Sıralama Algoritmaları Uygulaması

Atayağız Uslu
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
191307062
191307062@kocaeli.edu.tr

Ömer Faruk Sevinç
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
191307007
191307007@kocaeli.edu.tr

Hüseyin Nas
Kocaeli Üniversitesi
Bilişim Sistemleri Mühendisliği
191307043
191307043@kocaeli.edu.tr

***Özet*—Bu rapor, Python programlama dili kullanılarak geliştirilen algoritma sıralama uygulamasının özetini içermektedir. Uygulama, çeşitli sıralama algoritmalarını uygulayarak verilerinizi sıralamanıza olanak tanır.**

Anahtar sözcükler—sıralama, algoritma, python, bubble sort, selection sort, insertion sort, merge sort, quick sort.

I. GIRIŞ

Python programlama dili kullanılarak geliştirilen ve çeşitli sıralama algoritmalarını içeren bir uygulamanın sunumunu içermektedir. Uygulama, kullanıcıların verilerini hızlı ve etkili bir şekilde sıralamalarına olanak sağlamak amacıyla tasarlanmıştır. Sıralama algoritmaları, bilgisayar biliminde önemli bir rol oynar ve bir dizi verinin düzenlenmesinde yaygın olarak kullanılır.

Uygulamamız, kullanıcıların farklı sıralama algoritmalarını kullanarak verilerini sıralama seçeneklerini keşfetmelerine olanak tanır. Sıralama algoritmaları, verilerin belirli bir düzene göre yeniden düzenlenmesini sağlar ve bu şekilde verilerin erişimi, analizi veya başka işlemler için daha kolay hale getirir. Uygulamamızda, kullanıcılara beş temel sıralama algoritması, animasyon hızı ve hamle sayısı bilgileri sunulmuştur. Böylece kullanıcının ve projenin bütün isterlerini karşılayan bir uygulama olmuştur.

II. SIRALAMA ALGORITMALARI

A. Bubble Sort(Kabarcık Sıralama)

Kabarcık sıralama, bir diziyi küçükten büyüğe (veya büyükten küçüğe) doğru sıralamak için kullanılan basit bir algoritmadır. Adını, her iterasyonda en büyük (veya en küçük) elemanın "kabarcık gibi" yükselmesinden alır.Kabarcık sıralama algoritması basit ve anlaşılır bir yapıya sahiptir. Ancak, büyük veri setlerinde veya veri sıralamasının zaten neredeyse sıralı olduğu durumlarda verimlilik açısından dezavantajlı olabilir. En kötü durumda, yani ters sıralı bir dizide, kabarcık sıralama algoritmasının zaman karmaşıklığı O(n^2) olur.

B. Selection Sort(Seçmeli Sıralama)

Seçmeli sıralama, bir dizideki en küçük (veya en büyük) elemanı bulup ilk konuma yerleştirerek çalışır. Ardından, ikinci en küçük (veya en büyük) elemanı bulup ikinci konuma yerleştirir ve bu işlemi dizinin tamamı sıralanana kadar tekrar eder. Seçmeli sıralama, her geçişte en küçük (veya en büyük) elemanı bulmak için tüm diziyi dolaşması nedeniyle performans açısından etkili değildir. Ortalama ve en kötü durumda zaman karmaşıklığı O(n^2)'dir, burada n dizinin eleman sayısını temsil eder.

C. Insertion Sort(Ekleme Sıralama)

- Ekleme Sıralama algoritması, bir dizi içindeki elemanları sıralanmış ve sırasız bölgelere ayırır. Başlangıçta, sıralanmış bölge boştur ve sadece ilk elemanı içerir. Sonraki adımlarda, sırası gelen eleman sıralanmış bölgedeki elemanlarla karşılaştırılır ve doğru konumuna yerleştirilir.

- Ekleme Sıralama algoritması, bir elemanın doğru konumunu bulmak için sıralanmış bölgedeki elemanları sağa doğru kaydırmak zorunda olabilir. Bu nedenle, en kötü durumda zaman karmaşıklığı O(n^2) olur. Ancak, en iyi durumda, yani sıralanmış bir dizide zaman karmaşıklığı O(n) olabilir.

