**Algoritmi pe grafuri**

**Autori:**

**Ardelean Andrei-Timotei**

**Ardelean Răzvan-Mitel**

# Cuprins

[Cuprins 1](#_Toc425589388)

[Aplicația 2](#_Toc425589389)

[Utilizarea aplicației 2](#_Toc425589390)

[Particularități 4](#_Toc425589391)

[Crearea unui algoritm 4](#_Toc425589392)

[Demonstrarea algoritmilor 5](#_Toc425589393)

[Posibilități de dezvoltare. 5](#_Toc425589394)

[Resurse Hardware și Software necesare. 6](#_Toc425589395)

[Bibliografie 6](#_Toc425589396)

# Aplicația

**Aplicația este una originală care facilitează învățarea structurilor de date de tip graf alături de aplicațiile acestora în nenumărați algoritmi.**

**Ea este foarte flexibilă, ea nu necesită o instalare, intrând astfel în categoria aplicațiilor portabile.**

**Aceasta se prezintă la un stadiu de „release”, fiind o versiune stabilă și curățată de orice eroare sau bug găsit. Interfața este una ușor de folosit, intuitivă, plăcută la aspect. Aceasta se adaptează la diferite rezoluții și permite utilizatorului să folosească eficient spațiul. Culorile alese sunt în nuanțe pastelate, nu sunt stridente, obositoare.**

**Prin adăugarea diferitor algoritmi sub forma unor extensii se permite actualizarea conținutului iar utilizatorul poate modela aplicația pentru a-i satisface propriile cerințe.**

# Utilizarea aplicației

Pentru demonstrație am creat un editor de grafuri, prin care utilizatorul poate deschide un fișier de tip .graf(graf salvat anterior) sau poate construi un nou graf creând noduri, muchii și definind valorile acestora.

Prin apăsarea butonului „Construiește” utilizatorul deschide un toolbox cu opțiuni pentru adăugarea unui nod nou, adăugarea unei noi muchii sau auto-aranjarea în spațiu a nodurilor sub forma unui poligon convex regulat cu n vârfuri.

Prin Click dreapta pe unul dintre noduri el este șters, împreună cu toate muchiile incidente acestuia.

Prin metoda „drag and drop” nodurile pot fi mutate și aranjate de către utilizator într-un mod care să faciliteze urmărirea algoritmului.

La apăsarea butonului „Generează” aplicația va genera un graf cu 7 noduri, cu muchii de costuri randomizate. Acestea vor fi aranjate în forma unui heptagon, după modelul auto-aranjare.

Un graf astfel creat poate fii modificat ulterior prin adăugarea altor noduri/muchii sau prin ștergerea lor, in submeniul „Construiește”.

Graful creat poate fi mai apoi salvat în fișier, pentru utilizări ulterioare.

Din primul submeniu, „Algoritmi” pot fi selectate două opțiuni: „Creează” si „Adaugă”. Opțiunea „Creează” va deschide o nouă fereastră în care se poate crea și salva cod sursă scris de utilizator, alături de partea teoretică-explicativă a codului, sub forma unui fișier .txt sau .algo care descrie un algoritm și poate fi adăugat ulterior in aplicație folosind opțiunea „Adaugă”.

Crearea unui algoritm de către utilizator sub forma unor astfel de plug-in-uri permite aplicația să fie extinsă iar conținutul poate fi actualizat și gestionat atât de developer cât și de un Third-party. Această opțiune este explicată mai pe larg la „Particularități – Crearea unui algoritm”

Prin opțiunea Adaugă se poate alege un fișier de tip .txt sau .algo. Acesta este mai întâi verificat ca fiind un algoritm valid. Dacă acesta corespunde undei verificări generale este adăugat în folder-ul aplicației si va fi adăugat ca submeniu in bara principală.

