**UNIVERSITETI I PRISHTINËS**

**FAKULTETI I SHKENCAVE MATEMATIKE-NATYRORE**

**DEPARTAMENTI I MATEMATIKËS**



**Projekt ne lenden inteligjence artificiale**

**TEMA:** Kruskal's algorithm

**Punuan:**

Arben Syla

Benita Mehmeti

Erona Gashi

**Prill 2023**

Përmbajtja

[**Abstrakt 3**](#_Toc104754710)

[**Hyrje 4**](#_Toc104754711)

[**Pershkrimi i Problemit 4**](#_Toc104754712)

[**Qasja në zgjidhjen e problemit 4**](#_Toc104754713)

[**Kruskal's algorithm 5**](#_Toc104754714)

[**Aplikimet e algoritmit të Kruskalit janë 6**](#_Toc104754715)

[**Rezultatet 8**](#_Toc104754716)

[**Diskutim Rezultateve 9**](#_Toc104754717)

[**Përfundim 9**](#_Toc104754718)

[**References 10**](#_Toc104754719)

**Abstrakt**

Në këtë punim do të trajtojmë problemin e gjetjes së rrugës më të shkurtër duke e përdorur algoritmin Kruskal. Në fillim do të njihemi shkurtimisht me grafet, pastaj do të tregojmë se qka janë “minimum spanning tree”, sa është numri i tyre që mundë të formohen nga një graf me n-kulme. Në vashdim do të tregojmë se si funksion algoritmi Kruskal,si do ta gjejë rrugën më të shkurtër dhe a mund të ketë ndonjë grafë që ky algoritëm nuk mund që të gjejë rrugën më të shkurtër. Si dhe do të përmendim disa algoritme të cilat janë të ngjajshme me algoritmin që e kemi marrë ne.

**Hyrje**

Një spanning tree (MST) minimale e grafit G, me V-kulme dhe E-brinjë, është numri minimal i nënbashkësive të brinjëve E, të grafit G me kosto më të vogël. Problemi MST ka qenë intensivisht i studiuar në të kaluarën pasi është një rrjet themelor problemi i projektimit me shumë aplikacione dhe për shkak se ai lejon përfundimin e algoritmeve në kohë polinomiale. Në praktikë (në makinat sekuenciale dhe me memorie te brendshme), dy algoritme të thjeshta që datojnë të paktën gjysmë shekulli me pare ende performojnë më mirë në shumicën e rasteve. Algoritmi i Kruskal gjen “minimum spanning forest” të një grafi te padrejtuar. Nëse grafi është i lidhur ai gjen numrin më të vogël të mundshëm të ”spanning tree“. Një ”spanning tree“ është një nënbashkësi e brinjëve që kanë kosto e që i lidhin të gjithë kulmet së bashku pa e formuar një cikel dhe me numrin minimal të kostos. Pra shuma e të gjithë brinjëve është më e vogla e mundshme. Më në përgjithësi, çdo grafik i padrejtuar me peshë (jo domosdoshmërisht i lidhur) ka një spanning forest, i cili është një bashkim i “tree“-ve minimale që shtrihen për komponentët e tij të lidhur. Ka shumë raste përdorimi për “spanning trees“. Një shembull është një kompani telekomunikacioni që bën shpërndarjen e rrjetit të telefonit, dhë është e udhës që për të gjithë shpërndarjen e kabllove të gjejë rrugën më të shkurtër, sepse do të punohet e njejta punë vetëm që do të ketë kosto më të vogël. Ky algoritëm u shfaq për herë të parë në Proceedings of the American Mathematical Society, f. 48-50 në 1956, dhe u shkrua nga Joseph Kruskal. Ideja e tij bazë ishte të zgjidhte n − 1 skaj një herë, dhe më pas përdori teknikën “greedy”, duke zgjedhur një skaj me kosto minimale që nuk mund të formojë cikël për t'u bashkuar me grupin që është zgjedhur. Ky algoritëm ndahet në x hapa, ku x është numri total i skajeve të pemës, duke gjykuar vetëm një skaj sipas rendit të rritjes së kostos së këtyre × skajeve në të njëjtën kohë. Kur njerëzit e konsideruan një avantazh dhe e shtuan atë në skajet e zgjedhura, nëse cikli shfaqet, atëherë braktiseni atë. Përndryshe, do të zgjidhej për grumbullim. Algoritme të tjera për këtë problem përfshijnë algoritmin e Prim, algoritmin e fshirjes së kundërt dhe algoritmin e Borůvka.

