Cuprins

1. Introducere

1.1. Motivație

Odată cu intrarea în lumea informaticii, o persoana are de învățat anumiți algoritmi sau structuri de date care îi vor fixa o bază și îl vor ajuta mai departe pentru rezolvarea problemelor care le întâmpină. De obicei, premisa care stă la baza învațării acestora este aceea că trebuie înțeles mai întâi modul de rulare a programului (funcții, bucle, recursivitate) după care trebuie abstractizate pentru o întelegere mai ușoară a unui program, prin faptul că nu mai suntem restrictionați de implementare ci mai mult de ideile principale. De aceea, în primă fază, se folosește un pseudocod pentru a explica un algoritm sau o structură de date.

Abstractizarea ne îndepărtează de lucrurile cu care suntem obișnuiți (ca exemplu, lucrul cu numere naturale versus lucrul cu grupuri) și ne pune câteodată în situații inconfortabile, în care ne este greu să și vizualizăm / imaginăm soluția unei probleme. Astfel, e mai ușor sa facem gradual această tranziție, prin exemple sau relatarea la unor cazuri mai simple.

1.2. Obiectiv

Această aplicație are scopul de a ușura și minimiza timpul de învățare a unui algoritm sau structuri de date descrise în următoarele pagini. Există situații când, chiar dacă codul scris pare corect, e mai ușor de vizualizat prin desene și rulare pas cu pas a programului. Abstractizarea dintre cum sunt ținute datele în calculator și ce semnificație le dăm sau cum le vizualizăm pentru a ușura logica, aceasta este una din principalele ținte cu care a fost scrisă aplicația.

Totodată, se pune la dispoziția utilizatorului un mediu de lucru flexibil, controale de flux ale programului (butoane de începere / pauză, un timer pentru fiecare pas al programului), o componentă de vizualizare a acelui algoritm sau structură de date, puncte de merit pentru răspunsuri corecte, pentru a face totul o experiență placută și provocatoare.

Obiectivul final este ca un utilizator să poată învăța în modul ales de el, mai încet sau mai rapid, și energia depusă să fie cât mai mare pe acel item și nu pe confuzia utilizării acestei aplicații.

1.3. Soluții existente

Există mai multe soluții de acest gen, gratis sau nu, iar din acestea am ales să discut și compar trei dintre ele. Când vorbim de o soluție pentru problema de învățare a unui algoritm sau structură de date, dorim sa ne uităm la:

- **flexibilitatea aplicației** (dacă utilizatorul poate schimba intrările, codul, pune breakpoints sau comentarii)
- **tip de vizualizare** (dacă se poate da zoom, are culori, este statică sau interactivă)
- **controlul de flux** (dacă putem opri programul la un anumit pas, inversa pasul curent, seta un interval de parcurgere a liniilor de cod)

Toate aceste caracteristici contează pentru un utilizator și modul de învățare care îl aplică.

[1] PathFinding.js

Aceasta este o aplicație care simulează diferiți algoritmi de găsire a drumurilor minime pe o matrice. Vizualizarea lor este pe o matrice, dar unii din ei pot fi aplicați și pe grafuri. Câțiva din acești algoritmi care pot fi selectați de utilizator sunt:

- A*
- Dijkstra
- Breadth-First-Search
- Jump Point Search

Utilizatorul poate alege pe oricare dintre aceștia și tipul de distanță care se folosește la calcularea costului unui drum (euclidiană, Manhattan). Pe lângă acestea, utilizatorul dispune de o

vizualizare interactiva care acceptă comenzi de mouse-click sau drag-and-drop. Există o matrice de dimensiuni fixe, cu un punct de start reprezentat prin verde si punct de oprire reprezentat prin roșu.

Mai mult, comanda de click pe o celulă comută o celulă liberă într-una învalidă, și invers.

