УНИВЕРЗИТЕТ „СВ. КИРИЛ И МЕТОДИЈ“ – СКОПЈЕ

ПРИРОДНО – МАТЕМАТИЧКИ ФАКУЛТЕТ

ИНСТИТУТ ЗА МАТЕМАТИКА

МРЕЖНО ПРОГРАМИРАЊЕ

ПРОБЛЕМ НА МИНИМАЛНО ОПФАЌАЧКО ДРВО, ПРИМОВ АЛГОРИТАМ И НЕГОВА ПРИМЕНА ВО КОНКРЕТЕН ПРОБЛЕМ

-труд од областа математика и информатика
презентиран на Приматијада 2015, Балатон, Унгарија-

Ментор:

Д-р Ирена Стојковска

Изработила:

Ивона Ѓероска

Април, 2015

ABSTRACT

In the design of electronic circuitry, it is often necessary to make a set of pins electrically equivalent by wiring them together. Network optimization problems have become one of the most pervasive fields of research in computer science in the past few decades. We can see hundreds of problems in real life that can be defined and solved in terms of network optimization.
In the next few chapters, our main topic is going to be the minimum spanning tree problem. Such a tree is defined as a connected, acyclic graph that carries a minimum total weight. For this purpose, first we give a brief introduction to this part of Graph theory, including the definitions of the main terms and the algorithmic approach to the stated problem.
We study an algorithm for solving a minimum spanning tree problem, known as Prim's algorithm, and we show its application on a real-life problem.

Keywords: minimum spanning tree problem; Prim's algorithm; graph theory; network optimization; minimum weight; acyclic; connected

АБСТРАКТ

При составување на електрични кола, често е потребно некое множество точки со жица да се поврзат под ист напон. Во последните неколку декади, проблемот за мрежна оптимизација стана едно од најпродорните полиња на истражување. Во секојдневниот живот сме сведоци на стотици проблеми кои може да се решат во рамките на мрежна оптимизација.

Во следните неколку поглавија, наша главна тема ќе биде проблемот на минимално опфаќачко дрво. Вакво дрво се дефинира како поврзана, нециклична мрежа која има минимална вкупна тежина. За оваа цел, прво ќе дадеме краток вовед за овој дел од теоријата на мрежи, вклучувајќи ги и дефинициите на главните поими и соодветен алгоритам за наведениот проблем.

Ќе проучиме и алгоритам за решавање на минимално опфаќачко дрво, познат како Примов алгоритам, и ќе ја демонстрираме неговата примена на реален проблем.

Клучни зборови: минимално опфаќачко дрво; Примов алгоритам; теорија на мрежи; мрежна оптимизација; минимална тежина; ациклична; поврзана

I ВОВЕД

Поимот за мрежи често се среќава во секојдневниот живот. Мрежните претставувања наоѓаат широка примена во производство, транспорт, инфраструктурни планови, комуникација и дистрибуција. Поради широката примена и високата ефикасност на визуелизација на врските помеѓу одредени делови од еден систем, мрежите се користат во скоро секое поле од науката и економијата.

Во компјутерската наука, како предмет на изучување сé повеќе се јавуваат мрежите како структури од податоци, и проучувањето на алгоритми за работа со нив е фундаментално за ова поле. Алгоритмите за решавање на задачи од областа на оптимизација, обично содржат низа од чекори, од кои секој дава множество можности. ***Алчен* алгоритам** (анг. *greedy algorithm*) е алгоритам кој во секој чекор ја избира онаа опција од понудените, која во моментот изгледа најсоодветно. Тоа значи, дека локално го избира оптималното решение, со надеж дека ваквата постапка ќе нé одведе до глобално оптимално решение. Ваквите алгоритми не секогаш даваат оптимално решение, но сепак во многу случаи се ефикасни.

