Algorithm Sorting Visualizer

Harun Şimsek
Bilişim Sistemleri Mühendisliği
harunsimsek1940@gmail.com

Ahmet Melih Türkmen
Bilişim Sistemleri Mühendisliği
turkmenmelih01@gmail.com

Dzenit Vildic
Bilişim Sistemleri Mühendisliği
dzenit6@gmail.com

I. Özet

Proje kapsamında algoritma sıralama programı tasarlamamız istenmektedir.Programımız seçilen veri yapısına (insertion sort,bubble sort,merge sort,quick sort) uygun algoritma sıralama görsel olarak gerçekleşmektedir.

II. Abstraction

Within the scope of the project, we are required to design an algorithm sorting program. Our program performs the algorithm sorting visually in accordance with the selected data structure (insertion sort, bubble sort, merge sort, quick sort).

Program Tasarimi

A. *Dashboard*

Programımızın panel kısmında algoritma seçme kısmı,algoritmanın işleyiş hızı,veri boyutunu,min ve max değerini,grafik tipini seçiyoruz.Özellikler seçildikten sonra oluşturulan ggrafik üzerinde seçili olan algoritma görsel şekilde ekrana yazdırılmış olur.

(Dashboard ekranı tasarım görüntüsü)

B. *Algoritmalar*

Programımızda 5 adet algoritma seçme seçeneği mevcuttur.Algoritma seçildiği zaman seçilen algoritmaya uygun algoritma görselleştirme işlemi gerçekleşir.Görselleştirdiğimiz algoritma seçili olan min,max değerler ve seçtiğimiz işleyiş hızına göre çalışmaya başlar.Programımız seçtiğimiz algoritma tamamlandığında sıralama işlemimiz bitmektedir.Programımızın sunduğu 5 farklı algoritma seçeneği mevuttur.

Insertion Sort: eklemeli sıralama algoritmasını uygular. Bu yöntem, bir dizi içindeki elemanları sırayla alır ve doğru konumlarına ekler. Döngüler ve karşılaştırmalar kullanarak elemanların sıralanmasını gerçekleştirir. drawData ve timeTick parametreleri, görselleştirmeyi güncellemek ve işlem arasında gecikme eklemek için kullanılır. Son olarak, sıralama tamamlandığında bir ses çalınır.

Bubble Sort:

Bubble sort, basit ve anlaşılır bir sıralama algoritmasıdır. Bir dizi içindeki elemanları sıralamak için kullanılır. Adını, elemanları karşılaştırırken küçük olanı yukarı "bubbles up" şeklinde taşımasından alır.

Bubble sort algoritması, listedeki elemanları çiftler halinde karşılaştırır ve gerektiğinde yerlerini değiştirir. Sıralama işlemi, listedeki tüm elemanlar karşılaştırılarak en büyük elemanın en sona yerleştirildiği bir geçit (pass) olarak adlandırılan adımlar dizisiyle gerçekleşir. Her geçitte, listedeki elemanlar çiftler halinde karşılaştırılır ve iki elemanın sırası yanlışsa yer değiştirilir. Bu işlem, listedeki tüm elemanlar sıralanana kadar tekrarlanır.

Merge Sort:

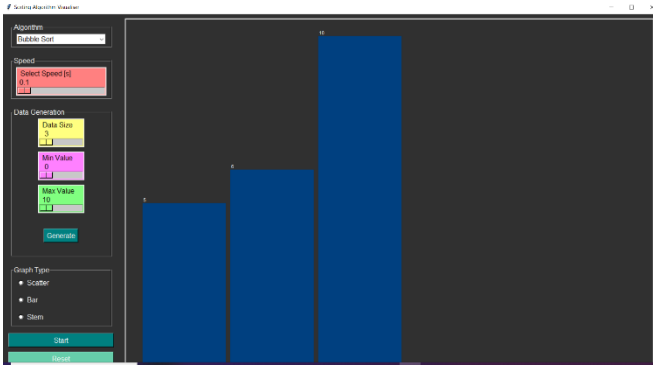
Merge sort, sıralama algoritmalarından biridir ve genellikle dizileri veya listeleri sıralamak için kullanılır. Merge sort, "böl ve yönet" (divide and conquer) yaklaşımına dayanır, yani büyük bir sorunu küçük parçalara böler, ardından bu parçaları sıralar ve son olarak birleştirir.

