri:
   - Kullanıcı arayüzünü daha kullanıcı dostu ve etkileşimli hale getirebilirsiniz.
   - İşlem adımlarını ve karşılaştırma sayılarını daha açık bir şekilde göstermek için bilgilendirici metinler veya animasyonlar ekleyebilirsiniz.
   - Animasyon hızı kontrolünü daha sezgisel hale getirmek için kaydırma çubukları veya düğmeler kullanabilirsiniz.
   - Kullanıcıya sıralama algoritmasının karmaşıklık analizi hakkında daha fazla bilgi sunabilirsiniz.
3. Performans İyileştirmeleri:
   - Büyük veri setleri üzerinde daha iyi performans elde etmek için sıralama algoritmalarını optimize edebilirsiniz.
   - Grafik animasyonlarını daha akıcı ve hızlı hale getirmek için grafik kütüphanelerinin özelliklerinden yararlanabilirsiniz.
   - Bellek yönetimini ve veri işleme sürelerini optimize etmek için geliştirme tekniklerini kullanabilirsiniz.
4. Hata Yakalama ve İyileştirme:
   - Kullanıcının hatalı veya geçersiz girişler yapmasını önlemek için doğrulama mekanizmaları ekleyebilirsiniz.
   - Uygulamanın istikrarını artırmak için hata yakalama ve hata düzeltme mekanizmalarını güçlendirebilirsiniz.
   - Kullanıcı geri bildirimlerini dikkate alarak iyileştirmeler yapabilir ve uygulamanın kullanılabilirliğini artırabilirsiniz.
5. Dokümantasyon ve Kullanım Kılavuzu:
   - Projenin kodunu ve işleyişini açıklayan detaylı bir dokümantasyon oluşturabilirsiniz.
   - Kullanıcılar için kapsamlı bir kullanım kılavuzu veya yardım bölümü hazırlayabilirsiniz.
   - Kullanıcıların projeyi kolayca anlamalarını ve kullanmalarını sağlamak için örnekler ve talimatlar ekleyebilirsiniz.
   - Bu öneriler, projenin geliştirme aşamasında daha iyi bir kullanıcı deneyimi sağlamak ve işlevselliği artırmak için faydalı olabilir.

KAYNAKLAR VE REFERANSLAR

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. MIT Press.

Sedgewick, R., & Wayne, K. (2011). Algorithms. Pearson Education.

Lafore, R. (2002). Data Structures and Algorithms in Java. Sams Publishing.

GeeksforGeeks. (2023). Sorting Algorithms. Erişim adresi: https://www.geeksforgeeks.org/sorting-algorithms/

Python.org. (2023). Python Documentation. Erişim adresi: https://docs.python.org/

W3Schools. (2023). Python Tutorial. Erişim adresi: https://www.w3schools.com/python/

Stack Overflow. (2023). Python Questions and Answers. Erişim adresi: https://stackoverflow.com/

Data Structures and Algorithms in Python - Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser.

Introduction to the Design and Analysis of Algorithms - Anany Levitin.

Python Data Structures and Algorithms - Benjamin Baka.

Real Python. (2023). Python Sorting Algorithms: A Comprehensive Guide. Erişim adresi: https://realpython.com/sorting-algorithms-python/

DataCamp. (2023). Sorting Algorithms in Python. Erişim adresi: https://www.datacamp.com/community/tutorials/sorting-algorithms-python

Tutorialspoint. (2023). Python - Data Structures. Erişim adresi: https://www.tutorialspoint.com/python_data_structure/index.htm

Tutsplus. (2023). Data Structures and Algorithms in Python. Erişim adresi: https://code.tutsplus.com/tutorials/data-structures-and-algorithms-in-python--cms-25617

Towards Data Science. (2023). Data Structures and Algorithms in Python: A Practical Guide. Erişim adresi: https://towardsdatascience.com/data-structures-and-algorithms-in-python-a-practical-guide-28c7e79f924b