Истражување на една мрежа претставува систематско следење на нејзините **рабови** (анг. *edges*) со цел да се посетат некои определени **темиња** (анг. *vertices*) на мрежата. Алгоритам кој го истражува текот на една мрежа може да даде многу информации за нејзината структура. Постојат стотици интересни реални проблеми кои може да се дефинираат и разработуваат со помош на мрежна оптимизација. Во следново излагање главно ќе се задржиме на **проблемот на минимално опфаќачко дрво** (анг. *minimum spanning tree problem*). Вакво дрво се дефинира како начин на поврзување на сите темиња со искористување на најмала можна „тежина“ (или најмало растојание), ако секој раб е зададен со определена тежина (должина). Исто така, ќе изложиме еден начин на доаѓање до оптимално решение, изработка на програма која го користи алгоритамот на Прим (Prim’s algorithm), и нејзина примена во еден конкретен случај „оптимален начин за мрежно поврзување на просториите од Институтот за математика, ПМФ, Скопје“.

II ОСНОВНИ ПОИМИ ОД ТЕОРИЈА НА МРЕЖИ

Како што кажавме, под **мрежа** подразбираме систем од парови од **темиња** меѓусебно поврзани со **рабови**. Секој таков пар од темиња, може да се гледа како подреден пар (во насочени мрежи) или пак неподреден пар (во ненасочени мрежи). Теоријата на мрежи се изучува со векови, во различни земји и на различни начини. Како последица од тоа, среќаваме многубројни различни термини со ист концепт. На пример, под зборот „мрежа“, најчесто се подразбира ненасочена мрежа и покрај тоа што тоа не е специфицирано. Сепак најсигурно е тоа да се заклучи од самиот концепт на текстот. Во овој дел ќе избереме по еден термин за секој поим, и под зборот „мрежа“ ќе подразбираме ненасочена мрежа, доколку не е поинаку напоменето. Сепак, треба да биде јасно дека во различни литератури се даваат различни интерпретации.

Мрежите се претставуваат како подредени парови од темиња и рабови, G = (V, E)[1]. Разликите помеѓу различни типови на мрежи е во дефинираноста на множеството *E*.

Во **насочени мрежи**, секој елемент од множеството *E* претставува подреден пар, и рабовите ги замислуваме како стрелки од едно, **почетно теме**, до друго, **крајно теме**. Секое од овие темиња се нарекува крајна точка на работ. Една насочена мрежа се нарекува **едноставна**, ако постои најмногу еден раб помеѓу еден пар темиња. Во спротивно, ако од едно теме во мрежата постојат повеќе рабови кои него го поврзуваат со друго теме (еднозначно определено), мрежата се нарекува **мултимрежа**.

Во **ненасочени мрежи**, секој раб претставува двоелементно подмножество на V. Едноставна ненасочена мрежа не содржи повеќе од еден раб помеѓу ист пар темиња, ниту пак јамки („рабови“ кои поврзуваат ендо теме само со себе). Во спротивно, исто така се нарекува мултимрежа. Предноста на едноставните, ненасочени мрежи е тоа што во контекст на релации, за нив важи нерефлексивност (без јамки) и симетричност (ненасочени рабови), па според тоа можеме да ги третираме како насочени мрежи со тоа што секои две поврзани темиња, претставуваат два подредени парови (два раба со спротивна насока).

Пред да преминеме на поимите кои ни се од главен интерес, потребно е да дефинираме уште неколку поими:

Под **цилкус** подразбираме мрежа за која важи дека за *n* темиња {*0, 1, 2, ..., n-1*} постои раб помеѓу *i* и *i+1* за секое *i* = 0, 1, 2, … *n*-1 и помеѓу 0 и *n*-1. Мрежа која не содржи циклуси се нарекува **ациклична**.

Една мрежа H е **подмрежа** од мрежата G, ако множеството точки $V_H \subseteq V_G$ и множеството рабови $E_H \subseteq E_G$.

Поврзаноста е една од основните особини на мрежите: дали мрежата е поделена на две или повеќе подмрежи кои меѓу себе не се поврзани со рабови. До овој поим се доаѓа со помош на

[1] Од англиски: G – graph (мрежа), V – vertices (темиња), E – edges (рабови)

поимот за **достапност**: дали од едно теме постои пат до некое друго теме, т.е. дали тоа е достапно за него. Под **пат** подразбираме низа од темиња $v_{0,\dots,}v_{n-1}$, така што постои раб $\{v_i, v_{i+1}\}$ во мрежата G.

