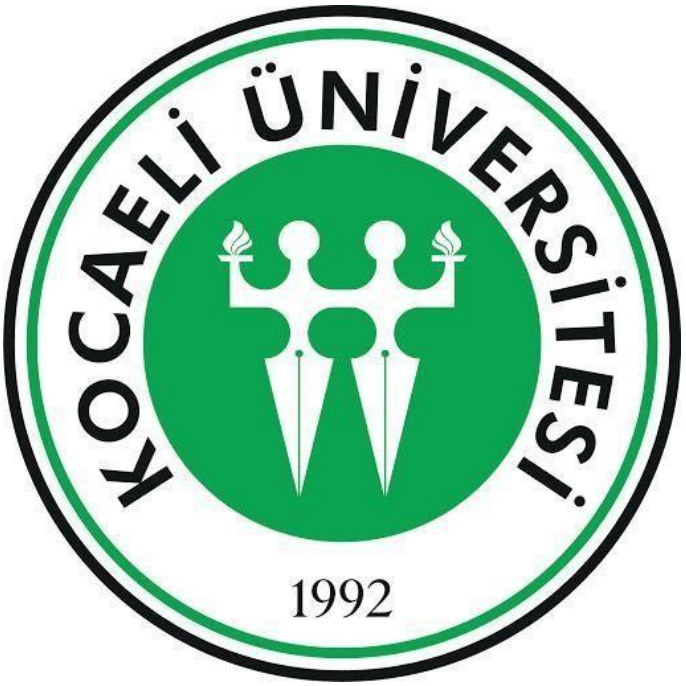


T.C. KOCAELİ ÜNİVERSİTESİ

YAZILIM GELİŞTİRME LABORATUVARI DERSİ-II

PROJE RAPORU

SIRALAMA ALGORİTMALARI GÖRSELLEŞTİRİCİSİ

201307034	201307058	201307057
Alparslan BAKIR	Sude Nur ELMAS	Abdullah İŞLER

KOCAELİ ÜNİVERSİTESİ

SIRALAMA ALGORİTMALARI GÖRSELLEŞTİRİCİSİ

Alparslan BAKIR, Sude Nur ELMAS, Abdullah İŞLER
201307034@kocaeli.edu.tr
201307058@kocaeli.edu.tr
201307057@kocaeli.edu.tr

Bilişim Sistemleri Mühendisliği – Teknoloji Fakültesi
Kocaeli Üniversitesi

Özet

Bu proje sıralama algoritmalarının çalışma şeklini gözlemlemek ve göstermek üzere geliştirilmiştir. Kullanıcıların belirledikleri boyuttaki bir dizi kullanıcın seçtiği grafik şekli ve sıralama algoritma çeşidi ile kullanıcıya gösterilir. Yine kullanıcı tarafından belirlenen hızda çalışmaya başlayan uygulama algoritmaların çalışma anını kullanıcıya arayüz (GUI) aracılığı ile gösterilir. Aynı zamanda manuel olarak dizi girmesine olanak tanır. Farklı sıralama algoritma çeşitleri ve farklı grafik çeşitleri ile kullanıcının işleyişi ve farklılıkları kavramasına yardımcı olmayı hedefler.

Abstract

This project has been developed to observe and demonstrate the functioning of sorting algorithms. A user-specified array of a chosen size is presented to the user with their selected graph type and sorting algorithm. The application, which starts working at a user-defined speed, shows the execution of algorithms to the user through a graphical user interface (GUI). Additionally, it allows manual input of arrays. The project aims to assist users in understanding the operations and differences through various sorting algorithm types and different graph types.

1. GİRİŞ

Projenin amacı farklı sıralama algoritmalarının çalışma şeklini GUI aracılıyla kullanıcılara göstermektir. Sıralama algoritması, bilgisayar bilimlerinde ya da matematikte kullanılan, herhangi bir sayıdaki tip verilerin sınırlı bellek ve işlem gücü ile belirli bir sıraya göre dizilmesinin sağlanmasıdır. Önemli olan en optimum bellek ve performans ikilisini verecek bir algoritmanın elde edilmesidir. Bu algoritmaların geliştirilmesindeki temel amaç büyük miktardaki verileri insanlar tarafından rahatça anlaşılabilmesi için olabildiğince hızlı bir şekilde sıralamaktır. Bu projede popüler olan Selection Sort, Bubble Sort, Insertion Sort, Merge Sort, Quick Sort algoritmları gösterilmiştir. Bu algoritmaların gösterimi için de dağılım, sütun ve kök grafikleri seçenekleri sunulmuştur.

