ГИМНАЗИЈА У КРАЉЕВУ

МАТУРСКИ РАД ИЗ РАЧУНАРСТВА И ИНФОРМАТИКЕ

МИНИМАЛНА СТАБЛА РАЗАПИЊАЊА У НЕУСМЕРЕНОМ ГРАФУ (MST)

Ментор:
Игор Кнежевић, проф.

Ученик:
Славковић Виктор, IV_7

Краљево
мај 2014. године

Садржај

Ова страна је намерно празна.

Увод

Историјски увод

Чешки научник **О. Борувка** је 1926. развио први познати алгоритам за налажење минималног стабла разапињања у графу. Тиме је намеравао да реши проблем проналажења најефикаснијег начина за снабдевање чешке области Моравске електричном енергијом. Поред Борувкиног алгоритма, најчешће коришћени алгоритми су Примов и Крускалов. Примов алгоритам је познат и као Јарник-Прим-Дијкстрин алгоритам јер су до њега независно дошла три научника и то су: **В. Јарник** (1930), **Р.К. Прим** (1957) и **Е.В. Дијкстра** (1959). Крускалов алгоритам носи назив по **Џ.Б. Крускалу** и објављен је 1956. године. Развојем структура података које ови алгоритми користе, развијали су се и они сами, а најпознатије су варијације Примовог алгоритма које користе *d*-арни хип (**Д. Џонсон**, 1975) и Фибоначијев хип (**М. Фредман** и **Р. Тарџан**, 1984).

У последњој четвртини прошлог века, као и почетком овог, објављени су алгоритми исте намене чија сложеност тежи линеарној што представља велики напредак у односу на претходне алгоритме линеаритмичке сложености. Имплементације ових алгоритама су јако комплексне због комплексности самих алгоритама, као и због комплексности помоћних структура података које користе, па због тога неће бити предмет овог матурског рада. Аутори тих алгоритама су: **Х.Н. Габов** и коаутори (1987), **Б. Чазел** (1997. и 1999) и **С. Петит** и **В. Рамачардан** (2001).

Такође, **Бадер** и **Конг** су 2003. године објавили алгоритам исте намене који је дизајниран као паралелни алгоритам и са линеарним бројем процесора (нити) проблем решава у логаритамској сложености.

О. Борувка (Otakar Borůvka) (1899 - 1995) је био чешки математичар. Упамћен је по свом доприносу теорији графова много пре њеног оснивања као математичке дисциплине.

Р.К. Прим (Robert Clay Prim) (1921) је амерички математичар и информатичар. Најпознатији је по свом раду у Беловим лабораторијама где је дао знатан допринос теорији графова и развоју рачунарских мрежа.

В. Јарник (Vojtěch Jarník) (1897 - 1970) је био чешки математичар. Упамћен је по свом доприносу областима математичке анализе, теорије бројева и торије графова.

Е.В. Дијкстра (Edsger Wybe Dijkstra) (1930 - 2002) је био Холандски информатичар. Упамћен је као један од највећих алгоритмичара свих времена. Добитник је Турнингове награде, а данас постоји награда за допринос у области дистрибуисаних система која носи његово име.

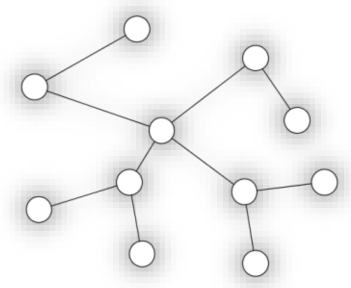
Џ.Б. Крускал (Joseph Bernard Kruskal, Jr.) (1928 - 2010) је био амерички математичар, статистичар и информатичар. У областима математике и информатике је познат по доприносу теорији графова и развоју поменутог алгоритма.

Р.Е. Тарџан (Robert Endre Tarjan) (1948) је амерички информатичар и велики алгоритмичар. Најпознатији је по Фибоначијевом хипу (енг. Fibonacci Heap) и раширеном бинарном стаблу претраге (енг. Spaly Binary Search Tree).

Теоријски увод

Напомена: У овом раду се термин „граф“ изједначава са термином „неусмерени граф“ због природе саме теме. Тема аналогна овој у области усмерених графова је комплекснија и неће бити разматрана у овом раду.

Дефиниција: Стабло је ациклични и повезани граф.



Теорема: Следећа тврђења су еквивалентна:

(а) G је стабло.

(б) G је ациклични граф, а додавање било какве гране би створило циклус.

(в) Чворови графа G немају сопствених цилуса и између свака два чвора постоји јединствена проста веза.

(г) G је повезан граф, али уклањање било које гране нарушава повезаност.

Доказ:

Овом исказу је еквивалентан исказ:

$$(а) \Longrightarrow (б) \Longrightarrow (в) \Longrightarrow (г) \Longrightarrow (а)$$

па ће доказ бити раздвојен на доказе појединачних импликација.