**Pershkrimi i Problemit**

Në projektin tone kemi trajtur problemin e “Kruskal Algorithm“. Pra problemi ynë është që në një graf të pa orientuar me n-kulme, të gjejmë rrugën me koston më të vogël të mundshme e cila formohet nga (n-1)-brinjë mirëpo me kusht që pasi që të lidhen kulmet nuk duhet të formohet një graf i mbyllur.

**Qasja në zgjidhjen e problemit**

Pra sic e cekëm më lartë do të merremi me zgjidhjen e problemit ku na duhet të gjejmë rrugën më të shkurtër të një grafi të pa orientuar i cili i ka të lidhura të gjitha kulmet e tij, mirëpo me kusht që të mos formohet një qark i cili përbëhet nga brinjët e rrugës më të shkurtër. Këtë problem do ta zgjidhim sipas logjikës së “Kruskal Algorithm“.

**Kruskal's algorithm**

Siç u përmend më herët, algoritmi Kruskal përdoret për të gjeneruar një “spanning tree” minimale që përfshin një grafik të caktuar. Por, çfarë është saktësisht një “spanning tree” me shtrirje minimale? Një pemë me shtrirje minimale është një nëngrup i një grafi me të njëjtin numër kulmesh si grafiku dhe skajet të barabarta me numrin e kulmeve -1. Ai gjithashtu ka një kosto minimale për shumën e të gjitha peshave të skajeve në një pemë.

Algoritmi i Kruskal i rendite të gjitha skajet sipas rendit sipas kostos nga me e vogla te me e larta dhe vazhdon të shtojë nyje në pemë vetëm nëse skaji i zgjedhur nuk formon ndonjë cikël. Gjithashtu, ai zgjedh skajin me një kosto minimale në fillim dhe skajin me një kosto maksimale në fund. Prandaj, mund të thuhet se algoritmi Kruskal bën një zgjedhje optimale në nivel lokal, duke synuar të gjejë zgjidhjen optimale globale. Kjo është arsyeja pse ai quhet Algoritem Greedy.

Në vijim kemi një fotografi ku është paraqitur një graf i pa orientuar:

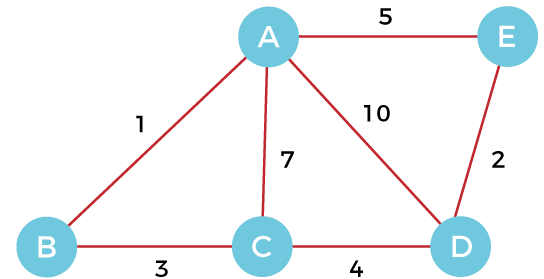


Fig .1. Grafi i pa orientuar

Algoritmi i Kruskal është një algoritëm i “spanning tree” me shtrirje minimale që gjen një skaj me “peshën” më të vogël të mundshme që lidh çdo dy pemë në pyll. Është një algoritëm i greedy në teorinë e grafikëve pasi gjen një “spanning tree” të shtrirjes për një grafik të pa orientuar të lidhur duke shtuar harqe në rritje të kostos në çdo hap. Kjo do të thotë se gjen një nëngrup të skajeve që formon një pemë që përfshin çdo kulm, ku pesha totale e të gjitha skajeve në pemë minimizohet.

