Cuprins

[1. Introducere 2](#__RefHeading___Toc875_274622961)

[1.1. Motivație 2](#__RefHeading___Toc877_274622961)

[1.2. Obiectiv 2](#__RefHeading___Toc985_274622961)

[1.3. Soluții existente 3](#__RefHeading___Toc983_274622961)

[Bibliografie 5](#__RefHeading___Toc989_274622961)

# 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 cănd, 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 rularea unei linii de cod), 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.

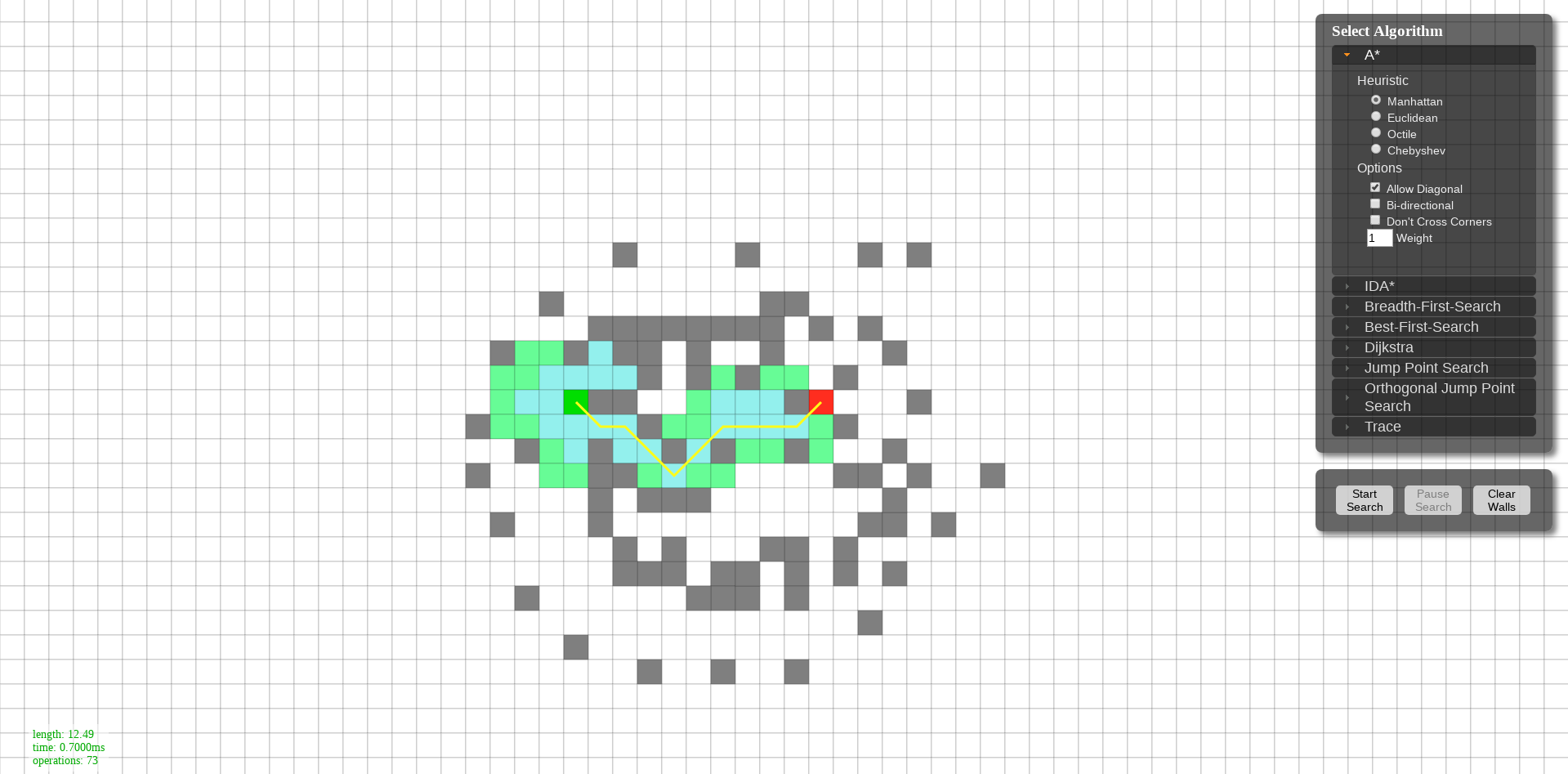
**[1] PathFinding.js**

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

Astfel, aplicația este flexibila, putând schimba cam tot înafara de dimensiunile matricei. ofera o vizualizare care diferențiaza diferitele tipuri de celule și care este interactiva (acțiunile facându-se din mouse), totuși la partea de control al fluxul are un subset incomplet, lipsând controlul de timp.

**[2] USF Data Structure Vizualizations**

Universitatea din San Francisco, departamentul de Computer Science, a construit o listă de vizualizări

# 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