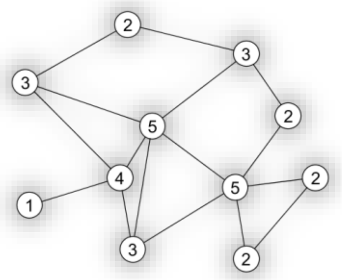
(а) $\Longrightarrow$ (б): Познато је да је G ацикличан и повезан граф и претпоставимо да се њему додаје нова грана e која повезује чворове A и B. Ако је $A = B$, онда e ствара прост циклус на том чвору, а ако је $A \neq B$, како је G већ повезан граф, е ствара прост циклус.

(б) $\Longrightarrow$ (в): Познато је да је G ацикличан граф и да додавање било које гране ствара циклус. Очигледно, чворови графа G немају сопствених циклуса. Нека су A и B било која два чвора графа G. Ако не постоји веза између њих, додавање гране која их повезује не ствара циклус, а то је контрадикторно полазној претпоставци. Значи, G је повезан граф и још преостаје доказ јединствености просте везе између чворова A и B. Претпоставимо, супротно тврђењу, да има више од једне просте везе између њих. Нека су две такве везе P і P'(различите). Како ове везе имају исти почетак и исти крај, или ће се разићи на неком чвору, чиме се добија циклус или ће бити идентичне, што је контрадикција са претпоставком.

(в) $\Longrightarrow$ (г): Из полазног тврђења је јасно да је G повезан граф. Претпоставимо да се уклања грана e која повезује чворове A и B. Како чворови графа G немају сопствених цилуса, важи $A \neq B$. Ако и након уклањања постоји проста веза између чворова A и B, онда постоје две просте путање које повезују чворове A и B, па је то контрадикција.

(г) $\Longrightarrow$ (а): Ако у графу G постоји прост циклус, уклањање гране циклуса неће нарушити повезаност, па је то контрадикторно полазном тврђењу. Значи, G је ацикличан граф.

Дефиниција: Степен чвора је број грана које излазе из њега (или улазе у њега - нема разлике у неусмереном графу).



Теорема: Нека је $G(V, E)$ коначан граф и $n = |V|$.Следећа тврђења су еквивалентна:

(а) G је стабло.
(б) G је ациклични граф и $|\mathrm{E}| = n - 1$.
(в) G је повезан граф и $|\mathrm{E}| = n - 1$.

Доказ:

За $n = 1$, доказ је тривијалан. Овом исказу је еквивалентан исказ:

$$(а) \Longrightarrow (б) \Longrightarrow (в) \Longrightarrow (а)$$

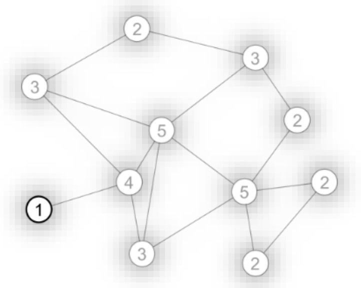
па ће доказ за $n \geq 2$ бити раздвојен на доказе појединачних импликација.

(а) $\Longrightarrow$ (б): По дефинцији стабла је G ацикличан граф, а може се показати индукцијом да у тим условима важи $|\mathrm{E}| = n - 1$. За базу индукције можемо узети тривијалне случајеве за $n = 1$ и $n = 2$. Под претпоставком да тврђење важи за свако $n < m$, показаћемо да важи и за m. Ако из дрвета са m грана избацимо било коју грану e , по еквиваленцији исказа (а) и (г) добијамо две дисјунктне компоненте повезаности од којих је свака ациклична, па је и стабло. По индуктивној претпоставци, свако од та тва стабла има број грана за један мањи од броја чворова, па је тај збир $m - 2$ и кад се врати e, то је $m - 1$.

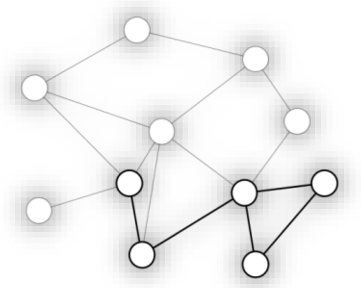
$(\text{б}) \Longrightarrow (\text{в})$: Докажимо прво да граф G има бар два чвора степена 1. Како је то ацикличан граф, $n \geq 2$ и $|\mathrm{E}| \geq 1$, овај доказ је тривијалан. Сада повезаност графа G доказујемо индукцијом. Поново, за базу индукције можемо узети тривијалне случајеве за $n = 1$ и $n = 2$. Под претпоставком да важи за n, доказујемо да важи за $n + 1$. Ако поменутом графу са n чворова додамо један чвор и једну грану тако да нова грана повезује нови чвор и један од чворова степена *1*, добијамо тражени повезани граф и потврђујемо коректност индукцијског корака и саме индукције.

$(\text{в}) \Longrightarrow (\text{а})$: Све док G има простих циклуса, можемо уклонити неку грану без нарушавања повезаности. На крају тог процеса се добија стабло које по $(\text{а}) \Longrightarrow (\text{б})$ има $n - 1$ грана, а то је број грана од кога смо по претпоставци кренули, па је G ацикличан граф, тј. G је стабло.

