Entwicklung einer Musik-App – Johannes Brenner

1. Einleitung

Im Rahmen des Projekts wurde eine Musikanwendung entwickelt, die die Implementierung und den Vergleich von zwei Such- und zwei Sortieralgorithmen zum Ziel hatte. Die Aufgabe bestand darin, eine Anwendung zu erstellen, die es ermöglicht, eine Musiksammlung realitätsnah zu durchsuchen, zu sortieren und Playlists zu verwalten.

Kernstück der Anwendung sind die beiden implementierten Suchalgorithmen Binary Search und Linear Search sowie die Sortieralgorithmen Quick Sort und Bubble Sort. Diese Algorithmen wurden aufgrund ihrer unterschiedlichen Komplexität und Einsatzmöglichkeiten ausgewählt, um ihre Effizienz und Praktikabilität im Umgang mit großen Datensätzen zu untersuchen.

Ziel dieser Dokumentation ist es, die Vorgehensweise bei der Entwicklung der Applikation sowie die Funktionsweise der verwendeten Algorithmen detailliert darzustellen. Ein besonderer Fokus liegt dabei auf der Analyse der Laufzeiten und der Performance der Such- und Sortierverfahren, um deren Vor- und Nachteile aufzuzeigen. Die Dokumentation soll somit als Grundlage für das Verständnis der Implementierung dienen und die theoretischen Konzepte der Algorithmen in einen praxisnahen Kontext stellen.

2. Konzept und Vorgehensweise

Für die Entwicklung der Musik-App wurde ein Datensatz von Kaggle verwendet, der eine Sammlung von ca. 114.000 Songs umfasst. Die Daten enthalten die Attribute **title**, **artist**, **release_year** und **genre**. Diese Daten bieten eine umfangreiche Grundlage, um die Funktionalitäten der Anwendung zu testen und die implementierten Such- und Sortieralgorithmen in einem realistischen Szenario zu evaluieren.

Die Daten wurden als CSV-Datei importiert und mit der Python-Bibliothek **Pandas** verarbeitet. Pandas bietet effiziente Methoden zur Datenmanipulation und ermöglichte ein einfaches Laden und Filtern der Musikinformationen. Die Visualisierung des Such- und Sortierfortschritts wurde mit Hilfe der Bibliothek **TQDM** realisiert, die Fortschrittsbalken anzeigt und so einen besseren Einblick in die Funktionsweise der Algorithmen ermöglicht.

Bei der Programmierung wurde schrittweise vorgegangen: Zunächst wurde die Grundstruktur der Anwendung erstellt, einschließlich der Hauptfunktionen wie das Laden der Daten und die Erstellung der Nutzeroberfläche. Anschließend wurden die Such- und Sortieralgorithmen in die Anwendung integriert und getestet. Besonderes Augenmerk wurde auf die Performance der Algorithmen gelegt, da die Anwendung in der Lage sein sollte, große Datenmengen effizient zu durchsuchen und zu sortieren.

Die Planung orientierte sich an klar definierten Zielen, die bereits in der Einleitung beschrieben wurden. Die Umsetzung erfolgte iterativ, wobei schrittweise Funktionen ergänzt und optimiert wurden, um den Nutzeranforderungen gerecht zu werden und die vorgegebenen Algorithmen korrekt zu integrieren. Herausforderungen wie der Umgang mit großen Datenmengen und die Optimierung der Algorithmen wurden durch schrittweises Testen und Anpassen des Codes bewältigt.

3. Architektur und Nutzerführung

Die Musik-App ist modular aufgebaut und besteht aus verschiedenen Funktionen, die jeweils spezifische Aufgaben erfüllen. Der Code ist in einem funktionalen Programmierstil ohne Verwendung von Klassen oder speziellen Modulen geschrieben, was eine klare Struktur und einfache Erweiterbarkeit ermöglicht. Die Architektur und die Nutzerführung der App sind eng miteinander verknüpft und bietet eine intuitive und übersichtliche Navigation durch die verschiedenen Funktionen der Anwendung.

