Bilişim Sistemleri Mühendisliği

YazLab II Proje Raporu

Muhammet Arif Eren   
201307077

201307077@kocaeli.edu.tr

Semih Aysu  
211307085

211307085@kocaeli.edu.tr

Proje GitHub Linki:<https://github.com/semihaysu/Visualising_SortingAlgorithms>

# GİRİŞ

Bu projede öncelikli amacımız kullanıcının seçili sıralama algoritmaları ve grafik seçenekleri arasında istediğini seçerek belli değerlerdeki tabloların sıralama grafiklerini sunan bir uygulama hazırlamaktır. Bu proje kapsamında python dilinin inceliklerini iyice kavramak ve gerekli kütüphaneleri kullanarak grafik görselleştirmesini yapmaktır.

# PROJENİN TANITIMI

Projemiz Selection Sort, Buble Sort, Insertion Sort, Merge Sort ve Quick Sort sıralama algoritmalarını kullanarak Scatter, Bar ve Stem grafik çeşitleri içerisinde kullanıcıya değer aralığını seçtiği grafiğini sunmayı amaçlıyor. Bu sıralama algoritmalarını kısaca tanıtmak gerekirse;

## Selection Sort

Selection sort çalışma mantığı olarak listedeki en küçük elemanı bulup, en küçük sayıyla baştaki sayıyı yer değiştirir. Daha sonra tekrar en küçük sayıyı arar ancak bu sefer en başa attığı sayıya bakmaz yani 2. elemandan (1. indexten ) itibaren aramaya başlar. Bu şekilde diziyi sıralı hale getirir.

## Buble Sort

* Kabarcık Sıralaması, bilgisayar birimlerinde kullanılan yalın bir sıralama algoritmasıdır. Sıralanacak dizinin üzerinde sürekli ilerlerken her defasında iki öğenin birbiriyle karşılaştırılıp, karşılaştırılan öğelerin yanlış sırada olmaları durumunda yerlerinin değiştirilmesi mantığına dayanır. Algoritma, herhangi bir değişiklik yapılmayıncaya kadar dizinin başına dönerek kendisini yineler. Adına *"Kabarcık"* sıralaması denmesinin nedeni büyük olan sayıların aynı suyun altındaki bir kabarcık gibi dizinin üstüne doğru ilerlemesidir.
* Başlangıçta yer *yer değiştirme sıralaması* olarak adlandırılan kabarcık sıralaması, dizi içindeki büyükelemanların algoritmanın her adımında dizinin sonuna doğru doğrusal olarak ilerlemesini sağlar. Bu ilerleme, seçmeli sıralama algoritmasındaki dizideki değeri küçük olan elemanların dizinin başında kümelenmesi yöntemine benzer şekilde gerçekleşir.

## Insertion Sort

Eklemeli Sıralama bilgisayar bilimlerinde kullanılan ve sıralı diziyi her adımda öğe öğe oluşturan bir sıralama algoritmasıdır. Büyük dizilerle çalışıldığında hızlı sıralama, birleştirmeli sıralama ve yığın sıralaması gibi daha gelişmiş sıralama algoritmalardan daha verimsiz çalışır. Ancak buna karşın bazı artıları da vardır:

* Uygulaması kolaydır.
* Küçük Veri kümeleri üzerinde kullanıldığında verimlidir.
* Çoğunluğu zaten sıralanmış olan diziler üzerinde kullanıldığında verimlidir.
* Karmaşıklığı olan seçmeli sıralama ve kabarcık sıralaması gibi çoğu yalın sıralama algoritmalarından daha verimlidir.
* Kararlı bir sıralama algoritmasıdır (değeri eşit olan öğelerin asıl listedeki göreceli konumlarını değiştirmez)
* Sıralanacak diziyi yerinde sıralar, ek bir bellek alanı gerektirmez.
* Sıralanacak dizinin hepsinin algoritmanın girdisi olmasına gerek yoktur. Dizi parça parça da alınabilir ve sıralama işlemi sırasında diziye yeni veriler eklenebilir.

