KRUSKAL ALGORITMI - MINIMAL ORALIQ DARAXTI TOPISH

Onarkulov Maksadjon Karimberdiyevich
Farg'ona davlat univarsiteti Amaliy matematika va informatika kafedrasi dotsenti (PhD)
maxmaqsad@gmail.com
Nurmatova Hushnozabonu To'ychiboy qizi
Farg'ona davlat universiteti 2-bosqich talabasi
nurmatovaxushnozabonu@gmail.com

Anatatsiya:

Kruskal algoritmi minimal oraliq daraxt (Minimum Spanning Tree - MST) topish uchun mo'ljallangan eng samarali va keng qo'llaniladigan ochko'z (greedy) algoritmlardan biridir. U 1956-yilda Joseph Kruskal tomonidan ishlab chiqilgan bo'lib, bog'langan og'irlikli grafda barcha tugunlarni bog'laydigan eng kichik umumiy og'irlikka ega daraxtni topadi. Algoritm qirralarni og'irligi bo'yicha saralab, har bir qaddamda eng kichik og'irlikli qirrani tanlab oladi, lekin tsikl hosil qilmaydigan qirralarni qo'shadi. Union-Find (Disjoint Set Union) ma'lumotlar strukturasi yordamida tsikllarni aniqlash va samarali ishlash ta'minlanadi. Ushbu maqolada Kruskal algoritmining ishlash mexanizmi, asosiy bosqichlari, murakkablik tahlili va amaliy qo'llanish sohalari batafsil yoritiladi.

Kalit so'zlar: *Kruskal algoritmi, minimal oraliq daraxt, MST, ochko'z algoritm, Union-Find, DSU, graf nazariyasi, qirra og'irligi, tsikl, bog'langan graf, saralash algoritmi, optimallashtirish, tarmoq dizayni, algoritm tahlili, ma'lumotlar strukturasi*

Annotation:

Kruskal's algorithm is one of the most efficient and widely used greedy algorithms for finding the Minimum Spanning Tree (MST). Developed by Joseph Kruskal in 1956, it finds a tree that connects all vertices in a connected weighted graph with the minimum total weight. The algorithm works by sorting edges by weight and selecting the smallest weight edge at each step, adding only edges that

do not create cycles. The Union-Find (Disjoint Set Union) data structure is used for efficient cycle detection. This paper provides a detailed examination of Kruskal's algorithm mechanism, main stages, complexity analysis, and practical applications.

Keywords: *Kruskal algorithm, minimum spanning tree, MST, greedy algorithm, Union-Find, DSU, graph theory, edge weight, cycle, connected graph, sorting algorithm, optimization, network design, algorithm analysis, data structures*

Аннотация:

Алгоритм Краскала является одним из наиболее эффективных и широко используемых жадных алгоритмов для поиска минимального остовного дерева (MST). Разработанный Джозефом Краскалом в 1956 году, он находит дерево, соединяющее все вершины связного взвешенного графа с минимальным общим весом. Алгоритм работает путём сортировки рёбер по весу и выбора ребра с наименьшим весом на каждом шаге, добавляя только те рёбра, которые не создают циклы. Структура данных Union-Find (система непересекающихся множеств) используется для эффективного обнаружения циклов. В данной статье подробно рассматриваются механизм работы алгоритма Краскала, основные этапы, анализ сложности и практические применения.

Ключевые слова: *алгоритм Краскала, минимальное остовное дерево, MST, жадный алгоритм, Union-Find, СНМ, теория графов, вес ребра, цикл, связный граф, алгоритм сортировки, оптимизация, проектирование сетей, анализ алгоритма, структуры данных*

Graf nazariyasining rivojlanishi bilan birga, turli amaliy masalalarni hal qilish uchun samarali algoritmlar yaratish zarurati ortib bormoqda. Minimal oraliq daraxt (Minimum Spanning Tree - MST) topish masalasi graf algoritmlarining eng muhim va keng qo'llaniladigan masalalaridan biri hisoblanadi. Bu masala tarmoq loyihalash, telekommunikatsiya tizimlari, transport infratuzilmasi, elektr

tarmoqlari va ko'plab boshqa sohalarda qo'llaniladi. MST masalasi bog'langan og'irlikli grafda barcha tugunlarni bog'laydigan eng kichik umumiy og'irlikka ega daraxtni topishni talab qiladi. Bunday daraxt grafning barcha tugunlarini bog'laydi, lekin hech qanday tsikl mavjud emas va umumiy og'irligi minimal bo'ladi.

