**UNIVERSITETI I PRISHTINËS**

***FAKULTETI I SHKENCAVE MATEMATIKE-NATYRORE***

***Departamenti i Matematikës***

Logo

Description automatically generated

**Prunim laboratorik**

***Lënda: Inteligjenca Artificiale  
Tema: Aplikimi i Uniform Cost Search-Dijkstra Algorithm***

***Profesori:***  ***Studenti:***

*Prof.Dr.Eliot Bytyci Leart Bllacaku   
Prof. Besnik Duriqi Gentin Morina Donat Gosalci*

Përmbajtja

[1. Hyrje 3](#_Toc132668996)

[2. Dijkstra algorithm 3](#_Toc132668997)

[3. Dokumentimi 3](#_Toc132668998)

[3.1 Metoda TSP 5](#_Toc132668999)

[3.2 Metoda getCityName 6](#_Toc132669000)

[3.3 Metoda main 6](#_Toc132669001)

[4. Rezultatet 7](#_Toc132669002)

[Figura 1, problemi i paraqitur si graf. 4](#_Toc132668670)

[Figura 2, matrica. 5](#_Toc132668671)

[Figura 3, variablat. 5](#_Toc132668672)

[Figura 4, kodi i metodës TSP. 6](#_Toc132668673)

[Figura 5, metoda main. 7](#_Toc132668674)

[Figura 6, konsola me rezultatin e marrë. 7](#_Toc132668675)

# Hyrje

Në këtë projekt, ne kemi implementuar algoritmin e njohur, Dijkstra, për të gjetur rrugën më të shkurtër që lidhë disa qytete të caktuara të Kosovës. Specifikisht, jemi fokusuar në 5 qytete: Prishtina, Mitrovica, Peja, Gjakova dhe Prizreni. Problemi i rrugëzimit është një problem i zakonshëm në shkencën kompjuterike që përfshin gjetjen e rrugës më të shkurtër midis dy ose më shumë nyjeve në një grafik. Implementimi ynë merr si hyrje distancat midis nyjeve dhe kthen rrugën më të shkurtër që i lidh ato. Rezultatet tregojnë efektivitetin e algoritmit në zgjidhjen e problemit të rrugëzimit. Ky dokument ofron një pasqyrë të algoritmit Dijkstra, përshkruan zbatimin tonë dhe paraqet rezultatet e eksperimentit.

# Dijkstra algorithm

Algoritmi Dijkstra është një algoritëm popullor për gjetjen e shtegut më të shkurtër midis dy nyjeve në një grafik me pesha jo negative. Algoritmi u përshkrua për herë të parë nga shpikësi i tij, Edsger W. Dijkstra, një shkencëtar kompjuterik holandez dhe pionier në fushën e programimit kompjuterik, në vitin 1959. Dijkstra e zhvilloi algoritmin ndërsa punonte në Qendrën Matematikore në Amsterdam, ku po përpiqej të zgjidhte problemin e rrugës më të shkurtër për një rrjet qytetesh në Holandë.

Algoritmi Dijkstra funksionon duke mbajtur një radhë prioritare të nyjeve, ku prioriteti bazohet në vlerësimin aktual të rrugës më të shkurtër. Fillimisht, vlerësimi i rrugës më të shkurtër për nyjen fillestare vendoset në 0, dhe të gjitha nyjet e tjera vendosen në pafundësi. Më pas, algoritmi zgjedh në mënyrë të përsëritur nyjen me përparësinë më të ulët nga radha e përparësisë dhe relakson skajet e saj dalëse. Nëse gjendet një shteg i ri më i shkurtër drejt një nyje, radha e përparësisë përditësohet me vlerësimin e ri të shtegut më të shkurtër. Ky proces përsëritet derisa nyja e destinacionit të zgjidhet nga radha prioritare, ose radha prioritare të jetë bosh.

Algoritmi i Dijkstra është një algoritëm i “greedy” që zgjedh gjithmonë nyjen me përparësinë më të ulët nga radha e përparësisë. Kjo siguron që algoritmi të zgjedhë gjithmonë vlerësimin e rrugës më të shkurtër nga nyja fillestare në nyjen aktuale. Megjithatë, algoritmi supozon se të gjitha peshat e skajeve janë jonegative, që do të thotë se mund të mos funksionojë siç duhet në grafikët me pesha negative, por meqenëse problemi jonë është i distancave mes qyteteve, atëherë nuk kemi të bëjmë me distanca negative.

