LICEUL TEORETIC O.C. TĂSLĂUANU

Vizualizator al algoritmilor de sortare

|  |  |
| --- | --- |
| Instructor | Elev |
| Butyka Mihai | Marc Rareș-Cristian |

Cuprins

Introducere 3

Alegerea temei 3

Conținutul lucrării 3

Tema lucrării 4

Algoritmi de sortare 4

Complexitatea algoritmilor de sortare 4

Complexitatea timpului 4

Complexitatea spațiului 4

Tehnologiile folosite pentru crearea aplicației 5

HyperText Markup Language (HTML) 5

JavaScript 5

Cascading Style Sheets (CSS) 5

Implementarea aplicației 6

page.html 6

sorting\_algs.js 7

Bubble sort 7

Selection sort 8

Insertion sort 8

Quick sort 9

Random sort 9

Merge sort 10

main.js……………………………....………………………………....………………………..10  
 Bibliografie ……………………………………………………………………………………..12

**Introducere**

Lucrarea mea de atestat constă în vizualizarea mai multor algoritmi de sortare cu scopul de a înțelege modul în care funcționează.

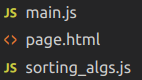
Alegerea temei

Am ales această temă deoarece algoritmii de sortare sunt utilizați foarte des în programare și consider că înțelegerea lor este necesară oricărui programator. Deoarece vreau să lucrez ca programator am decis că trebuie să mă familiarizez mai mult cu ei, așadar proiectul a fost creat cu scopuri educaționale personale.

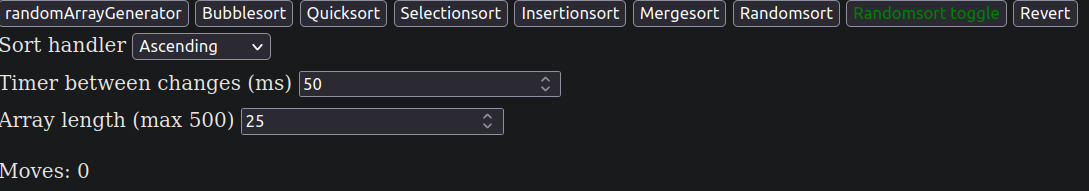
Conținutul lucrării

Proiectul a fost realizat folosind HTML (Hypertext markup-language), CSS (Cascade style-sheets) și limbajul de programare JavaScript.

Fișierele necesare funcționării aplicației:

  
  
  
Pentru a accesa aplicația utilizatorul trebuie să acceseze o pagină web în care vor apărea butoanele care cheamă funcțiile, câmpurile prin care se transmite input-ul și șirul generat după completarea diferitelor câmpuri.

Exemplu de cum arată interfața paginii principale:

După completarea câmpurilor utilizatorul poate apăsa pe unul dintre butoanele care cheamă funcțiile de sortare pentru a vizualiza sortarea șirului.

**Tema lucrării**

Tema lucrării este vizualizarea algoritmilor de sortare.

Algoritmi de sortare

* Un algoritm de sortare este folosit pentru a rearanja un șir sau o listă de elemente în funcție de un operator de comparare.
  + Operatorul de comparare este folosit pentru a decide noua ordine a elementelor în structura de date respectivă.

Exemplu:

|  |  |  |
| --- | --- | --- |
| Mod de sortare | Șir inițial | Șir final |
| Crescător | 0, 7, 5, 6, 3 | 0, 3, 5, 6, 7 |
| Descrescător | 0, 7, 5, 6, 3 | 7, 6, 5, 3, 0 |

Eficiența unui algoritm de sortare depinde de doi factori:

1. Complexitatea timpului
2. Complexitatea spațiului

Complexitatea algoritmilor de sortare

Complexitatea timpului

Complexitatea timpului este definită ca numarul de ori de care un set particular de instrucțiuni este executat, în loc de timpul total al execuției, deoarece timpul total al execuției depinde de factori externi ca viteza procesorului, compilatorul folosit, etc.

Complexitatea spațiului

Complexitatea spațiului este spațiul total de memorie necesar programului pentru execuția sa.

Un lucru important aici este că în ciuda acestor factori eficiența unui algoritm depinde și de natura și mărimea datelor de intrare.