Kullanıcı istediği sıralama algoritmasını ve gösterilmesini istediği grafik şeklini seçer. Daha sonra kullanıcının seçtiği dizi boyutunda isteğe bağlı olarak rastgele belirlenmiş bir liste ve ya manuel olarak girilmiş bir liste hazırlanır. “Play” tuşuna basıldığı anda arayüz (GUI) aracılığıyla belirlenmiş olan hız ile algoritma çalışmaya başlar. Bu kısımda renk farklılıkları ile yerleştirilmiş ve karşılaştırılmaya devam eden değerler birbirlerinden ayrılırlar. Böylece kullanıcının takip etmesinin kolaylaşması hedeflenir. Aynı zamanda karşılaştırma sayısı gibi bilgiler de GUI’de yer alır.

2. KAPSAM

Bu projede, sıralama algoritmalarının görselleştirilmesini amaçlayan bir yazılım uygulaması gerçekleştirilmiştir. Uygulama, kullanıcılara çeşitli sıralama algoritmalarını interaktif bir şekilde keşfetme ve anlama imkanı sağlamaktadır. Uygulama arayüzünde, kullanıcılar farklı sıralama algoritmaları arasından seçim yapabilir ve bir veri kümesini belirleyebilir. Seçilen algoritma ve veri kümesi üzerinde gerçekleştirilen sıralama işlemi, adım adım görselleştirilmektedir. Her adımda, verilerin durumu grafikler üzerinde animasyonlarla kullanıcıya aktarılmaktadır. Projede kullanılan araçlar, yöntemler ve çalışma aşamaları devam eden başlıklarda anlatılmıştır.

2.1 Kullanılan Teknolojiler

Projede Python dili ve aynı zamanda Python dilinin çeşitli kütüphanelerinden biri olan Tkinter kütüphanesi kullanıldı. Masaüstü uygulamasına çevirmek için AutoPyToExe aracı kullanıldı. Geliştirme ortamı olarak Visual Studio Code uygulaması tercih edildi. Github ile kodların paylaşılması ve kodların takibi gerçekleştirildi.

2.1.1 Python (3.11.3)

Python, birçok amaç için kullanılabilen, dinamik bir söz dizime sahip, yaygın olarak kullanılan, nesne yönelimli ve üst düzey bir programlama dilidir. Dil, okunabilir ve gerçek İngilizce'ye yakın olacak şekilde tasarlanmıştır. Veri görselleştirme, veri analitiği, makine öğrenimi, uygulama geliştirme, güvenlik uygulamaları gibi bir çok alanda yaygın olarak tercih edilir. Diğer birçok dile kıyasla çok daha az kod satırıyla istenilen sonuçlar elde edilebilir.

2.1.2 Tkinter

Tkinter Python diline ait bir kütüphanedir. Python için arayüz geliştirme, düzenleme işlemlerini gerçekleştirir. Aslında basit GUI uygulamaları ve algoritmalar geliştimek için kullanılır. Ancak projede grafik arayüz programı geliştirmek için yeterli olduğundan bu kütüphane tercih edilmiştir.

2.1.3 AutoPyToExe

AutoPyToExe, Windows dosyalarına dönüştürmek için kullanılan ücretsiz bir araçtır. Python betiğinizi derleyerek tek bir yürütülebilir (.exe) dosya oluşturur. Bu, Python kaynak kodunun kullanıcının sistemindeki Python yorumlayıcısı olmadan çalıştırılabilmesini sağlar. Oluşturulan .exe dosyası, Python yorumlayıcısını içerir ve kullanıcının Python ortamını bilmesine veya kurmasına gerek kalmadan Python betiğini çalıştırabilir.