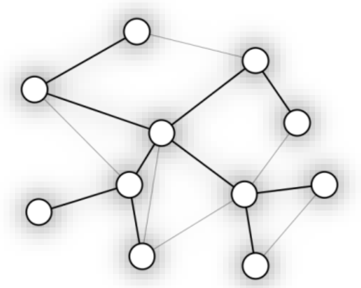
Дефиниција: Чвор чији је степен 1 се назива лист.



Дефиниција: Граф $G'(V', E')$ је подграф графа $G(V, E)$ ако $V' \subset V$ је и $\mathrm{E}' \subset E$.

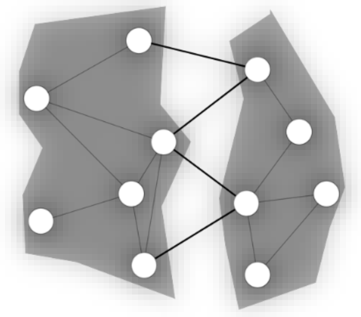


Дефиниција: Подграф $\mathrm{T}(V, E')$ графа $G(V, E)$ је његово стабло разапињања ако је T стабло.



Дефиниција: Минимало стабло разапињања (енг. MST – Minimum Spanning Tree или MCST - Minimum-Cost Spanning Tree) графа $G(V, E)$ је оно од његових стабала разапињања за које је сума тежина додељених његовим гранама минимална.

Дефиниција: Рез је било каква подела чворова графа у два дисјунктна и непреазна подскупа, а грана реза је свака грана која повезује два чвора који су у различитим скуповима реза.



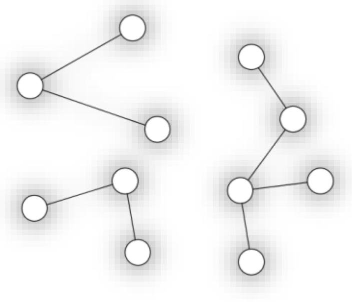
Теорема: Својство реза: Грана најмање тежине произвољног реза графа код кога су тежине додељене гранама међусобно различите припада минималном стаблу разапињања тог графа.

Доказ:

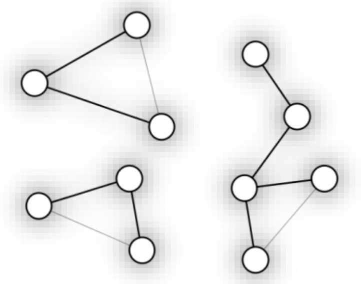
Нека је e грана најмање тежине произвољног реза, а T минимално стабло разапињања. Претопоставимо да, супротно тврђењу, T не садржи грану e. Додавањем гране e у стабло T добија се прост циклус који осим гране e садржи бар још једну грану реза (означавамо са f) која има већу тежину од e. Можемо добити повезујуће стабло мање тежине ако уклонимо грану f и додамо грану e, па долазимо до контрадикције.

Напомена: У претходној теореми, као и у остатку рада, у обзир ће се узимати графови код којих су тежине додељене гранама међусобно различите и сматраће се да граф има јединствено минимално стабло разапињања. Ово је важно јер поједностављује теоријске доказе, а у пракси нема разлике у примени алгоритама који следе тј. алгоритми који следе ће увек наћи једно од минималних стабала разапињања.

Дефиниција: Шума је граф код кога свака компонента повезаности представља стабло.



Дефиниција: Шума разапињања неповезаног графа је скуп појединачних стабала повезаности његових компонената повезаности (по једно стабло разапињања из сваке компоненте повезаности).



Дефиниција: Минимална шума разапињања неповезаног графа је она шума разапињања чије је свако стабло минимално стабло разапињања компоненате повезаности којој припада.

Напомена: У наставку ће се у обзир узимати само повезани графови јер у супротном не постоји минимално стабло разапињања већ минимална шума разапињања. Сврха овога је такође да поједностави теоријске доказе, а у пракси нема разлике у примени алгоритама који следе тј. алгоритми који следе ће у случају неповезаног графа наћи минималну шуму разапињања (јенду од њих).