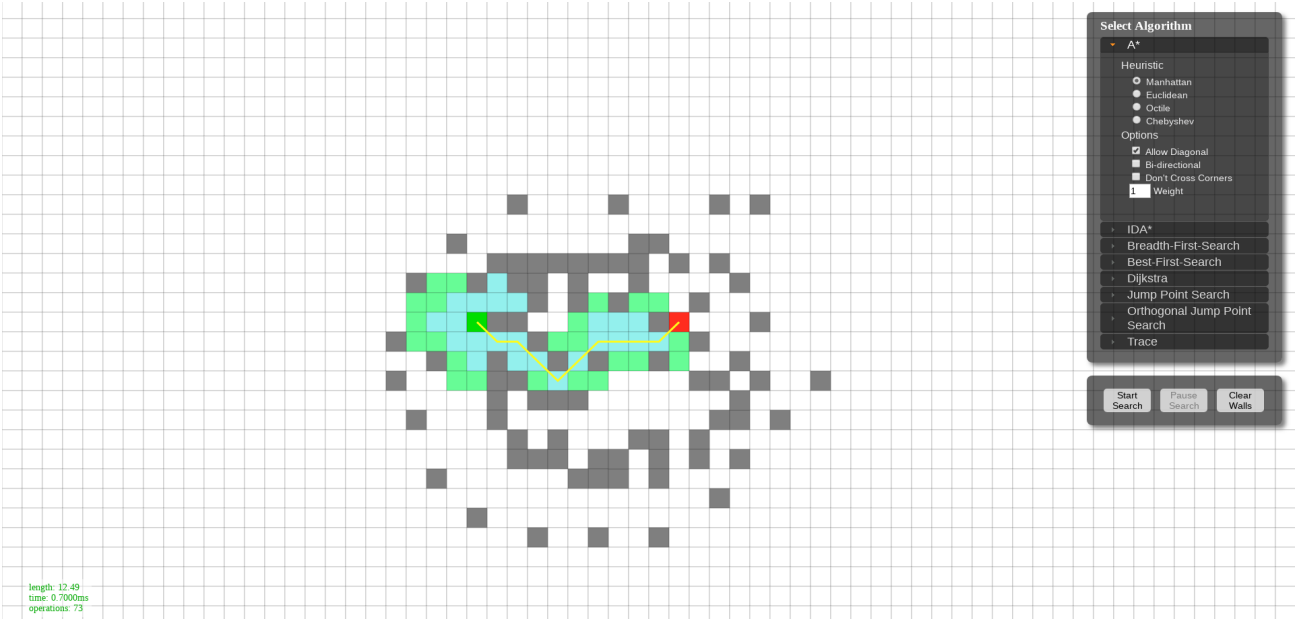

Figura 1-1 PathFinding.js

Aplicația are butoane de începere și oprire a algoritmului, dar ruleaza foarte rapid pentru o matrice de dimensiunile ilustrate, astfel că utilizatorul nici nu apucă să apese butonul de oprire. Totuși, oferă o funcționalitate în plus și folositoare.

Astfel, aplicația este flexibilă, putând schimba aproape tot înafară de dimensiunile matricei. oferă o vizualizare care diferențiază diferitele tipuri de celule și care este interactivă (acțiunile facându-se din mouse), totuși la partea de control al fluxul are un set incomplet, lipsând controlul de timp și altele.

[2] USF Data Structure Vizualizations

Universitatea din San Francisco, departamentul de Computer Science, a construit o listă de vizualizări pentru diverși algoritmi și structuri de date:

- Sortări (radix sort, bucket sort, heap sort)
- Structuri de Indexare (tabele hash, arbori indexați binar, arbori AVL)

- Algoritmi de Grafuri (Depth-First-Search, Breadth-First-Search, componente conexe, arbore parţial de cost minim)
- Programare Dinamică (numere Fibonacci, problema celui mai lung subşir comun)

Fiind o listă largă de analizat, ne vom lega doar de principalele componente care se găsesc în vizualizări şi ce avantaje / dezavantaje are această aplicaţie.