2.1.4 Github

GitHub, özellikle açık kaynak yazılım projeleri için önemli bir platformdur. Projeler, kullanıcılar tarafından oluşturulan "repository" adı verilen merkezi bir yerde saklanır ve kullanıcılar, bu depolardaki kaynak kodlarını düzenleyebilir, çeşitli değişiklikler yapabilir ve bu değişiklikleri "commit" adı verilen bir işlemle kaydedebilirler. Bu projede kodların yayımlanması ve paylaşılması adına Github uygulaması kullanıldı. Böylece yapılan değişiklikler bu platform üzerinden paylaşılıp kaydedildi. Projenin Github linki aşağıda verilmiştir.
Github Linki: https://github.com/AlparslanBakir/Sorting-Algorithms-Visualization

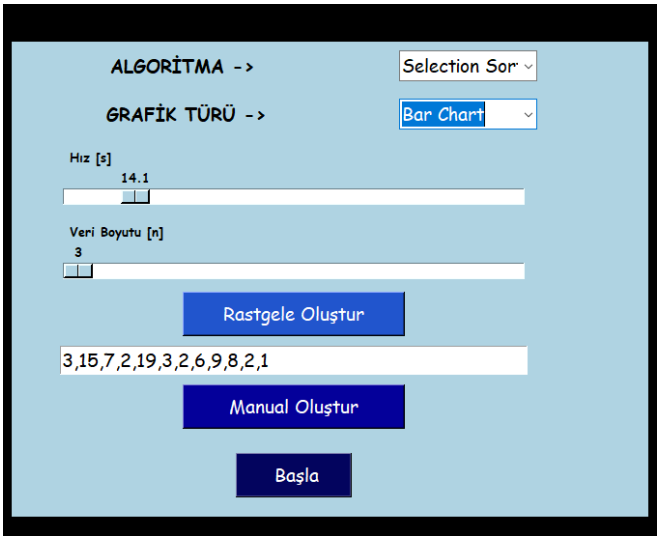
2.1.5 VS Code

Visual Studio Code, Microsoft tarafından Windows, Linux ve MacOS için geliştirilen bir kaynak kodu düzenleyicisidir. Hata ayıklama, gömülü Git kontrolü, sözdizimi vurgulama, akıllı kod tamamlama, snippet'ler ve kod yeniden yapılandırma desteği içerir. Projede bu kod düzenleme aracı tercih edildi.

2.2 Arayüz ve Kullanım

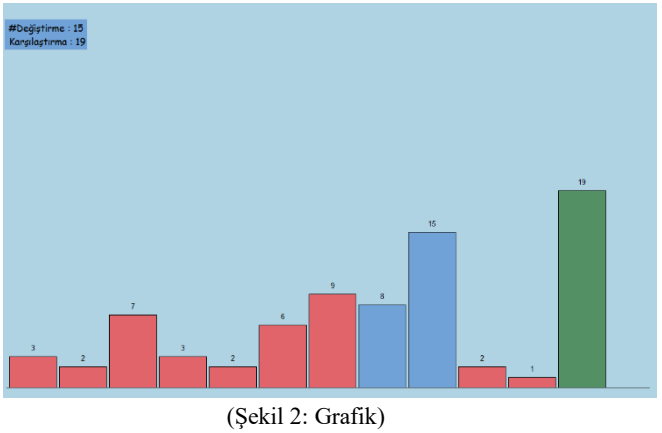
Arayüz üzerinde kullanıcının sıralama algoritması, grafik türü ve hız gibi seçenekleri belirlemesini sağlar. Ayrıca kullanıcıya rastgele veya manuel olarak bir dizi oluşturma seçeneği sunar. Ardından seçilen algoritmayı ve grafik türünü kullanarak verileri görsel olarak sıralar.

Uygulamanın sol panelinde ilk olarak kullanıcının bir sıralama algoritması ve grafik türü seçmesi için alanlar bulunur. Hız çubuğu animasyonun hızını belirlemek için kullanılır. Veri boyutu çubuğu ile kullanıcı 3 ile 100 arasında bir boyuta sahip rastgele bir dizi oluşturabilir. Veri giriş alanına ise manuel olarak değerler girilip dizi oluşturulabilir. İlgili generate butonuna basıldığında diziye ait grafik ekranda gösterilir. Play tuşunda basıldığında ise animasyon çalışır. Burada bahsedilen elementler Şekil 1'de gösterilmiştir.



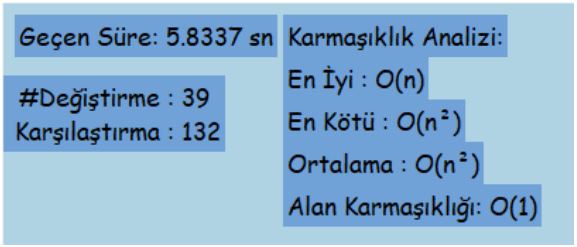
(Şekil 1: Girdi Paneli)

Grafik ve animasyon Şekil 2'deki gibi ana ekranda gösterilir. Algoritma adım adım ilerler ve algoritmadaki verinin değiştirilme durumuna göre anlık olarak belirli bir renk alır. Örneğin bubble sort algoritması çalışırken henüz sıralanmamış değerler kırmızı, karşılaştırılan değerler mavi ve sıralanan değerler yeşil renk alır.