Toți algoritmii adăugați vor fi vizibili în oricare redeschidere a aplicației până când se alege opțiunea „Ștergere” a algoritmului. Această modificare este vizibilă atât instant cât și pe durata întrucât și fișierul .algo corespunzător algoritmului este șters din folder-ul aplicației.

Există doi algoritmi predefiniți: „Kruskal” și „Prim” care rezolvă problema Arborelui Parțial de Cost Minim. Aceștia nu pot fi șterși și pot manipula controalele aplicației si elementele grafului în moduri care nu sunt valabile într-un algoritm creat de utilizator.

La apăsarea butonului „Demonstrație” din submeniul „Kruskal”, submeniul „Prim” sau alt submeniu care se refera la un algoritm(adăugat de către utilizator) se va deschide în partea dreaptă interfața de control a algoritmului. Prin butonul „Setup” sunt preluate datele grafului și începe rularea propriu zisă a algoritmului.

Prin folosirea succesivă a butonului „Înainte” Utilizatorul poate rula programul pas cu pas, observând la fiecare iterație modificările petrecute în memorie, dar și direct pe graf. De asemenea, poate fi utilizată opțiunea „Derulare rapidă” pentru a ajunge direct la rezultat.

Interfața prezintă un chenar în care este explicat pasul curent și cel următor. Sub acesta sunt dispuse 3 coloane cu rol de „Watch”. În primul sunt reprezentate nodurile și informația incidentă acestora. În al doilea și al treilea se afla câte o lista de muchii prin care este facilitată o observare amănunțită a algoritmului, pas cu pas.

Pentru o vizualizare mai bună a algoritmului nodurile sunt colorate. Inițial toate au culori diferite. In decursul algoritmului acestea se pot modifica pentru ca să se evidențieze anumite grupări de noduri, care vor avea aceeași culoare. Astfel, derularea algoritmului poate fi urmărită vizual, ușor de înțeles.

Din submeniurile pentru algoritmi mai poate fi accesată explicația completă a acestora. În submeniul predefinit „Kruskal” există încă o categorie, „Optimizări” unde este explicat algoritmul de Păduri de Mulțimi Disjuncte, prin care complexitatea algoritmului scade considerabil.

# Particularități

## Crearea unui algoritm

Pentru a crea un nou algoritm utilizatorul trebuie să intre în submeniul „Algoritmi”, opțiunea „Crează”. O nouă fereastra este deschisă. Aceasta conține niște imagini alternate cu Textbox-uri unde utilizatorul poate scrie cod sursă pentru a crea algoritmul dorit. Acesta poate controla atât explicațiile cât și „watch”-urile (informația nodurilor, listele de muchii, culoarea nodurilor).

Codul scris de utilizator, împreună cu codul standard(rezumat în imaginile văzute de utilizator) este compilat sub standardul C# 4.0 (În viitor aplicația poate fi extinsa pentru ca utilizator să poată crea algoritmul si folosind limbajul C++ sau alte limbaje).

Exista 4 submeniuri: „Save”, „Load”, „Verifică codul” și „Adaugă teorie”.

* Save: Utilizatorul are opțiunea de a salva algoritmul sub forma unui fișier .txt sau .algo ambele având același conținut care constă în teorie + codul complet, formatat. Înainte ca salvarea să fie finalizată se produce o verificare a codului, dacă se întâmpină erori algoritmul nu va putea fi salvat.
* Load: Utilizatorul are opțiunea de a relua un algoritm și a îl îmbunătăți. Poate alege un fișier .txt sau .algo. Codul va fi împărțit în casetele de text corespunzătoare și se va salva și teoria din acest fișier.
* Verifică codul: După apăsarea acestei opțiuni codul este compilat. Dacă există erori la compilare acestea vor apărea în josul ferestrei cu mesajele corespunzătoare produse de compilator. În cazul în care codul poate fi compilat cu succes se va afișa un mesaj corespunzător.
* Adaugă teorie: Folosind această opțiune utilizatorul poate adăuga o parte teoretică care să însoțească algoritmul în sine. O data ce un fișier text este adăugat va apărea un mesaj care înștiințează utilizatorul ca a fost adăugată teorie.