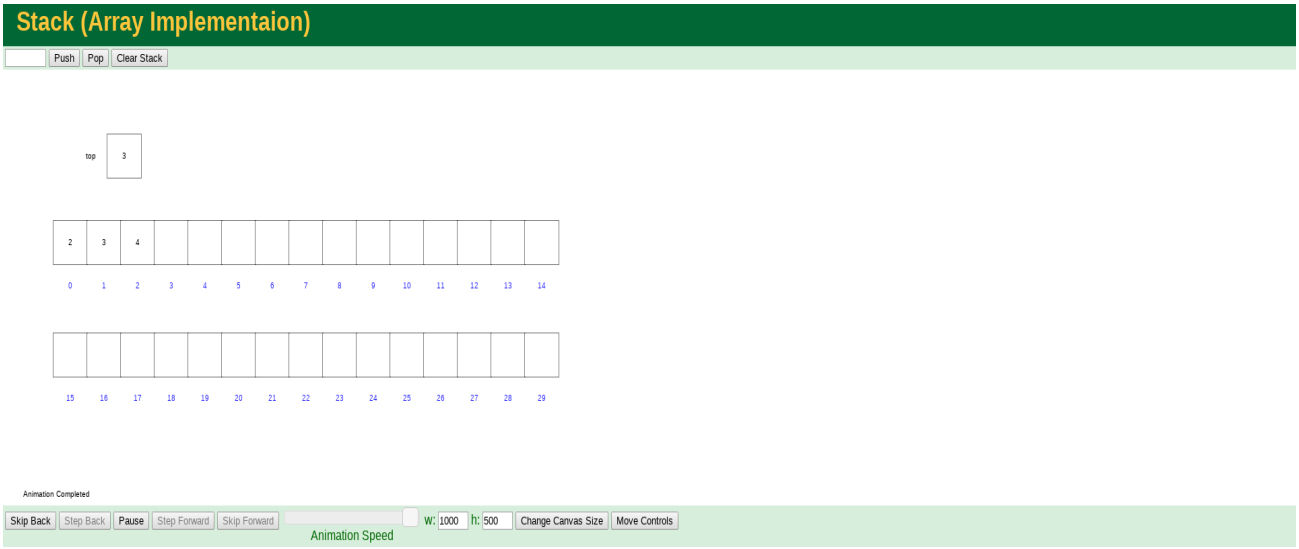

Figura 1-2 USF - Stack implementation

Ca şi controale de flux, are un limitator de timp pentru viteza cu care se întâmplă animaţia unei operaţii din cadrul exerciţiului, butoane de începere şi pauză, butoane care îţi oferă posibilitatea să mergi doar câte un pas în algoritm, înainte sau înapoi, şi nişte butoane speciale care depind de exerciţiu şi definesc operaţiile care se pot face.

Majoritatea vizualizărilor nu sunt interactive, ci doar operaţiile specifice acelui exerciţiu schimbă vizualizarea în vreun mod. Totuşi are animaţii destul de interesante şi diverse, plăcute utilizatorului. Vizualizarea este conţinută într-un bloc canvas care se poate redimensiona după plăcerea utilizatorului şi restricţiilor browser-ului.

Aplicaţia este relativ flexibilă, putând lua ca date de intrare şi ieşire orice în contextul exerciţiului. Un dezavantaj este faptul că nu se pot da mai multe operaţii deodată, deci dacă s-a pierdut sesiunea, trebuie refăcute operaţiile, de mână, din nou. Deobicei, există mai multe operaţii care se fac asupra unei structuri de date sau algoritm, şi ar fi natural să se poată spune numărul de

operații pe prima linie din datele de intrare, iar după aceea să se descrie fiecare operație în parte pe următoarele linii.

În concluzie, aplicația dezvoltată de Universitatea din San Francisco oferă o gamă largă de algoritmi și structuri de date și o pagină de simulare destul de completă. Cu toate acestea, există câteva lucruri dezavantajoase, cum ar fi vizualizări care trebuie redimensionate de mână în caz că există prea multe date de afișat, sau operațiile care trebuie procesate una câte una.

[3] CsAcademy Applications

Acest site este folosit în principal pentru concursuri de algoritmică, dar are niste sub-aplicații pe care orice utilizator le poate folosi caz că dorește să vizualizeze o structură de date.

Cele mai importante aplicații sunt "Graph Editor" și "Geometry Widget". Aceste aplicații construiesc o vizualizare bazată pe date de intrare scrise de utilizator.