3.1 Modularer Aufbau der Anwendung

Die Hauptfunktionen der App lassen sich in folgenden Bereiche unterteilen:

Datenverarbeitung: Die Daten werden mit der Funktion `load_data()` aus einer CSV-Datei geladen. Die Python-Bibliothek Pandas wird verwendet, um die Musikdaten effizient zu verarbeiten, zu filtern und zu manipulieren. Diese Datenverarbeitung bildet die Grundlage für die Such- und Sortieralgorithmen.

Suchfunktionen: Die App bietet zwei Suchalgorithmen, Binary Search und Linear Search, die vom Nutzer ausgewählt werden können. Diese Algorithmen durchsuchen die Musiksammlung nach Titeln, die mit einem bestimmten Suchbegriff beginnen. Während Binary Search eine schnelle Suche in sortierten Daten ermöglicht, durchsucht Linear Search die Musiksammlung sequentiell, was für unsortierte Daten geeignet ist.

Sortierfunktionen: Die Anwendung implementiert zwei Sortieralgorithmen, Quick Sort und Bubble Sort, um Musiktitel nach Titelnamen zu sortieren. Der Nutzer kann seinen bevorzugten Sortieralgorithmus auswählen. Quick Sort bietet eine effiziente Sortiermethode, während Bubble Sort aufgrund seiner Einfachheit langsamer ist und sich besser für kleinere Datensätze geeignet.

Favoriten- und Playlist-Verwaltung: Die App ermöglicht es dem Nutzer, Songs zu seinen Favoriten hinzuzufügen und Playlists zu erstellen, zu bearbeiten oder zu löschen. Diese

Verwaltung bietet eine flexible Organisation der gespeicherten Musiksammlung und erlaubt es, Songs gezielt anzuhören und nach Künstler oder Genre zu ordnen.

Musikwiedergabe und Interaktion: Die Anwendung simuliert das Abspielen von Musiktiteln und ermöglicht das Pausieren und Fortsetzen von Songs sowie das direkte Hinzufügen oder Entfernen von Titeln zu Favoriten oder Playlists. Diese Funktionen verbessern das Nutzererlebnis und stellen sicher, dass die Interaktion intuitiv und verständlich ist.

Entdecken: Mit der Entdecken-Funktion kann der Nutzer ein Genre seiner Wahl auswählen, um neue Songs zu entdecken und seine Musiksammlung zu erweitern. Diese Funktion unterstützt den Nutzer bei der gezielten Suche nach neuen Musiktiteln innerhalb eines bestimmten Genres und beim Kennenlernen neuer Künstler und Stile.

3.2 Nutzeroberfläche

Die Nutzeroberfläche der Musikanwendung ist textbasiert und bietet dem Nutzer übersichtliche Menüs zur Navigation durch die verschiedenen Funktionen. Das Hauptmenü dient als zentraler Einstiegspunkt, von dem aus der Nutzer auf alle wichtigen Funktionen zugreifen kann.

3.3 Navigation und Menüsteuerung

Die Navigation erfolgt über numerische Eingaben, die den Nutzer durch die verschiedenen Menüoptionen führen. Die Menüs sind so gestaltet, dass sie nur die relevanten Optionen anzeigen, die für den aktuellen Kontext wichtig sind. Eine besondere Stärke der App ist die dynamische Anpassung der Menüoptionen: Optionen werden nur dann angezeigt, wenn sie sinnvoll sind. So wird beispielsweise die Option, einen Song zu den Favoriten hinzuzufügen, nur dann angezeigt, wenn der Song noch nicht in den Favoriten enthalten ist. Befindet sich der Song bereits in den Favoriten, wird stattdessen die Option zum Entfernen des Lieds angezeigt. Gleiches gilt für die Verwaltung von Playlists, wo nur dann bestehende Playlists angezeigt werden, wenn auch welche vorhanden sind. Diese dynamische Anpassung der Menüs sorgt für eine übersichtliche und nutzerfreundliche Bedienung.

3.4 Nutzerfreundlichkeit und Interaktion

Die Nutzeroberfläche ermöglicht verschiedene Interaktionen, die dem Nutzer eine intuitive Steuerung der Musiksammlung erlauben. Die wichtigsten Interaktionen sind:

Musikwiedergabe und -steuerung: Der Nutzer kann Songs suchen, abspielen, pausieren und fortsetzen sowie diesen Titel zu seinen Favoriten hinzufügen oder entfernen.