(Şekil 2: Grafik)

Ana ekranın sol üst kısmında, animasyon bittiğinde algoritmanın çalışması sırasında geçen karşılaştırma ve değiştirme sayısı, geçen süre ve karmaşıklık analizi bilgileri Şekil 3'teki gibi gösterilmektedir.



(Şekil 3: Değerler)

2.3 Algoritmalar [1] [2]

Bu uygulamada aşağıda tanımları ve özellikleri verilen algoritmalar gözlenebilir:

Bubble Sort:
Çalışma Prensibi: Komşu elemanlar karşılaştırılarak gerektiğinde yer değiştirilir, böylece büyük elemanlar yavaş yavaş dizinin sonuna doğru kayar.
Karmaşıklık Analizi: En kötü durumda $O(n^2)$, ortalama ve en iyi durumlarda $O(n^2)$.
Avantajları: Basit ve anlaşılır bir algoritmadır, ekstra bellek gerektirmez.
Dezavantajları: Büyük veri kümelerinde yavaş çalışır, veri sıralıysa bile tüm elemanları kontrol eder.

Selection Sort:
Çalışma Prensibi: Minimum elemanı bulup dizinin başına yerleştirir, ardından kalan alt dizide bu işlemi tekrar ederek sıralama yapar.
Karmaşıklık Analizi: En kötü durumda $O(n^2)$, ortalama ve en iyi durumlarda O(n^2).
Avantajları: Basit bir algoritmadır, yer değiştirme işlemi azdır.
Dezavantajları: Büyük veri kümelerinde yavaş çalışır, tekrar eden elemanlar için uygun değildir.

Insertion Sort:
Çalışma Prensibi: Dizi, sıralı ve sıralanmamış bölgelere ayrılır. Sıralanmamış bölgedeki her eleman sıralı bölgeye yerleştirilir.
Karmaşıklık Analizi: En kötü durumda $O(n^2)$, ortalama ve en iyi durumlarda O(n).
Avantajları: Küçük veri kümelerinde etkili çalışır, daha az yer değiştirme yapar.
Dezavantajları: Büyük veri kümelerinde yavaş çalışır, verinin neredeyse sıralı olması durumunda bile tüm elemanları kontrol eder.

Merge Sort:
Çalışma Prensibi: Dizi sürekli olarak ikiye bölünür, ardından ayrı ayrı sıralanan alt diziler birleştirilerek sıralı bir dizi oluşturulur.
Karmaşıklık Analizi: En kötü durumda O(n log n), ortalama ve en iyi durumlarda da aynı performansa sahiptir.
Avantajları: Büyük veri kümeleri için etkili bir sıralama algoritmasıdır.
Dezavantajları: Ek bellek kullanımına ihtiyaç duyabilir.

Quick Sort:
Çalışma Prensibi: Bir eleman pivot olarak seçilir ve diğer elemanlar pivotun solunda veya sağındaki bölgeye yerleştirilir. Ardından sol ve sağ bölgeler için aynı işlem tekrarlanarak sıralama gerçekleştirilir.
Karmaşıklık Analizi: En kötü durumda $O(n^2)$, ortalama ve en iyi durumlarda O(n log n).
Avantajları: Genellikle hızlı bir sıralama algoritmasıdır, ek bellek kullanımına ihtiyaç duymaz.
Dezavantajları: En kötü durumda performansı düşük olabilir, kararlı bir sıralama algoritması değildir.

Heap Sort:
Çalışma Prensibi: Dizi, bir binary heap veri yapısı kullanılarak sıralanır. İlk adımda dizi bir maksimum heap veya minimum heap olarak düzenlenir. Ardından en büyük veya en küçük eleman kök olarak alınarak diziye yerleştirilir ve kök düğümü yeniden düzenlenerek heap özelliği korunur. Bu adımlar tekrarlanarak tüm elemanlar sıralanır.
Karmaşıklık Analizi: En kötü durumda, ortalama ve en iyi durumlarda O(n log n) karmaşıklıkla çalışır.
Avantajları: Stabil bir sıralama algoritmasıdır. Yani aynı değere sahip elemanların sıralanmış hali, orijinal girişin sıralanmış haliyle aynı sıra düzenini korur. Ayrıca, ek bellek kullanımına ihtiyaç duymaz.
Dezavantajları: Diğer bazı sıralama algoritmalarına göre daha yavaş olabilir. Heap yapısının oluşturulması ve elemanların yer değiştirmesi işlemleri nedeniyle ek işlem süresi gerektirebilir.