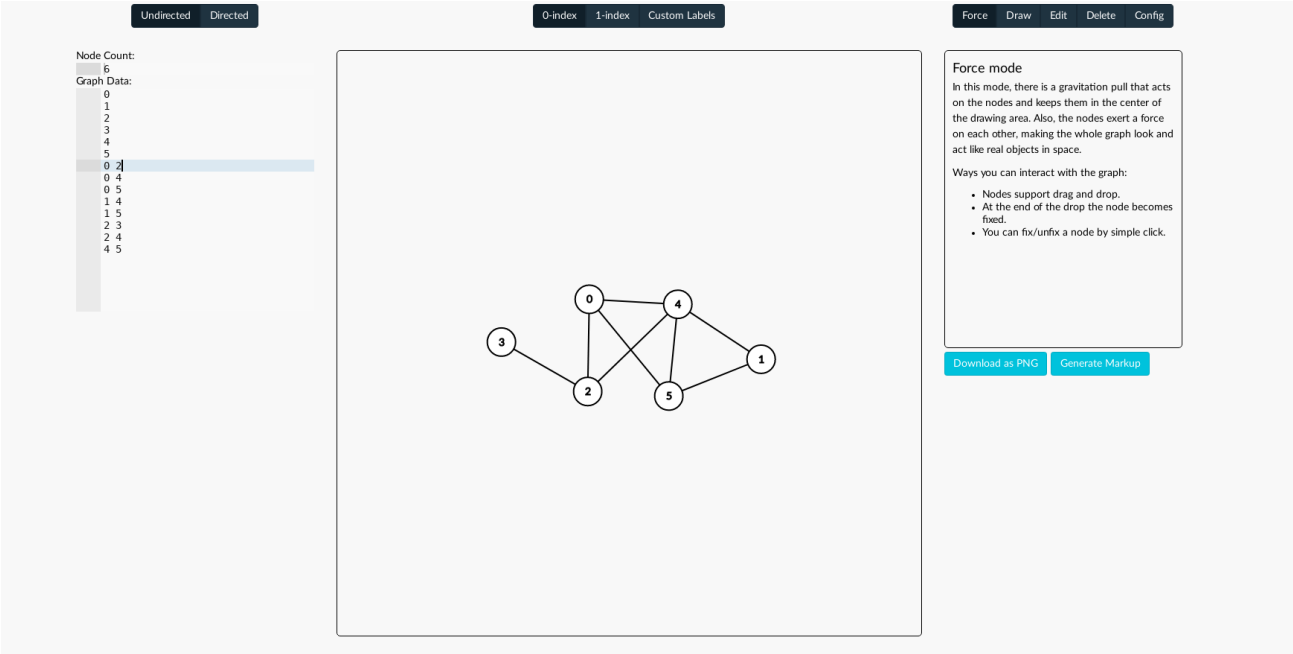

Figura 1-3 CsAcademy - Graph Editor

Astfel, la partea de control al fluxului aplicațiile nu au nimic utlizabil. În schimb, vizualizările sunt interactive, pot fi redimensionate prin utilizarea mouse-ului (scroll, drag-and-drop) și au suport pentru dispozitive mobile.

Mai mult, datele de intrare pot fi scrise în totalitate de utilizator, doar că trebuie scrise într-un format predefinit pentru o bună funcționare a aplicației. Există și mai multe moduri pentru editarea vizualizării fară ajutorului datelor de intrare.

În concluzie, chiar dacă aplicațiile nu sunt făcute pentru a învăța user-ul un anumit algoritm sau o anumită structură de date, ele pot fi folosite pentru a face vizualizari cu diferite scopuri (prezentări, lecții la școală etc.).

3. Tehnologii utilizate

3.1 Javascript

Fiind o aplicație web, limbajul principal folosit, pe lângă HTML și CSS, este Javascript.

Javascript este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Totodată, este un limbaj dinamic, unde orice proprietate a unui obiect se poate schimba la *run-time*.

Astfel, este un limbaj flexibil și este ideal pentru o aplicație unde nu se cunosc datele cu care se va lucra (fiind o aplicație de tip client-server).

Este asemănător limbajelor ca C++ sau Java, prin faptul că are funcții, obiecte, tipuri de date, dar diferă prin alte module cum ar fi obiectul *document* care este folosit pentru a comunica cu elementele de pe pagina ce formează DOM (*Document Object Model*).