## Demonstrarea algoritmilor

În demonstrarea unui algoritm sunt cuprinse ecranele de afișare a datelor. Instrucțiunile și explicațiile sunt scrise într-un textbox, iar datele sunt afișate în 3 controale de tip ListView diferite.

Există 2 algoritmi predefiniți care pot accesa funcții speciale inaccesibile unui algoritm creat de utilizator, acești algoritmi nu pot fi șterși: „Krukal” si „Prim”.

La apăsarea butonului „Setup”, datele grafului sunt preluate de control, și începe rularea algoritmului. Într-un algoritm creat de utilizator funcția „start” este apelată.

La apăsarea butonului „Înainte”, este apelată funcția „executa” implementată de utilizator.

„Derulare Rapida” face ca funcția „executa” să fie apelată până când se apelează funcția „finish” din clasa „algoClass”. Dacă se fac execuții numeroase fără ca această funcție să fie apelată, utilizatorul va fi întrebat dacă dorește să continue. Aceasta măsură previne intrarea programului într-o buclă infinită care ar putea strica buna execuție a aplicației noastre în instanța curentă.

Odată ce este apelată funcția „finish” se va afișa un MessageBox cu mesajul trimis ca parametru de tip String funcției. Această utilitate poate fi folosită de implementatorul algoritmului pentru a informa utilizatorul cu privire la desfășurarea algoritmului; echivalentul unui „exit code”.

La fiecare pas sunt actualizate „Watch”-urile, nodurile și muchiile în funcție de noile valori ale variabilelor corespunzătoare din algoritm și de funcțiile utilitare apelate de acesta.

# Posibilități de dezvoltare.

Aplicația oferă posibilități multiple în vederea unei dezvoltări ulterioare.

O dezvoltare ar fi ca aplicația să suporte construirea unui algoritm si folosind cod sursa în alte limbaje cum ar fi C++, python sau chiar limbaje de scriptare.

O idee de dezvoltare este aceea de a crea o bază de date online cu diferiți algoritmi. Astfel dacă cineva a creat un algoritm si l-a explicat într-o formă prezentabilă să poată să îl uploadeze in baza de date pentru a fi disponibil si altor utilizatori interesați de acel algoritm. Acest lucru ar scuti munca multor utilizatori de a construi anumiți algoritmi cunoscuți și care probabil au fost implementați deja într-un mod corespunzător de o altă persoană. Pentru a evita riscul downloadări anumitor programe de risc se poate crea un sistem de votare a acestora iar algoritmii necorespunzători să fie scoși în mod automat. Astfel algoritmii folositori și bine explicați vor fi ușor de găsit, iar șansele de a găsi un pseudo-algoritm cu cod dăunător scad considerabil.

În mod evident, aplicația poate fi extinsă și în sensul în care să se permită unui utilizator să controleze mai multe funcții cu privire la graf cum ar fi ștergerea/adăugarea unui nod sau să customizeze „Watch”-urile.

# Resurse Hardware și Software necesare.

Aplicația folosește puține resurse hardware: aproximativ 30 MB(RAM) reali, 5MB pe disc și poate fi rulată pe orice platformă Windows XP sau mai nouă care suporta .NET 4.0

Configurație minimă de exemplu:

* Procesor : Intel Core2 Duo CPU 2.0 GHz
* Memorie RAM 2 GB
* SO: Windows: XP, VISTA, 7, 8, 10
* Necesită .NET versiunea 4.0 sau mai recentă să fie instalată.

# Bibliografie

1. Microsoft - Programarea orientată pe obiecte si programarea vizuala .Net Framework
2. http://stackoverflow.com/
3. https://msdn.microsoft.com/en-us/library/
4. https://ro.wikipedia.org/wiki/

O prezentare mai detaliată a resurselor externe folosite se afla în fișierul README