Ациклична, поврзана мрежа се нарекува **дрво**. *Една мрежа G е дрво ако и само ако е поврзана мрежа и има точно |V| - 1 рабови* (|V| е бројот на темиња). Ова својство на мрежите се заснова не една лема која изнесува една важна особина на поврзаните мрежи: *Секоја поврзана мрежа G = (V, E) има најмалку |V| - 1 рабови, т.е. |E| ≥ |V| - 1 (доказ: James Aspnes, NOTES ON GRAPH THEORY, December, 2010, Yale University).* Искористените рабови од една дадена мрежа при формирање на дрво во неа, ги нарекуваме **врски** (анг. *links*). Од овие тврдења следува дека со отстранување на една врска од формирано дрво во една мрежа, добиваме точно 2 неповрзани компоненти.

III ПРОБЛЕМ НА МИНИМАЛНО ОПФАЌАЧКО ДРВО

Опфаќачко дрво во една мрежа G е подмрежа од G која ги содржи (ги опфаќа) сите темиња, и претставува дрво (т.е. е *поврзана* и *ациклична* мрежа). Со математичка индукција и со помош на претходно изложените тврдења, се покажува и следнава теорема: *Секоја непразна, поврзана мрежа содржи опфаќачко дрво*.

Нам од интерес ни е **проблемот на минимално опфаќачко дрво**. Како што индицира самиот наслов, овој проблем налага формирање на опфаќачко дрво во една дадена мрежа G = (V, E) во која секој од рабовите (*u, v*) има своја т.н. **тежина** *w*(*u, v*) (која може да претставува било каква особина, на пример должина, капацитет, носивост, тежина, време, цена...), така што вкупната тежина е минимална. Значи, бараме подмножество T од E, така што

$$w(T) = \min \sum_{(u,v)\in T} w(u,v)$$

Тежината на рабовите е секогаш позитивен број. Една мрежа може да има повеќе различни опфаќачки дрва, а од сите нив, оптимално решение на проблемот на минимално опфаќачко дрво е она кое има помала или еднаква вкупна тежина во споредба со сите останати опфаќачки дрва во мрежата.

Од самата дефиниција и својствата за дрво во една мрежа, гледаме дека во овој проблем ни се потребни точно n – 1 врски. Во продолжение ќе разгледаме еден од алгоритмите за наоѓање на тие врски и индуцирање на оптимално решение, познат под името „Алгоритам на Прим“.

Овој проблем наоѓа широка примена во секојдневниот живот. Некои такви примери се:

- Одредување на транспортни мрежи, како железници и патишта, кои поврзуваат одредени релации со притоа минимална искористеност на потребни материјали, т.е. минимална потрошувачка.
- Оптимален план за поставување на телефонски жици и далеководи.
- Водоводна инсталација во поголеми подрачја низ канали со минимална должина, минимална потрошувачка или цевки со најмал можен капацитет.

И примената која подоцна ќе ја прикажеме во реален случај:

- Мрежно поврзување на електронски уреди со при тоа минимализирање на должината на искористените кабли.

Да претпоставиме дека дадена ни е една поврзана, ненасочена мрежа G = (V, E) со функција на тежина w: E → $\mathbb{R}$, и целта ни е да најдеме минимално опфаќачко дрво за G. Алгоритамот на Прим за кој подоцна ќе стане збор, е таканаречен алчен алгоритам. Ова значи дека опфаќачкото дрво се „разгранува“ постапно, избирајќи ги темињата едно по едно. Алгоритамот работи со едно множество A, така што во цуклусот важи инваријантата: *Пред секоја итерација, A е подмножество од некое минимално опфаќачко дрво.*

Во секој чекор избираме по еден раб (u, v) која смее да се додаде на множеството A, без при тоа да се наруши инваријантата, односно A ∩ {(u, v)} е исто така подмножество од некое минимално опфаќачко дрво. Ваквиот раб го нарекуваме **безбеден раб** за A, бидејќи тој безбедно може да се додаде на множеството A и при тоа да важи инваријантата.

