DEPTH FIRST SEARCH

Rreze Vrapçani[#1]
FSHMN - Shkencë Kompjuterike
Universiteti i Prishtinës
Prishtinë, Kosovë, 10000
1 rreze.vrapcani@student.uni-pr.edu

Erë Dedinca[#2]
FSHMN - Shkencë Kompjuterike
Universiteti i Prishtinës
Prishtinë, Kosovë, 10000
2 ere.dedinca@student.uni-pr.edu

Rubina Berisha[#3]
FSHMN - Shkencë Kompjuterike
Universiteti i Prishtinës
Pejë, Kosovë, 30000
3 rubina.berisha@student.uni-pr.edu

Abstrakt: Në këtë punim do të përfshihen disa detaje në lidhje me algoritmin e kërkimit Depth first (Kërkimi sipas thellësisë), do të spjegohet algoritmi përkatës, do ceken disa nga përparësitë e mangësitë e përdorimit të këtij algoritmi dhe më tutje përfshihet implementimi I këtij algoritmi duke përdorur gjuhën programuese Java. Gjithashtu paraqiten edhe shembuj me anë të të cilëve është ilustruar se si bëhet kërkmi sipas thellësisë.

Fjalë kyçe: Kërkim, thellësi, algoritëm, kompleksitet

I. ALGORITMI

Çka është algoritmi? Algoritmi është procedurë hap pas hapi, përzgjidhjen e problemit. Algoritmi është proceudra e kryerjes së ndonjë detyre të caktuar. Algoritmi është idea prapa cilitdo program kompjuterik. Këto do të ishin disa prej definicioneve më të thjeshta lidhur me atë se çka është algoritmi. Përndryshe ekzistojnë edhe shumë definicione të tjera, të cilat në mënyra tëndryshme e japin shpjegimin ose mundohen ta sqarojnë se çka është algoritmi. Algoritmi definohet edhe si: Algoritmi është bashkësi e rregullave për kryerjen ellogaritjeve me dorë ose me ndonjë pajisje [1]. Algoritmi është një procedurë e përcaktuar hap pas hapi për arritjen e një rezultati të caktuar. Algoritmi është njëvarg i hapave llogaritës që e transformojnë hyrjen në dalje. Algoritmi është njëvarg i operacioneve të kryera në të dhënat që duhet të jenë të organizuara nëstruktura të të dhënave. Algoritmi është një abstraksion i programit që duhet të ekzekutohet në një makinë fizike (modeli i llogaritjes), etj. Algoritmi më I njohur në histori daton që nga koha e Greqisë antike: ky është "Algoritmi iEuklidit" për llogaritjen e pjestuesit më të madh të përbashkët të dy numrave të plotë. Shumica e algoritmeve të rëndësishme përfshijnë metodat për organizimin e tëdhënave të përfshira në llogaritje. Objektet e krijuara në këtë mënyrë quhet ***struktura të të dhënave*** dhe këto janë gjithashtu objekte qëndrore të studimitnë shkencat kompjuterike [2]. Prandaj, algoritmet dhe strukturat e të dhënave, shkojnë "dorë për dore" (së bashku). Pra, për të kuptuar algoritmet duhetstudiuar edhe strukturat e të dhënave. Ka raste kur algoritmet e thjeshta "nxjerrinnë pah" struktura të komplikuara dhe anasjelltas, algoritmet e komplikuaramund të përdorin struktura të thjeshta të të dhënave. Parimisht, do të studiohendhe prezentohen tiparet (vetitë, karakteristikat) e shumë strukturave të tëdhënave. Kur përdorim kompjuterin për të zgjidhur një problem, zakonisht ballafaqohemime një numër të qasjeve të ndryshme të mundshme për zgjidhjen e problemit. Për problemet e vogla, rrallë herë është me rëndësi se cila qasje përdoret, përderisa e kemi atë që e zgjidhë problemin