Favoriten filtern: Gespeicherte Favoriten können nicht nur als einfache Liste angezeigt werden, sondern auch gezielt nach Künstlern oder Genres gefiltert werden. Diese Filteroptionen bieten dem Nutzer eine schnelle Möglichkeit, bestimmte Songs zu finden und seine Favoriten besser zu organisieren. So kann der Nutzer beispielsweise auf „Meine Künstler“ oder „Meine Genres“ zugreifen, um eine übersichtliche Darstellung der gespeicherten Songs nach Künstler oder Musikrichtung zu erhalten.

Playlist-Verwaltung: Playlists können erstellt, bearbeitet und gelöscht werden. Nutzerinnen und Nutzer können Titel direkt zu bestehenden Playlists hinzufügen oder neue Playlists erstellen, wenn keine passenden vorhanden ist.

Neue Genres entdecken: Mit der Entdecken-Funktion können Nutzer gezielt Genres auswählen, um neue Musik zu entdecken, was die App auch für Nutzer interessant macht, die ihre Musiksammlung erweitern möchten.

Diese Interaktionen wurden so einfach und intuitiv wie möglich gestaltet. Die klare Struktur der Menüs und die logische Abfolge der Aktionen erleichtern es dem Nutzer, sich in der App zurechtzufinden und alle gewünschten Funktionen schnell zu erreichen.

4. Implementierte Algorithmen

4.1 Suchalgorithmen

In der Musik-App wurden zwei Suchalgorithmen und zwei Sortieralgorithmen implementiert, um die Musiksammlung effizient zu durchsuchen und zu sortieren. Die Funktionsweise der Algorithmen wird im Folgenden beschrieben.

4.1.1 Binary Search

Binary Search ist ein effizienter Suchalgorithmus für sortierte Daten. Der Algorithmus arbeitet, indem er die Liste wiederholt halbiert und den Suchbereich schnell eingrenzt, indem er den mittleren Wert mit der Suchanfrage vergleicht. Dadurch eignet sich Binary Search besonders für große Datensätze, die bereits sortiert sind.

4.1.2 Linear Search

Linear Search durchsucht die Liste von oben nach unten und vergleicht jeden Eintrag mit der Suchanfrage. Diese Methode ist einfach zu implementieren und kann direkt auf unsortierte Daten angewendet werden, was sie flexibel, aber bei großen Datensätzen weniger effizient macht.

4.2 Sortieralgorithmen

4.2.1 Quick Sort
Quick Sort teilt die Liste rekursiv in kleinere Teillisten auf, die nach einem Pivotelement sortiert werden. Durch die Aufteilung und rekursive Sortierung der Teillisten erreicht der Algorithmus eine schnelle und effiziente Sortierung.

4.2.2 Bubble Sort
Bubble Sort ist ein einfacher Sortieralgorithmus, der benachbarte Elemente vertauscht, bis die gesamte Liste sortiert ist. Der Algorithmus ist intuitiv und leicht verständlich, aber bei großen Datensätzen ineffizient, da er viele Vergleiche und Vertauschungen benötigt.

5. Laufzeitanalyse der Algorithmen

Die Leistungsfähigkeit der Such- und Sortieralgorithmen wurde sowohl theoretisch als auch praktisch untersucht, um ihre Effizienz in verschiedenen Szenarien zu bewerten.

5.1 Theoretische Laufzeitkomplexität

- **Binary Search**: Hat eine Laufzeit von O(log n), da der Algorithmus die Liste bei jedem Schritt halbiert. Dies macht ihn besonders effizient im Vergleich zu linearen Suchmethoden, vor allem bei großen, sortierten Datensätzen.

- **Linear Search**: Die Laufzeit beträgt O(n), da jedes Element der Liste sequenziell durchsucht wird. Diese Methode ist praktikabel bei kleinen oder unsortierten Datensätzen, zeigt jedoch deutliche Leistungseinbußen bei großen Datenmengen.

- **Quick Sort**: Mit einer durchschnittlichen Laufzeit von O(n log n) bietet Quick Sort eine effiziente Sortiermethode für große Datensätze. Im ungünstigsten Fall, abhängig von der Wahl des Pivotelements, kann die Laufzeit jedoch $O(n^2)$ betragen.