Merge sort algoritmasının adı, sıralanacak diziyi sürekli ikiye bölmek ve ardından birleştirmek için kullanılan "birleştirme" (merge) adımından gelir. İşlem, sıralanacak dizi tek elemanlı parçalara ayrılana kadar tekrarlanır. Daha sonra, birleştirme adımı başlar.

Birleştirme adımında, iki küçük parçayı birleştirirken sıralar ve daha büyük bir parçayı oluşturur. Bu işlem, parçalar tamamen birleşene kadar tekrarlanır ve sonunda sıralanmış bir dizi elde edilir.

Quick Sort

Quick sort (hızlı sıralama), verileri sıralamak için yaygın olarak kullanılan bir sıralama algoritmasıdır. Hızlı ve etkili bir şekilde çalışabilmesiyle bilinir. Quick sort, "böl ve fethet" (divide and conquer) yöntemini kullanarak çalışır.



III. ALGORİTMA TASARIMI

A. *Sorting Algoritması*

Bu Python kodu, farklı sıralama algoritmalarını içeren bir "Sortings" sınıfını tanımlar. Bu sınıf, insertion sort (ekleme sıralaması), bubble sort (kabarcık sıralaması), selection sort (seçim sıralaması), quick sort (hızlı sıralama) ve merge sort (birleştirme sıralaması) algoritmalarını içerir.

Her bir sıralama metodu, girdi olarak bir dizi, bir çizim fonksiyonu (drawData) ve bir zaman gecikmesi (timeTick) alır. "drawData" fonksiyonu, sıralama işlemi sırasında dizinin nasıl çizileceğini kontrol eder ve "timeTick" parametresi, her adımda beklenmesi gereken zaman aralığını belirler.

Sorting Algoritmasının İşleyişi

Ekleme sıralaması (insertion_sort): Her elemanı sırasıyla alarak, elemanın uygun yerine yerleştirir.

Kabarcık sıralaması (bubble_sort): Komşu elemanları karşılaştırarak, doğru sıralamaya yerleştirilene kadar elemanları yer değiştirir.

Seçim sıralaması (selection_sort): Minimum değeri bulup listenin başına yerleştirerek, elemanları sıralar.

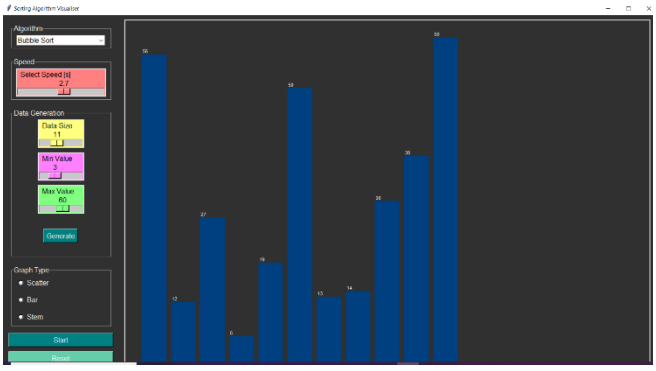
Hızlı sıralama (quick_sort): Pivot elemanını seçerek, liste içindeki diğer elemanları pivot elemanına göre böler ve ayrı ayrı sıralar.

Birleştirme sıralaması (merge_sort): Liste içindeki elemanları sürekli olarak ikiye böler ve ardından birleştirerek sıralar.

Ayrıca, "getColorArray" ve "getColorArrayMerge" fonksiyonları, görselleştirmelerde kullanılan renk dizisini döndürür.

Kod ayrıca, işlemlerin gerçekleştiğini göstermek için winsound modülünü kullanarak ses efektleri de çalar.

Bu sınıf, her sıralama algoritması için sıralama işlemini gerçekleştiren yöntemler sağlar. Bu yöntemleri kullanarak veri dizisini sıralayabilir ve görsel bir şekilde adımları takip edebilirsiniz.