Kruskal algoritmi 1956-yilda amerikalik matematik Joseph Kruskal tomonidan ishlab chiqilgan bo'lib, MST masalasini hal qilishda ochko'z (greedy) yondashuvni qo'llaydi. Bu algoritm har bir qaddamda mahalliy jihatdan eng yaxshi tanlovni amalga oshiradi va oxir-oqibatda global optimumga erishadi. Kruskal algoritmining asosiy g'oyasi qirralarni og'irligi bo'yicha o'sish tartibida saralab, har bir qaddamda eng kichik og'irlikli qirrani tanlab olishdan iborat. Biroq, algoritm faqat tsikl hosil qilmaydigan qirralarni qo'shadi, bu esa Union-Find ma'lumotlar strukturasi yordamida samarali amalga oshiriladi.

Algoritmning ushbu yondashuvi uni boshqa MST algoritmlaridan, masalan, Prim algoritmidan farqlaydi. Prim algoritmi bitta tugunden boshlayib, daraxtni bosqichma-bosqich kengaytirib borsa, Kruskal algoritmi qirralar bilan ishlaydi va global nuqtai nazardan eng optimal tanlovlarni amalga oshiradi. Bu xususiyat Kruskal algoritmini ayrim hollarda Prim algoritmiga nisbatan afzal qiladi, ayniqsa siyrak graflarda yuqori samaradorlik ko'rsatadi. Algoritm O(E log E) vaqt murakkabligiga ega bo'lib, bu asosan qirralarni saralash bosqichi bilan bog'liq.

Hozirgi kunda Kruskal algoritmi nafaqat nazariy jihatdan, balki amaliy dasturlashda ham keng qo'llaniladi. U kompyuter tarmoqlari loyihalashda, yo'l-transport infratuzilmasini rejalashtirish, elektr uzatish liniyalarini optimallashtirish, klasterlash algoritmlarida va ko'plab boshqa sohalarda foydalaniladi. Algoritmning soddaligi va samaradorligi uni graf algoritmlari o'rganayotgan talabalar uchun ham, amaliy loyihalarda ishlovchi dasturchilar uchun ham muhim qiladi. Ushbu maqolada Kruskal algoritmining chuqur tahlili, uning ishlash mexanizmi, afzalliklari va real hayotdagi qo'llanishlari batafsil ko'rib chiqiladi.

Kruskal algorimining ishlash prinsipi uning ochko'z (greedy) strategiyasiga asoslanadi. Algoritm ishlashini boshlashdan oldin, barcha qirralar og'irligi bo'yicha o'sish tartibida saralanadi. Boshlang'ich holatda har bir tugun alohida komponent sifatida qaraladi, ya'ni V ta tugun uchun V ta alohida to'plam mavjud bo'ladi. Algoritm asosiy tsiklda saralangan qirralar ro'yxatini ketma-ket ko'rib chiqadi va har bir qirra uchun quyidagi amallarni bajaradi.

Har bir qirra (u, v) uchun algoritm birinchi navbatda u va v tugunlari bir xil komponentda joylashganligini tekshiradi. Agar ular turli komponentlarda bo'lsa, bu qirrani qo'shish tsikl hosil qilmaydi va qirra minimal oraliq daraxtga qo'shiladi. So'ngra ushbu ikki komponent birlashtiriladi. Agar u va v tugunlari allaqachon bir xil komponentda bo'lsa, bu qirrani qo'shish tsikl hosil qiladi va u rad etiladi. Bu jarayon barcha qirralar ko'rib chiqilgunga yoki minimal oraliq daraxt to'liq qurilgunga qadar davom etadi.

Tugunlarning qaysi komponentga tegishli ekanligini aniqlash va komponentlarni samarali birlashtirish uchun Union-Find (Disjoint Set Union - DSU) ma'lumotlar strukturasi ishlatiladi. DSU ikki asosiy operatsiyani qo'llab-quvvatlaydi: Find - berilgan tugun qaysi to'plamga tegishliligini aniqlash va Union - ikkita to'plamni birlashtirishadi. Path compression va union by rank kabi optimizatsiyalar yordamida bu operatsiyalar deyarli O(1) vaqtda bajariladi.

