Javascript Υλοποίηση Αλγορίθμων

Επιμέλεια: Α.Δρακόπουλος

Υλοποίηση Αλγορίθμων

Ο Ευκλείδειος αλγόριθμος για την εύρεση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ)

Ο Ευκλείδειος αλγόριθμος για την εύρεση του Μέγιστου Κοινού Διαιρέτη (ΜΚΔ) δύο αριθμών είναι ένας πολύ αποδοτικός αλγόριθμος που βασίζεται στην επαναληπτική διαίρεση. Ο αλγόριθμος αυτός επαναλαμβάνει τη διαδικασία διαίρεσης μέχρι να βρεθεί το υπόλοιπο ίσο με το μηδέν. Ο τελευταίος μη μηδενικός διαιρέτης είναι ο ΜΚΔ.

Παρακάτω δίνεται ένα ολοκληρωμένο πρόγραμμα σε JavaScript, HTML και CSS που υλοποιεί τον Ευκλείδειο αλγόριθμο και παρουσιάζει τη διαδικασία βήμα προς βήμα στο browser.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ευκλείδειος Αλγόριθμος ΜΚΔ</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            width: 300px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ccc;
            border-radius: 10px;
```

```
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
        }
        input {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        button {
            width: 100%;
            padding: 10px;
            background-color: #28a745;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #218838;
        }
        .steps {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #ddd;
            background-color: #f9f9f9;
            border-radius: 5px;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Ευκλείδειος Αλγόριθμος ΜΚΔ</h2>
    <input type="number" id="num1" placeholder="Εισάγετε τον πρώτο αριθμό" />
    <input type="number" id="num2" placeholder="Εισάγετε τον δεύτερο αριθμό" />
    <button onclick="findGCD()">Βρες τον ΜΚΔ</button>
    <div id="steps" class="steps">Η διαδικασία θα εμφανιστεί εδώ...</div>
</div>

<script src="gcd.js"></script>
```

```
</body>
</html>
```

JavaScript (gcd.js)

```
// Συνάρτηση για την εύρεση του Μέγιστου Κοινού Διαιρέτη χρησιμοποιώντας τον Ευκλείδειο αλγόριθμο.
function findGCD() {
    // Λήψη των τιμών από τα πεδία εισόδου.
    let num1 = parseInt(document.getElementById('num1').value);
    let num2 = parseInt(document.getElementById('num2').value);

    // Έλεγχος αν έχουν εισαχθεί τιμές.
    if (isNaN(num1) || isNaN(num2)) {
        document.getElementById('steps').innerHTML = "Παρακαλώ εισάγετε και τους δύο αριθμούς.";
        return;
    }

    // Μεταβλητή για να αποθηκεύσουμε τη διαδικασία.
    let steps = "";

    // Εκτελούμε τον Ευκλείδειο Αλγόριθμο και αποθηκεύουμε κάθε βήμα.
    let a = num1;
    let b = num2;

    steps += `Ξεκινάμε με a = ${a}, b = ${b}<br>`;

    while (b !== 0) {
        let remainder = a % b;
        steps += `a = ${a}, b = ${b}, υπόλοιπο = ${remainder}<br>`;
        a = b;
        b = remainder;
    }

    steps += `<strong>Ο ΜΚΔ είναι: ${a}</strong>`;

    // Εμφάνιση των βημάτων στο HTML.
    document.getElementById('steps').innerHTML = steps;
}
```

Επεξήγηση του Προγράμματος

1. HTML:

Η HTML σελίδα έχει ένα απλό interface όπου ο χρήστης μπορεί να εισάγει δύο αριθμούς.

Υπάρχουν δύο input πεδία για τους δύο αριθμούς και ένα κουμπί που εκτελεί τη διαδικασία εύρεσης του ΜΚΔ όταν πατηθεί.

Ένα div με id steps χρησιμοποιείται για να εμφανίζει τη διαδικασία του Ευκλείδειου αλγορίθμου βήμα προς βήμα.

2. CSS:

Η CSS εφαρμόζει βασική μορφοποίηση για το layout και styling της σελίδας.

Η σελίδα είναι κεντραρισμένη και οι φόρμες έχουν καθαρό στυλ με σκιά και στρογγυλεμένες γωνίες.

Το κουμπί έχει ένα ελαφρύ εφέ hover για καλύτερη εμπειρία χρήστη.

3. JavaScript:

Η συνάρτηση findGCD() καλείται όταν ο χρήστης πατά το κουμπί "Βρες τον ΜΚΔ".

Η τιμή των αριθμών λαμβάνεται από τα πεδία εισόδου και αποθηκεύεται στις μεταβλητές num1 και num2.

Η διαδικασία εύρεσης του ΜΚΔ εκτελείται χρησιμοποιώντας τον Ευκλείδειο αλγόριθμο, και κάθε βήμα της διαδικασίας καταγράφεται και εμφανίζεται στον χρήστη.

Το τελικό αποτέλεσμα (ΜΚΔ) εμφανίζεται με έντονα γράμματα στο τέλος της διαδικασίας.

Ευκλείδειος Αλγόριθμος

Ο αλγόριθμος βασίζεται στην εξής λογική:

1. Ξεκινάμε με δύο αριθμούς a και b.

2. Υπολογίζουμε το υπόλοιπο της διαίρεσης του a με το b (δηλαδή a % b).

3. Αν το υπόλοιπο είναι μηδέν, ο ΜΚΔ είναι το b. Αν όχι, αντικαθιστούμε τον a με τον b και τον b με το υπόλοιπο.

4. Επαναλαμβάνουμε τη διαδικασία μέχρι το υπόλοιπο να γίνει μηδέν.

Παράδειγμα Χρήσης

Εισάγοντας τους αριθμούς 56 και 98, η διαδικασία που θα εμφανιστεί θα είναι:

Ξεκινάμε με a = 56, b = 98
a = 56, b = 98, υπόλοιπο = 56
a = 98, b = 56, υπόλοιπο = 42
a = 56, b = 42, υπόλοιπο = 14
a = 42, b = 14, υπόλοιπο = 0
Ο ΜΚΔ είναι: 14

Συμπέρασμα

Αυτό το πρόγραμμα παρουσιάζει τη χρήση του Ευκλείδειου αλγορίθμου με έναν απλό και διαδραστικό τρόπο, ο οποίος επιτρέπει στον χρήστη να εισάγει δύο αριθμούς και να παρακολουθεί τη διαδικασία υπολογισμού του ΜΚΔ βήμα προς βήμα. Το περιβάλλον είναι φιλικό και εύχρηστο, και η JavaScript εξασφαλίζει την άμεση ανταπόκριση στα δεδομένα που εισάγει ο χρήστης.

Ο αλγόριθμος Selection Sort

Ο αλγόριθμος Selection Sort είναι ένας αλγόριθμος ταξινόμησης που λειτουργεί βρίσκοντας το μικρότερο στοιχείο σε μια μη ταξινομημένη περιοχή και τοποθετώντας το στην αρχή της λίστας. Η διαδικασία αυτή επαναλαμβάνεται για κάθε στοιχείο της λίστας μέχρι να ταξινομηθεί πλήρως. Ο αλγόριθμος Selection Sort δεν είναι πολύ αποδοτικός για μεγάλες λίστες, καθώς έχει πολυπλοκότητα $O(n^2)$, αλλά είναι εύκολος στην κατανόηση και υλοποίηση.

Παρακάτω θα γράψουμε έναν κώδικα JavaScript που θα υλοποιεί τον αλγόριθμο Selection Sort, ενώ θα παρουσιάζει κάθε βήμα της διαδικασίας μέσα στον browser. Για να είναι πιο κατανοητή η διαδικασία, θα χρησιμοποιηθεί HTML και CSS για την εμφάνιση της λίστας και την ενημέρωση των βημάτων της ταξινόμησης.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Selection Sort Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            width: 600px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ccc;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
```

```
    text-align: center;
}
input, button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #28a745;
    color: white;
    cursor: pointer;
}
button:hover {
    background-color: #218838;
}
.steps {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    background-color: #f9f9f9;
    border-radius: 5px;
    max-height: 300px;
    overflow-y: scroll;
}
.list {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}
.list div {
    padding: 10px;
    border: 1px solid #ccc;
    width: 50px;
    text-align: center;
    background-color: #f0f0f0;
    border-radius: 5px;
}
.highlight {
    background-color: #ffeb3b;
}
.sorted {
    background-color: #8bc34a;
```

```
            color: white;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Selection Sort Visualization</h2>
    <input type="text" id="arrayInput" placeholder="Εισάγετε αριθμούς χωρισμένους με κόμμα" />
    <button onclick="startSort()">Εκκίνηση Ταξινόμησης</button>

    <div class="list" id="arrayDisplay"></div>

    <div id="steps" class="steps">Εδώ θα εμφανιστούν τα βήματα του αλγορίθμου...</div>
</div>

<script src="selectionsort.js"></script>

</body>
</html>
```

