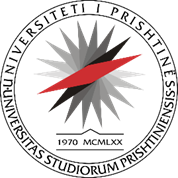
**UNIVERSITETI I PRISHTINËS**

**FAKULTETI I SHKENCAVE MATEMATIKO-NATYRORE**

**DEPARTAMENTI I MATEMATIKËS**



**Punim seminarik**

***LËNDA: Inteligjenca Artificiale***

***TEMA:* *Algoritmi Depth-First Search***

***Profesori:                                                                      Studentët:***

*Prof. Dr. Eliot Bytyçi* *Donjeta Sverka*

*Ass. Besnik Duriqi* *Shkurtesa Krosa*

*Shpat Spahiu*

*Prill 2023, Prishtinë*

**Përmbajtja**

[**Çfarë është një algoritëm Depth First Search?** 3](#_Toc135585650)

[**Si punon** 4](#_Toc135585651)

[**Pseudokodi** 7](#_Toc135585652)

[**Kodi** 7](#_Toc135585653)

[**Kompleksiteti i Depth First Search** 9](#_Toc135585654)

[**Përparësitë dhe mangësitë e Depth First Search** 9](#_Toc135585655)

[**Referencat** 9](#_Toc135585656)

# **Çfarë është një algoritëm Depth First Search?**

Depth First Search (DFS) është një algoritëm traversal i përdorur si për strukturat e të dhënave Tree dhe Graph. Traversal do të thotë të vizitosh të gjitha nyjet e një grafi.

Kur përshkohet një grafik, një algoritëm DFS gjithmonë fillon në kulmin burimor të një peme dhe më pas shkon poshtë një shteg të vetëm për të arritur kulmin e destinacionit. Për të kuptuar më mirë se si funksionon, imagjinoni se po përpiqeni të zgjidhni një labirint si ai më poshtë.

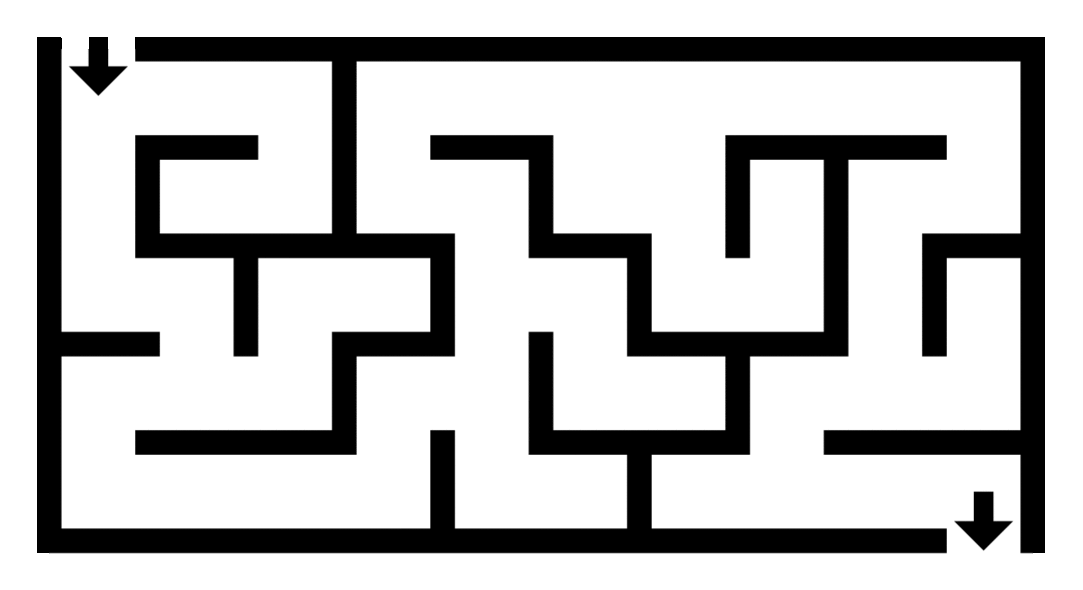


Figura 1.Një labirint [1]