Algoritm tugashi bilan V-1 ta qirra tanlanadi (V - tugunlar soni), chunki har qanday daraxtda tugunlar sonidan bitta kam qirra bo'ladi. Natijada hosil bo'lgan qirralar to'plami minimal oraliq daraxtni tashkil etadi. Kruskal algoritmining matematik isboti ochko'z tanlov xususiyati (greedy choice property) va optimal substructure prinsiplariga asoslanadi. Har bir qaddamda eng kichik og'irlikli qirrani tanlash global optimumga olib keladi, chunki MST masalasi matroid strukturasiga ega va ochko'z algoritm uchun mos keladi.

Algoritmning to'g'riligi cut xususiyati orqali ham isbotlanishi mumkin. Grafning har qanday cut'i uchun, ushbu cut'ni kesib o'tadigan eng kichik og'irlikli qirra albatta minimal oraliq daraxtning bir qismi bo'ladi. Kruskal algoritmi aynan shu printsipni amalga oshiradi va har doim to'g'ri natija beradi.

Kruskal algoritmi zamonaviy graf algoritmlarining eng foydali namunalaridan biri sifatida bir qator muhim afzalliklarga ega. Birinchi va asosiy afzalligidan biri uning **soddaligidir**. Algoritmni tushunish va amalga oshirish nisbatan oson bo'lib, asosiy g'oya - qirralarni saralab, eng kichigidan boshlab tsikl hosil qilmaydigan qirralarni tanlash - intuituiv va tushunarli. Bu xususiyat uni o'quv jarayonida va amaliy loyihalarda keng qo'llanishiga sabab bo'ladi.

Ikkinchi muhim afzalligi - **siyrak graflarda yuqori samaradorligi**dir. Kruskal algoritmi O(E log E) vaqt murakkabligiga ega bo'lib, bu qirralar soni kam bo'lgan graflarda juda samarali. Masalan, qirralar soni O(V) tartibida bo'lgan graflarda algoritm O(V log V) vaqtda ishlaydi, bu esa Prim algoritmining $O(V^2)$ murakkabligidan yaxshiroq. Uchinchidan, algoritm **parallellashtirish imkoniyatiga** ega. Qirralarni saralash bosqichi parallel algoritmlarda amalga oshirilishi mumkin, shuningdek, Union-Find operatsiyalari ham ma'lum darajada parallellashtirish mumkin.

Algoritm shuningdek **xotira samaradorligi** jihatidan ham afzallikka ega. U faqat qirralar ro'yxati va Union-Find strukturasi uchun xotira talab qiladi, qo'shimcha murakkab ma'lumot strukturalari kerak emas. **Deterministik natija** berishi ham muhim afzallik - bir xil kiritish ma'lumotlari uchun doimo bir xil MST hosil qiladi (qirralar og'irliklari turlicha bo'lsa). Shuningdek, algoritm **dinamik graflarda** ham qo'llanishi mumkin - yangi qirralar qo'shilganda yoki mavjud qirralar o'zgartirilganda, MST'ni yangilash nisbatan oson.

Biroq Kruskal algoritmining ba'zi **kamchiliklari** ham mavjud. Asosiy kamchilik - **qirralarni saralash zarurati**. Barcha qirralarni avvaldan saralash kerak bo'lib, bu O(E log E) vaqt talab qiladi va katta graflarda sezilarli vaqt sarfini keltirib chiqaradi. Ikkinchidan, **zich graflarda** (qirralar soni V^2 ga yaqin bo'lganda) Prim algoritmi Kruskal algoritmidan samaraliroq ishlashi mumkin.

Uchinchidan, **Union-Find strukturasining murakkabligi** algoritm tushunishni qiyinlashtirishi mumkin, ayniqsa optimizatsiyalar (path compression, union by rank) qo'llanilganda. **Real vaqtda ishlash** uchun mos emas, chunki barcha qirralar ma'lum bo'lishi kerak. **Katta hajmdagi graflarda xotira**

muammosi ham yuzaga kelishi mumkin, chunki barcha qirralarni xotirada saqlash kerak. Shuningdek, algoritm **og'irligi teng qirralar** mavjud bo'lganda turli MST'lar berishi mumkin, bu ba'zi hollarda muammo tug'diradi.