IV. PYTHON

Python Tkinter, Python programlama dilinin standart kütüphanelerinden biridir ve kullanıcı arayüzü oluşturmak için kullanılır. Tkinter, Tk GUI toolkit'inin Python sarmalayıcısıdır ve çeşitli widget'ları (butonlar, etiketler, giriş kutuları vb.) ve diğer GUI bileşenlerini oluşturmak ve yönetmek için kullanılır.
İşte Tkinter hakkında bazı temel bilgiler:

Identify applicable funding agency here. If none, delete this text box.

Tkinter'ı İçe Aktarma:

Tkinter'ı kullanabilmek için Python programınızın başına aşağıdaki gibi bir kod satırı eklemeniz gerekmektedir:

```
import tkinter as tk
```

Ana Pencere Oluşturma
Tkinter ile bir GUI uygulaması oluşturmak için, genellikle bir ana pencere (main window) oluşturulur. Bu ana pencere, uygulamanın temel çerçevesini sağlar. Ana pencereyi oluşturmak için şu adımları izleyebilirsiniz:

```
root = tk.Tk()
root.mainloop()
```

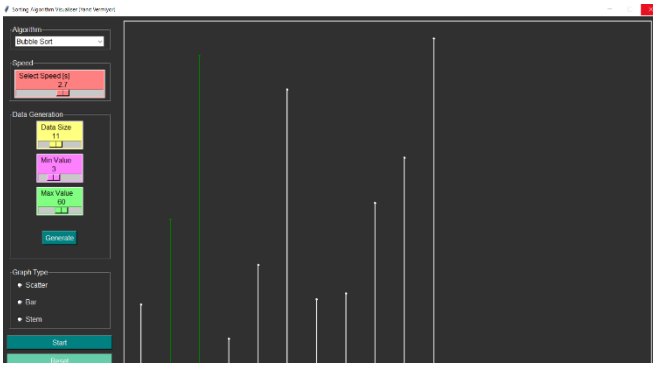
Widget'lar
Tkinter, çeşitli widget'ları oluşturmak için kullanılan sınıflar sağlar. Örneğin, bir düğme oluşturmak için tk.Button sınıfını kullanabilirsiniz. Bu widget'ları ana pencereye eklemek ve yönetmek için yöntemler kullanılır. Örneğin:

```
button = tk.Button(root, text="Click Me")
button.pack()
```

Olaylar ve İşlevler
Tkinter, kullanıcının widget'lara etkileşimde bulunduğunda gerçekleşen olayları yakalayabilmenizi sağlar. Örneğin, bir düğmeye tıklandığında bir işlevin çalışmasını sağlayabilirsiniz. Bu işlevler, kullanıcı olaylarını ele almak için kullanılır. Örneğin:
def button_click():

```
    print("Button Clicked!")
button = tk.Button(root, text="Click Me", command=button_click)
button.pack()
```

Diğer Özellikler
Tkinter, widget'lar için birçok farklı özelliği de destekler. Örneğin, widget'ların boyutlarını ayarlamak, metin veya görüntü eklemek, arka plan rengini değiştirmek gibi özellikleri kullanabilirsiniz.



Widget Yerleştirme

Tkinter, widget'ları pencerede yerleştirmek için farklı yöntemler sunar. En yaygın yöntemlerden biri pack()

yöntemidir. Bu yöntem, widget'ları pencerede yukarıdan aşağıya sıralı bir şekilde yerleştirir. Başka bir yöntem ise grid() yöntemidir. Bu yöntemde widget'ları hücrelere yerleştirerek bir ızgara oluşturabilirsiniz.

```
button1 = tk.Button(root, text="Button 1")
button1.pack()
button2 = tk.Button(root, text="Button 2")
button2.pack()
label = tk.Label(root, text="Hello, Tkinter!")
label.pack()
```

Widget Özellikleri

Tkinter widget'ları, çeşitli özelliklere sahip olabilir. Özellikler widget'ın görünümünü ve davranışını değiştirmek için kullanılır. Örneğin, bir düğmenin metnini değiştirmek, bir etiketin rengini ayarlamak, bir giriş kutusuna varsayılan bir değer atamak gibi özellikleri kullanabilirsiniz.

```
button = tk.Button(root, text="Click Me", bg="blue", fg="white")
entry = tk.Entry(root, width=20, bd=3)
label = tk.Label(root, text="Hello, Tkinter!", font=("Arial", 14))
```

