

# Cuprins

1. Introducere.....	2
1.1. Motivație.....	2
1.2. Obiectiv.....	2
1.3. Soluții existente.....	3
[1] PathFinding.js.....	3
6. Bibliografie.....	5
7. Lista de figuri.....	5

# 1. Introducere

## 1.1. Motivație

Odată cu intrarea în lumea informaticii, o persoană 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 învățării acestora este aceea că trebuie înțeles mai întâi modul de rulare a programului (funcții, bucle, recursivitate) după care trebuie abstractizate pentru o înțelegere mai ușoară a unui program, prin faptul că nu mai suntem restricționați 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 să 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 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 plăcută 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 să 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.

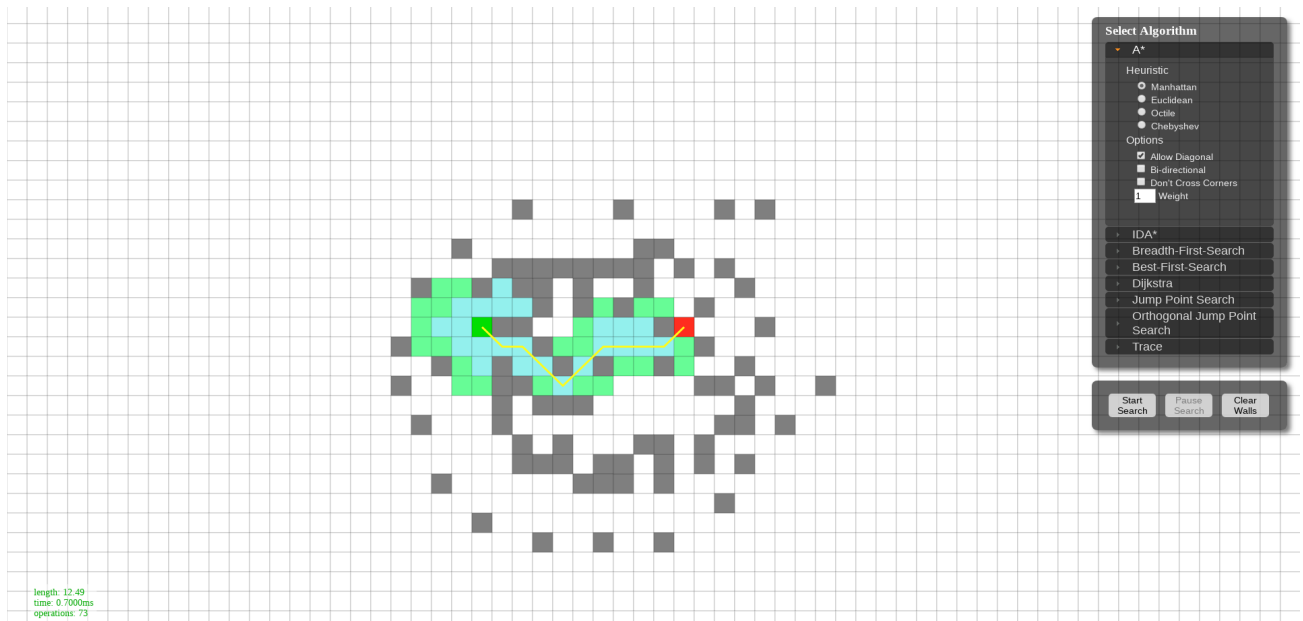
### [1] PathFinding.js

Aceasta este o aplicație care simulează diferiți algoritmi de găsim 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 interactivă care acceptă comenzi de mouse-click sau drag-and-drop. Există o matrice de dimensiuni fixe, cu un punct de start reprezentat prin verde și punct de oprire reprezentat prin roșu.

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



**Figura 1-1 PathFinding.js**

## **6. Bibliografie**

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

## **7. Lista de figuri**