Tabelul complexității timpului al algoritmilor de sortare standard

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritm | Complexitatea timpului | | |
|  | Best | Average | Worst |
| Selection sort | Ω(n2) | ~~0~~(n2) | 0(n2) |
| Bubble sort | Ω(n) | ~~0~~(n2) | 0(n2) |
| Insertion sort | Ω(n) | ~~0~~(n2) | 0(n2) |
| Heap sort | Ω(n \* log(n)) | ~~0~~(n \* log(n)) | 0(n \* log(n)) |
| Quick sort | Ω(n \* log(n)) | ~~0~~(n \* log(n)) | 0(n2) |
| Merge sort | Ω(n \* log(n)) | ~~0~~(n \* log(n)) | 0(n \* log(n)) |

**Tehnologiile folosite pentru crearea aplicației**

Pentru dezvoltarea aplicației am folosit HTML, CSS și JavaScript.

HyperText Markup Language (HTML)

HyperText Markup Language este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser. În cadrul dezvoltării web de tip front-end, HTML este utilizat împreună cu CSS și JavaScript.

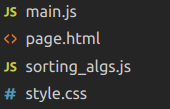
JavaScript

JavaScript este un limbaj de programare orientat pe programarea pe obiecte bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este bine cunoscut pentru folosirea sa în construirea site-urilor web.

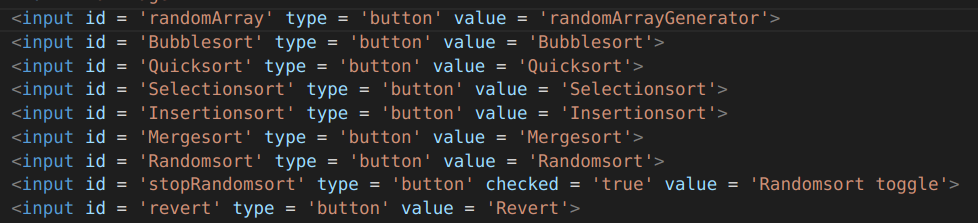
Cascading Style Sheets (CSS)

CSS sau Cascading Style Sheets este un standard pentru formatarea elementelor unui document HTML. Stilurile se pot atașa elementelor HTML prin intermediul unor fișiere externe sau în cadrul documentului, prin elementul <style> și/sau atributul style. CSS este una dintre tehnologiile de bază utilizate in procesul de dezvoltare web, împreuna cu HTML și JavaScript.

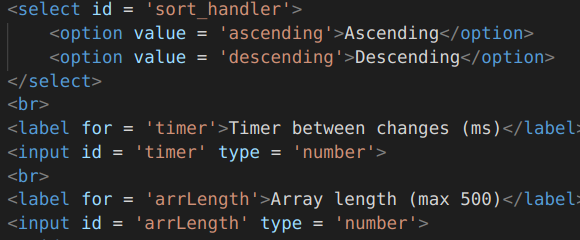
**Implementarea aplicației**

Aplicația este formată din patru fișiere. Acestea sunt main.js, page.html, style.css sorting\_algs.js.

page.html

Acest fișier conține elementele HTML care apar pe pagina web cum ar fi butoanele care cheamă funcțiile de sortare, butonul de oprire al algoritmului de sortare random sort, butonul de revenire la șirul inițial, butonul de generare a unui șir aleatoriu

și elementele de tip input prin care se transmit datele de intrare introduse de utilizator împreună cu elementul de tip select prin care se alege modul de sortare al șirului.



Pentru afișarea șirului de elemente pe pagina web am folosit elementul <canvas> care creează o pânză pe care se poate desena folosind diferite funcții predefinite în JavaScript. Atributele width și height ale elementului definesc lățimea și înălțimea pânzei.

De asemenea aici se declară și relația dintre acest document și fișierul style.css



sorting\_algs.js

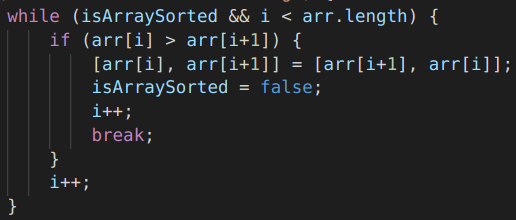
Acest fișier are rolul de modul, așadar aici sunt definite toate functiile de sortare care sunt exportate către fișierul principal.