si duhet. Mirëpo, për problemet e mëdha (ose për aplikacionet ku duhet njënumër i madh i problemeve të vogla),shpejt motivohemi që të krijojmë metoda të cilat përdorin kohën dhe hapësirën(memorike) në mënyrë sa më efikase të mundshme. Arsyeja kryesore për studimin e dizajnit të algoritmeve është se kjo disciplinë na jep potencialin për të bërë kursime të shumta edhe deri në pikën e mundësimit tëkryerjes së detyrave të cilat ndryshe do të ishte e pamundur të kryhen. Në një aplikacion ku procesohen miliona objekte, nuk është e pazakontë që të bëhet një program miliona herë më i shpejtë, duke përdorur një algoritëm të dizajnuarmirë. Kurse, investimi në blerjen e kompjuterit të ri me performansa më të mira, për të njëjtin problem, ka potencial të përshpejtimit me faktor prej vetëm 10 ose 100 herë. Dizajni i kujdesshëm i algoritmit është pjesë jashtëzakonisht efektive e procesit të zgjidhjes së problemeve të zgjidhjes së problemeve të mëdha, në çdo sferë të aplikimit.

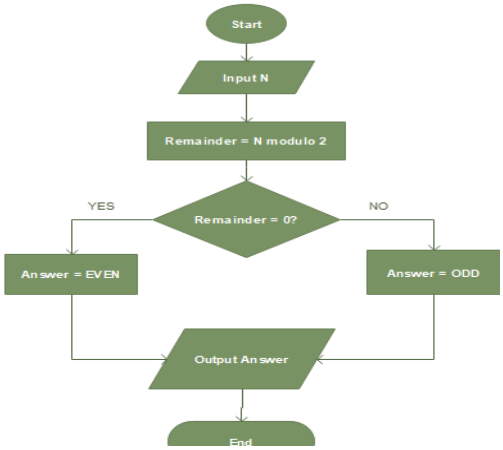


Figure 1 Mënyra e funksiomimit të algoritmit

KOMPLEKSITETI KOHOR

Kompleksiteti kohor është sasia e kohës që i duhet një algoritmi për tu ekzekutuar, në funksion të gjatësisë së hyrjes. Ai e mat kohën për të ekzekutuar çdo deklaratë të kodit në një algoritëm.

Ekziston një lidhje midis madhësisë së të dhënave hyrëse (n) dhe një numri operacionesh të kryera (N) në lidhje me kohën. Kjo lidhje njihet si *rendi i rritjes në kompleksitetin kohor* dhe jepet me O[n] ku O është rendi i rritjes dhe n është gjatësia e hyrjes [3]. Ekzistojnë lloje të ndryshme të kompleksiteteve kohore, si:

1. Koha konstante – O (1)
2. Koha lineare – O(n)
3. Koha logaritmike – O (log n)
4. Koha kuadratike – O (n^2)
5. Koha kubike – O (n^3)

Disa nga funksionet shumë më komplekse janë: koha eksponenciale, koha kuazilineare, koha faktoriale, etj. Të cilat përdoren varësisht nga lloji i funksioneve të përcaktuara.

Algoritmi ***Brute Force*** është një teknikë tipike e zgjidhjes së problemit ku zgjidhja e mundshme për një problem zbulohet duke kontrolluar secilën përgjigje një nga një, duke përcaktuar nëse rezultati plotëson deklaratën e një problemi apo jo [4]. Algoritmi i forcës brutale zgjidhet në mënyrën më të drejtpërdrejtë, pa përfituar nga ndonjë ide që mund ta bëjë algoritmin më efikas. Kompleksiteti kohor i forcës brutale është O(m*n).

KOMPLEKSITETI HAPËSINOR

Kompleksiteti i hapësirës së një algoritmi paraqet hapësirën e marrë nga algoritmi në lidhje me madhësinë e hyrjes. Kompleksiteti hapësinor përfshin dy hapësira: ndihmëse dhe hapësirën e përdorur nga inputi. Kompleksiteti hapësinor është një koncept paralel me kompleksitetin kohor [5]. Nëse krijojmë një grup me madhësi n, ky grup do të kërkojë hapësirë O(n). Nëse krijojmë një grup dy-dimensional me madhësi n*n, kjo kërkon hapësirë $O(n^2)$ [3]. Është më se e nevojshme të përmendet se kompleksiteti i hapësirës varet nga një një numër faktorësh si: gjuha programuese, përpiluesi, apo edhe makina që drejton algoritmin.