Birden Fazla Pencere Oluşturma

Tkinter ile birden fazla pencere oluşturabilirsiniz. İkinci bir pencere oluşturmak için yeni bir Tk nesnesi oluşturabilir veya ana pencerenin yerine Toplevel sınıfını kullanabilirsiniz.

```
second_window = tk.Toplevel(root)
```

Dizin Yapısı

Tkinter ile karmaşık arayüzler oluştururken widget'ları hiyerarşik bir yapıda düzenlemek önemlidir. Ana pencereye eklenen widget'lar, diğer widget'ların üstünde veya altında yer alabilir. Bu hiyerarşik yapıyı kullanarak widget'ların düzenini ve davranışını daha iyi kontrol edebilirsiniz.

Özelleştirme ve Stil

Tkinter, widget'ları özelleştirmek ve stil vermek için çeşitli seçenekler sunar. Örneğin, widget'ların arka planını, metin rengini, fontunu, boyutunu ve daha fazlasını değiştirebilirsiniz. Ayrıca, temalar kullanarak uygulamanın genel görünümünü değiştirebilirsiniz.

Tkinter Widget'ları

Tkinter, çeşitli widget'ları destekler. İşte bazı yaygın kullanılan widget'lar:

Label (Etiket): Metin veya görüntüleri görüntülemek için kullanılır.

Button (Düğme): Kullanıcının tıklamasına tepki olarak bir işlevi tetikler.

Entry (Giriş): Kullanıcıdan veri girmek için kullanılan bir metin kutusudur.

Text (Metin): Çok satırlı metin girişi ve düzenlemesi içinkullanılır.

Listbox (Liste Kutusu): Bir listedeki öğeleri görüntülemek ve seçmek için kullanılır.

Checkbutton (Onay Kutusu): Bir veya birden çok seçeneği işaretlemek için kullanılır.

Radiobutton (Seçenek Düğmesi): Yalnızca bir seçeneğin seçilebileceği bir grup seçenek için kullanılır.

Combobox: Bir açılır kutu ve metin girişi kombinasyonudur.

Canvas: Çizimler, grafikler ve özel arayüz bileşenlerini oluşturmak için kullanılır.

Olay Yönetimi

Tkinter, kullanıcı etkileşimlerini yakalamak ve bunlara tepki vermek için olay yönetimini destekler. Örneğin, bir düğmeye tıklandığında bir işlevin çalışmasını sağlayabilirsiniz. Bu olayları yakalamak için bind() yöntemini kullanabilir veya widget'lara doğrudan bir olay işleyici (event handler) atayabilirsiniz.

```
def button_click():
print("Button Clicked!")
button = tk.Button(root, text="Click Me")
button.bind("<Button-1>", lambda event: button_click())
```

Diyalog Pencereleri

Tkinter, kullanıcıya mesajlar göstermek, dosya seçimi yapmak, onay almak gibi işlemler için diyalog pencereleri sağlar. tkinter.messagebox ve tkinter.filedialog modüllerini kullanarak kullanıcıyla etkileşime geçebilirsiniz.

```
import tkinter.messagebox as messagebox
import tkinter.filedialog as filedialog
messagebox.showinfo("Bilgi", "İşlem tamamlandı.")
file_path = filedialog.askopenfilename()
```

Stil Yönetimi

Tkinter, widget'ların görünümünü özelleştirmek için stil yönetimi sunar. Tkinter stil yönetimini kullanarak widget'ların font, renk, boyut ve diğer özelliklerini merkezi bir şekilde ayarlayabilirsiniz.

```
style = tk.Style()
style.configure("MyLabel.TLabel", background="blue", foreground="white", font=("Arial", 12))
label = tk.Label(root, text="Hello, Tkinter!", style="MyLabel.TLabel")
```

KAYNAKÇA

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

[2]https://www.pythontutorial.net/tkinter/