İnsanlar herhangi bir şeyi (örneğin, iskambil kartları) sıralarken, çoğu durumda eklemeli sıralamaya benzer bir yöntem kullanırlar.

## Merge Sort

Merge Sort (Birleştirme Sıralaması), diziyi ardışık olarak en küçük alt dizilerine kadar yarılayan sonra da onları sıraya koyarak bireştiren özyineli bir algoritmadır. Yarılama işlemi en büyük alt dizi en çok iki öğeli olana kadar sürer. Sonra "Merge (Birleştirme)" işlemiyle altdiziler ikişer ikişer bölünüş sırasıyla sıralı olarak bir üst dizide bireşir. Süreç sonunda en üstte sıralı diziye ulaşılır.

## Quick Sort

Hızlı sıralama, günümüzde yaygın olarak kullanılan bir sıralama algoritmasıdır. Hızlı sıralama algoritması *n* adet sayıyı, ortalama bir durumda,karmaşıklığıyla, en kötü durumda ise karmaşıklığıyla sıralarAlgoritmanın karmaşıklığı aynı zamanda yapılan karşılaştırma sayısına eşittir.

Bu sıralamalardan sonra kullanıcıya grafik çeşitlerinden, Scatter, Bar ve Stem grafikleri sunulmaktadır. Kısaca bu grafik türlerini açıklıcak olursak;

1. *Scatter*

Dağılım grafiği, farklı hesaplamalardaki değerleri noktalar koleksiyonu olarak tek bir boyut olarak sunar. Çoğu grafikte, boyutunuzu eksenlerden birinde bulursunuz, ancak dağılım grafiği için boyut grafikteki noktalarla gösterilir ve hesaplamalar iki eksenden birinde bulunur. İsteğe bağlı bir üçüncü hesaplama kullanıldığında, değeri kabarcık boyutunda gösterilir. Büyük veri kümelerini analiz ediyor ve sıkıştırılmış verileri görüntülüyorsanız, veri noktalarının yoğunluğu renkle gösterilir.

1. *Bar*

Çubuk grafiği olarak da adlandırılan sütun grafiği, daha uzun sütunların daha büyük sayıları gösterdiği, farklı değerleri karşılaştıran bir grafik türüdür. Sütun grafiği, ayrıca bar grafiği olarak da bilinmektedir. Yatay veya dikey sütun grafikleri oluşturabilirsiniz. Farklı şeylerin boyut veya değer açısından nasıl karşılaştırıldığını göstermek istediğinizde bir sütun grafiği kullanın.

1. *Stem*

Sap-yaprak grafiği olarak da adlandırılan bir gövde-yaprak diyagramı, bireysel veri noktalarını korurken verileri hızlı bir şekilde özetleyen bir diyagram. Böyle bir şemada, "kök", son basamağı çıkardıktan sonra benzersiz veri öğelerinin bir sütunudur.

Benzer şekilde, bir kök grafiği nasıl okursunuz? 'Kök' üzerinde sol ilk basamağı görüntüler veya rakamlar. 'Yaprak' sağdadır ve son rakamı gösterir. Örneğin, 543 ve 548 bir gövde ve yaprak üzerinde birlikte.

Bu seçnekleri kullanıcı istediği gibi doldurduktan sonra algoritmanın ve grafiğin oluşma hızını ayarlıyabilicektir. Bunun yanında grafikte bulunabilicek değer sayısını da belirleyebilmektedir. Grafiklendirmek için seçili değerler uyan grafik modelini “Oluştur” tuşu ile oluştura bilir, “Başlat” tuşu ile hesaplamayı başlatabilir, “Duraklat” tuşu ile hesaplamayı olduğu devrede durdurabilir son olarak “Durdur” tuşu ile işlemi tamamen durdurmuş olur.

# PROJEYE BAŞLARKENKİ HAZIRLIKLAR

Projenin asıl dili olan pythonda daha önce hiçbir pratiğimiz olmadığı için öncelikle gerekli kaynaklardan projemizi yürütmemize yarayıcak bilgileri edindik. Bu bilgiler ışığında kullanıcağımız kütüphaneleri ve yapıları öğrenmiş olduk.Gerekli kütüphaneleri istemciye indirdik.