Циклусот е зададен на следниов начин:

ГЕНЕРИРАЧКИ АЛГОРИТАМ ЗА МИНИМАЛНО ОПФАЌАЧКО ДРВО

1. A □∅
2. **while** A не формира минимално опфаќачко дрво
3. **do** најди раб (u, v) кој е безбеден за A
4. A □ {(u, v)}
5. **return** A

После првиот ред, множеството A тривијално ја задоволува инваријантата. Циклусот од 2 до 4 ја одржува инваријантата со тоа што на множеството A додава само безбедни рабови. Сите рабови кои се додаваат на A припаѓаат на минимално опфаќачко дрво.
Најкомплициран дел овде е, секако, третиот ред каде се бара новиот безбеден раб кој ќе биде следната врска во опфаќачкото дрво. Таков раб мора да постои, бидејќи кога ќе се негира овој ред, инваријантата наведува на тоа дека имаме опфаќачко дрво T, така што A ⊆ T. Во **while** циклусот, A мора да биде вистинско подмножество од T, што значи мора да постои раб од T кој не припаѓа на A и е воедно безбеден за A.

За претставување на алегоритамот на Прим, потребно е да дефинираме уште неколку поими и да наведеме едно тврдење.

Пресек (S, V – S) на една ненасочена мрежа G = (V, E) претсавува партиција на множеството V. За еден раб (u, v) велиме дека го **прекршува** пресекот (S, V – S) ако едната од неговите крајни точки припаѓа на S, а другата на V – S. Велиме дека пресекот го **почитува** множеството A, ако ниту еден раб од A не го прекршува пресекот. Еден раб е **лесен раб** кој го прекршува пресекот ако неговата тежина е најмала од сите рабови кои го прекршуваат пресекот.

Нашето правило за препознавање на безбедни рабови е дадено со следнава теорема: *Нека G = (V, E) е поврзана, ненасочена мрежа со реално-вредносна функција на тежина w дефинирана на множеството E. Нека A е подмножество од E кое се содржи во некое минимално опфаќачко дрво за G, нека (S, V – S) е кој било пресек од G кој го почитува множеството A, и нека (u, v) е лесен раб кој го прекршува (S, V – S). Тогаш работ (u, v) е безбеден за A.*