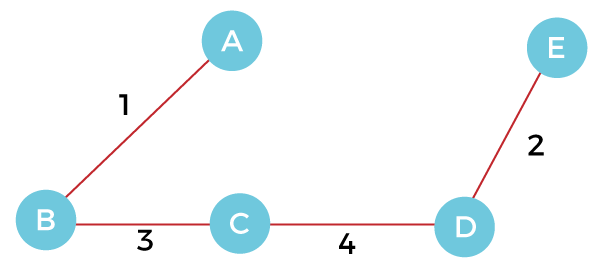


Fig.2. Spanning tree

Kompleksiteti kohor i ketij algoritmi eshte O(E logE), ku E eshte numri I skajeve ne graf. Ky kompleksitet vjen si pasoje e sortimit te skajeve te grafit nga algoritmi, dhe pastaj performimi I operacioneve te bashkesise disjunkte ne keto skaje gje qe merr kohe mesatarisht O(E logE).

Algoritmi I Kruskalit eshte nje greedy search algorithm, pasi qe ben zgjidhje lokalisht optimale me shpresen e gjetjes se nje zgjidhje globalisht optimale. Mirepo, suksesi I gjetjes se MST me kete algoritem eshte I garantuar.

Hapat qe ndjek ky algoritem jane:

1. Sortimi I skajeve te grafit sipas kostos ne renditje rritese.
2. Inicializimi I nje bashkesie te zbrazet ku do ruhen me pas skajet e MST.
3. Iterimi neper skajet e sortuara dhe shtimi I skajit ne bashkesi nese ky veprim nuk krijon nje cikel.
4. Ndalimi kur te gjitha kulmet jane te lidhura mes vete apo kur nuk kane mbetur me skaje per tu shqyrtuar.

**Perse nuk funksionon algoritmi I Kruskalit me grafe te orientuara?**

Ne algoritmin e Kruskalit, ne secilin hap behet kontrollimi I skajeve nee ato formojne nje cikel apo jo. Kjo shkakton deshtimin e ketij algoritmi ne grafet e orientuara pasi qe ne keto grafe ka raste kur nuk ka cikel fare mes kulmeve, mirepo algoritmi I Kruskalit gjithnje supozon se ekziston diku nje cikel dhe disa skaje nuk I merr parasysh fare, pra ato mbesin te pashqyrtuara dhe per kete arsye ky algoritem nuk funksionon per gjetjen e zgjidhjes optimale ne grafet e orientuara.

Në vijim paraqesim një demo:

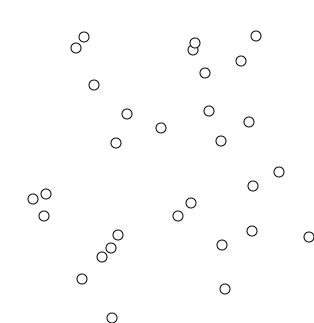


Fig.3. Demo

Një demonstrim për algoritmin e Kruskal-it në një grafik të plotë me pesha të bazuara në distancën Euklidiane.

**Aplikimet e algoritmit të Kruskalit janë**

* Algoritmi i Kruskal mund të përdoret për vendosjen e instalimeve elektrike midis qyteteve.
* Mund të përdoret për të vendosur lidhjet LAN.
* Per grumbullimin e te dhenave ne subgrupe me atribute te ngjashme apo segmentimin e imazheve.

|  |
| --- |
|  |

**Algoritmi 1.** Pseudokodi i algoritmit Kruskal

|  |
| --- |
| 1. Renditni të gjitha brinjët nga më e madhja e deri te më e vogla ne varësi të peshës.  2. Zgjidhni brinjën më të vogël. Kontrolloni nëse ajo formon një cikël me pemën që shtrihet deri tani. Nëse cikli nuk është formuar, përfshini këtë skaj. Përndryshe, hidheni atë.  3. Përsëriteni hapin #2 derisa të ketë skaje (V-1) në pemën që përfshin. |

|  |
| --- |
|  |

**Algoritmi 2.** Pseudokodi i metodës Graph()

|  |
| --- |
| Graph(**int** v, **int** e)         V = v;          E = e;          edge = **new** Edge[E];  **for** **int** i = 0 to e              edge[i] = **new** Edge(); |

Metoda Graph e cila është konstruktor i klasës Graph, krijon një graf me V-kulme dhe E-brinjë.