Exemplu: Exportarea funcției bubblesort  
Lista de parametri a funcțiilor de sortare diferă în funcție de algoritm, dar câțiva parametri sunt folosiți peste tot, aceștia fiind arr, moves și show.

* arr – ia referința către șirul care urmează să fie sortat
* moves – primește 0 și are rolul de a reține numărul de mișcări făcute pentru sortarea șirului
* show – primește referința către funcția folosită pentru a afișa schimbările făcute pe parcursul sortării

Bubble sort

Bubble sort-ul este cel mai simplu algoritm de sortare care funcționează prin schimbarea repetată a elementelor adiacente dacă sunt în ordinea greșită.

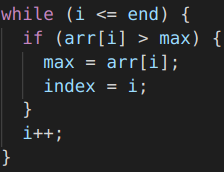
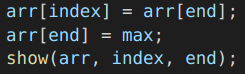


În această secvență de cod se iterează prin șir până când are nevoie de o schimbare, se face interschimbarea elementelor, după care se afișează schimbarea.

  
Funcția este chemată recursiv până când șirul nu mai are nevoie de nici o schimbare, ceea ce înseamnă că este sortat.

  
  
Selection sort

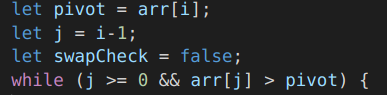
Algoritmul de sortare selection sort sortează un șir prin găsirea repetată a elementului minim (pentru ordinea crescătoare) sau maxim (pentru ordinea descrescătoare) din partea nesortată a șirului și punerea lui la început (pentru ordinea crescătoare) sau la sfârșit (pentru ordinea descrescătoare).

În secvențele următoare de cod se găsesc maximul șirului și poziția sa care se rețin în variabilele max și index, urmând punerea maximului la sfârșitul subșirului nesortat și afișarea schimbării.

Insertion sort

Insertion sort-ul este un algoritm de sortare simplu similar cu modul în care sortezi cărțile de joc în mână. Șirul este împărțit într-o parte sortată și una nesortată. Valorile din partea nesortată sunt alese și plasate la poziția corectă în partea sortată.

Algoritmul pentru a sorta un șir de mărime n în ordine crescătoare

1. Iterează de la arr[1] la arr[n] prin șir
2. Compară elementul curent (cheie) cu predecesorul său
3. Dacă elementul curent (cheie) este mai mic decât predecesorul său, c ompară-l cu elementele dinaintea sa. Mută elementele mai mari cu o poziție la dreapta pentru a face spațiu pentru elementul schimbat.

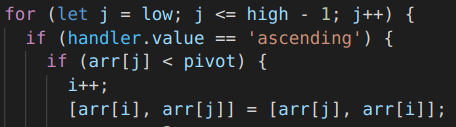
Quick sort

Quick sort-ul alege un element ca pivot și partitionează șirul în jurul pivotului ales. Există multe versiuni ale acestui algoritm care aleg pivotul în mod diferit.

1. Întotdeauna alege primul element ca pivot
2. Întotdeauna alege ultimul element ca pivot
3. Alege un element aleatoriu ca pivot
4. Alege elementul din mijloc ca pivot

Procesul cheie în quick sort este partitionarea. Scopul partiționării este cel de a pune elementul ales ca pivot pe poziția corectă în șirul sortat, cu toate elementele mai mici decât pivotul ales în stânga sa, și cu toate elementele mai mari decât el în dreapta sa.