D. Quick Sort(Hızlı Sıralama)

Hızlı Sıralama, böl ve fethet yaklaşımını izleyen oldukça verimli bir sıralama algoritmasıdır. Ortalama durum zaman karmaşıklığı O(n log n)'dir. Hızlı Sıralamanın verimliliği büyük ölçüde pivot seçimine bağlıdır. İdeal senaryo, pivotun diziyi kabaca eşit yarılara böldüğü zamandır. Bununla birlikte, pivot, diziyi sürekli olarak oldukça dengesiz alt dizilere bölerse, performans en kötü durum zaman karmaşıklığı olan O(n^2)'ye düşebilir. Genel olarak Hızlı Sıralama, ortalama vaka süresi karmaşıklığı ve uygulamadaki iyi performansı nedeniyle yaygın olarak kullanılmaktadır.

E. Merge Sort(Birleştirme Sıralaması)

- Merge Sort, böl ve fethet yaklaşımını izleyen popüler bir sıralama algoritmasıdır. Verimliliği ve kararlılığı ile bilinir. Merge Sort, giriş dizisini daha küçük alt dizilere böler, bunları ayrı ayrı sıralar ve ardından son sıralanmış diziyi elde etmek için bunları birleştirir.Merge Sort, her durumda O(n log n) zaman karmaşıklığına sahiptir; burada n, giriş dizisindeki öğelerin sayısını temsil eder. Bu, Birleştirme Sıralamasını büyük girdi boyutları için verimli hale getirir. Ek olarak, Merge Sort kararlı bir sıralama algoritmasıdır, yani eşit değerlere sahip öğelerin göreli sırasını korur.
- Merge Sort iyi performans özelliklerine sahip olsa da, sıralama işlemi sırasında alt dizileri depolamak için ek bellek alanı gerektirir. Bu, çok büyük dizilerle uğraşırken veya sınırlı bellek kaynaklarına sahip durumlarda dikkate alınabilir.

III. KULLANILAN TEKNOLOJILER

A. *PyCharm:*

PyCharm, JetBrains tarafından geliştirilen bir entegre geliştirme ortamı (IDE) olarak bilinen bir yazılımdır. Özellikle Python programlama dili için tasarlanmış olan PyCharm, geliştiricilere kod yazma, hata ayıklama, test etme, sürüm kontrolü gibi birçok işlemi kolaylaştıran kapsamlı bir geliştirme ortamı sunar.

PyCharm, kullanıcı dostu arayüzü ve zengin özellikleriyle geliştiricilerin verimli bir şekilde çalışmalarını sağlar. Kod tamamlama, otomatik düzeltme, semantik analiz gibi özellikler sayesinde hızlı ve hatasız kod yazma imkanı sunar. Ayrıca, projelerin yönetimi için gelişmiş proje yapısı, paket yönetimi araçları ve entegre bir terminal gibi özellikler sunar.

B. *Github*

GitHub, geliştiricilerin projelerini paylaşabilecekleri, işbirliği yapabilecekleri ve sürdürülebilir yazılım geliştirme süreçlerini destekleyen bir bulut tabanlı platformdur. Yazılım geliştirme projeleri için sürüm kontrolü, iş takibi, kod incelemesi ve işbirliği gibi özellikleri sunan bir hizmettir.

GitHub, dağıtılmış sürüm kontrol sistemi olan Git'i temel alır ve bu sayede projelerinize kolayca sürüm kontrolü uygulayabilirsiniz. Git, kodunuzun tüm geçmişini izler ve değişiklikleri kaydeder, böylece projenizin herhangi bir noktasına geri dönebilir ve farklı sürümleri kolayca yönetebilirsiniz.

C. *PyGame*

Pygame, Python programlama dilinde başlıca oyun geliştirmek için kullanılan bir kütüphanedir. Oyunların grafik, ses, giriş kontrolleri ve diğer özelliklerini yönetmek için geliştirilmiştir. Pygame, kullanıcı dostu bir API (Application Programming Interface) sağlayarak oyun geliştirme sürecini kolaylaştırır.

Pygame, sadece oyun geliştirmek için değil, aynı zamanda Python dilini kullanarak konsol uygulamaları geliştirmek için de kullanılabilir. Konsol uygulamalarında, Pygame kullanarak kullanıcı dostu arayüzler oluşturabilir, metin tabanlı grafikleri görüntüleyebilir ve kullanıcı girişlerini yakalayabilirsiniz.