|  |
| --- |
|  |

**Algoritmi 3**. Pseudokodi i metodës find()

|  |
| --- |
| int find subset a [],int i          if subsets[i].parent != i              subsets[i].parent        = find subsets, subsets[i].parent  end if          return subsets[i].parent |

Metoda find() – gjen bashkësin e elementeve të i-ta

|  |
| --- |
|  |

**Algoritmi 4**. Pseudokodi për metodën Union

|  |
| --- |
| **int** xroot = find(subsets, x)  **int** yroot = find(subsets, y)    **if** subsets[xroot].rank              < subsets[yroot].rank              subsets[xroot].parent = yroot  **else** **if** subsets[xroot].rank                   > subsets[yroot].rank              subsets[yroot].parent = xroot  **else**              subsets[yroot].parent = xroot              subsets[xroot].rank++  end if |

Metoda Union() e bën unionin e dy bashkësive qe i fitojm nga metoda e mëhershme find().

|  |
| --- |
|  |

**Algoritmi 5**. Pseudokodi i Kruskal()

|  |
| --- |
| KruskalMST()            Edge result[] = **new** Edge[V]  **for** i = 0 to V               result[i] = **new** Edge()    end for          Arrays.sort(edge)          subset subsets[] = **new** subset[V]  **for** i = 0 to V               subsets[i] = **new** subset()    end for  **while** e < V  **if** (x != y)                  result[e++] = next\_edge                  Union subsets, x, y           end if |

Në këtë metodë do të ruhet numri i “minal spanning tree”, do të sortohen të gjitha brinjët sipas kostos që kanë, dhe përderisa numri brinjëve është më I vogel se numri i kulmeve do të selektohet një brinjë dhe i njejti process do të iterohet përsëri.

**Rezultatet**

Në vijim paraqesim rezultatet që kemi bërë nga testimi i programit

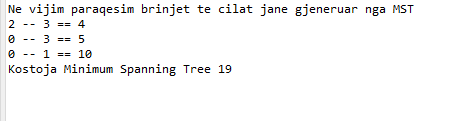


Fig .4. Rezultati nga testimi parë

Në figuren 4 kemi paraqitur rezultatet të cilat janë gjetur nga algoritmi minum spanning tree, ku I kemi caktuar një kosto të cfardoshme brinjëve.

Në këtë fotografi eshte llogaritur kostoja ku është e ndryshme nga testimi i parë.

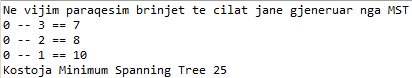


Fig.5. Rezultati nga testimi i dytë

**Diskutim Rezultateve**

Rezultatet që kemi marrë nga testimi i aplikacionit tone kemi vërejtur se ky algoritëm ka një performance të mire në shumicën e rasteve. Problemi që ekziston dhe që është i pazgjidhshëm është që nëse kemi një graf i cili është i ndarë në dy pjesë të cilat nuk lidhen me njëra tjetrën atëhere ky algoritëm nuk mund të zgjidhe këtë problem.

**Përfundim**

Në punimin tone kemi trajtuar problemin e gjetjes së rrugës më të shkurtër të një garfi me n-kulme, nga i cili mund të konstruktohen (n-1)-brinjë që kanë kosto. Algoritmi që ne kemi përdorur fillimisht gjen brinjën që ka koston më të vogël dhe pastaj fillon gjetjen e brinjëve të tjera që kanë koston më të vogël të mundshme.

# References

[1]. *geeksforgeeks.* [Online]   
Available at: https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

[2]. *www.javatpoint.com.* [Online]   
Available at: https://www.javatpoint.com/kruskal- h.  
[Accessed 28 May 2022].

[3]. *www.simplilearn.com.* [Online]   
Available at: https://www.simplilearn.com/tutorials/data-structure-tutorial/kruskal-algorithm#:~:text=Introduction%20to%20Kruskal%20Algorithm,-

[4]*.* [Online]   
Available at: https://en.wikipedia.org/wiki/Kruskal's\_algorithm  
[Accessed 28 May 2022].