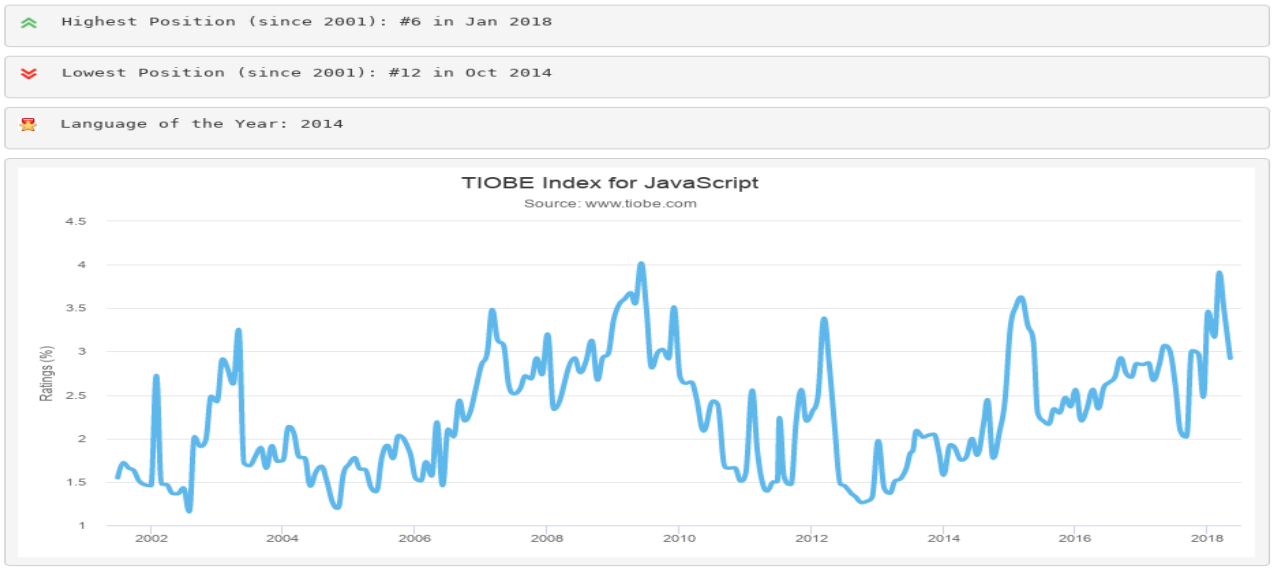

Figura 1-4 Javascript TIOBE index

Javascript în combinație cu HTML / CSS și mediul oferit de un browser (cookies, browser storage) formeaza un grup matur și puternic de tehnologii, comunitatea este mare și foarte ajutătoare.

Se observă din graficul obținut de cei de la TIOBE că Javascript, chiar și dupa 20 de ani de folosire în lumea dezvoltării aplicațiilor web, rămâne în top 10 cele mai folosite limbaje de programare de-a lungul globului.

3.2 Angular

Angular [4] este succesorul popularului framework **AngularJs.**

AngularJs este un framework bazat pe Javascript, menținut în principal de Google, și permite dezvoltarea de *single-page applications.* Principalele obiective de design sunt:

- să decupleze manipularea DOM-ului (*Document Object Model*) față de logica aplicației.
- să decupleze frontend-ul de backend. (are rolul de a împărți sarcinile separat, acceptând ideea lucrului în paralel)
- să furnizeze o structură coerentă pe masură ce se vor dezvolta diferitele module ale aplicației (UI, logica de business, testarea)

Angular se bazează pe caracteristicile și flexibilitatea oferita de AngularJs, oferind un pachet complet (routing, componente, module, dependency injection) și posibilitatea de a crea aplicații pentru platforme mobile dar scriind Javascript.

Angular este tânăr în comparație cu AngularJs (2016 versus 2010) dar s-a înălțat rapid în rândul proiectelor de pe Github [5], crescând în popularitate an după an.

Majoritatea avantajelor față de Angularjs sunt:

- **Typescript**. este un limbaj statically-typed și este superset a lui Javascript. Angular te obligă să folosesti Typescript pentru a crea o aplicație web. Unul din marele avantaje ale lui Typescript este faptul ca, fiind tradus în Javascript, se poate configura să fie compatibil cu browsere vechi (cum ar fi Internet Explorer 8)

- **Angular-CLI:** este un utilitar pentru linia de comandă care te ajută la dezvoltarea de aplicații în Angular. Are comenzi pentru generare de componente sau directive, testare și are inclus un utilitar numit *Webpack* pentru împachetarea diferitelor biblioteci folosite în aplicație.
- **Modularitate:** Angular se bazează mult pe modularitate deoarece ofera o logică mai bună și te scapă de aplicații cu dimensiuni imense. Totodată este mentenabil și oferă extensibilitate.

3.3 NodeJs

NodeJs [6] este un mediu de dezvoltare pentru Javascript, construit cu ajutorul motorului V8 al browser-ului Chrome.

Bibliografie

[1] "PathFinding.js". Available: http://qiao.github.io/PathFinding.js/visual/

[2] "Data Structure Visualizations", University of San Francisco.

[3] "CsAcademy Applications". Avalaible: https://csacademy.com/

[4] "Angular", Google. Avalaible: https://angular.io/

[5] "Github", Github, Inc. Avalaible: https://github.com/

[6] "NodeJs", Joyent. Avalaible: https://nodejs.org/en/

Avalaible: https://www.cs.usfca.edu/~galles/visualization/Algorithms.html