II. GRAFET

Graf është objekt që në mënyrë më të lirshme mund të përkufizohet si bashkësi nyjash (anglisht vertex) të lidhura ndërmjet vete, ashtu që lidhjet paraqesin relacionet përkatëse ndërmjet nyjave të cilat quhen degët (anglisht edge) [6]. Është e zakonshme në teorinë e grafeve që nyjat dhe degët të shënohen me shkronjat e vogla të alfabetit latin: a, b, c... Në qoftë se dega e i lidh nyjat a dhe b, ajo mund të shënohet si e = {a, b} (në rast se nuk është e rëndësishme të theksohet kahja e degës dhe për një nocion të tillë do të bëhet fjalë më vonë) ose e = ab.

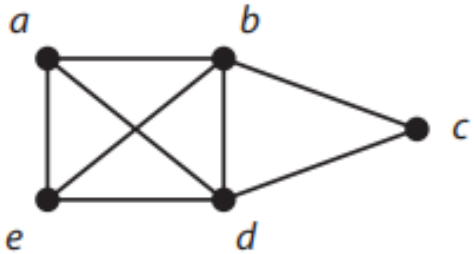


Figure 2 Grafi

Kështu në figurë është paraqitur grafi që përbëhet nga nyjat *a, b, c, d* dhe *e* dhe me degët *ab, ad, ae, bd, be, bc, cd dhe de.* Grafet janë shumë interesant, meqenëse nëpërmjet tyre në mënyrë shumë të thjeshtë mund të modelohen problemet e ndërlikuara nga sistemet reale. P.sh. në qoftë se shqyrtojmë një hartë gjeografike me një bashkësi të madhe qytetesh që janë të lidhura me anë të rrugëve – përftojmë një graf, në mënyrë që nyjet e grafit janë në të vërtetë qytetet, kurse degët e grafit paraqesin rrugët ndërmjet qyteteve në atë hartë. Shembull tjetër paraqesin njerëzit që përcjellin një shfaqje në një teatër. Nyjat e grafit përkatës janë njerëzit kurse degët formohen nga çiftet e njerëzve që njihen ndërmjet tyre. Formula strukturale e një molekule ose një kompozimi gjithashtu paraqet një graf. Skema e qarkut elektrik në teorinë e qarqeve ose në elektronikë gjithashtu paraqet një graf.

PARAQITJA E GRAFEVE

Ekzistojnë dy mënyra standarde për paraqitjen e grafeve: **koleksioni i listave të lidhshmërisë (fqinjësisë) dhe matricave të lidhshmërisë (fqinjësisë apo incidencës)** [7]. Të dy këto mënyra mund të zbatohen në grafet e orientuara dhe të paorientuara, mirëpo, meqenëse nuk jeni takuar deri tani me listat si struktura, do të prezantojmë vetëm mënyrën e paraqitjes nëpërmjet matricave. Për paraqitjen e grafeve me n nyja, nevojitet matrica me n rende (rreshta) dhe n kolona (shtylla), ashtu që secilës nyje "i përgjigjet" saktësisht një rend dhe një kolonë. Supozojmë se nyjat janë shënuar nëpërmjet numrave dhe se nyjës i i përgjigjet rendi i i-të dhe shtylla e i-të. Atëherë matrica e incidentës përkufizohet si më poshtë:

- vlera e fushës [i, j] është 1 në qoftë se ekziston dega që lidh nyjat i dhe j, në të kundërtën vlera është 0.

- te grafi me peshë, vlera e fushës [i, j] është e barabartë me peshën e degës përkatëse që lidh nyjën i me nyjën j në qoftë se ajo degë ekziston, në të kundërtën vlera është $-\infty$.

Para se të takohemi me algoritmat kryesore për punën me grafe, të fusim në përdorim nocionet e mëposhtme [7]:

- **dimensioni i grafit** është numri i nyjeve në graf (dimensioni i grafit nga figura 18.7. është 8);
- **nyjat fqinje** janë ato nyje që janë të lidhura me anë të një dege (në grafin nga figura 18.7. nyjat A dhe C janë fqinjë, kurse nyjat B dhe C nuk janë fqinjë);
- **shkalla e nyjës** është numri i degëve që arrijnë në një nyjë (shkalla e nyjës A në grafin nga figura 18.7. është 3);
- **nyja e izoluar** është ajo nyjë, shkalla e së cilës është 0, d.m.th. deri te ajo nyjë nuk arrin asnjë degë (nyja H është nyjë e izoluar në grafin nga figura 18.7.);
- **nyja pezulluese** është ajo nyje, shkalla e së cilës është 1, d.m.th. deri te ajo arrin vetëm një degë (nyja H është nyjë pezulluese në grafin nga figura 18.7.);
- **rruga** (udhëtimi) në graf (në graf me peshë apo pa peshë) është një varg i çfarëdoshëm nyjash fqinje. Nyja e parë dhe nyja përfundimtare janë fundet e rrugës, kurse, në qoftë se nyja e parë përputhet me nyjën përfundimtare, atëherë rruga është e mbyllur (shpeshherë thuhet se bëhet fjalë për një cikël, gjatësia e të cilit është e barabartë me numrin e degëve në rrugë) (në grafin nga figura 18.7. rruga C-A-F-E-G nuk është e mbyllur, kurse rruga D-E-F-D është e mbyllur, prandaj paraqet ciklin me gjatësi 3);
- **graf i lidhur** është ai graf i tillë që për çdo dy nyje të tij ekziston rruga që i lidh (grafi nga figura 18.7. nuk është i lidhur, sepse nuk ekziston rruga ndërmjet nyjave H dhe A);
- **komponenti i lidhshmërisë** së grafit është nëngrafi i tij më i madh i lidhur (një komponent lidhshmërie i grafit nga figura 18.7. është nëngrafi me nyjat A, B, C, D, E, F, G; komponenti i dytë është nëngrafi që përbëhet nga nyja H)

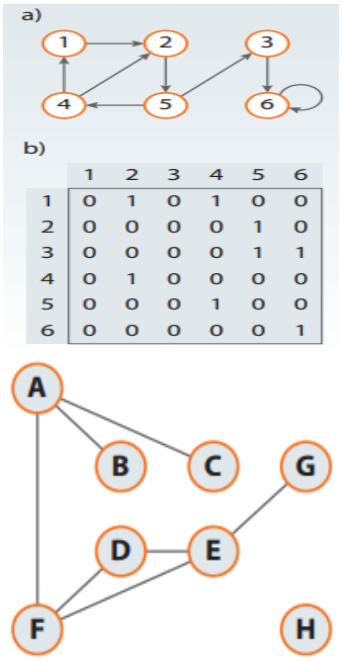


Figure 3 Paraqitja e grafeve

PËRSHKRIMI I GRAFIT

Algoritmet e para me të cilat do të takohemi janë algoritmet për përshkimin e grafeve. Ekzistojnë dy algoritma themelorë për përshkimin e grafeve: **përshkimi sipas gjerësisë** (anglisht breadth-first) dhe

përshkimi sipas thellësisë (anglisht deapth-first) [8].

Përshkimi sipas gjerësisë përbëhet nga përshkimi i të gjithë fqinjëve të drejtpërdrejtë të nyjës së dhënë. Nyjat fqinje shënohen dhe në secilin prej tyre zbatohet përshkimi i njëjtë gjithnjë deri sa ka nyje të pavizituara. Nyja fillestare mund të jetë çdo nyjë e grafit, kurse algoritmi përfundon, kur nuk ka më nyja të pavizituara që janë të arritshme nga nyjat e vizituara më herët.