JavaScript (selectionsort.js)

```
// Συνάρτηση για την εκκίνηση της ταξινόμησης.
function startSort() {
    // Παίρνουμε τους αριθμούς από το input.
    let input = document.getElementById("arrayInput").value;
    let array = input.split(',').map(Number);

    // Εμφανίζουμε το αρχικό array στο browser.
    displayArray(array);

    // Εκκίνηση του αλγορίθμου selection sort.
    selectionSort(array);
}

// Συνάρτηση που εμφανίζει τον πίνακα (array) στο browser.
function displayArray(array, highlightIndex = -1, sortedIndices = []) {
    let arrayDisplay = document.getElementById("arrayDisplay");
    arrayDisplay.innerHTML = "";  // Καθαρίζουμε την προηγούμενη εμφάνιση.

    array.forEach((value, index) => {
        let element = document.createElement("div");
```

```
        element.textContent = value;

        // Χρωματίζουμε το στοιχείο που είναι υπό εξέταση.
        if (index === highlightIndex) {
            element.classList.add("highlight");
        }

        // Χρωματίζουμε τα στοιχεία που είναι ήδη ταξινομημένα.
        if (sortedIndices.includes(index)) {
            element.classList.add("sorted");
        }

        arrayDisplay.appendChild(element);
    });
}

// Συνάρτηση για να προσθέσουμε τα βήματα στη λίστα των βημάτων.
function addStep(step) {
    let stepsDiv = document.getElementById("steps");
    stepsDiv.innerHTML += step + "<br>";
}

// Συνάρτηση που υλοποιεί τον αλγόριθμο Selection Sort.
async function selectionSort(array) {
    let n = array.length;

    // Πίνακας με τους ταξινομημένους δείκτες.
    let sortedIndices = [];

    for (let i = 0; i < n - 1; i++) {
        // Αρχικά, υποθέτουμε ότι το ελάχιστο στοιχείο βρίσκεται στη θέση i.
        let minIndex = i;

        // Εμφανίζουμε το array με τον τρέχοντα δείκτη.
        displayArray(array, minIndex, sortedIndices);
        addStep(`Βήμα ${i + 1}: Ψάχνουμε για το μικρότερο στοιχείο από τη θέση ${i} και μετά.`);

        // Αναζητούμε το ελάχιστο στοιχείο στον υπόλοιπο πίνακα.
        for (let j = i + 1; j < n; j++) {
            if (array[j] < array[minIndex]) {
                minIndex = j;
            }

            // Εμφάνιση του array με το νέο στοιχείο υπό εξέταση.
```

```
            displayArray(array, j, sortedIndices);
            await delay(500); // Καθυστέρηση για να δούμε το βήμα.
        }

        // Αν το μικρότερο στοιχείο δεν είναι ήδη στη θέση i, κάνουμε την ανταλλαγή.
        if (minIndex !== i) {
            let temp = array[i];
            array[i] = array[minIndex];
            array[minIndex] = temp;

            addStep(`Ανταλλαγή του στοιχείου στη θέση ${i} με το ελάχιστο στη θέση ${minIndex}.`);
        } else {
            addStep(`Το στοιχείο στη θέση ${i} είναι ήδη το μικρότερο.`);
        }

        // Προσθήκη του τρέχοντος στοιχείου στους ταξινομημένους δείκτες.
        sortedIndices.push(i);
        displayArray(array, -1, sortedIndices);
        await delay(500);
    }

    // Το τελευταίο στοιχείο είναι ήδη ταξινομημένο.
    sortedIndices.push(n - 1);
    displayArray(array, -1, sortedIndices);
    addStep(`Η ταξινόμηση ολοκληρώθηκε!`);
}

// Συνάρτηση για καθυστέρηση (χρησιμοποιείται για να βλέπουμε τα βήματα αργά).
function delay(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
```

Επεξήγηση του Κώδικα

1. HTML:

Υπάρχει ένα πεδίο εισαγωγής όπου ο χρήστης μπορεί να εισάγει μια λίστα αριθμών, διαχωρισμένων με κόμμα (,).

Ένα κουμπί "Εκκίνηση Ταξινόμησης" ενεργοποιεί την εκτέλεση του αλγορίθμου και εμφανίζει τα βήματα της ταξινόμησης.

Οι αριθμοί εμφανίζονται δυναμικά στο div με id arrayDisplay, και η διαδικασία της ταξινόμησης εμφανίζεται στο div με id steps.

2. CSS:

Βασική μορφοποίηση του περιβάλλοντος χρήστη με σκιές, στρογγυλεμένες γωνίες και χρωματισμένα κουμπιά.

Τα στοιχεία του πίνακα είναι εμφανισμένα ως κουτιά σε οριζόντια σειρά, και τα επιλεγμένα στοιχεία (υπό εξέταση ή ταξινομημένα) χρωματίζονται αντίστοιχα.

3. JavaScript (selectionsort.js) :

Η συνάρτηση displayArray() χρησιμοποιείται για να εμφανίζει τον πίνακα στο browser, ανανεώνοντας κάθε φορά την κατάσταση του πίνακα κατά τη διάρκεια της εκτέλεσης του αλγορίθμου.

Μπορούμε να επισημάνουμε ένα συγκεκριμένο στοιχείο χρησιμοποιώντας τη μεταβλητή highlightIndex, η οποία το χρωματίζει κίτρινο.

Τα στοιχεία που έχουν ήδη ταξινομηθεί αποθηκεύονται στο sortedIndices και χρωματίζονται πράσινα.

Η συνάρτηση addStep() προσθέτει τα βήματα του αλγορίθμου σε μορφή κειμένου μέσα στο div που εμφανίζει τα βήματα της διαδικασίας, ώστε ο χρήστης να μπορεί να παρακολουθεί κάθε κίνηση του αλγορίθμου.

Η συνάρτηση selectionSort() περιέχει την υλοποίηση του αλγορίθμου Selection Sort:

Για κάθε θέση i του πίνακα, ο αλγόριθμος υποθέτει ότι το μικρότερο στοιχείο είναι αυτό που βρίσκεται στη θέση i.