Алгоритми

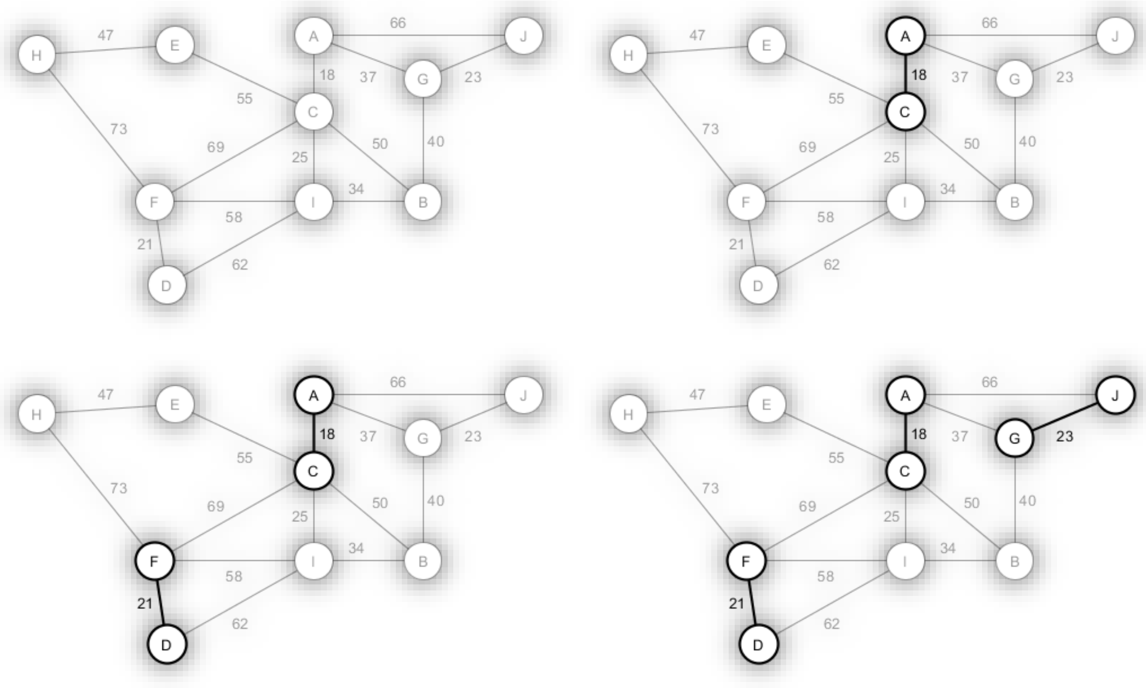
Крускал

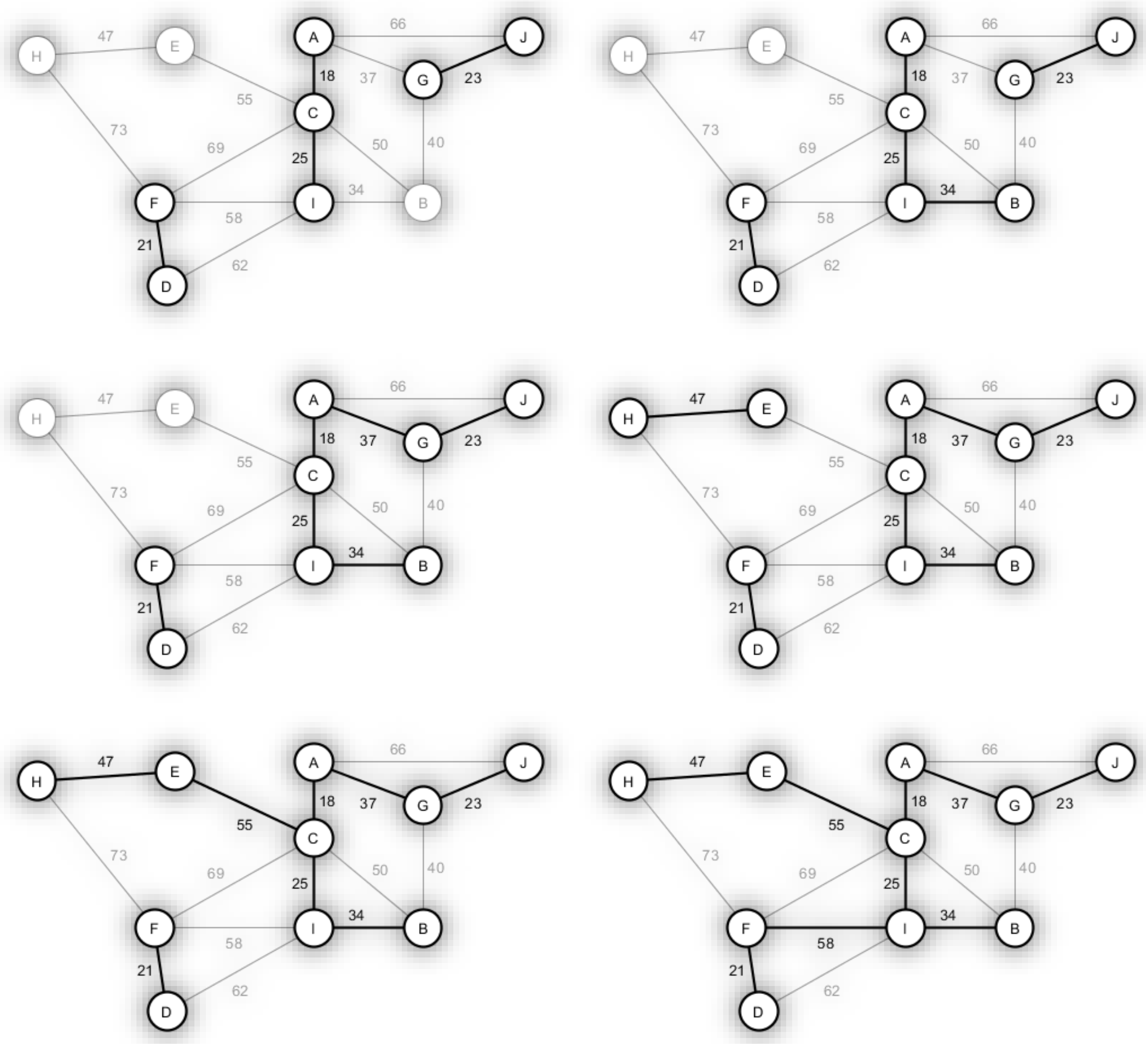
Крускалов алгоритам подразумева сортирање свих грана графа по тежини и њихов пролазак (од гране најмање тежине ка грани највеће тежине) у ком се тренутна грана додаје у тражено стабло уколико њено додавање не формира циклус са претходно додатим гранама. Одабране гране формирају шуму чија се стабла даљим додавањем грана спајају у једно тражено стабло. Одабир грана се прекида када број убачених грана буде за 1 мањи од броја чворова. Ово је пример грамзивог алгоритма.