# Dokumentimi

Dokumentimi për algoritmin Dijkstra do të japë një shpjegim të detajuar se si funksionon algoritmi dhe si mund të zbatohet në kod. Dokumentimi do të fillojë duke paraqitur një foto që paraqet problemin si grafik dhe më pas do të demonstrojë sesi funksionon algoritmi hap pas hapi derisa të gjendet një zgjidhje. Qëllimi i dokumentimit është të ofrojë një shpjegim të qartë dhe konciz të algoritmit në mënyrë që ai të kuptohet dhe zbatohet lehtësisht në aplikime të ndryshme. Duke e paraqitur problemin si grafik, dokumentimi do t'i lejojë lexuesit të vizualizojnë problemin dhe të kuptojnë më mirë se si funksionon algoritmi për të gjetur rrugën më të shkurtër midis dy nyjeve.

Në vijim është dhënë procedura se si funksionon algoritmi Dijkstra ose ndryshe (Uniform Cost Search).

* Hapi 1: Të gjitha nyjet duhet të shënohen si të pavizituara.
* Hapi 2: Të gjitha nyjet duhet të inicializohen me distancën "infinite" (një numër i madh). Nyja fillestare duhet të inicializohet me zero.
* Hapi 3: Shënoni nyjen fillestare si nyjen aktuale.
* Hapi 4: Nga nyja aktuale, analizoni të gjithë fqinjët e saj që nuk janë vizituar ende dhe llogaritni distancat e tyre duke shtuar peshën e skajit, e cila vendos lidhjen midis nyjes aktuale dhe nyjes fqinje me distancën aktuale të nyjes aktuale.
* Hapi 5: Tani, krahasoni distancën e llogaritur së fundi me distancën e caktuar për nyjen fqinje dhe trajtojeni atë si distancën aktuale të nyjes fqinje,
* Hapi 6: Pas kësaj, fqinjët përreth të nyjës aktuale, e cila nuk është vizituar, merren parasysh dhe nyjet aktuale shënohen si të vizituara.
* Hapi 7: Kur nyja mbaruese shënohet si e vizituar, atëherë algoritmi ka bërë punën e tij ndryshe.
* Hapi 8: Zgjidhni nyjen e pavizituar të cilës i është caktuar distanca minimale dhe trajtojeni si nyjen e re aktuale. Pas kësaj, filloni përsëri nga hapi 4.

Radar chart

Description automatically generated with low confidence

Figura , problemi i paraqitur si graf.

Ky implementim përdor një matricë fqinjësie për të përfaqësuar grafikun, ku rreshtat dhe kolonat përfaqësojnë qytetet, dhe vlerat përfaqësojnë distancat ndërmjet tyre.

Algoritmi fillon në qytetin fillestar dhe viziton qytetet e tij fqinjë, duke përditësuar distancat e tyre me burimin nëse gjendet një shteg më i shkurtër. Ky proces vazhdon derisa të vizitohen të gjitha qytetet dhe të finalizohen distancat e tyre nga burimi. Për të mbajtur gjurmët e qyteteve të vizituara dhe distancat e tyre, përdoren dy grupe: - të vizituara: një grup boolean për të shënuar qytetet e vizituara- distanca: një grup për të ruajtur distancat minimale nga burimi në çdo qytet

Text

Description automatically generated

Figura , matrica.

Text

Description automatically generated

Figura , variablat.

## Metoda TSP

Metoda tsp është një funksion rekurziv që merr tre parametra: pos, dist dhe visitedCount. Funksioni i secilës prej tyre, është dhënë më poshtë:

* **pos**- Pozicioni aktual në grafik. Ky është një numër i plotë që përfaqëson indeksin e qytetit aktual në grupin e grafikëve.
* **dist**- Distanca aktuale e udhëtuar deri tani. Ky është një dyfish që përfaqëson shumën e distancave midis qyteteve të vizituara deri më tani.
* Numri i vizituar: Numri i qyteteve të vizituara deri më tani. Ky është një numër i plotë që fillon nga 1 (qyteti fillestar konsiderohet gjithmonë i vizituar).