Ψάχνει στον υπόλοιπο πίνακα για να βρει το μικρότερο στοιχείο και, αν βρει κάποιο μικρότερο, το ανταλλάσσει με το στοιχείο στη θέση i.

Μετά από κάθε βήμα της ταξινόμησης, το στοιχείο που τοποθετείται σωστά προστίθεται στο sortedIndices για να εμφανίζεται ως ταξινομημένο.

Η λειτουργία καθυστέρησης delay() χρησιμοποιείται για να επιτρέψει στον χρήστη να βλέπει τη διαδικασία της ταξινόμησης σταδιακά, χωρίς να εκτελείται στιγμιαία όλη η διαδικασία.

Τέλος, η συνάρτηση delay() δημιουργεί μια καθυστέρηση με τη χρήση Promise για να παρουσιάζει τα βήματα με χρονική διαφορά.

Παράδειγμα Λειτουργίας

Ας υποθέσουμε ότι ο χρήστης εισάγει τον ακόλουθο πίνακα αριθμών: 29, 10, 14, 37, 13.

Η διαδικασία που θα εμφανιστεί έχει ως εξής:

1. Βήμα 1: Ψάχνουμε το μικρότερο στοιχείο από τη θέση 0 και μετά.

Επισημαίνεται το στοιχείο 29, και το μικρότερο στοιχείο είναι το 10. Τα δύο στοιχεία ανταλλάσσονται.

Νέος πίνακας: [10, 29, 14, 37, 13]

2. Βήμα 2: Ψάχνουμε το μικρότερο στοιχείο από τη θέση 1 και μετά.

Επισημαίνεται το στοιχείο 29, και το μικρότερο στοιχείο είναι το 13. Τα δύο στοιχεία ανταλλάσσονται.

Νέος πίνακας: [10, 13, 14, 37, 29]

3. Βήμα 3: Ψάχνουμε το μικρότερο στοιχείο από τη θέση 2 και μετά.

Επισημαίνεται το στοιχείο 14. Είναι ήδη το μικρότερο, οπότε δεν γίνεται καμία αλλαγή.

Νέος πίνακας: [10, 13, 14, 37, 29]

4. Βήμα 4: Ψάχνουμε το μικρότερο στοιχείο από τη θέση 3 και μετά.

Επισημαίνεται το στοιχείο 37, και το μικρότερο στοιχείο είναι το 29. Τα δύο στοιχεία ανταλλάσσονται.

Νέος πίνακας: [10, 13, 14, 29, 37]

5. Βήμα 5: Ο πίνακας είναι ταξινομημένος, καθώς το τελευταίο στοιχείο είναι ήδη στη θέση του.

Συμπέρασμα

Αυτό το παράδειγμα δείχνει τον τρόπο λειτουργίας του αλγορίθμου Selection Sort με μια οπτικοποίηση των βημάτων του μέσα σε έναν browser. Ο αλγόριθμος εμφανίζει κάθε βήμα ξεχωριστά, επισημαίνοντας το στοιχείο που εξετάζεται και ενημερώνοντας τον χρήστη για την τρέχουσα κατάσταση του πίνακα. Η υλοποίηση αυτή χρησιμοποιεί JavaScript για την επεξεργασία και προβολή των βημάτων, ενώ η HTML και η CSS χρησιμοποιούνται για την εμφάνιση και το στυλ της διεπαφής χρήστη.

Αυτός ο τύπος αλγορίθμου είναι εύκολος στην κατανόηση και μπορεί να βοηθήσει τους χρήστες να μάθουν πώς λειτουργούν οι βασικοί αλγόριθμοι ταξινόμησης.

Ο αλγόριθμος Insertion Sort

Ο αλγόριθμος Insertion Sort είναι ένας απλός αλγόριθμος ταξινόμησης που λειτουργεί με τον εξής τρόπο: διατρέχει τη λίστα από αριστερά προς τα δεξιά, τοποθετώντας κάθε στοιχείο στην σωστή του θέση σε μια ήδη ταξινομημένη υπολίστα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να ταξινομηθεί ολόκληρη η λίστα. Ο αλγόριθμος είναι ιδιαίτερα αποδοτικός για μικρούς πίνακες ή για πίνακες που είναι σχεδόν ταξινομημένοι.

Παρακάτω δίνεται ένα ολοκληρωμένο πρόγραμμα σε JavaScript, HTML και CSS που υλοποιεί τον αλγόριθμο Insertion Sort και παρουσιάζει τα βήματα της διαδικασίας μέσα στον browser.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Insertion Sort Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            width: 600px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ccc;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
        }
```

```
input, button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #28a745;
    color: white;
    cursor: pointer;
}
button:hover {
    background-color: #218838;
}
.steps {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    background-color: #f9f9f9;
    border-radius: 5px;
    max-height: 300px;
    overflow-y: scroll;
}
.list {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}
.list div {
    padding: 10px;
    border: 1px solid #ccc;
    width: 50px;
    text-align: center;
    background-color: #f0f0f0;
    border-radius: 5px;
}
.highlight {
    background-color: #ffeb3b;
}
.sorted {
    background-color: #8bc34a;
    color: white;
}
```

```
    </style>
</head>
<body>

<div class="container">
    <h2>Insertion Sort Visualization</h2>
    <input type="text" id="arrayInput" placeholder="Εισάγετε αριθμούς χωρισμένους με κόμμα" />
    <button onclick="startSort()">Εκκίνηση Ταξινόμησης</button>

    <div class="list" id="arrayDisplay"></div>

    <div id="steps" class="steps">Εδώ θα εμφανιστούν τα βήματα του αλγορίθμου...</div>
</div>

<script src="insertionsort.js"></script>

</body>
</html>
```

JavaScript (insertionsort.js)

```
// Συνάρτηση για την εκκίνηση της ταξινόμησης.
function startSort() {
    // Παίρνουμε τους αριθμούς από το input.
    let input = document.getElementById("arrayInput").value;
    let array = input.split(',').map(Number);

    // Εμφανίζουμε το αρχικό array στο browser.
    displayArray(array);

    // Εκκίνηση του αλγορίθμου insertion sort.
    insertionSort(array);
}

// Συνάρτηση που εμφανίζει τον πίνακα (array) στο browser.
function displayArray(array, highlightIndex = -1, sortedIndices = []) {
    let arrayDisplay = document.getElementById("arrayDisplay");
    arrayDisplay.innerHTML = "";  // Καθαρίζουμε την προηγούμενη εμφάνιση.

    array.forEach((value, index) => {
        let element = document.createElement("div");
        element.textContent = value;
```

```
        // Χρωματίζουμε το στοιχείο που είναι υπό εξέταση.
        if (index === highlightIndex) {
            element.classList.add("highlight");
        }

        // Χρωματίζουμε τα στοιχεία που είναι ήδη ταξινομημένα.
        if (sortedIndices.includes(index)) {
            element.classList.add("sorted");
        }

        arrayDisplay.appendChild(element);
    });
}

// Συνάρτηση για να προσθέσουμε τα βήματα στη λίστα των βημάτων.
function addStep(step) {
    let stepsDiv = document.getElementById("steps");
    stepsDiv.innerHTML += step + "<br>";
}

// Συνάρτηση που υλοποιεί τον αλγόριθμο Insertion Sort.
async function insertionSort(array) {
    let sortedIndices = [];

    for (let i = 1; i < array.length; i++) {
        let currentValue = array[i];
        let j = i - 1;