Доказ ефикасности:

> Ако разматрана грана не формира циклус са претходно одабраним гранама, онда се она може посматрати као грана реза у коме чворови повезани одабраним гранма са различитих страна разматране гране припадају различитим скуповима реза. Како разамтрана грана не образује поменути циклус, то је прва грана тог реза која се разматра, а како их разматрамо у растућем поретку, она је минимална грана тог реза. Према теореми о својству реза, ова грана припада минималном стаблу разапињања.

Пример поступне примене алгоритма:





Приликом имплементације Крускаловог алгоритма за континуални одабир гране најмање тежине се као елегантно решење користе редови за приоритетом (енг. **Priority Queues**) или се гране могу стандардно сортирати. Што се тиче дела који проверава да ли размтрана грана формира циклус са претходно додатим гранама, проблем се своди на проверу повезаности два чвора, а то се најбрже реализује употребом структуре дисјунктних скупова (**Disjoint-Set Union-Find**) и то уз **WQUPC** (**Weighted Quick-Union with Path Compression**) имплементацију.

Сложеност сортирања грана је у сваком поменутом случају линеаритмичка, док је сложеност операција уније и претраге WQUPC дисјунктних скупова амортизовано константна (сложеност је реда итеративног логаритма), па је целокупна сложеност алгоритма:

$$O(\mathrm{E}\log\mathrm{E}) + O(\log* E) = O(\,\mathrm{E}\log\mathrm{E}^{\uparrow})$$