Teknikisht, ka vetëm një pikë hyrje dhe një dalje në një labirint si ky. Megjithatë, nga hyrja në pikën e daljes, mund të ketë rrugë të shumta që mund të merrni për ta zgjidhur atë. Në mënyrë tipike, qasja më praktike për të zgjidhur problemin është të provoni të gjitha shtigjet. Pra, nga pika e fillimit, do të marrësh një rrugë derisa të arrish në fund. Nëse është një rrugë zgjidhje, atëherë do të provoni një rrugë tjetër. Do të ktheheshit në një pikë ku keni bërë një nga zgjedhjet më herët dhe do të provonit një tjetër. Këtë do ta bëni disa herë derisa të arrini në pikën e daljes, duke zgjidhur kështu problemin. [1]

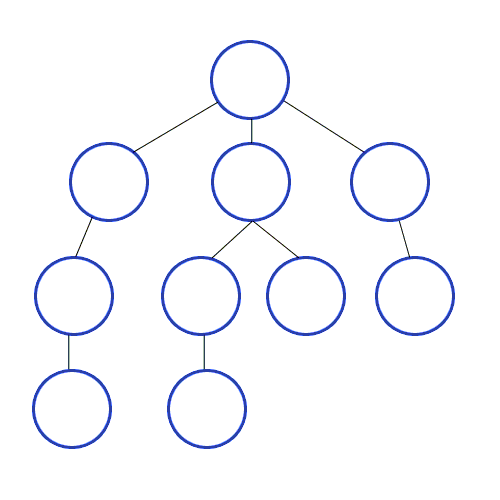
Nga një këndvështrim algoritmik, proceset që keni përdorur për të zgjidhur labirintin zakonisht quhen si kthim prapa (backtracking) dhe rekursion. Këto janë dy komponentë të rëndësishëm të algoritmit Depth First Search.

# **Si punon**

Një algoritëm DFS përshkon një pemë ose grafik nga kulmi prind deri te kulmet e fëmijëve dhe nipërve në një rrugë të vetme derisa të arrijë në një rrugë pa krye. Kur nuk ka më kulme për të vizituar në një shteg, algoritmi DFS do të kthehet në një pikë ku mund të zgjedhë një rrugë tjetër për të marrë. Do të përsërisë procesin pa pushim derisa të vizitohen të gjitha kulmet.

Kur kontrolloni dhe përditësoni çdo kulm, ekzistojnë tre gjëra që duhet të bëjë një algoritëm DFS:

* Lexoni të dhënat e ruajtura në nyjen që kontrollohet ose përditësohet.
* Kontrolloni kulmin në të majtë të nyjës që po kontrollohet.
* Kontrolloni kulmin në të djathtë të nyjës që po kontrollohet. [2]



***Strategjitë e Depth First Search***

Duhet theksuar se ka strategji që mund të përdoren në varësi të rendit në të cilin algoritmi dëshiron të ekzekutojë. Këto urdhra quhen:

* Preorder
* Inorder
* Post Order

Në *preorder* depth first search, algoritmi do të lexojë të dhënat e ruajtura duke filluar nga nyja rrënjë, pastaj do të lëvizë poshtë në nënpemën e nyjes së majtë në nënpemën e nyjes së djathtë. Kjo strategji zakonisht quhet DLR. [3]

##### Root -> Left Subtree -> Right Subtree

preOrder(node)

if (node != null)

visit(node)

preOrder(node.left)

preOrder(node.right)

Në *inorder* depth first search, algoritmi do të vizitojë nënpemën e majtë dhe më pas do të lexojë të dhënat e ruajtura në nyjen rrënjë përpara se të kalojë në nënpemën e djathtë. Kjo strategji njihet si LDR. [3]

##### Left Subtree -> Root -> Right Subtree

inOrder(node)

if (node != null)

inOrder(node.left)

visit(node)

inOrder(node.right)