        // Προσθέτουμε τα βήματα της αρχικής επιλογής του στοιχείου.
        addStep(`Βήμα ${i}: Εξετάζουμε το στοιχείο ${currentValue}.`);

        // Κίνηση του στοιχείου προς τα αριστερά όσο είναι μικρότερο από τα προηγούμενα.
        while (j >= 0 && array[j] > currentValue) {
            array[j + 1] = array[j];
            displayArray(array, j, sortedIndices);
            await delay(500);  // Καθυστέρηση για να βλέπουμε τα βήματα.
            j--;
        }
        array[j + 1] = currentValue;

        // Ενημέρωση του array και των ταξινομημένων στοιχείων.
        sortedIndices.push(i);
        displayArray(array, -1, sortedIndices);
        addStep(`Τοποθετούμε το στοιχείο ${currentValue} στη σωστή του θέση.`);
```

```
        await delay(500);
    }

    addStep("Η ταξινόμηση ολοκληρώθηκε!");
}

// Συνάρτηση για καθυστέρηση (χρησιμοποιείται για να βλέπουμε τα βήματα αργά).
function delay(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
```

Επεξήγηση του Κώδικα

1. HTML:

Ένα πεδίο εισαγωγής επιτρέπει στον χρήστη να εισάγει μια λίστα αριθμών, διαχωρισμένων με κόμμα.

Ένα κουμπί "Εκκίνηση Ταξινόμησης" ενεργοποιεί τη διαδικασία εκτέλεσης του αλγορίθμου.

Τα βήματα της διαδικασίας εμφανίζονται στο div με id steps, και τα στοιχεία της λίστας εμφανίζονται στο div με id arrayDisplay.

2. CSS:

Παρέχεται βασική μορφοποίηση του περιβάλλοντος χρήστη για μια καθαρή και ευχάριστη εμφάνιση.

Τα στοιχεία του πίνακα εμφανίζονται ως κουτιά σε οριζόντια σειρά, και τα ενεργά στοιχεία χρωματίζονται αντίστοιχα.

Τα ταξινομημένα στοιχεία εμφανίζονται πράσινα, ενώ τα στοιχεία υπό επεξεργασία εμφανίζονται κίτρινα.

3. JavaScript (insertionsort.js):

Η συνάρτηση startSort() παίρνει την είσοδο του χρήστη, τη μετατρέπει σε πίνακα και ξεκινά τον αλγόριθμο ταξινόμησης.

Η συνάρτηση displayArray() εμφανίζει τα στοιχεία του πίνακα στον browser. Επισημαίνει τα στοιχεία που βρίσκονται υπό επεξεργασία ή είναι ήδη ταξινομημένα.

Η συνάρτηση addStep() προσθέτει περιγραφή των βημάτων της διαδικασίας στη λίστα.

Η συνάρτηση insertionSort() υλοποιεί τον αλγόριθμο Insertion Sort:

Για κάθε στοιχείο, τοποθετείται στη σωστή του θέση στην ήδη ταξινομημένη υπολίστα αριστερά του.

Το στοιχείο που εξετάζεται μετακινείται αριστερά όσο είναι μικρότερο από τα προηγούμενα στοιχεία, και κάθε κίνηση εμφανίζεται στον browser.

Η συνάρτηση delay() προσθέτει μια καθυστέρηση μεταξύ των βημάτων, ώστε να μπορούμε να παρακολουθούμε τη διαδικασία σε πραγματικό χρόνο.

Παράδειγμα Λειτουργίας

Ας υποθέσουμε ότι ο χρήστης εισάγει τους αριθμούς: 12, 11, 13, 5, 6.

Η διαδικασία του Insertion Sort θα εκτελεστεί ως εξής:

1. Βήμα 1: Το πρώτο στοιχείο, 12, θεωρείται ήδη ταξινομημένο.

Δεν γίνεται καμία αλλαγή, καθώς είναι το πρώτο στοιχείο.

Ταξινομημένος πίνακας: [12]

2. Βήμα 2: Το δεύτερο στοιχείο είναι 11.

Το 11 συγκρίνεται με το 12 και επειδή είναι μικρότερο, τοποθετείται μπροστά από το 12.

Ταξινομημένος πίνακας: [11, 12]

3. Βήμα 3: Το τρίτο στοιχείο είναι 13.

Το 13 συγκρίνεται με το 12 και επειδή είναι μεγαλύτερο, παραμένει στη θέση του.

Ταξινομημένος πίνακας: [11, 12, 13]

4. Βήμα 4: Το τέταρτο στοιχείο είναι 5.

Το 5 συγκρίνεται με το 13, το 12, και το 11. Επειδή είναι μικρότερο από όλα, μετακινείται στην αρχή του πίνακα.

Ταξινομημένος πίνακας: [5, 11, 12, 13]

5. Βήμα 5: Το πέμπτο στοιχείο είναι 6.

Το 6 συγκρίνεται με το 13, το 12, και το 11. Τοποθετείται μεταξύ του 5 και του 11.

Ταξινομημένος πίνακας: [5, 6, 11, 12, 13]

Στο τέλος της διαδικασίας, ο πίνακας θα είναι πλήρως ταξινομημένος.

Λειτουργία στον Browser

Κατά τη διάρκεια της εκτέλεσης του αλγορίθμου:

Ο πίνακας θα εμφανίζεται σε πραγματικό χρόνο με τα στοιχεία που εξετάζονται να χρωματίζονται κίτρινα και τα ήδη ταξινομημένα στοιχεία να γίνονται πράσινα.

Τα βήματα της διαδικασίας θα εμφανίζονται στη λίστα με κείμενο, για να είναι εύκολο να παρακολουθήσουμε τι συμβαίνει σε κάθε στάδιο.

Συμπέρασμα

Αυτό το παράδειγμα υλοποιεί τον αλγόριθμο Insertion Sort σε JavaScript με HTML και CSS για οπτικοποίηση στον browser. Καθώς η ταξινόμηση εκτελείται, εμφανίζονται όλες οι ενδιάμεσες καταστάσεις του πίνακα και κάθε βήμα περιγράφεται αναλυτικά. Η προσθήκη της καθυστέρησης (με την delay()) επιτρέπει την οπτική παρακολούθηση των αλλαγών βήμα προς βήμα, καθιστώντας την εφαρμογή κατάλληλη για την εκμάθηση και κατανόηση του τρόπου λειτουργίας του αλγορίθμου.

Ο αλγόριθμος Binary Search (Δυαδική Αναζήτηση)

Ο αλγόριθμος Binary Search (Δυαδική Αναζήτηση) είναι ένας αποδοτικός τρόπος εύρεσης ενός στοιχείου σε έναν ταξινομημένο πίνακα. Αντί να διατρέχει τον πίνακα σειριακά, ο αλγόριθμος χωρίζει τον πίνακα στα δύο και εξετάζει αν το ζητούμενο στοιχείο βρίσκεται στο αριστερό ή στο δεξί υποπίνακα, επαναλαμβάνοντας τη διαδικασία έως ότου βρει το στοιχείο ή εξαντλήσει τις πιθανότητες.

Παρακάτω θα δημιουργήσουμε ένα πρόγραμμα σε JavaScript που υλοποιεί τον αλγόριθμο Binary Search και θα παρουσιάσουμε τη διαδικασία βήμα προς βήμα στον browser, ώστε να φαίνεται η λειτουργία του.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Binary Search Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            width: 600px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ccc;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
```

```
}
input, button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #28a745;
    color: white;
    cursor: pointer;
}
button:hover {
    background-color: #218838;
}
.steps {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    background-color: #f9f9f9;
    border-radius: 5px;
    max-height: 300px;
    overflow-y: scroll;
}
.list {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}
.list div {
    padding: 10px;
    border: 1px solid #ccc;
    width: 50px;
    text-align: center;
    background-color: #f0f0f0;
    border-radius: 5px;
}
.highlight {
    background-color: #ffeb3b;
}
.left {
    background-color: #f44336;
    color: white;
```

```
        }
        .right {
            background-color: #2196f3;
            color: white;
        }
        .found {
            background-color: #4caf50;
            color: white;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Binary Search Visualization</h2>
    <input type="text" id="arrayInput" placeholder="Εισάγετε ταξινομημένους αριθμούς
χωρισμένους με κόμμα" />
    <input type="number" id="targetInput" placeholder="Εισάγετε τον αριθμό προς αναζήτηση" />
    <button onclick="startSearch()">Ξεκινήστε την αναζήτηση</button>

    <div class="list" id="arrayDisplay"></div>

    <div id="steps" class="steps">Εδώ θα εμφανιστούν τα βήματα του αλγορίθμου...</div>
</div>

<script src="binarysearch.js"></script>

</body>
</html>
```