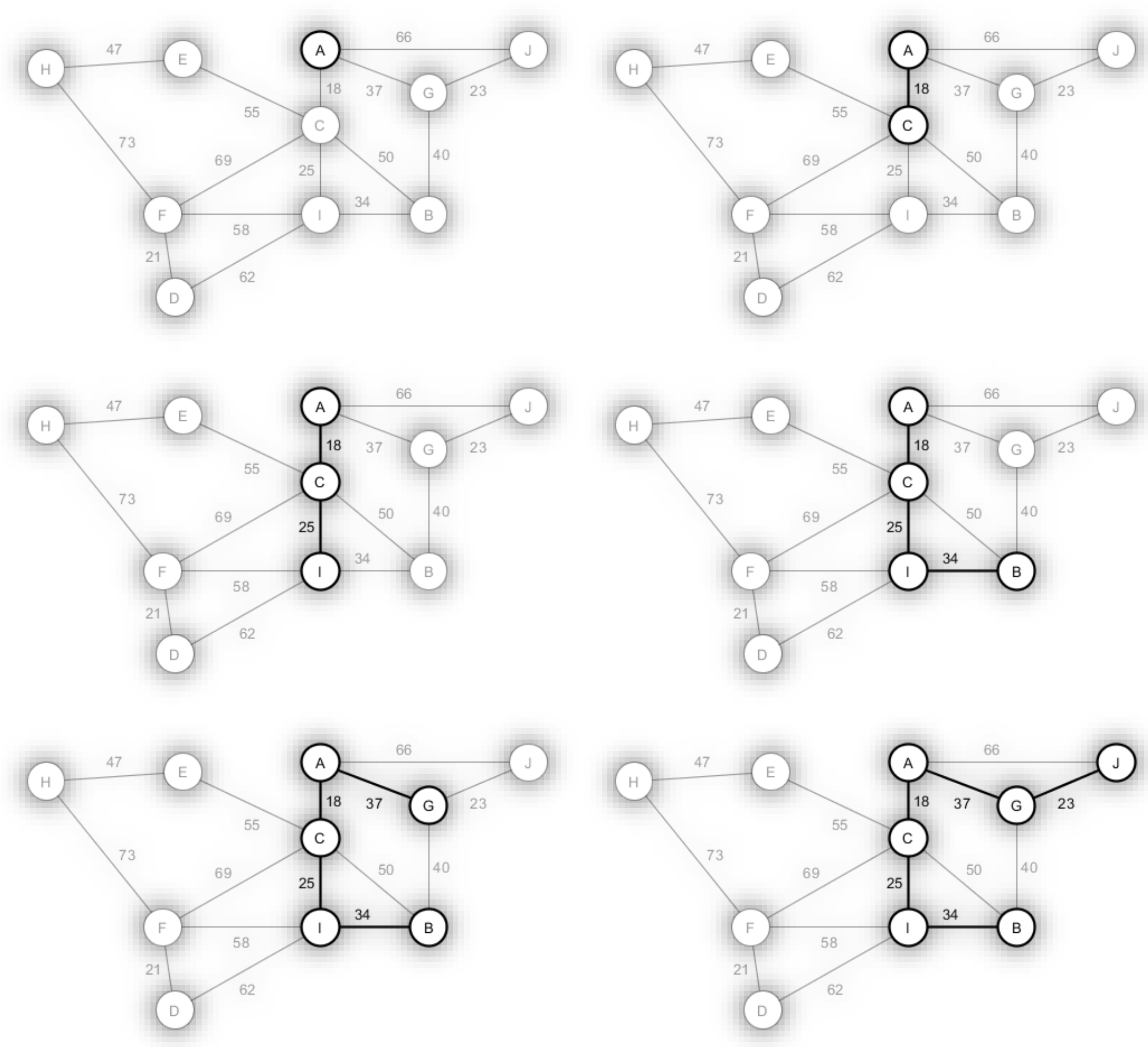
Прим

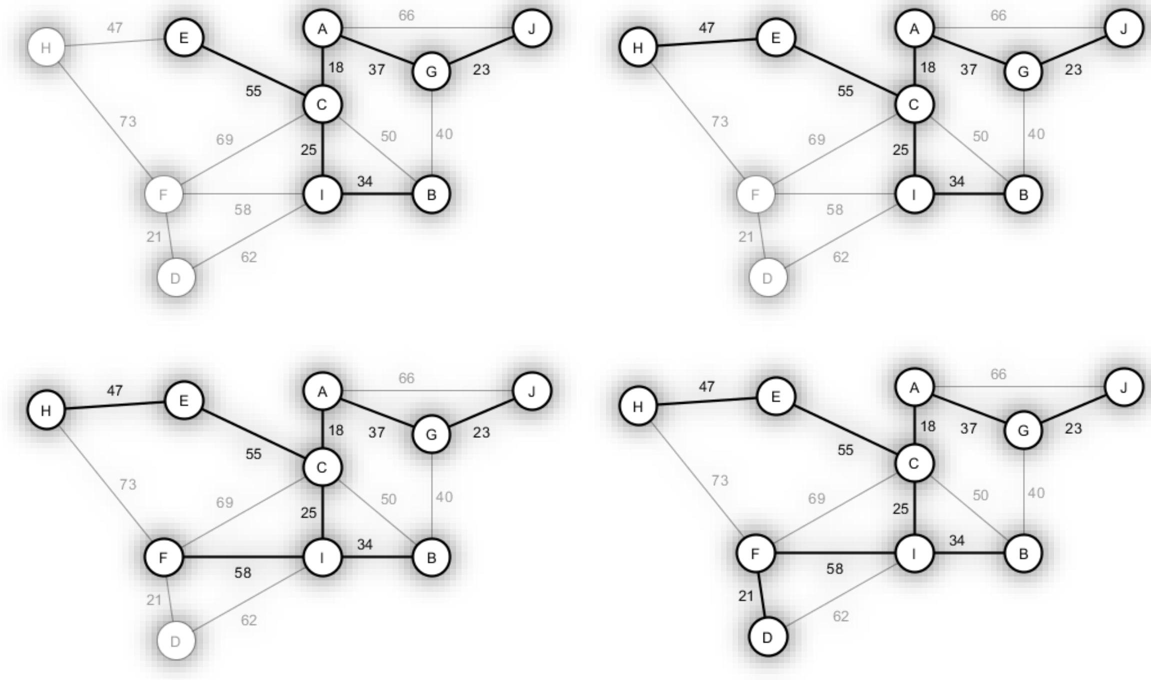
Примов алгоритам подразумева изградњу минималног стабла разапињања поласком од једночворног стабла (произвољни чвор) и додавањем грана најмање тежине које повезују растуће стабло са неким чвором ван њега. Додавање се прекида када сви чворови буду повезани у тражено стабло.

Доказ ефикасности:

> У сваком тренутку графа можемо направити рез такав да један скуп реза чине чворови који су тренутно у стаблу, а други сви опстали. Грана која се додаје је грана реза са минималном тежином, па је то директна последица теореме о својству реза.