Në *post order* depth first search, mundëson algoritmit të përshkojë pemën së pari duke filluar nga nënpema e majtë në nënpemën e djathtë përpara se të lexojë të dhënat e ruajtura në nyje. Kjo strategji DFS quhet LRD. [3]

##### Left Subtree -> Right Subtree -> Root

postOrder(node)

if (node != null)

postOrder(node.left)

postOrder(node.right)

visit(node)

Për t’i kuptuar më mirë tri strategjitë e DFS, marrim një shembull:

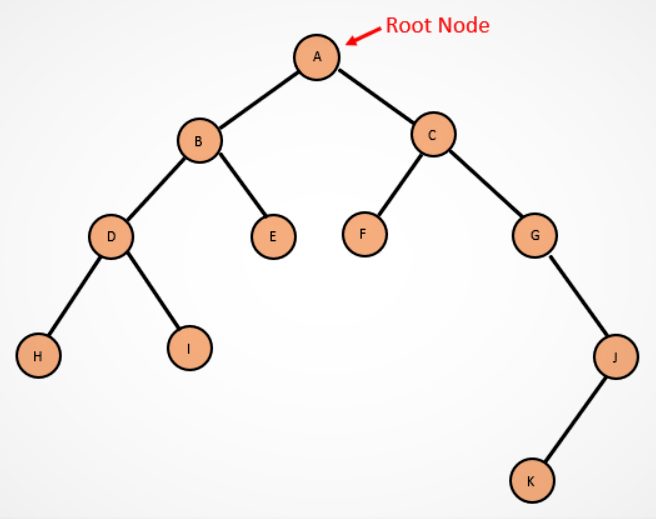


Figura 2. Sample Tree [2]

**Preorder DFS**

Algoritmi DFS do të përshkojë nyjet në rendin e mëposhtëm:

* Lexoni të dhënat në nyjen rrënjë A
* Nyjet përshkojnë në nënpemën e majtë sipas renditjes B, D, H, E, I
* Përshkoni nyjet në nënpemën e djathtë sipas renditjes C, F, G, J, K. [2]

**Inorder DFS**

Algoritmi DFS do të përshkojë nyjet në rendin e mëposhtëm:

* Vizitoni të gjitha nyjet në nënpemën e majtë duke filluar nga H në D, I, B, E
* Lexoni të dhënat në nyjen rrënjë A
* Përshkoni nyjet në nënpemën e djathtë në rendin e F, C, G, J, K. [2]

**Post Order DFS**

Algoritmi DFS do të përshkojë nyjet në rendin e mëposhtëm:

* Vizitoni nyjet në nënpemën e majtë duke filluar me nyjen H, I, D, E
* Përshkoni nyjet në nënpemën e djathtë sipas renditjes B, F, K, J, G, C
* Lexoni të dhënat në nyjen rrënjë A. [2]

***Rekursioni në Depth First Search***

Rekursioni është procesi i thirrjes së një metode brenda së njëjtës metodë, duke lejuar që një veprim të përsëritet vazhdimisht. Siç mund ta shihni, strategjitë e algoritmit DFS rrotullohen të gjitha rreth tre gjërave: leximi i të dhënave dhe kontrollimi i nyjeve në nënpemën e majtë dhe nënpemën e djathtë.

Ndonjëherë, mund të ndodhin ndryshime të vogla në varësi të rendit të procesit. Sidoqoftë, në përgjithësi, ne do të bëjmë të njëjtat gjëra në mënyrë të përsëritur derisa të vizitohen të gjitha kulmet në pemë. Në thelb, një pemë ka tre pjesë, të dhënat, një referencë të majtë dhe një referencë të djathtë. Me këtë informacion, është e lehtë të shihet se ne duhet të përsërisim procesin e leximit të tre pjesëve të një nyje për secilën nyje në tre.

Pa rekursion, algoritmi DFS nuk do të jetë në gjendje të kontrollojë të gjitha nyjet në një pemë, sepse asnjë funksion nuk do ta lejojë atë të përsërisë veprimin e tij. Thënë kështu, përfundimi i procesit të kontrollit të rrënjës ose nyjës prind nuk do të jetë i mundur.