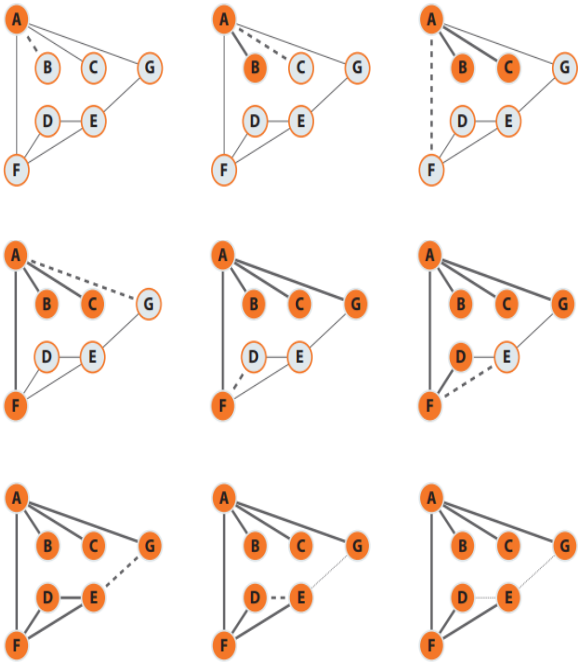


Figure 4 Përshkrimi sipas gjerësisë

dalin. Kur vizitohen të gjitha lidhjet nga nyja e dhënë v, përshkimi kthehet një hap më prapa, me qëllim që të vizitohen të gjitha lidhjet që dalin nga nyja prej të cilës është arritur deri te nyja v. Ky proces vazhdohet gjithnjë deri sa të vizitohen të gjitha nyjat që janë të arritshme nga nyja burimore. Në qoftë se ekziston një nyjë e pavizituar, ajo nyjë merret si nyjë fillestare dhe përshkimi përsëritet nga ajo nyjë. Procesi i tërë përsëritet gjithnjë deri sa të ketë nyje të pavizituara në graf.

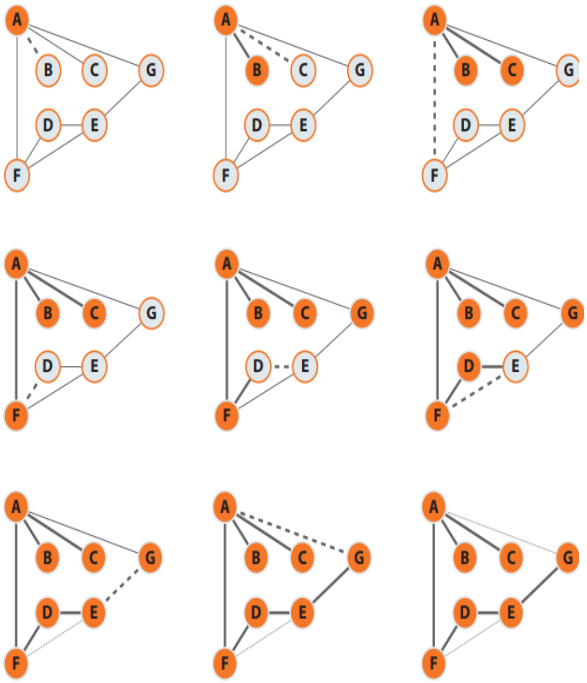


Figure 5 Përshkrimi sipas thellësisë

Strategjia gjatë **përshkimit sipas thellësisë** nënkupton përshkimin në thellësi të grafit, deri sa një proces i tillë është i mundshëm. Në përshkim sipas thellësisë, përshkohen lidhjet prej nyjës së fundit të vizituar v, gjithnjë deri sa të ketë lidhje të pavizituara që nga ajo nyjë

III. DEPTH FIRST SEARCH

Depth First Search (DFS) është një algoritëm kërkimi që përdoret në fusha të ndryshme të informatikës, duke përfshirë grafikën, inteligjencën artificiale dhe shumë fusha të

tjera. DFS eksploron grafin duke shkuar thellë në strukturën e saj dhe kërkon të gjejë një zgjidhje për problemin e dhënë. Algoritmi DFS fillon nga një nyjë e caktuar në graf dhe eksploron secilën nyjë që lidhet me këtë nyjë në mënyrë të thellë, deri sa të arrihet në një nyjë që nuk ka më lidhje me nyjet e tjera [9]. Në këtë moment, DFS kthehet prapa dhe vazhdon eksplorimin e nyjeve të tjera. Për të realizuar DFS, është e nevojshme që grafiku të paraqitet në formën e një liste lidhjesh, ku secila nyjë në graf ka një listë të gjitha nyjeve me të cilat është e lidhur. Në këtë mënyrë, DFS mund të shkojë nga një nyjë në tjetrën duke përdorur këtë listë. DFS mund të përdoret për të gjetur rrugën më të shkurtër midis dy nyjeve në graf, për të gjetur ciklet në një graf, ose për të zgjidhur probleme të tjera të ngjashme. Në implementimin e DFS, është e rëndësishme të përdoren struktura të dhënash efikase, si steka dhe tabele hash, për të ndihmuar në shënuarjen e nyjeve që janë vizituar dhe për të zbuluar ciklet në graf. Në përgjithësi, algoritmi DFS ka një kohë ekzekutimi O(V+E), ku V është numri i nyjeve në graf dhe E është numri i kapërcimeve midis tyre.