Pygame, konsol uygulamalarında metin tabanlı grafikler oluşturmak için kullanılan karakter tabanlı bir yaklaşım sunar. Örneğin, konsol penceresinde şekiller, çizgiler, renkli metinler gibi görsel öğeleri temsil etmek için karakterleri kullanabilirsiniz. Bu, konsol uygulamanızın daha etkileşimli ve görsel olarak daha çekici olmasını sağlar.

IV. GÖREV PAYLAŞIMI

Bu proje ödevi için üç kişilik bir ekip olarak çalıştık. Her birimiz farklı sorumluluklar üstlendik ve projenin başarıyla tamamlanmasını sağladık. Aşağıda, her bir ekibimizin görevlerini ve katkılarını belirtiyoruz:

1. Atayağız Uslu:
- Sıralama algoritmalarının Python dilinde kodlanmasını gerçekleştirdi.
- Proje yönetimi ve koordinasyonunu üstlendi.
- Kullanıcı gereksinimlerini analiz etti ve bunları ekibe iletti.
- Sıralama algoritmalarının araştırılması ve seçimi için kaynakları inceledi.
- Kodun genel yapısını ve uygulamanın temel işleyişini belirledi.

2. Ömer Faruk Sevinç:

- Sıralama algoritmalarının Python dilinde kodlanmasını gerçekleştirdi.
- Kabarcık sıralama, seçim sıralaması, ekleme sıralaması ve birleştirme sıralaması gibi temel algoritmaları uyguladı.
- Algoritmaların doğruluğunu test etmek için gerekli test senaryolarını oluşturdu.
- Performans analizi için zaman ve adım sayısı gibi metrikleri hesapladı.

3. Hüsyin Nas:

- Sıralama algoritmalarının Python dilinde kodlanmasını gerçekleştirdi.
- Kullanıcı arayüzünün tasarımını ve geliştirmesini yaptı.
  Kullanıcının veri girişi yapabileceği bir form oluşturdu.
- Sıralama sonuçlarını kullanıcıya gösteren çıktıyı oluşturdu.
- Uygulamanın kullanıcı dostu ve hata toleranslı bir şekilde çalışmasını sağladı.

Bu görev paylaşımı sayesinde, proje ekibimiz işbirliği içinde çalışarak Python ile sıralama algoritmalarını içeren uygulamamızı başarıyla tamamladık. Her birimizin farklı yetenekleri ve katkıları, proje sürecinde etkili bir şekilde kullanıldı. İşbirliği içinde çalışmanın önemi vurgulandı ve projenin amacına uygun olarak tamamlanmasına yardımcı oldu.

V. UYGULAMA GELIŞTIRME AŞAMALARI

A. Gereksinim Analizi ve Tasarım:

İlk olarak, proje ekibi olarak kullanıcı gereksinimlerini anlamak için bir gereksinim analizi gerçekleştirdik. Kullanıcıların verilerini sıralamak için hangi sıralama algoritmalarını kullanmak istediklerini ve kullanıcı arayüzünün nasıl olması gerektiğini belirledik. Ardından, uygulamanın tasarımını oluşturduk, veri girişi, sıralama işlemi ve sonuçların gösterilmesi gibi temel bileşenleri içeren bir yapı oluşturduk.

B. *Algoritma Seçimi ve Analizi:*

Algoritma seçimi aşamasında, kabarcık sıralama, seçim sıralaması, ekleme sıralaması ve birleştirme sıralaması gibi temel sıralama algoritmalarını araştırdık ve bu algoritmaların avantajlarını ve dezavantajlarını değerlendirdik. Ardından, her bir algoritmayı Python dilinde uyguladık. Algoritmaları doğru bir şekilde çalıştıklarından emin olmak için test senaryoları oluşturduk ve doğruluklarını kontrol ettik.

C. Kullanıcı Arayüzü Geliştirme:

Kullanıcı arayüzü geliştirme aşamasında, kullanıcıların veri girişi yapabilecekleri bir form oluşturduk. Kullanıcılar, sıralanmasını istedikleri verileri girebildikleri ve istedikleri sıralama algoritmasını seçebildikleri bir kullanıcı arayüzü tasarladık. Ayrıca, sıralama sonuçlarını kullanıcıya gösterecek bir çıktı düzeni oluşturduk.