Qëllimi i metodës TSP është të gjejë rrugën më të shkurtër të mundshme që viziton të gjitha qytetet saktësisht një herë, duke filluar dhe duke përfunduar në të njëjtin qytet. Ai e bën këtë duke provuar në mënyrë rekursive të gjitha shtigjet e mundshme dhe duke mbajtur gjurmët e më të mirave të gjetura deri më tani.

Në çdo thirrje për TSP, metoda kontrollon nëse të gjitha qytetet janë vizituar. Nëse po, ai kontrollon nëse ka një rrugë nga qyteti aktual për në qytetin fillestar dhe nëse kjo rrugë është më e shkurtër se shtegu aktual më i mirë. Nëse është, ai përditëson shtegun më të mirë me shtegun aktual dhe distancën e tij.

Nëse të gjitha qytetet nuk janë vizituar ende, metoda kalon nëpër të gjitha qytetet që nuk janë vizituar ende dhe përpiqet t'i vizitojë ato duke thirrur në mënyrë rekursive TSP me qytetin e ri si pozicion aktual. Më pas, metoda nuk e viziton qytetin përpara se të provojë një tjetër.

Ky proces vazhdon derisa të jenë provuar të gjitha rrugët e mundshme dhe të gjendet rruga më e mirë dhe distanca e saj. Si përmbledhje, metoda TSP përdor një qasje rekursive për të gjetur rrugën më të shkurtër të mundshme që viziton të gjitha qytetet saktësisht një herë, duke filluar dhe duke përfunduar në të njëjtin qytet. Ai e bën këtë duke provuar të gjitha shtigjet e mundshme dhe duke mbajtur gjurmët e më të mirave të gjetura deri më tani.

Text

Description automatically generated

Figura , kodi i metodës TSP.

## Metoda getCityName

Metoda getCityName përdoret për të kthyer indeksin e qytetit (0-4) në emrin e tij përkatës. Merr një numër të plotë si hyrje dhe kthen një varg që përfaqëson emrin e qytetit. Në metodë, një deklaratë if-else përdoret për të kontrolluar vlerën e numrit të plotë të hyrjes i. Nëse është e barabartë me 0, metoda kthen emrin "Prishtina". Nëse është e barabartë me 1, metoda kthen emrin "Mitrovicë". Nëse është e barabartë me 2, metoda kthen emrin "Peja". Nëse është e barabartë me 3, metoda kthen emrin "Gjakova". Dhe nëse është e barabartë me 4, metoda kthen emrin "Prizren". Kjo metodë përdoret në metodën kryesore për të printuar emrat e qyteteve në rrugën më të mirë të gjetur nga algoritmi.

## Metoda main

Metoda kryesore është ku ne inicializojmë variablat, thërrasim algoritmin TSP dhe printojmë daljen. Së pari, vendosim pikën e fillimit si të vizituar dhe e shtojmë atë në shteg. Pastaj, ne e quajmë algoritmin TSP me pikën e fillimit, distancën e deritanishme (e cila fillimisht është 0) dhe numrin e nyjeve të vizituara (që fillimisht është 1 sepse ne kemi vizituar tashmë nyjen fillestare).

Algoritmi TSP do të vizitojë në mënyrë rekursive çdo nyje të pavizituar, do të përditësojë shtegun aktual dhe distancën më të mirë aktuale dhe do të kthejë shtegun më të mirë dhe distancën më të mirë në fund.

Pasi të përfundojë algoritmi TSP, ne printojmë shtegun më të mirë duke përsëritur mbi grupin bestPath dhe duke konvertuar indekset e nyjeve në emrat e qyteteve duke përdorur metodën getCityName. Ne gjithashtu shtypim distancën totale të shtegut më të mirë. Metoda kryesore është ajo që inicializon variablat e nevojshme, thërret algoritmin TSP dhe printon daljen.

Text

Description automatically generated

Figura , metoda main.

# Rezultatet

Pas ekzekutimit të kodit me të dhënat që kemi dhënë, si rezultat kemi marrë:

A screenshot of a computer

Description automatically generated

Figura , konsola me rezultatin e marrë.