JavaScript (binarysearch.js)

```
// Συνάρτηση που ξεκινά τη δυαδική αναζήτηση
function startSearch() {
    // Παίρνουμε τους αριθμούς από το input και τον αριθμό προς αναζήτηση.
    let input = document.getElementById("arrayInput").value;
    let target = parseInt(document.getElementById("targetInput").value);
    let array = input.split(',').map(Number);

    // Εμφανίζουμε το αρχικό array στο browser.
    displayArray(array);

    // Ξεκινάμε τον αλγόριθμο δυαδικής αναζήτησης.
```

```
    binarySearch(array, target);
}

// Συνάρτηση που εμφανίζει τον πίνακα (array) στο browser
function displayArray(array, highlightIndex = -1, left = -1, right = -1, foundIndex = -1) {
    let arrayDisplay = document.getElementById("arrayDisplay");
    arrayDisplay.innerHTML = "";  // Καθαρίζουμε την προηγούμενη εμφάνιση.

    array.forEach((value, index) => {
        let element = document.createElement("div");
        element.textContent = value;

        // Χρωματίζουμε το στοιχείο που εξετάζεται.
        if (index === highlightIndex) {
            element.classList.add("highlight");
        }

        // Χρωματίζουμε το εύρος των αριστερών στοιχείων.
        if (index < left) {
            element.classList.add("left");
        }

        // Χρωματίζουμε το εύρος των δεξιών στοιχείων.
        if (index > right && right !== -1) {
            element.classList.add("right");
        }

        // Χρωματίζουμε το στοιχείο που έχει βρεθεί.
        if (index === foundIndex) {
            element.classList.add("found");
        }

        arrayDisplay.appendChild(element);
    });
}

// Συνάρτηση για να προσθέσουμε τα βήματα στη λίστα των βημάτων
function addStep(step) {
    let stepsDiv = document.getElementById("steps");
    stepsDiv.innerHTML += step + "<br>";
}

// Συνάρτηση που υλοποιεί τον αλγόριθμο δυαδικής αναζήτησης
async function binarySearch(array, target) {
```

```
    let left = 0;
    let right = array.length - 1;

    addStep(`Αναζητούμε το ${target} στον πίνακα.`);

    while (left <= right) {
        let mid = Math.floor((left + right) / 2);

        // Εμφάνιση της τρέχουσας κατάστασης του array
        displayArray(array, mid, left, right);
        await delay(1000);  // Καθυστέρηση για να παρακολουθήσουμε τα βήματα

        if (array[mid] === target) {
            addStep(`Το στοιχείο ${target} βρέθηκε στη θέση ${mid}.`);
            displayArray(array, -1, left, right, mid);  // Χρωματίζουμε το βρέθηκε στοιχείο
            return;
        } else if (array[mid] < target) {
            addStep(`Το στοιχείο στη μέση (${array[mid]}) είναι μικρότερο από το ${target}.
Αναζητούμε στο δεξί υποπίνακα.`);
            left = mid + 1;  // Αναζήτηση στο δεξί μέρος
        } else {
            addStep(`Το στοιχείο στη μέση (${array[mid]}) είναι μεγαλύτερο από το ${target}.
Αναζητούμε στο αριστερό υποπίνακα.`);
            right = mid - 1;  // Αναζήτηση στο αριστερό μέρος
        }

        await delay(1000);  // Καθυστέρηση για να παρακολουθήσουμε τα βήματα
    }

    addStep(`Το στοιχείο ${target} δεν βρέθηκε στον πίνακα.`);
}

// Συνάρτηση για καθυστέρηση (χρησιμοποιείται για να βλέπουμε τα βήματα αργά)
function delay(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
```

Επεξήγηση του Κώδικα

1. HTML:

Υπάρχει ένα πεδίο εισαγωγής για την είσοδο των αριθμών σε μορφή πίνακα, διαχωρισμένων με κόμμα.

Ένα δεύτερο πεδίο επιτρέπει στον χρήστη να εισάγει τον αριθμό που θέλει να αναζητήσει.

Ένα κουμπί "Ξεκινήστε την αναζήτηση" εκτελεί τον αλγόριθμο και τα βήματα εμφανίζονται στο κάτω μέρος της σελίδας.

Τα στοιχεία του πίνακα εμφανίζονται οπτικά σε οριζόντια διάταξη με χρωματισμούς για καλύτερη παρακολούθηση των βημάτων.

2. CSS :

Τα δεξιά στοιχεία που δεν εξετάζονται πια γίνονται μπλε.

Το στοιχείο που εξετάζεται στη μέση του πίνακα χρωματίζεται κίτρινο.

Αν το στοιχείο βρεθεί, χρωματίζεται πράσινο.

3. JavaScript (binarysearch.js):

Η συνάρτηση startSearch():

Παίρνει τους αριθμούς από το πεδίο εισαγωγής και τους μετατρέπει σε έναν πίνακα αριθμών.

Ξεκινάει τη διαδικασία της αναζήτησης καλώντας τη binarySearch().

Εμφανίζει τον πίνακα οπτικά στον browser με την displayArray().

Η συνάρτηση displayArray():

Εμφανίζει τα στοιχεία του πίνακα σε κουτιά στο div με id arrayDisplay.

Χρησιμοποιεί διαφορετικούς χρωματισμούς για τα στοιχεία που βρίσκονται στα αριστερά και δεξιά του τρέχοντος στοιχείου (χρωματισμένα με κόκκινο και μπλε αντίστοιχα).

Χρωματίζει το στοιχείο που εξετάζεται (highlight) και το στοιχείο που βρέθηκε (found).

Η συνάρτηση addStep():

Προσθέτει μια περιγραφή του βήματος στη λίστα των βημάτων (div με id steps), ώστε να είναι ξεκάθαρα τα βήματα που ακολουθούνται σε κάθε φάση του αλγορίθμου.

Η συνάρτηση binarySearch():

Υλοποιεί τον αλγόριθμο δυαδικής αναζήτησης (Binary Search):

1. Ξεκινά με τον πίνακα και ορίζει τις αρχικές τιμές των μεταβλητών left και right για το αριστερό και δεξιό άκρο του πίνακα.

2. Σε κάθε βήμα υπολογίζει τη μέση τιμή (mid) του πίνακα και συγκρίνει το στοιχείο της μέσης με τον στόχο (target).