Umuman olganda, Kruskal algoritmi o'zining soddaligidir, siyrak graflardagi samaradorligi va keng qo'llanish imkoniyatlari bilan MST topish masalasida eng mashhur tanlovlardan biri bo'lib qolmoqda.

Kruskal algoritmi zamonaviy texnologiya va muhandislikda juda keng qo'llaniladi. Uning eng muhim amaliy qo'llanish sohalaridan biri **tarmoq infratuzilmasi loyihalash**dir. Telekommunikatsiya kompaniyalari shaharlar orasida optik kabel liniyalarini o'tkazishda, eng kam xarajat bilan barcha shaharlarni bog'laydigan tarmoqni qurish uchun Kruskal algoritmidan foydalanadilar. Bu yerda shaharlar graf tugunlari, ular orasidagi masofalar yoki qurilish xarajatlari esa qirra og'irliklari sifatida qaraladi.

Elektr energetika tizimlarida ham algoritm muhim rol o'ynaydi. Elektr stantsiyalarini shahar va qishloqlar bilan bog'laydigan uzatish liniyalarini loyihalashda, eng kam xarajat bilan barcha iste'molchilarni elektr bilan ta'minlaydigan tarmoqni yaratish uchun MST algoritmlari, xususan Kruskal algoritmi qo'llaniladi. **Transport va logistika**da yo'l qurilishi, temir yo'l liniyalari rejalashtirish va yuk tashish marshrutlarini optimallashtirish uchun ham ushbu algoritm ishlatiladi.

Kompyuter tarmoqlari va internet infratuzilmasida Kruskal algoritmi marshrut protokollarida, Local Area Network (LAN) dizaynida va tarmoq topologiyasini optimallashtirish uchun qo'llaniladi. **Klasterlash algoritmlari**da ma'lumotlarni guruhlash, o'xshash obyektlarni birlashtirish va tasniflash masalalarida ham MST yondashuvi, jumladan Kruskal algoritmi ishlatiladi. **Bioinformatika**da genetik ma'lumotlarni tahlil qilish, filogenetik daraxtlar qurish va protein strukturalarini o'rganishda ham algoritm qo'llaniladi.

Rasm va grafika ishlov berishda Kruskal algoritmi segmentatsiya, kontur aniqlash va rasm sifatini yaxshilash uchun ishlatiladi. Piksellar orasidagi o'xshashlik darajasini qirra og'irligi sifatida qabul qilib, MST orqali rasm

qismlarini ajratish mumkin. **Robotika va avtomatlashtirish**da yo'l rejalashtirish, sensorlar tarmoqini dizayn qilish va robot harakatini optimallashtirish uchun ham qo'llaniladi.

Dasturlash tillarida Kruskal algoritmi turli kutubxonalar orqali amalga oshirilgan. Python'da NetworkX, Java'da JGraphT, C++'da Boost Graph Library kabi kutubxonalarda algoritm mavjud. **Ma'lumotlar bazasi optimallashtirish**da indekslar yaratish va so'rovlarni tezlashtirish uchun ham MST yondashuvi ishlatiladi. **IoT (Internet of Things) tizimlarda** sensorlar tarmoqini energiya samarali qilib loyihalash, ma'lumot uzatish yo'llarini optimallashtirish va tarmoq yashash muddatini uzaytirish uchun Kruskal algoritmi qo'llaniladi.

Amaliy amalga oshirishda algoritm ko'pincha optimizatsiyalar bilan birgalikda ishlatiladi: saralash algoritmini yaxshilash, Union-Find strukturasini optimallashtirish, parallel hisoblash texnikalari va xotira samaradorligini oshirish usullari.

Kruskal algoritmi hozirgi zamonaviy dasturlash va graf nazariyasida muhim o'rin egallab, minimal oraliq daraxt topish masalasini hal qilishda samarali va ishonchli yechim taqdim etadi. Uning ochko'z yondashuvi - qirralarni og'irligi bo'yicha saralab, eng kichik og'irlikli va tsikl hosil qilmaydigan qirralarni ketma-ket tanlash - matematik jihatdan to'g'ri va amaliy jihatdan samarali ekanligini isbotlaydi. Union-Find ma'lumotlar strukturasi bilan kombinatsiyasi algoritmni O(E log E) murakkablik bilan amalga oshirish imkonini beradi, bu ko'plab real vaziyatlarda optimal hisoblanadi.