Pseudokodi i Depth First Search

- Fillimisht shënojmë të gjitha nyjet si të pazbuluara.
- Përzgjedhim një nyje fillestare.
- Shënojmë këtë nyje si të zbuluar.
- Përsërisim për secilën nyje fqinje të kësaj nyje që ende nuk është zbuluar:
- Thërrasim DFS për këtë nyje fqinje.
- Përsërisim këtë proces për secilën nyje fqinje të kësaj nyje që ende nuk është zbuluar [10].

KOMPLEKSITETI

Kompleksiteti hapesinor dhe kohor i Depth First Search (DFS) varet nga struktura e grafit që po kërkohet dhe nga implementimi i algoritmit. Kompleksiteti hapesinor i DFS varet nga madhësia e strukturës së grafitit dhe nga thellësia e shkarkimit në kërkim të zgjidhjeve [9]. Në rastin më të keq, kur grafi është i plotë me n nyje, kompleksiteti hapesinor i DFS është O(n). Megjithatë, në raste të tjera, mund të ketë një numër të madh të nyjeve që nuk duhet vizituar, duke reduktuar kështu kompleksitetin hapesinor. Kompleksiteti kohor i DFS varet nga numri i lidhjeve të secilës nyje në grafin e dhënë, dhe nga thellësia e shkarkimit. Në rastin më të keq, kur DFS eksploron gjithë grafin, kompleksiteti kohor është O(n+e), ku n është numri i nyjeve dhe e është numri i lidhjeve në grafin e dhënë. Megjithatë, në raste të tjera, DFS mund të ndalojë shkarkimet kur gjen zgjidhjen, duke reduktuar kështu kompleksitetin kohor.

IV. SHEMBUJ TË DEPTH FIRST SEARCH

Në këtë graf, nëse fillon DFS nga nyja 1, në atëherë DFS do të vijojë në mënyrën e mëposhtme:

1. Zbulohet nyja 1.
2. Eksplorohet nyja 2.
3. Zbulohet nyja 2.
4. Eksplorohet nyja 4.
5. Zbulohet nyja 4.

6. Kthehet prapa në nyjen 2.
7. Eksplorohet nyja 5.
8. Zbulohet nyja 5.
9. Kthehet prapa në nyjen 2.
10. Kthehet prapa në nyjen 1.
11. Eksplorohet nyja 3.
12. Zbulohet nyja 3.
13. Kthehet prapa në nyjen 1.

Figure 6. Ilustrimi i shembullit

Kështu, DFS kërkon në thellësi duke filluar nga nyja 1, dhe eksploron çdo nyje sa më thellë që është e mundur.

SHEMBULL

Një shembull tjetër është nëse kemi një graf të ndërlidhur me 6 nyje dhe 8 kërcime të mundshme midis tyre si në figurën e poshtme:

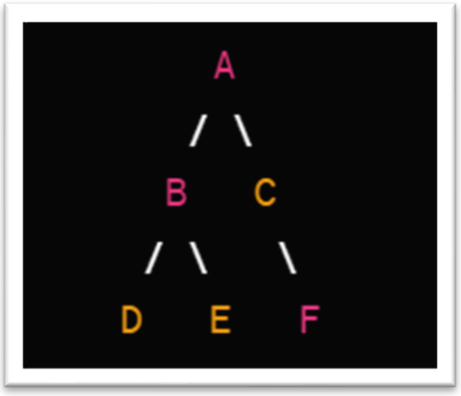
Në këtë graf, nëse fillon DFS nga nyja 1, në atëherë DFS do të vijojë në mënyrën e mëposhtme:

Figure 7. Ilustrim i shembullit

1. Zbulohet nyja 1.
2. Eksplorohet nyja 2.
3. Zbulohet nyja 2.
4. Eksplorohet nyja 5.
5. Zbulohet nyja 5.
6. Eksplorohet nyja 3.
7. Zbulohet nyja 3.
8. Kthehet prapa në nyjen 5.
9. Kthehet prapa në nyjen 2.
10. Eksplorohet nyja 4.
11. Zbulohet nyja 4.
12. Eksplorohet nyja 5 (që tashmë është zbuluar).
13. Kthehet prapa në nyjen 2.
14. Kthehet prapa në nyjen 1.
15. Eksplorohet nyja 4 (që tashmë është zbuluar).
16. Kthehet prapa në nyjen 1.

