Fachinformatiker für Anwendungsentwicklung

Dokumentation zur Projektarbeit im Fach EBA

Abgabedatum: Chemnitz, den 04.06.2021

Sortieralgorithmen

Projektteilnehmer:

Annika Michl
Dominik Angel
Paulo Weber

Ausbildungsbetriebe:

Schaeffler Digital Solution GmbH

LKS– Landwirtschaftliche Kommunikations- und Servicegesellschaft mbH

Projektbetreuer: Herr Martin Rein

Inhaltsverzeichnis

Vorwort

Wir haben uns in diesem Projekt mit dem Thema Sortieralgorithmen auseinandergesetzt. In der Informatik nimmt das Sortieren von Datenmengen einen wichtigen Stellenwert ein, da häufig Daten bzw. Informationen ungeordnet vorliegen. Sortiervorgänge haben das Ziel, eine bestimmte Ordnung innerhalb einer Menge zu schaffen. Diese sortierten Mengen werden zur vereinfachten und beschleunigten Suche verwendet. Sortiervorgänge nehmen heute einen Großteil der Rechenzeit in Anspruch, schätzungsweise 20% bis 25% der Rechenzeit im kommerziellen Bereich.

In diesem Zusammenhang haben wir uns mit den Sortieralgorithmen Bubble-Sort, Merge-Sort und Quick-Sort beschäftigt. Ziel des Projektes war es eine Anwendung zu schaffen, welche die Effizienz der genannten Sortieralgorithmen darstellt.

1 Einleitung

1.1 Problemstellung

Unsere Aufgabe bestand darin eine Webandwendung unter Einsatz von HTML, CSS und JavaScript zu entwickeln. Die Anwendung soll eine Reihe von Karten mit den Sortieralgorithmen Bubble-Sort, Merge-Sort und Quick-Sort in aufsteigender Reihenfolge ordnen und darstellen. Dabei muss die Effizienz der jeweiligen Algorithmen klar erkennbar sein. Der Nutzer soll in der Lage sein eine beliebige Kartenreihe zu erzeugen. Das Projekt muss vollständig dokumentiert und nachvollziehbar sein.

2 Planung

2.1 Aufgabenteilung

Unsere Projektgruppe besteht aus drei Mitglieder. Nachdem wir unser Vorgehen besprochen haben, konnten wir unser Projekt in 6 Bereiche aufteilen. Paulo Weber übernahm die Implementierung des Quick-Sort Algorithmus und leistete Unterstützung bei der Implementierung des Frontend. Dominik Angel implementierte den Merge-Sort Algorithmus, sowie den Schrittzähler zur Darstellung der Effizienz und leistete ebenfalls Unterstützung beim Frontend. Annika Michl übernahm die Implementierung des Bubble-Sort Algorithmus und des Frontend auf Grundlage eines zusammen erstellten Mockups. Das Erstellen der Dokumentation erfolgte in gemeinschaftlicher Arbeit.

2.2 Zeitplanung

Zum Start unseres Projektes haben wir ein zweistündiges Meeting abgehalten. Schwerpunkt war die Projektplanung und die Vorbereitung des GitHub Repository. In der ersten Präsenzstunde erstellten wir zusammen ein Mockup und sammelten Ideen. Der Hauptteil unserer Arbeit erfolgte in der Zeit des Homeschooling. Im ersten Schritt implementierte jeder seinen gewählten Sortieralgorithmus. Im nächsten Schritt erfolgte die Implementierung des Schrittzählers und des Frontend. In dem darauffolgenden Präsenzunterricht behoben wir letzte Fehler im Code und erarbeiteten die Dokumentation, welche im Homeschooling beendet wurde.

2.3 Verwendete Technologien

Wie in der Problemstellung beschrieben arbeiteten wir nur mit der Auszeichnungssprache HTML, der Formatierungssprache CSS und der Skriptsprache JavaScript. Da jedoch in der Bearbeitungszeit viel Homeschooling gemacht wurde haben wir uns darauf geeinigt bei GitHub ein Repository für dieses Projekt zu erstellen. Damit konnten alle beteiligten gleichzeitig an dem Projekt arbeiten und die Arbeit der anderen Teammitglieder sehen und eventuelle Verbesserungen vorschlagen. Da jedoch nicht alle mit Git vertraut waren gab es an dieser Stelle Probleme. Da Dominik Angel aber ausreichend Erfahrung mit Git mitbrachte wurden diese Probleme schnell behoben. Probleme bei Git waren unter anderem das Rebasen.