Algoritmning soddaligi uni o'rganish va amalga oshirish uchun ideal tanlovga aylantiradi, shu bilan birga uning keng qo'llanish sohalari - tarmoq loyihalash, telekommunikatsiya, transport, energetika va ko'plab boshqa sohalardagi amaliy qiymati uni zamonaviy muhandislikning ajralmas qismiga aylantiradi. Kruskal algoritmi ochko'z algoritmlarga yorqin misol bo'lib, mahalliy optimal tanlovlar orqali global optimumga erishish mumkinligini ko'rsatadi.

Kelajakda Kruskal algoritmini yanada takomillashtirish yo'nalishlari mavjud. Parallel va taqsimlangan hisoblash texnologiyalaridan kengroq

foydalanish, kvant algoritmlar bilan integratsiya, sun'iy intellekt yondashuvlari bilan kombinatsiyalash va big data muhitida samaradorlikni oshirish istiqbolli tadqiqot yo'nalishlari hisoblanadi. Shuningdek, IoT, 5G tarmoqlari, blokchejn va edge computing kabi yangi texnologiyalar Kruskal algoritmi uchun yangi qo'llanish imkoniyatlarini ochmoqda.

Tavsiya sifatida, dasturlash va graf algoritmlari bilan shug'ullanuvchi mutaxassislar Kruskal algoritmini nafaqat nazariy jihatdan, balki turli amaliy stsenariyalarda ham chuqur o'rganishlari zarur. Algoritmni turli dasturlash tillarida amalga oshirish, Union-Find strukturasining optimizatsiya usullarini o'rganish va real ma'lumotlar ustida algoritm samaradorligini tahlil qilish muhim amaliy ko'nikmalarni rivojlantiradi. Shuningdek, algoritmni boshqa MST algoritmlari (Prim, Borůvka) bilan solishtirib o'rganish, ularning kuchli va zaif tomonlarini aniqlash ham foydali bo'ladi.

Xulosa qilib aytganda, Kruskal algoritmi nafaqat samarali graf algoritmi, balki algoritmik fikrlashning mukammal namunasi sifatida ham qadrlanadi. Uning o'rganilishi dasturchilar va tadqiqotchilarga ochko'z algoritmlar prinsiplarini tushunish, murakkab masalalarni sodda qismlarga bo'lish va matematik nazariyani amaliy masalalarda qo'llash usullarini o'rgatadi. Bu algoritm graf nazariyasi va optimallashtirish masalalarini o'rganishda mustahkam poydevor bo'lib xizmat qiladi.

Foydalanilgan adabiyotlar

1. Kruskal, Joseph B. "On the shortest spanning subtree of a graph and the traveling salesman problem". Proceedings of the American Mathematical Society, 1956.

2. Thomas H. Cormen et al. *Introduction to Algorithms*. MIT Press, 4th Edition.

3. Robert Sedgewick, Kevin Wayne. *Algorithms*. Addison-Wesley, 4th Edition.

4. Kleinberg, Jon; Tardos, Éva. *Algorithm Design*. Addison-Wesley.

5. Tarjan, Robert E. "Efficiency of a Good But Not Linear Set Union Algorithm". Journal of the ACM, 1975.

6. West, Douglas B. *Introduction to Graph Theory*. Prentice Hall, 2nd Edition.

7. Bang-Jensen, Jørgen; Gutin, Gregory. *Digraphs: Theory, Algorithms and Applications*. Springer.

8. Ahuja, Ravindra K., et al. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

9. GeeksforGeeks. *Kruskal's Minimum Spanning Tree Algorithm*. https://www.geeksforgeeks.org

10. Wikipedia. *Kruskal's Algorithm*. https://en.wikipedia.org/wiki/Kruskal%27s_algorithm

11. LeetCode Discussions. *Minimum Spanning Tree Algorithms Comparison*.

12. Stack Overflow. *Union-Find Data Structure Implementation*.

13. NetworkX Documentation. *Minimum Spanning Tree in Python*.

14. Boost Graph Library Documentation. *Kruskal Minimum Spanning Tree*.

15. Algorithm Visualizer. *Kruskal's Algorithm Step-by-Step Visualization*.

Ilm fan taraqqiyotida raqamli iqtisodiyot va zamonaviy ta'limning o'rni hamda rivojlanish omillari