Partiționarea șirului:

  
  
  
  
Apelurile recursive:

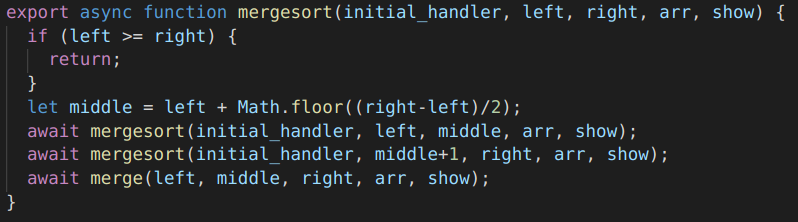
  
Random sort  
Random sort-ul nu este de fapt o metodă de sortare folosită în practică și nu garantează sortarea șirului dat, însă a fost adăugată pentru amuzament și diferite teste (de exemplu, dacă algoritmul poate sorta un șir de 10 elemente și în câte mișcări o face).

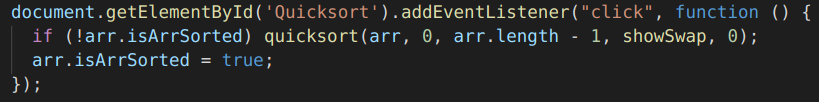
Algoritmul constă în alegerea a două elemente aleatorii din șir (pentru a realiza asta am folosit metoda random) care vor fi interschimbate, lucru care se repetă până la sortarea șirului.

Alegerea a două elemente aleatorii din șir:  
  
  
Interschimbarea elementelor alese:

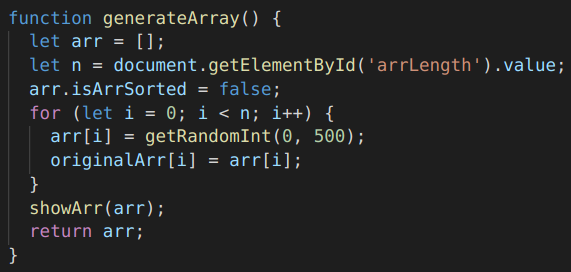
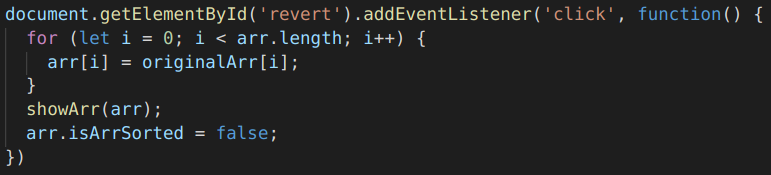
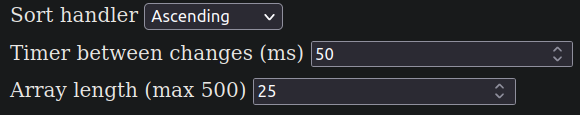
Merge sort

Merge sort-ul împarte șirul în două jumătăți, se apelează recursiv pentru cele două jumătăți și apoi unește subșirurile sortate. Funcția merge este folosită pentru a uni cele două subșiruri.

main.js

Fișierul main.js are rolul de a îmbina partea de HTML cu funcțiile definite în sorting\_algs.js pentru a crea interfața paginii web împreună cu funcționalitatea ei. Funcțiile definite în sorting\_algs.js pot fi folosite în acest fișier datorită faptului ca au fost exportate și importate.  
Fiecare buton din HTML are o funcție corespondentă care se execută atunci când o condiție este îndeplinită (apăsarea butonului respectiv în cele mai multe cazuri).  
După chemarea funcției șirul se marchează ca sortat pentru a nu putea chema alte metode de sortare în timp ce șirul este sortat.

Pe lângă funcțiile de sortare am implementat și:

1. O funcție care să genereze un șir nou cu elemente aleatorii:
2. O funcție care să revină la starea inițială a ultimului șir generat
3.  O funcție care să oprească random sort-ul
4. Trei câmpuri în care utilizatorul poate introduce datele de intrare pentru modul de sortare (crescător/descrescător), timpul dintre afișările schimbărilor din șir și lungimea șirului.
5.  O variabilă care reține numărul de mișcări făcut de algoritmul de sortare și-l afișează pe pagină

Bibliografie

https://www.geeksforgeeks.org/fundamentals-of-algorithms/?ref=shm

https://stackoverflow.com/questions/5185864/javascript-quicksort

https://javascript.info/

https://developer.mozilla.org/en-US/docs/Web/JavaScript

https://developer.mozilla.org/en-US/docs/Learn/HTML

https://developer.mozilla.org/en-US/docs/Learn/CSS