- **Bubble Sort**: Bubble Sort hat eine Laufzeit von $O(n^2)$, da jedes Element mehrfach mit seinen Nachbarn verglichen und vertauscht wird. Dies macht den Algorithmus ungeeignet für große Listen.

5.2 Messung der realen Laufzeiten

Die tatsächlichen Laufzeiten der Algorithmen wurden mit Hilfe von Python-Funktionen zur Zeitmessung (`time.time()`) und der Bibliothek TQDM ermittelt. Die Messungen wurden an der Musiksammlung durchgeführt, um die Leistungsfähigkeit der Algorithmen bei einer praxisnahen Datengröße von über 114.000 Songs zu testen. Die gemessenen Laufzeiten sind:

- **Suchalgorithmen**:

  Suche nach "love":

  Binary Search: 0,31 Sekunden

  Linear Search: 2,04 Sekunden

  Suche nach "night":

  Binary Search: 0,12 Sekunden

  Linear Search: 2,06 Sekunden

- **Sortieralgorithmen (alle Lieder A-Z sortieren):**

  Quick Sort: 1,67 Sekunden

  Bubble Sort: ca. 1 Stunde und 20 Minuten

5.3 Vergleich der Laufzeiten und Analyse

Die gemessenen Laufzeiten zeigen deutliche Unterschiede in der Effizienz der Algorithmen:

Binary Search zeigte für beide getesteten Suchbegriffe ("love" und "night") konstant schnelle Laufzeiten, wobei die Zeit für "love" aufgrund der höheren Trefferzahl etwas höher lag (0,31 Sekunden). Dies unterstreicht die Stärke der binären Suche.

Linear Search hatte für beide Suchbegriffe längere Laufzeiten (ca. 2 Sekunden), was die erwartete O(n)-Komplexität widerspiegelt. Der Algorithmus ist robust gegenüber unsortierten Daten, zeigt aber deutliche Leistungseinbußen bei großen Datensätzen.

Quick Sort aller Lieder von A bis Z dauerte 1,67 Sekunden, was seine Effizienz bei großen unsortierten Datenmengen bestätigt. Die Laufzeit entspricht der erwarteten O(n log n)-Leistung.

Bubble Sort benötigte für die gleiche Sortieraufgabe ca. 1 Stunde und 20 Minuten, was seine Ineffizienz bei großen Datensätzen unterstreicht. Diese Ergebnisse stimmen mit der theoretischen Laufzeit von $O(n^2)$ überein und zeigen, dass Bubble Sort für große Listen ungeeignet ist.

Die realen Messwerte bestätigen die theoretischen Laufzeitanalysen und verdeutlichen, warum Binary Search und Quick Sort die bevorzugten Algorithmen für große Datensätze sind. Die deutlichen Unterschiede zwischen den Algorithmen unterstreichen die Wichtigkeit der Auswahl geeigneter Such- und Sortierverfahren, insbesondere bei der Verarbeitung großer Datenmengen.

6. Zusammenfassung

In diesem Projekt wurde eine Musikanwendung entwickelt, die das Suchen, Sortieren und Verwalten von Musiktiteln ermöglicht. Zwei Suchalgorithmen (Binary Search und Linear Search) und zwei Sortieralgorithmen (Quick Sort und Bubble Sort) wurden implementiert und hinsichtlich ihrer Laufzeit analysiert. Die Ergebnisse zeigen, dass Binary Search und Quick Sort insbesondere bei großen Datensätzen effizienter sind, während Linear Search und Bubble Sort aufgrund ihrer längeren Laufzeiten weniger geeignet sind.

Die Anwendung bietet eine nutzerfreundliche, textbasierte Oberfläche mit dynamischen Menüs, die sich an die Aktionen des Nutzers anpassen. Zu den wichtigsten Funktionen gehören das Filtern der Favoriten nach Künstler und Genre, sowie das Verwalten von Playlists. Die Kombination aus effizienten Algorithmen und intuitiver Nutzerführung macht die Anwendung leistungsfähig und einfach zu bedienen.