Пример поступне примене алгоритма:





Размотрићемо више имплементација овог алгоритма. Прва имплементација користи ред са приоритетом имплементиран бинарним хипом (енг. **Binary Heap**) у који се додају гране које повезују последњи убачен чвор са чворовима ван стабла. Овај ред даје тражену грану у свакој итерацији, међутим, мана је што се приликом додавања нове гране која повезује исти чвор као и нека претходна грана са стаблом долази до нагомилавања непотребних грана у реду, а то успорава цео процес. Сложеност ове имплементације је:

$$O(\mathrm{E} \log \mathrm{E})$$

Друга имплементација овог алгоритма користи ред са приоритетом (имплементиран бинарним хипом) који има операцијоу смањивања приоритета појединачних чланова (енг. **Decrease Key Priority Queue**). У сваком тренутку се у реду налазе сви чворови графа, а приоритет је минимална тежина гране која повезује чвор са растућим стаблом. Уколико таква грана још није пронађена или је чвор већ укључен у стабло, његов приоритет је бесконачан тј. ставља се на крај реда. Како се чворови додају у стаблуо и обилазе нове гране, тако се самњује приоритет чворова до којих воде гране које обилазимо и овиме се постиже да се у сваком тренутку у реду налази минимална веза сваког чвора са стаблом. Сложеност ове имплементације је:

$$O(\mathrm{E} \log V)$$

Трећа имплементација је варијација претходне позната као Џонсонов алгоритам. Разлика је у имплементацији реда са приоритетом где се уместо бинарног хипа користи Џонсонов d-арни хип (енг. **d-ary Heap**). Сложеност оваквог алгоритма је:

$$O(\mathrm{E} \log_d V)$$

Четврта имплементација је поново варијација исте природе позната као Фредман - Тарџанов алгоритам. Овде се за имплементацију реда са приоритетом користи њихов Фибоначијев хип (енг. **Fibonacci Heap**). Сложеност овог алгоритма је:

$$O(\mathrm{E} + V \log V)$$

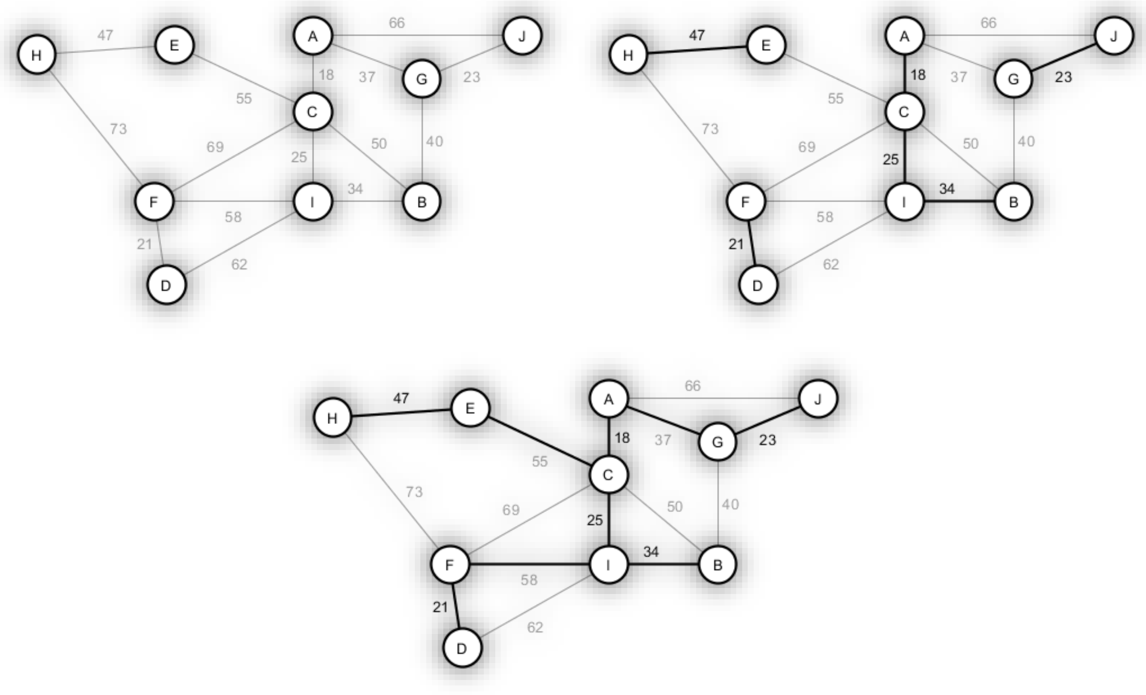
Борувка

Борувкин алгоритам се може назвати мешавином Примовог и Крускаловог алгоритма. Полази се од $|V|$ једночворних стабала (по једно за сваки чвор) и у свакој итерацији се сваком стаблу додаје грана најмање тежине која повезује то стабло и чвор ван њега. Процес траје све док стабло није обједињено тј. док број додатих грана није $|V| - 1$.