##### SONUÇ

Gerekli isterlere uygun bi sonuç elde ettik ve bu konuda gereksinimleri tamamladık. Bu projeyi yaparken kullandığımız kütüphaneleri kullanmayı öğrendik ve bu kütüphaneler içinde verilen değerleri değiştirebimeyi öğrendik. Yapılan yanlışlardan yola çıkarak proje başlangıç öncesi düzgün planlama yapmayı öğrendik.

##### KAYNAKÇA

1. *https://tr.wikipedia.org/wiki/Kabarc%C4%B1k\_s%C4%B1ralamas%C4%B1*
2. http://cagataykiziltan.net/algoritmalar/1-siralama-algoritmalari/2-secmeli-siralama/#:~:text=Selection%20sort%20%C3%A7al%C4%B1%C5%9Fma%20mant%C4%B1%C4%9F%C4%B1%20olarak,%C5%9Fekilde%20diziyi%20s%C4%B1ral%C4%B1%20hale%20getirir.
3. https://tr.wikipedia.org/wiki/Eklemeli\_s%C4%B1ralama#:~:text=Eklemeli%20S%C4%B1ralama%20(%C4%B0ngilizce%3A%20Insertion%20Sort,s%C4%B1ralama%20algoritmalar%C4%B1ndan%20daha%20verimsiz%20%C3%A7al%C4%B1%C5%9F%C4%B1r.
4. https://tr.wikipedia.org/wiki/Eklemeli\_s%C4%B1ralama#:~:text=Eklemeli%20S%C4%B1ralama%20(%C4%B0ngilizce%3A%20Insertion%20Sort,s%C4%B1ralama%20algoritmalar%C4%B1ndan%20daha%20verimsiz%20%C3%A7al%C4%B1%C5%9F%C4%B1r..
5. https://bidb.itu.edu.tr/seyir-defteri/blog/2013/09/08/merge-sort-(bile%C5%9Ftirme-s%C4%B1ralamas%C4%B1)-algoritmas%C4%B1#:~:text=Merge%20Sort%20(Birle%C5%9Ftirme%20S%C4%B1ralamas%C4%B1)%2C,iki%20%C3%B6%C4%9Feli%20olana%20kadar%20s%C3%BCrer.
6. https://bilgisayarkavramlari.com/2008/08/09/hizli-siralama-algoritmasi-quick-sort-algorithm/
7. https://help.qlik.com/tr-TR/sense/February2023/Subsystems/Hub/Content/Sense\_Hub/Visualizations/ScatterPlot/scatter-plot.htm
8. https://help.qlik.com/tr-TR/sense/February2023/Subsystems/Hub/Content/Sense\_Hub/Visualizations/Bar-Chart/bar-chart.htm
9. https://www.canva.com/tr\_tr/sema/bar-semasi/
10. https://reviews.tn/tr/wiki/what-is-a-stem-graph/
11. https://www.python.org/downloads/
12. https://www.w3schools.com/python/default.asp
13. https://www.w3schools.com/python/python\_ml\_scatterplot.asp
14. https://github.com/search?l=Python&o=asc&p=2&q=Sorting+Algorithms+visualiser&s=updated&type=Repositories
15. https://github.com/Newspaperman57/visualisingSortingAlgorithms
16. https://github.com/raspberrypilearning/visualising-sorting-with-python
17. https://github.com/raspberrypilearning/visualising-sorting-with-python
18. https://github.com/zurakin/sort-visualisation
19. https://github.com/search?q=Sorting+Algorithms+visualiser&type=Repositories
20. https://github.com/ad-1/SortingVisualisation
21. https://github.com/DeepakJha01/SortingVisualizer
22. https://github.com/Azanul/Sorting-Visualized
23. https://github.com/anirudhlakhotia/visual-sorting-algorithms
24. https://github.com/Zircoz/Sorting-Algorithms-in-Python
25. https://ogr.kocaeli.edu.tr/KOUBS/Ogrenci/index.cfm
26. https://edestek3.kocaeli.edu.tr/my/courses.php