3. Αν το στοιχείο στη μέση είναι ίσο με το target, ο αλγόριθμος σταματά και το στοιχείο χρωματίζεται πράσινο.

4. Αν το στοιχείο στη μέση είναι μικρότερο από το target, ο αλγόριθμος συνεχίζει να ψάχνει στο δεξί μέρος του πίνακα (αυξάνει τη μεταβλητή left).

5. Αν το στοιχείο στη μέση είναι μεγαλύτερο από το target, ο αλγόριθμος συνεχίζει να ψάχνει στο αριστερό μέρος του πίνακα (μειώνει τη μεταβλητή right).

6. Η διαδικασία συνεχίζεται μέχρι είτε να βρεθεί το στοιχείο, είτε να εξαντληθούν οι πιθανότητες.

Η binarySearch() καλεί τη displayArray() και την addStep() σε κάθε βήμα για να εμφανίσει τα τρέχοντα δεδομένα και την πορεία της αναζήτησης.

Χρησιμοποιείται η delay() για να προστεθεί μια καθυστέρηση (1 δευτερόλεπτο) μεταξύ των βημάτων, ώστε ο χρήστης να μπορεί να παρακολουθήσει τη διαδικασία σε πραγματικό χρόνο.

Η συνάρτηση delay():

Προσθέτει καθυστέρηση μεταξύ των βημάτων της αναζήτησης, ώστε να μπορούν να προβληθούν τα ενδιάμεσα βήματα πριν προχωρήσει το πρόγραμμα στο επόμενο βήμα.

Παράδειγμα Λειτουργίας

Ας υποθέσουμε ότι ο χρήστης εισάγει τον ταξινομημένο πίνακα: 5, 12, 18, 23, 35, 42, 56, 72, 91 και ο στόχος είναι να βρει τον αριθμό 35.

Η διαδικασία που θα εμφανιστεί έχει ως εξής:

1. Βήμα 1: Αναζητούμε το στοιχείο 35.

Εξετάζουμε το στοιχείο στη μέση (42).

Επειδή το 42 είναι μεγαλύτερο από το 35, συνεχίζουμε την αναζήτηση στο αριστερό μισό του πίνακα.

Ο πίνακας εμφανίζεται με τα στοιχεία από το δεξί μισό (δεξιά του 42) να χρωματίζονται μπλε.

2. Βήμα 2: Αναζητούμε στο αριστερό μισό του πίνακα.

Εξετάζουμε το στοιχείο στη νέα μέση (18).

Επειδή το 18 είναι μικρότερο από το 35, συνεχίζουμε την αναζήτηση στο δεξί μισό του πίνακα.

Ο πίνακας εμφανίζεται με τα στοιχεία από το αριστερό μισό (αριστερά του 18) να χρωματίζονται κόκκινα.

3. Βήμα 3: Αναζητούμε στο δεξί μισό του πίνακα.

Εξετάζουμε το στοιχείο στη μέση (35).

Το στοιχείο 35 βρέθηκε! Χρωματίζεται πράσινο.

Η διαδικασία σταματά και το πρόγραμμα εμφανίζει ότι το στοιχείο 35 βρέθηκε στη θέση 4.

Συμπέρασμα

Αυτό το πρόγραμμα υλοποιεί τον αλγόριθμο Binary Search με οπτικοποίηση των βημάτων σε πραγματικό χρόνο. Κάθε βήμα της διαδικασίας αναζήτησης εμφανίζεται στον browser, με χρωματισμούς που δείχνουν τα στοιχεία που εξετάζονται και τα διαμερίσματα του πίνακα που απορρίπτονται. Ο κώδικας είναι οργανωμένος ώστε να παρέχει πλήρη επεξήγηση της διαδικασίας, επιτρέποντας στον χρήστη να κατανοήσει την πορεία του αλγορίθμου και να δει πώς ο αλγόριθμος διαιρεί συνεχώς τον πίνακα μέχρι να βρει το ζητούμενο στοιχείο.

Ο αλγόριθμος Merge Sort

Ο αλγόριθμος Merge Sort είναι ένας από τους πιο αποδοτικούς αλγορίθμους ταξινόμησης με πολυπλοκότητα O(n log n). Ο αλγόριθμος αυτός ανήκει στην κατηγορία των διαίρει και βασίλευε (divide and conquer), όπου χωρίζει τον πίνακα σε δύο υποπίνακες, τους ταξινομεί και τους συγχωνεύει ξανά.

Παρακάτω θα δημιουργήσουμε έναν κώδικα που υλοποιεί τον αλγόριθμο Merge Sort σε JavaScript και θα δείχνει την οπτικοποίηση της διαδικασίας σε έναν browser, εμφανίζοντας τα βήματα της διαδικασίας συγχώνευσης και ταξινόμησης.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Merge Sort Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f4;
            margin: 0;
            padding: 20px;
        }
        .container {
            width: 600px;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ccc;
            border-radius: 10px;
            box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
        }
```

```
input, button {
    width: 100%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    background-color: #28a745;
    color: white;
    cursor: pointer;
}
button:hover {
    background-color: #218838;
}
.steps {
    margin-top: 20px;
    padding: 10px;
    border: 1px solid #ddd;
    background-color: #f9f9f9;
    border-radius: 5px;
    max-height: 300px;
    overflow-y: scroll;
}
.list {
    display: flex;
    justify-content: space-between;
    margin-top: 20px;
}
.list div {
    padding: 10px;
    border: 1px solid #ccc;
    width: 50px;
    text-align: center;
    background-color: #f0f0f0;
    border-radius: 5px;
}
.highlight {
    background-color: #ffeb3b;
}
.merged {
    background-color: #8bc34a;
    color: white;
}
```

```
    </style>
</head>
<body>

<div class="container">
    <h2>Merge Sort Visualization</h2>
    <input type="text" id="arrayInput" placeholder="Εισάγετε αριθμούς χωρισμένους με κόμμα" />
    <button onclick="startSort()">Εκκίνηση Ταξινόμησης</button>

    <div class="list" id="arrayDisplay"></div>

    <div id="steps" class="steps">Εδώ θα εμφανιστούν τα βήματα του αλγορίθμου...</div>
</div>

<script src="mergesort.js"></script>

</body>
</html>
```

JavaScript (mergesort.js)

```
// Συνάρτηση για την εκκίνηση της διαδικασίας Merge Sort.
function startSort() {
    // Παίρνουμε τους αριθμούς από το input.
    let input = document.getElementById("arrayInput").value;
    let array = input.split(',').map(Number);

    // Εμφανίζουμε το αρχικό array στο browser.
    displayArray(array);

    // Εκκίνηση του αλγορίθμου Merge Sort.
    mergeSort(array, 0, array.length - 1);
}

// Συνάρτηση που εμφανίζει τον πίνακα (array) στο browser.
function displayArray(array, highlightIndices = []) {
    let arrayDisplay = document.getElementById("arrayDisplay");
    arrayDisplay.innerHTML = "";  // Καθαρίζουμε την προηγούμενη εμφάνιση.

    array.forEach((value, index) => {
        let element = document.createElement("div");
        element.textContent = value;
```

```
        // Χρωματίζουμε τα στοιχεία που βρίσκονται υπό επεξεργασία.
        if (highlightIndices.includes(index)) {
            element.classList.add("highlight");
        }

        arrayDisplay.appendChild(element);
    });
}

// Συνάρτηση που προσθέτει τα βήματα στη λίστα των βημάτων.
function addStep(step) {
    let stepsDiv = document.getElementById("steps");
    stepsDiv.innerHTML += step + "<br>";
}