# **Pseudokodi**

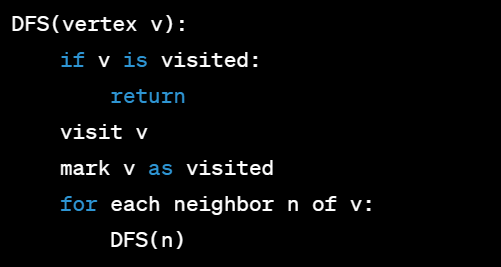
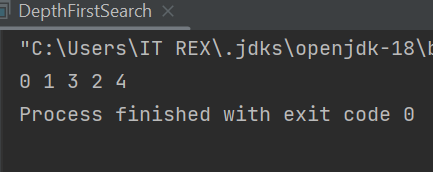


Figura 3. Pseudokodi

# **Kodi**

import java.util.ArrayList;  
import java.util.List;  
  
public class DepthFirstSearch {  
 private List<Integer>[] graph;  
 private boolean[] visited;  
  
  
 public DepthFirstSearch(int numVertices) {  
 graph = new List[numVertices];  
 visited = new boolean[numVertices];  
  
 for (int i = 0; i < numVertices; i++) {  
 graph[i] = new ArrayList<>();  
 }  
 }  
  
  
 public void addEdge(int u, int v) {  
 graph[u].add(v);  
 graph[v].add(u);  
 }  
  
  
  
 public void dfs(int v) {  
 visited[v] = true;  
 System.*out*.print(v + " ");  
  
  
 for (int n : graph[v]) {  
 if (!visited[n]) {  
 dfs(n);  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 DepthFirstSearch dfs = new DepthFirstSearch(5);  
 dfs.addEdge(0, 1);  
 dfs.addEdge(0, 2);  
 dfs.addEdge(1, 3);  
 dfs.addEdge(2, 4);  
  
  
 dfs.dfs(0);  
 }  
}

**Output:**



# **Kompleksiteti i Depth First Search**

Kompleksiteti kohor i algoritmit DFS paraqitet në formën e O(V + E), ku V është numri i nyjeve dhe E është numri i skajeve.

Kompleksiteti hapësinor i algoritmit është O(V). [4]

# **Përparësitë dhe mangësitë e Depth First Search**

Përparësitë e DFS:

* Kërkon më pak memorie pasi ruhen vetëm nyjet në shtegun aktual.
* Mund të gjejë një zgjidhje pa shqyrtuar fare pjesën më të madhe të hapësirës së kërkimit.
* Më pak kompleksitet në kohë dhe hapësirë sesa BFS. [5]

Mangësitë e DFS:

* Mund të gjejë një zgjidhje nën-optimale (një që është më e thellë ose më e kushtueshme se zgjidhja më e mirë)
* E paplotë: pa një kufi thellësie, mund të mos gjejë një zgjidhje edhe nëse ekziston. [5]

# **Referencat**

|  |  |
| --- | --- |
| [1] | K. J. a. J. K. Karleigh Moore, "Brilliant," [Online]. Available: https://brilliant.org/wiki/depth-first-search-dfs/. [Accessed 13 April 2023]. |
| [2] | A. D. Ridder, "Edgy," 05 June 2019. [Online]. Available: https://edgy.app/depth-first-search-algorithm-what-it-is-and-how-it-works. [Accessed 14 April 2023]. |
| [3] | A. Walker, "GURU99," [Online]. Available: https://www.guru99.com/tree-traversals-inorder-preorder-and-postorder.html. [Accessed 13 April 2023]. |
| [4] | "Programiz," [Online]. Available: https://www.programiz.com/dsa/graph-dfs. [Accessed 14 April 2023]. |
| [5] | V. M. H, "VTUPulse," [Online]. Available: https://www.vtupulse.com/artificial-intelligence/depth-first-search-example-advantages-and-disadvantages/. [Accessed 14 April 2023]. |