Cuprins

1. Introducere

1.1. Motivaţie

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

Abstractizarea ne îndepărtează de lucrurile cu care suntem obişnuiţi (ca exemplu, lucrul cu numere naturale versus lucrul cu grupuri) şi ne pune câteodată în situaţii inconfortabile, în care ne este greu să şi vizualizăm / imaginăm soluţia unei probleme. Astfel, e mai uşor sa facem gradual această tranziţie, prin exemple sau relatarea la unor cazuri mai simple.

1.2. Obiectiv

Această aplicaţie are scopul de a uşura şi minimiza timpul de învăţare a unui algoritm sau structuri de date descrise în următoarele pagini. Există situaţii când, chiar dacă codul scris pare corect, e mai uşor de vizualizat prin desene şi rulare pas cu pas a programului. Abstractizarea dintre cum sunt ţinute datele în calculator şi ce semnificaţie le dăm sau cum le vizualizăm pentru a uşura logica, aceasta este una din principalele ţinte cu care a fost scrisă aplicaţia.

Totodată, se pune la dispoziţia 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 experienţă placută şi provocatoare.

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

1.3. Soluţii existente

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

- **flexibilitatea aplicaţiei** (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 şi modul de învăţare care îl aplică.

[1] PathFinding.js

Aceasta este o aplicaţie care simulează diferiţi algoritmi de găsire a drumurilor minime pe o matrice. Vizualizarea lor este pe o matrice, dar unii din ei pot fi aplicaţi şi pe grafuri. Câţiva din aceşti algoritmi care pot fi selectaţi de utilizator sunt:

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

Utilizatorul poate alege pe oricare dintre aceştia şi tipul de distanţă care se foloseşte 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 roşu.

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

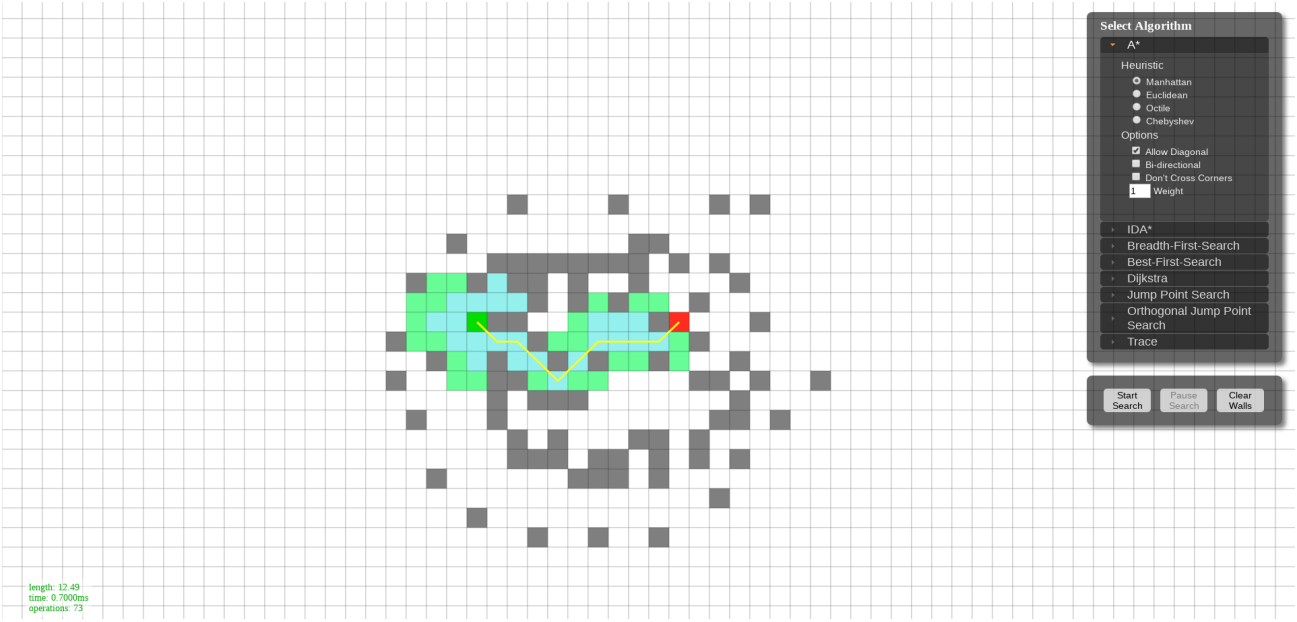

Figura 1-1 PathFinding.js

Aplicația are 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 parțial de cost minim)
- Programare Dinamică (numere Fibonacci, problema celui mai lung subșir comun)

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

Bibliografie

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

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

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

Listă de figuri