// Συνάρτηση για τον αλγόριθμο Merge Sort.
async function mergeSort(arr, left, right) {
    if (left >= right) {
        return; // Σταματάμε αν το τμήμα έχει μόνο ένα στοιχείο.
    }

    const mid = Math.floor((left + right) / 2);
    addStep(`Διαίρεση του πίνακα από ${left} έως ${right}, με μέση τιμή ${mid}.`);

    // Διαίρεση των υποπινάκων (αριστερό και δεξί μέρος).
    await mergeSort(arr, left, mid);
    await mergeSort(arr, mid + 1, right);

    // Συγχώνευση των υποπινάκων.
    await merge(arr, left, mid, right);
}

// Συνάρτηση που συγχωνεύει δύο υποπίνακες.
async function merge(arr, left, mid, right) {
    let leftPart = arr.slice(left, mid + 1);
    let rightPart = arr.slice(mid + 1, right + 1);
    let i = 0, j = 0, k = left;

    addStep(`Συγχώνευση από ${left} έως ${right}.`);

    while (i < leftPart.length && j < rightPart.length) {
        if (leftPart[i] <= rightPart[j]) {
            arr[k] = leftPart[i];
            i++;
```

```
        } else {
            arr[k] = rightPart[j];
            j++;
        }
        displayArray(arr, [k]);
        await delay(500);  // Καθυστέρηση για να βλέπουμε τα βήματα.
        k++;
    }

    // Αν υπάρχουν υπόλοιπα στοιχεία στο αριστερό μέρος.
    while (i < leftPart.length) {
        arr[k] = leftPart[i];
        displayArray(arr, [k]);
        await delay(500);  // Καθυστέρηση για να βλέπουμε τα βήματα.
        i++;
        k++;
    }

    // Αν υπάρχουν υπόλοιπα στοιχεία στο δεξί μέρος.
    while (j < rightPart.length) {
        arr[k] = rightPart[j];
        displayArray(arr, [k]);
        await delay(500);  // Καθυστέρηση για να βλέπουμε τα βήματα.
        j++;
        k++;
    }

    // Χρωματίζουμε το συγχωνευμένο τμήμα.
    displayArray(arr);
    addStep(`Ολοκληρώθηκε η συγχώνευση από ${left} έως ${right}.`);
}

// Συνάρτηση για καθυστέρηση (χρησιμοποιείται για να βλέπουμε τα βήματα αργά).
function delay(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}
```

Επεξήγηση του Κώδικα

1. HTML:

Υπάρχει ένα πεδίο εισαγωγής για την εισαγωγή αριθμών, οι οποίοι διαχωρίζονται με κόμμα.

Ένα κουμπί "Εκκίνηση Ταξινόμησης" ξεκινάει την εκτέλεση του αλγορίθμου Merge Sort.

Τα βήματα του αλγορίθμου εμφανίζονται σε μια λίστα στο κάτω μέρος της σελίδας και τα στοιχεία του πίνακα εμφανίζονται δυναμικά σε οριζόντια διάταξη στο div με id arrayDisplay.

2. CSS:

Τα στοιχεία του πίνακα εμφανίζονται ως κουτιά με πλαίσια και στρογγυλεμένες γωνίες.

Τα στοιχεία που βρίσκονται υπό επεξεργασία χρωματίζονται κίτρινα.

Τα συγχωνευμένα τμήματα μπορούν να χρωματίζονται πράσινα, αν το επιθυμούμε.

3. JavaScript (mergesort.js):

Η συνάρτηση startSort():

Παίρνει την είσοδο του χρήστη και την μετατρέπει σε έναν πίνακα αριθμών.

Ξεκινάει τον αλγόριθμο Merge Sort καλώντας τη συνάρτηση mergeSort().

Η συνάρτηση displayArray():

displayArray():

Αυτή η συνάρτηση εμφανίζει τον πίνακα στο div με id arrayDisplay.

Καθαρίζει την προηγούμενη εμφάνιση του πίνακα και δημιουργεί νέα κουτιά για κάθε στοιχείο του πίνακα.

Τα στοιχεία που είναι υπό επεξεργασία (π.χ., αυτά που συγχωνεύονται) χρωματίζονται κίτρινα.

addStep():

Αυτή η συνάρτηση προσθέτει τα περιγραφικά βήματα που εκτελούνται στον αλγόριθμο, ώστε να φαίνεται η διαδικασία στον browser.

Κάθε βήμα εμφανίζεται ως κείμενο μέσα στο div με id steps.

mergeSort():

Αυτή η συνάρτηση υλοποιεί τον αλγόριθμο Merge Sort.

Χωρίζει τον πίνακα σε δύο υποπίνακες, τους ταξινομεί αναδρομικά και στη συνέχεια συγχωνεύει τα ταξινομημένα τμήματα.

Η λειτουργία διαίρεσης του πίνακα παρουσιάζεται στα βήματα του αλγορίθμου, και κάθε αναδρομική κλήση της mergeSort() καλείται με καθυστέρηση, ώστε να μπορούμε να παρακολουθήσουμε τη διαδικασία.

merge():

Αυτή η συνάρτηση συγχωνεύει τους δύο ταξινομημένους υποπίνακες σε έναν ενιαίο πίνακα.

Σταδιακά συγκρίνει τα στοιχεία του αριστερού και δεξιού τμήματος και τα τοποθετεί στην σωστή τους θέση.

Κατά τη διάρκεια της συγχώνευσης, τα στοιχεία που μετακινούνται εμφανίζονται με κίτρινο χρώμα.

Μετά τη συγχώνευση, τα στοιχεία μπορούν να εμφανιστούν χρωματισμένα για να δείξουν ότι ολοκληρώθηκε η διαδικασία.

delay():

Αυτή η συνάρτηση χρησιμοποιείται για την προσθήκη καθυστέρησης μεταξύ των βημάτων της ταξινόμησης, ώστε η διαδικασία να είναι πιο εύκολα ορατή στον χρήστη.

Χρησιμοποιεί Promise για να δημιουργήσει μια καθυστέρηση καθορισμένη σε χιλιοστά του δευτερολέπτου (ms).

Παράδειγμα Λειτουργίας

Ας υποθέσουμε ότι ο χρήστης εισάγει τον ακόλουθο πίνακα αριθμών: 38, 27, 43, 3, 9, 82, 10.

Ο αλγόριθμος Merge Sort θα εκτελέσει τα εξής βήματα:

1. Διαίρεση: Χωρίζει τον πίνακα στο κέντρο.

Χωρίζει τον πίνακα από 38, 27, 43, 3, 9, 82, 10 σε δύο τμήματα: 38, 27, 43, 3 και 9, 82, 10.

2. Περαιτέρω διαίρεση: Συνεχίζει να χωρίζει κάθε τμήμα στα δύο, μέχρι να μείνει με μονομερή στοιχεία:

Το πρώτο τμήμα 38, 27, 43, 3 χωρίζεται σε 38, 27 και 43, 3.

Το δεύτερο τμήμα 9, 82, 10 χωρίζεται σε 9 και 82, 10, και στη συνέχεια το 82, 10 χωρίζεται σε 82 και 10.

3. Συγχώνευση και ταξινόμηση:

Το σύστημα ξεκινά να συγχωνεύει τα μονομερή στοιχεία.

Για παράδειγμα, το 38 και το 27 συγχωνεύονται για να δημιουργήσουν το 27, 38.

Ομοίως, το 43 και το 3 συγχωνεύονται για να δημιουργήσουν το 3, 43.

4. Ολοκλήρωση: Συνεχίζει με τη συγχώνευση των μεγαλύτερων υποπινάκων, μέχρι να ολοκληρωθεί η ταξινόμηση:

Το 27, 38 και το 3, 43 συγχωνεύονται για να δημιουργήσουν το 3, 27, 38, 43.

Το 9 συγχωνεύεται με το 10, 82 για να δημιουργήσει το 9, 10, 82.

Τέλος, τα δύο μεγάλα τμήματα 3, 27, 38, 43 και 9, 10, 82 συγχωνεύονται για να δώσουν τον πλήρως ταξινομημένο πίνακα 3, 9, 10, 27, 38, 43, 82.

Συμπέρασμα

Αυτό το πρόγραμμα υλοποιεί τον αλγόριθμο Merge Sort σε JavaScript με HTML και CSS για την οπτικοποίηση της διαδικασίας στον browser. Οι χρήστες μπορούν να εισάγουν μια λίστα αριθμών και να παρακολουθήσουν σε πραγματικό χρόνο πώς ο αλγόριθμος χωρίζει τον πίνακα, τον ταξινομεί και τον συγχωνεύει βήμα προς βήμα. Η χρήση της καθυστέρησης και των χρωματισμένων στοιχείων καθιστά τη διαδικασία εύκολα παρακολουθήσιμη, προσφέροντας μια καλή ευκαιρία για να μάθουν οι χρήστες τον τρόπο λειτουργίας του Merge Sort.

Bubble Sort σε JavaScript

Το παρακάτω πρόγραμμα υλοποιεί τον αλγόριθμο ταξινόμησης Bubble Sort σε JavaScript, με ένα περιβάλλον χρήστη (interface) σε HTML, ώστε να επιτρέπεται η εισαγωγή αριθμών από τον χρήστη και η ταξινόμησή τους. Το πρόγραμμα θα δείχνει τα βήματα της ταξινόμησης και θα επιτρέπει στον χρήστη να εισάγει μια λίστα αριθμών, οι οποίοι θα ταξινομούνται όταν πατήσει ένα κουμπί.

Ολοκληρωμένο Πρόγραμμα

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bubble Sort in JavaScript</title>
    <style>
        body {
            font-family: Arial, sans-serif;
        }
        .container {
            width: 300px;
            margin: 0 auto;
            padding: 20px;
            text-align: center;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        input, button {
            padding: 10px;
            margin: 5px;
            width: 80%;
        }
        .result {
            margin-top: 20px;
            padding: 10px;
            border: 1px solid #000;
```

```
            min-height: 50px;
        }
    </style>
</head>
<body>

<div class="container">
    <h2>Bubble Sort</h2>
    <!-- Input για την εισαγωγή αριθμών -->
    <input type="text" id="number-input" placeholder="Enter comma separated numbers" />
    <button onclick="sortNumbers()">Sort Numbers</button>

    <!-- Περιοχή για εμφάνιση του αποτελέσματος -->
    <h3>Sorted Numbers:</h3>
    <div id="sorted-display" class="result">No numbers sorted yet</div>
</div>

<script src="bubblesort.js"></script>
</body>
</html>
```