D. Performans Analizi:

Uygulamada kullanılan sıralama algoritmalarının performansını analiz etmek için performans metrikleri belirledik. Bu metrikler arasında zaman karmaşıklığı, karşılaştırma adım sayısı ve yer değiştirme adım sayısı gibi faktörler yer aldı. Algoritmaların performansını test etmek ve karşılaştırmak için farklı boyutlardaki veri setlerini kullanarak performans analizi gerçekleştirdik.

E. Hata Kontrolü ve İyileştirme:

Uygulama geliştirme sürecinde hata kontrolüne özen gösterdik ve olası hata senaryolarını ele aldık. Hataları tespit etmek ve gidermek için hata ayıklama araçlarını kullandık. Ayrıca, kullanıcı geri bildirimlerini değerlendirdik ve kullanıcı deneyimini iyileştirmek için gerekli düzenlemeleri yaptık.

F. Exe Dosyasına Çevirme:

Projenin tamamlanmasının ardından, Python kodunu kullanıcılar için daha erişilebilir hale getirmek amacıyla projeyi exe dosyasına dönüştürme adımını gerçekleştirdik. Bu, kullanıcıların Python yüklü olmasını gerektirmeyen bir çalıştırılabilir dosya sağlayarak uygulamayı daha kolay kullanmalarını sağladı. Exe dosyasına dönüştürme işlemi için PyInstaller gibi araçları kullanarak, Python kodunu bağımsız bir çalıştırılabilir dosyaya çevirdik. Bu sayede kullanıcılar, sadece exe dosyasını çalıştırarak sıralama algoritmaları uygulamasını kullanabilirler.

VI. Proje Esnasında Karşılaşılan Zorluklar

Proje sürecinde bazı zorluklarla karşılaştık ve bu zorlukları aşmak için çaba harcadık. İşte projede zorlandığımız kısımlar:

- Algoritma Karmaşıklığı: Sıralama algoritmalarının karmaşıklığı ve çalışma mantığını anlamak, projenin başlangıcında biraz zorluk yaşadığımız bir noktaydı. Özellikle daha karmaşık algoritmaların implementasyonu ve optimizasyonu bazı zorluklar doğurdu. Bu noktada, algoritmalara ayrıntılı bir şekilde araştırma yaparak, kaynaklar ve dokümantasyonlar kullanarak bu zorluğun üstesinden gelmeye çalıştık.

- Kod Entegrasyonu: Projeyi üç kişi birlikte yürüttüğümüz için, her birimizin geliştirdiği kodu birleştirmek ve uyumlu bir şekilde entegre etmek bazı zorluklar oluşturdu. Farklı parçalar üzerinde çalışırken uyumlu bir kod tabanı oluşturmak ve çakışmaları önlemek için dikkatli bir şekilde çalıştık. Sürekli iletişim ve işbirliğiyle, değişiklikleri uyumlu bir şekilde birleştirerek ve gerektiğinde kod revizyonları yaparak bu zorluğun üstesinden geldik.

- Hata Ayıklama ve Sorun Giderme: Projede karşılaştığımız hataları tespit etmek ve sorunları gidermek bazen zaman alıcı olabildi. Özellikle büyük veri setleriyle çalışırken veya karmaşık algoritmaları uygularken hataların kaynağını bulmak ve düzeltmek için ekstra çaba harcadık. Hata ayıklama araçlarını etkin bir şekilde kullanarak, hataları izleyerek ve adım adım sorunları tespit ederek bu zorluğun üstesinden geldik.

Bu zorluklara rağmen, ekip olarak sürekli iletişimde kalarak ve birbirimize destek sağlayarak projeyi başarıyla tamamladık. Bu deneyim, zorlukların üstesinden gelme becerilerimizi geliştirmemize ve daha sağlam bir kod tabanı oluşturmamıza yardımcı oldu.

VII. Kaynakça

[1] https://docs.python.org/3/

[2] - https://dev.to/edwardcashmere/sorting-algorithms-2541

[3] https://betterprogramming.pub/5-basic-sorting-algorithms-you-must-know-9ef5b1f3949c

[4] - https://www.javatpoint.com/sorting-algorithms

[5] - https://www.programiz.com/dsa/sorting-algorithm

[6] - https://levelup.gitconnected.com/sorting-algorithms-comparing-the-best-and-the-rest-2448129f5b4a

[7] - https://cs.lmu.edu/~ray/notes/sorting/