2.4 Karmaşıklık Analizi

Karmaşıklık analizi, bir algoritmanın çalışma süresini ve kaynak kullanımını değerlendirmek için kullanılan bir yöntemdir. Sıralama algoritmaları da karmaşıklık analizi ile değerlendirilebilir. Bu analiz, bir algoritmanın zaman ve bellek açısından ne kadar verimli olduğunu belirlemek için önemlidir.

Karmaşıklık analizinde en yaygın kullanılan ölçüt, sıralama algoritmasının zaman karmaşıklığıdır. Zaman karmaşıklığı, algoritmanın çalışma süresinin veri boyutuna bağlı olarak nasıl değiştiğini ifade eder. Genellikle, sıralama algoritmalarının en kötü durum zaman karmaşıklığı dikkate alınır, çünkü bu durumda algoritma en yavaş şekilde çalışır.

Sıralama algoritmalarının zaman karmaşıklığı büyük O (Big O) gösterimiyle ifade edilir. Örneğin, $O(n^2)$, O(n log n) gibi gösterimler sıkça kullanılır. Burada "n", veri kümesinin boyutunu temsil eder. [3] [4]

3. ÖNERİLER

Ek Sıralama Algoritmaları: Projenin çeşitliliğini artırmak için farklı sıralama algoritmalarının eklenmesi önerilebilir. Örneğin, Counting Sort veya Radix Sort gibi daha özel ama etkili algoritmaların dahil edilmesi kullanıcılara daha geniş bir yelpaze sunabilir.

Görsel İyileştirmeler: Kullanıcı deneyimini artırmak ve etkileşimi daha ilgi çekici hale getirmek için görsel iyileştirmelere odaklanılabilir. Örneğin, renk geçişleri veya animasyonlar gibi öğelerin eklenmesi kullanıcıların ilgisini daha fazla çekebilir.

Öğretici Modu: Kullanıcıların sıralama algoritmalarını daha iyi anlamalarına yardımcı olmak için öğretici bir mod ekleyebilirsiniz. Bu modda, kullanıcılara adım adım açıklamalar ve rehberlik sunarak sıralama algoritmalarının nasıl çalıştığını daha iyi anlamalarını sağlayabilirsiniz.

4. SONUÇ

Bu proje, kullanıcıların sıralama algoritmalarını görsel olarak keşfetmelerine ve anlamalarına olanak tanımaktadır. Kullanıcılar, algoritmaların çalışma sürelerini, karşılaştırma/değiştirme sayılarını ve performanslarını gözlemleyerek algoritmalara daha iyi bir bakış açısı kazanabilirler. Ayrıca, farklı veri setleri ve grafik tipleri kullanarak gerçek dünya senaryolarında sıralama algoritmalarını test etme imkanı sunulmaktadır.

6. KAYNAKLAR

[1] A. Arora, «Sorting Algorithms-Properties/Pros/Cons/Comparisons,» medium.com, 18 Dec 2019. [Çevrimiçi]. Available: https://ayush-arora.medium.com/sorting-algorithms-properties-pros-cons-comparisons-1b43599d587b. [Erişildi: 26 May 2023].

[2] J. Wandy, «The Advantages & Disadvantages of Sorting Algorithms,» sciencing.com, 27 Jun 2018. [Çevrimiçi]. Available: https://sciencing.com/the-advantages-disadvantages-of-sorting-algorithms-12749529.html. [Erişildi: 26 May 2023].

[3] Ş. E. Şeker, «youtube.com,» BilgisayarKavramlari, 2 Kasım 2016. [Çevrimiçi]. Available: https://www.youtube.com/watch?v=XQqjUSOi45o&t=3s. [Erişildi: 26 Mayıs 2023].

[4] C. Ninjas, «youtube.com,» Coding Ninjas, 2 Ağustos 2022. [Çevrimiçi]. Available: https://www.youtube.com/watch?v=Q-xyF-aEKXc&t=5s. [Erişildi: 26 Mayıs 2023].