3 Durchführung

3.1 Implementierung der Sortieralgorithmen

3.1.1 Bubble-Sort Algorithmus

Beim Start der Funktion wird ein unsortiertes Array übergeben. Das erste Array Element wird nun mit seinem Nachbar verglichen. Ist das Nachbarelement kleiner, so werden die Elemente getauscht. Ist es größer oder gleich so bleibt die Reihenfolge bestehen. Das neu entstandene Array wird nun wieder in die Funktion gegeben. Die Schleife wiederholt sich so lange bis nicht mehr getauscht werden kann und die Elemente somit alle sortiert sind.

3.1.2 Merge-Sort Algorithmus

In die Startfunktion für den Merge-Sort wird das unsortierte Array übergeben. Dieses wird so lange in der Hälfte geteilt bis das Array in mehrere, ein bis zwei Felder große Arrays, aufgeteilt wurde. Danach werden die zwei Felder großen Arrays miteinander verglichen. Dabei wird überprüft, ob das Feld Null von links kleiner ist als das rechte. Wenn das der Fall sein sollte, wird das linke Element in ein neues Array gepusht, wenn dem nicht der Fall sein sollte, wird das andere Element in das Array gepusht. Mit dem vollständigen neuen Array wird dann der ganze Prozess wiederholt, bis das neue Array in einer sortierten Reihenfolge besteht.

3.1.3 Quick-Sort Algorithmus

In die Start Funktion für den Quick-Sort wird das unsortierte Array übergeben. Wenn das Array nur ein Feld hat, wird es sofort wieder ausgegeben, wenn nicht werden fünf neue Variablen deklariert. Hierbei handelt es sich um drei neue Arrays (left, right, newArray), die Pivot Zahl und die Länge des Arrays. Jetzt folgt eine Schleife die solang ist wie das unsortierte Array. In dieser Schleife wird überprüft, ob das aktuelle Element des unsortierten Arrays kleiner oder die Pivot Zahl ist. In diesem Fall wird das Element in left gepusht, sonst wird es in right gepusht. Diese Funktion wird im Return sowie beim Zusammenfügen der Arrays in newArray mit left und right nochmal ausgeführt. Dieser ganze Prozess wird so lange wiederholt, bis das newArray in einer sortierten Reihenfolge besteht.

3.2 Implementierung des Frontend

In der HTML-Datei mit dem Titel „Sort algorithm“ werden zwei externe Dateien geladen, einmal „sort.css“ und die „sort.js“ die, die Sortieralgorithmen beinhaltet. Überschrift im Body ist Sortieralgorithmen. Darunter ist ein Paragraf der einen dazu bittet eine beliebige Reihenfolge von Karten auszuwählen. In einem Div-Container mit der ID „selectableCards“ sind 13 Buttons mit der Klasse „cards“ die bei Klick die Funktion „saveCards“ in „sort.js“ mit dem dementsprechenden Wert aufruft. Darauf folgt die zweite Überschrift „Ausgewählte Karten“. Danach sind fünf Div-Container, die jeweils eine ID haben. Die ID´s sind „selectetCards“, „sortCards“, „bubbleSteps“, „mergeSteps“ und „quickSteps“. Darauf folgt ein Button mit der ID „sortButton“ die beim Klicken die Funktion „run“ in „sort.js“ mit der unsortierten Kartenliste ausführt. Abschließend kommt ein Anker mit der ID „loadButton“ und dem Inhalt „Zurücksetzen“ der bei Klick die Seite neu lädt.

In der CSS-Datei „sort.css“ erfolgt die Formatierung der einzelnen HTML Elemente. Hier werden die Position und das Design der Elemente festgelegt.