JavaScript (bubblesort.js)

```
// Συνάρτηση Bubble Sort που ταξινομεί έναν πίνακα από αριθμούς.
function bubbleSort(arr) {
    let n = arr.length;
    let swapped;
    // Εξωτερικός βρόχος που επαναλαμβάνεται μέχρι να μην χρειάζονται άλλες ανταλλαγές.
    for (let i = 0; i < n - 1; i++) {
        swapped = false;
        // Εσωτερικός βρόχος για την αντιπαράθεση γειτονικών στοιχείων.
        for (let j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Αν το στοιχείο στα αριστερά είναι μεγαλύτερο, τα ανταλλάσσουμε.
                let temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true; // Σημειώνουμε ότι έγινε ανταλλαγή.
            }
        }
        // Αν δεν έγινε καμία ανταλλαγή, η λίστα είναι ήδη ταξινομημένη.
        if (!swapped) {
            break;
```

```
        }
    }
    return arr;
}

// Συνάρτηση που καλείται όταν ο χρήστης πατά το κουμπί για να ταξινομήσει τους αριθμούς.
function sortNumbers() {
    // Λαμβάνουμε την είσοδο του χρήστη από το πεδίο εισόδου.
    let input = document.getElementById("number-input").value;

    // Διαχωρίζουμε την είσοδο σε έναν πίνακα αριθμών, αφαιρώντας τα κενά και μετατρέποντας σε ακέραιους.
    let numberArray = input.split(',').map(Number);

    // Ελέγχουμε αν η είσοδος είναι έγκυρη.
    if (numberArray.some(isNaN)) {
        alert("Please enter a valid list of numbers separated by commas.");
        return;
    }

    // Εκτελούμε τον αλγόριθμο ταξινόμησης Bubble Sort.
    let sortedArray = bubbleSort(numberArray);

    // Εμφανίζουμε το ταξινομημένο αποτέλεσμα.
    document.getElementById("sorted-display").textContent = sortedArray.join(', ');
}
```

Περιγραφή της Υλοποίησης

1. HTML:

Δημιουργεί ένα απλό περιβάλλον χρήστη με ένα πεδίο εισόδου όπου ο χρήστης μπορεί να εισάγει αριθμούς, διαχωρισμένους με κόμμα (,).

Ένα κουμπί επιτρέπει στον χρήστη να ξεκινήσει την ταξινόμηση όταν πατηθεί.

Ένα div εμφανίζει το αποτέλεσμα μετά την ταξινόμηση.

2. JavaScript:

Η συνάρτηση bubbleSort(arr) υλοποιεί τον αλγόριθμο ταξινόμησης Bubble Sort:

Ελέγχει κάθε ζεύγος γειτονικών στοιχείων και τα ανταλλάσσει αν δεν είναι στη σωστή σειρά.

Επαναλαμβάνει τη διαδικασία μέχρι να μην υπάρχουν άλλες ανταλλαγές.

Η συνάρτηση sortNumbers() λαμβάνει την είσοδο του χρήστη, την μετατρέπει σε πίνακα αριθμών, και καλεί τη bubbleSort για να ταξινομήσει τους αριθμούς.

Το αποτέλεσμα εμφανίζεται στο div sorted-display.

3. Σχόλια στον Κώδικα:

Ο κώδικας έχει αναλυτικά σχόλια που εξηγούν τη λειτουργία του κάθε κομματιού του προγράμματος, από την εισαγωγή του χρήστη μέχρι την εμφάνιση του αποτελέσματος.

Παράδειγμα Χρήσης

1. Ο χρήστης εισάγει μια λίστα αριθμών, για παράδειγμα: 34, 12, 24, 9, 5.

2. Πατώντας το κουμπί "Sort Numbers", οι αριθμοί ταξινομούνται και εμφανίζονται στο κάτω μέρος, π.χ.: 5, 9, 12, 24, 34.

Συμπέρασμα

Αυτό το πρόγραμμα δείχνει πώς μπορούμε να υλοποιήσουμε τον αλγόριθμο ταξινόμησης Bubble Sort με JavaScript, σε μια HTML σελίδα, με απλό και κατανοητό τρόπο. Η διαδραστικότητα με τον χρήστη επιτρέπει την εισαγωγή μιας λίστας αριθμών, και η ταξινόμηση πραγματοποιείται με τη χρήση του κλασικού αλγορίθμου Bubble Sort, με τα αποτελέσματα να εμφανίζονται άμεσα στον browser.