SHEMBULLI 3

Le të jetë grafi:



Nëse zgjedhim "B" si nyje fillimi, DFS do të shkojë në këtë mënyrë:

1. Shtojeni "B" në stivë.
2. Merrni "B" nga stiva dhe vizitoni të gjitha fqinjët e saj, "D" dhe "E".
3. Shtoni "D" dhe "E" në stivë.
4. Merrni "E" nga stiva dhe vazhdoni me hapin 2.
5. Merrni "D" nga stiva dhe vazhdoni me hapin 2.

Rezultati i DFS do të jetë: B, D, E.

SHEMBULLI 4

Le të kemi një graf të mëposhtëm:

Figure 8.Ilustrimi i grafit

Nëse zgjedhim "A" si nyjen fillimi, DFS do të shkojë në këtë mënyrë:

1. Shtojeni "A" në stivë.
2. Merrni "A" nga stiva dhe vizitoni të gjitha fqinjët e saj, "B" dhe "C".
3. Shtoni "B" dhe "C" në stivë.
4. Merrni "C" nga stiva dhe vizitoni fqinjën e saj, "E".
5. Shtoni "E" në stivë.
6. Merrni "E" nga stiva dhe vizitoni të gjitha fqinjët e saj, "H" dhe "I".
7. Shtoni "H" dhe "I" në stivë.
8. Merrni "I" nga stiva dhe vazhdoni me hapin 2.
9. Merrni "H" nga stiva dhe vazhdoni me hapin 2.
10. Merrni "B" nga stiva dhe vizitoni të gjitha fqinjët e saj, "D" dhe "F".
11. Shtoni "D" dhe "F" në stivë.
12. Merrni "F" nga stiva dhe vazhdoni me hapin 2.
13. Merrni "D" nga stiva dhe vizitoni të gjitha fqinjët e saj, "G".
14. Shtoni "G" në stivë.
15. Merrni "G" nga stiva dhe vazhdoni me hapin 2.

Rezultati i DFS do të jetë: A, B, D, F, G, C, E, H, I.

V. KONKLUZIONI

Siç mund të shihni, algoritmi DFS mund të implementohet në mënyra të ndryshme në varësi të gjuhës së programimit dhe strukturave të dhënash që përdoren. Në përgjithësi, implementimi i DFS kërkon një kujdes të madh për shënuarjen e nyjeve të vizituara dhe për të shmangur vizitimin e tyre përsëri. Në përgjithësi, DFS është një algoritëm i rëndësishëm kërkimi që përdoret gjerësisht në informatikë. Për më tepër, DFS është një prej algoritmeve bazë të shumë algoritmave të tjerë në fushën e inteligjencës artificiale dhe në fusha të tjera të matematikës diskrete.

REFERENCAT

[1] "TechTarget," 2023. [Online]. Available: https://www.techtarget.com/whatis/definition/algorithm.

[2] "JavaTPoint," 2021. [Online]. Available: https://www.javatpoint.com/data-structure-tutorial.

[3] "InterviewKickStart," [Online]. Available: https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms. [Accessed March 2023].

[4] "CodeAcademy," 2023. [Online]. Available: https://www.codecademy.com/learn/learn-data-structures-and-algorithms-with-python/modules/brute-force-algorithms/cheatsheet. [Accessed 2023].

[5] "StudyTonight," 2023. [Online]. Available: https://www.studytonight.com/data-structures/space-complexity-of-algorithms.

[6] "SplashLearn," [Online]. Available: https://www.splashlearn.com/math-vocabulary/geometry/graph. [Accessed April 2023].

[7] "JavaTPoint," 2023. [Online]. Available: https://www.javatpoint.com/graph-theory-graph-representations#:~:text=In%20graph%20theory%2C%20a%20graph,to%20it%20by%20an%20edge)..

[8] "GeeksForGeeks," [Online]. Available: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/. [Accessed April 2023].

[9] "InterviewCake," 2023. [Online]. Available: https://www.interviewcake.com/concept/java/dfs.

[10] "Programiz," 2023. [Online]. Available: https://www.programiz.com/dsa/graph-dfs. [Accessed April].

[11] https://sq.wikipedia.org/wiki/Algoritmi, "Algoritmi".