Доказ ефикасности:

> У свакој итерацији се сваком појединачном стаблу шуме која касније формира тражено стабло додаје грана најмање тежине која га повезује са чвором ван њега, а валидност овог корака (подкорака итерације) је доказана у доказу ефикасности Примовог алгоритма. Такође, овај доказ се може свести и на доказ ефикасности Крускаловог алгоритма јер се у свакој итерацији у више подкорака у стабло додаје „наупотребљена“ грана најмање тежине, па је то део доказа ефикасности Крускаловог алгоритма.

Пример поступне примене алгоритма:



Ефикасна имплементација користи структуру WQUPC дисјунктних скупова као и код Крускаловог алгоритма за утврђивање повезаности, а ред са приоритетом за бирање гране најмање тежине која води до чвора ван стабла. Сложеност овог алгоритма је:

$$O(\mathrm{E}\log V^{\uparrow})$$

Помоћне структуре података

Поменуте помоћне структуре података нису предмет овог рада, па због тога овде нису детаљно објашњене већ су њихове карактеристике и карактеристике њихових операција укратко описане у овом одељку. Такође, ове структуре су детаљно имплементиране уз главне алгоритме овог рада и достављене у прилогу.

Дисјунктни скупови

Дисјунктни скупови представљају структуру која прати скуп елемената одређеног типа који међусобно могу бити повезани и формирају дисјунктне подскупове главног скупа. Основне операције над дисјунктним скуповима и сложености њихових имплементација различитим алгоритмима су представљене у следећој табели:

Операција	**Опис**	**Сложеност**			
		Quick-Find	Quick-Union	Weighted Quick-Union	Weighted Quick-Union with Path Compression
Union	повезује два елемента	$O(N)$	$O(N)$	$O(\log N)$	$O(\log * N)$ $O(1^{\dagger})$
Find	проналази индекс компоненте повезаности којој елемент припада	$O(1)$	$O(N)$	$O(\log N)$	$O(\log * N)$ $O(1^{\dagger})$

Редови са приоритетом

Редови са приоритетом представљају структуру која је слична обичној структури реда. Разлика је у томе што ови редови не функционишу по принципу FIFO (First-In-First-Out), већ је сваком члану реда додењен приоритет и члан са највећим приоритетом први излази из реда. Основне операције над редовима са приоритетом и сложености њихових имплементација различитим структурама су представљене у следећој табели:

Операција	**Опис**	**Сложеност**		
		Binary Heap	d-ary Heap	Fibonacci Heap
Insert	убацује елемент у ред	$O(\log N)$	$O(\log_d N)$	$O(1)$
Delete	избацује елемент највећег приоритета из реда	$O(\log N)$	$O(\log_d N)$	$O(\log N)$
Decrease Key	мења приоритет већ убаченог елемента елемента	$O(\log N)$	$O(\log_d N)$	$O(1)$
Top	враћа елемент највећег приоритета	$O(1)$	$O(1)$	$O(1)$

Имплементација

У овом одељку су као прилог у виду CD-а достављене имплементације главних алгоритама овог рада као и неких од поменутих помоћних структура и алгоритама који њима манипулишу. Коришћен је програмски језик Java због поједностављеног референцирања.

Референце

Even, S., Even, G. (2012) *Graph Algorithms*, 2nd edn., New York: Cambridge University Press.

Živković, M. (2000) *Algoritmi*, Beograd: Matematički fakultet.

Cormen,T.H.,Leiserson,C.E.,Rivest R.L.,Stein,C. (2012) *Graph Algorithms*, 3rd edn., London: The MIT Press.

Dasgputa, S., Papadimitriou, C.H., Vazirani U.V. (2006) *Algorithms*, California: McGraw-Hill.

Sedgewick, R., Wayne, K. (2011) *Algorithms*, 4th edn., Boston: Addison-Wesley.

Wikipedia (2013) *Bernard Chazelle,* Available at:*http://en.wikipedia.org/wiki/Bernard_Chazelle*.

Wikipedia (2013) *Borůvka's algorithm,* Available at:*http://en.wikipedia.org/wiki/Bor%C5%AFvka%27s_algorithm*.

Wikipedia (2013) *d-ary heap,* Available at:*http://en.wikipedia.org/wiki/D-ary_heap*.

Wikipedia (2013) *Donald B. Johnson,* Available at:*http://en.wikipedia.org/wiki/Donald_B._Johnson*.

Wikipedia (2013) *Robert C. Prim,* Available at:*http://en.wikipedia.org/wiki/Robert_C._Prim*.

Wikipedia (2014) *Edsger W. Dijkstra,* Available at:*http://en.wikipedia.org/wiki/Edsger_W._Dijkstra*.

Wikipedia (2014) *Otakar Borůvka,* Available at:*http://en.wikipedia.org/wiki/Otakar_Bor%C5%AFvka*.

Wikipedia (2014) *Robert Tarjan,* Available at:*http://en.wikipedia.org/wiki/Robert_Tarjan*.

Wikipedia (2014) *Vojtěch Jarník,* Available at:*http://en.wikipedia.org/wiki/Vojt%C4%9Bch_Jarn%C3%ADk*.