In der JavaScript-Datei werden zuerst acht Variablen deklariert und initialisiert. Dabei wird „disabled“ auf „false“ gesetzt, „stepCountBubble“, „stepCountMerge“ und „stepCountQuick“ auf 0. Die Variablen „unsortetUnicode“, „cards“ sowie „sortetUnicode“ sind leere Arrays. Die Variable „unicodeArray“ beinhaltet die Unicodes der Karten, in der Reihenfolge Ass bis König, die in der Oberfläche angezeigt werden. Danach folgen die Sortieralgorithmen Bubble-Sort, Merge-Sort und Quick-Sort. Im Anschluss wird die Funktion „saveCards“ implementiert. Dort wird beim Klicken auf eine Karte der übergebene Wert in das „cards“ Array gepusht und die Funktion „getUnicodes“ mit dem Parameter „cards“ aufgerufen und in „unsortedUnicode“ gespeichert. Dieses Array wird dann mithilfe von „innerHTML“ in dem Element mit der ID „selectedCards“ angezeigt. Die Funktion „getUnicodes“ bekommt das unsortierte Karten Array und leert das neue Array „unsortedUnicode“. Mithilfe der Länge des Parameters läuft eine Schleife. In dieser wird der Wert, welcher der Index des Parameters ist, minus eins gerechnet, da der Wert minus eins die Position des Unicodes in „unicodeArray“ ist. Im nächsten Schritt wird der entsprechende Unicode in das Array „unsortedUnicode“ gepusht und außerhalb der Schleife zurückgegeben. Darauf folgt die „getSteps“ Funktion die mit „innerHTML“ die Werte in den Elementen mit der ID „selectetCards“, „sortCards“ und „bubbleSteps“ ausgibt. Zum Schluss kommt noch die „run“ Funktion, die für jeden Sortieralgorithmus eine Kopie der unsortierten Kartenliste macht. Danach werden die Sortieralgorithmen ausgeführt und das Ergebnis der Bubble-Sort Funktion wird in „sortValue“ gespeichert. Dann wird „getSteps“ ausgeführt. Danach wird „get unicodes“ mit „sortValue“ aufgerufen und in „sortCards“ gespeichert. Das Array „sortCards“ wird anschließend mit „innerHTML“ im Element mit der ID „sortCards“ ausgegeben. Zum Schluss wird der „sortButton“ deaktiviert.

4 Funktionsweise

4.1 Technische Funktionsweise

Wenn eine Karte ausgewählt wird, wird diese durch die Funktion „saveCards“ in einem Array gespeichert. In dieser Funktion wird jedes Mal, wenn eine Karte ausgewählt wird die Funktion „getUnicodes“ ausführt. Dort wird der dementsprechende Unicode für den Wert rausgesucht und in einem Array gespeichert. Nach der Ausführung der Funktion „getUnicodes“ wird das dadurch gespeichert Array angezeigt. Wird in der Oberfläche auf Sortieren geklickt, wird die „run“ Funktion ausgeführt und die Sortieralgorithmen mit einer Kopie der unsortierten Kartenliste aufgerufen. Danach wird „getSteps“ ausgeführt die die Schritte der Sortieralgorithmen mit „getElementByID().innerHTML“ ausgibt. Mit den sortierten Karten wird nochmal „getUnicodes“ aufgerufen, um die Unicodes der sortierten Karten zu speichern. Dieses Array mit den sortierten Unicodes werden wieder mit „getElementByID().innerHTML“ angezeigt. Zum Schluss wird der Sortier-Button mit „getElementByID().disabled = true“ deaktiviert.

4.2 Nutzergerechte Funktionsweise

Wir haben eine Webanwendung geschrieben, die einfach nutzbar und verständlich ist. Wenn die Seite aufgerufen wird, sehen wir Spielkarten von Ass bis König. Man kann in einer beliebigen Reihenfolge beliebig viele Karten auswählen. Diese werden darunter angezeigt. Wir haben einen Button, der die drei verschiedenen Sortieralgorithmen ausführt. Wenn dieser Button betätigt worden, wird unter der unsortierten Kartenliste die sortierte Kartenliste angezeigt. Dazu kommen drei Felder, die die Anzahl der Schritte für den jeweiligen Sortieralgorithmus beinhaltet. Dies soll visualisieren welcher von den drei Sortieralgorithmen der Effizienteste ist. Dazu kommt noch der Button „Zurücksetzen“ welcher alles wieder auf den Ausgangszustand zurücksetzt.

5 Fazit

Wir haben eine benutzerfreundliche Webanwendung entwickelt, welche durch die Sortierung von Karten die Effizienz der Sortieralgorithmen Bubble-Sort, Merge-Sort und Quick-Sort verdeutlicht. Man erkennt in der Anwendung an den Schritten, die die einzelnen Algorithmen benötigen, dass der Bubble-Sort Algorithmus der Effizienteste ist, da dieser nur wenige Schritte benötigt. Darauf folgt der Quick-Sort Algorithmus und der Ineffizienteste ist der Merge-Sort Algorithmus. Wir fanden das Projekt sehr interessant und auch herausfordernd, da wir ohne weitere Frameworks gearbeitet haben, welche die Arbeit oft erleichtern.

6 Literaturverzeichnis

Als Literarische Hilfe haben wir die Mitschriften benutzt welche wir im Unterricht von Herr Rein angefertigt haben.