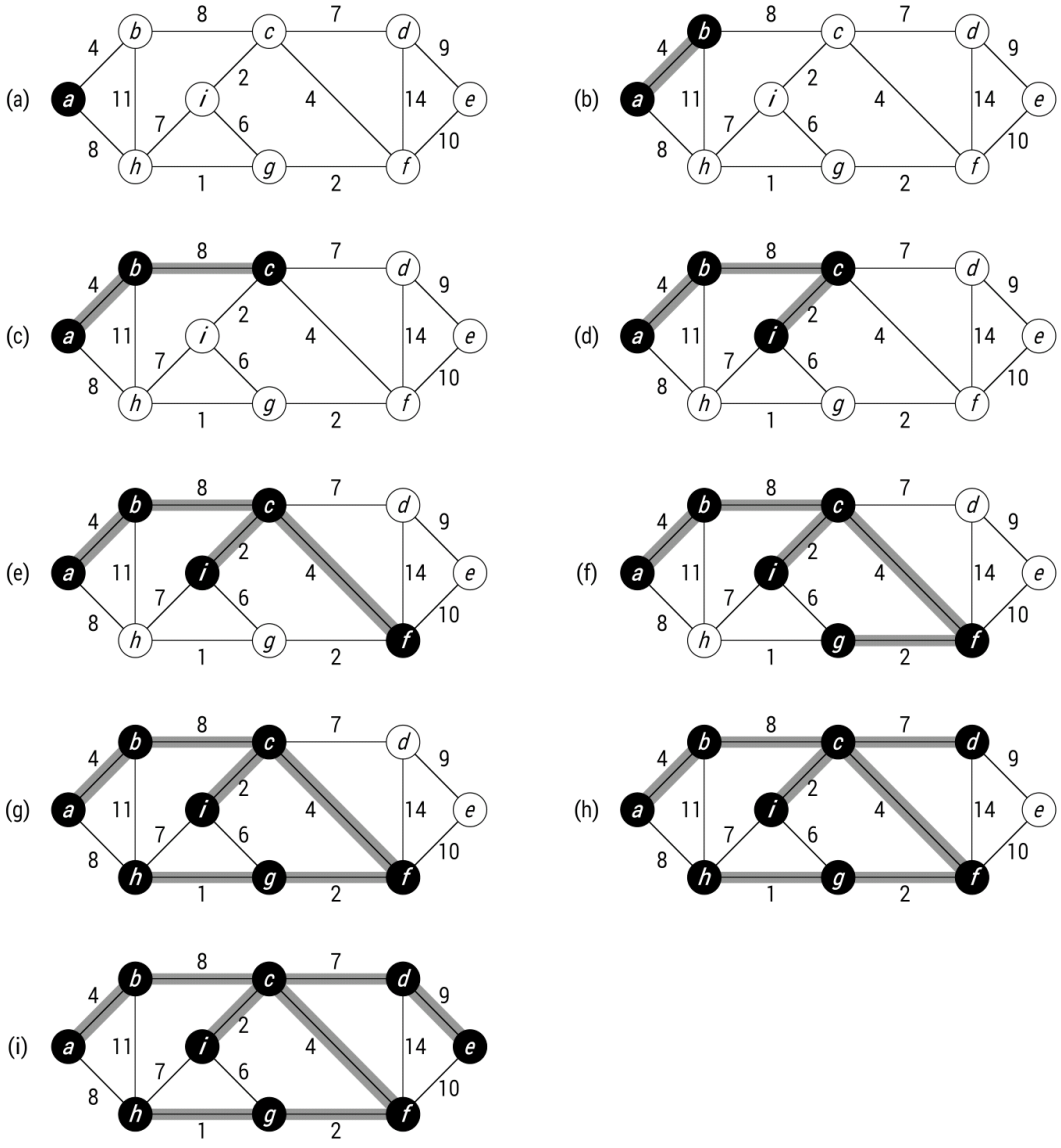
Оваа теорема ни помага подобро да го разбереме погоре наведениот генерирачки алгоритам на поврзана мрежа G = (V, E). Како што продолжува постапката, множеството A е секогаш ациклично, бидејќи во спротивно минималното опфаќачко дрво би содржело циклус, што е контрадикција. Во секој момент од текот на постапката, мрежата G_A = (V, A) се нарекува шума, а секоја поврзана компонента од мрежата претставува дрво. Во ова тврдење земаме во предвид дека и секое „неповрзано“ теме претставува едно дрво. Исто така, секој следен раб кој се додава на множеството A поврзува две различни дрва од мрежата, бидејќи во спротивно би формирал циклус.Претходно изложениот циклус се повторува точно |V| - 1 пати, со тоа што во секој чекор се додава по една врска во минималното опфаќачко дрво.

IV ПРИМОВ АЛГОРИТАМ

Постојат повеќе алгоритми кои се специјален случај на генерирачкиот алгоритам за минимално опфаќачко дрво за кој зборувавме погоре. Други такви алгоритми се и Крускаловиот алгоритам и алгоритамот на Борувка. Примовиот алгоритам првично бил развиен во 1930 година од страна на чешкиот математичар Војтеч Јарник, но подоцна на него дополнително работеле и американскиот математичар Роберт Прим во 1957 и холандскиот математичар Едсгер Дијкстра во 1959 година, поради што често се среќава под името ДЈП алгоритам.

Примовиот алгоритам работи по принципот дека во секој момент, рабовите во множеството A формираат единствено дрво. На слика 1 сликовито е образложен овој принцип. Дрвото започнува од едно произволно теме r, и понатаму продолжува да се разгранува сé додека не ги опфати сите темиња од мрежата. Во секој чекор, на дрвото A му се додава по еден лесен раб кој го поврзува A со едно изолирано теме од G_A = (V, A). Од петходно изнесените тврдења, гледаме дека со ова правило се додаваат само рабови кои се безбедни за A, што значи, кога алгоритамот ќе заврши, рабовите од A формираат минимално опфаќачко дрво. Овде зборуваме за алчна стратегија, бидејќи во секој чекор, дрвото се надополнува со нов раб кој вкупната тежина на дрвото ја надополнува со најмала можна тежина.

Клучниот дел во ефикасното развивањето на Примовиот алгоритам е да се олесни избирањето на нов раб кој ќе се додаде на дрвото A.



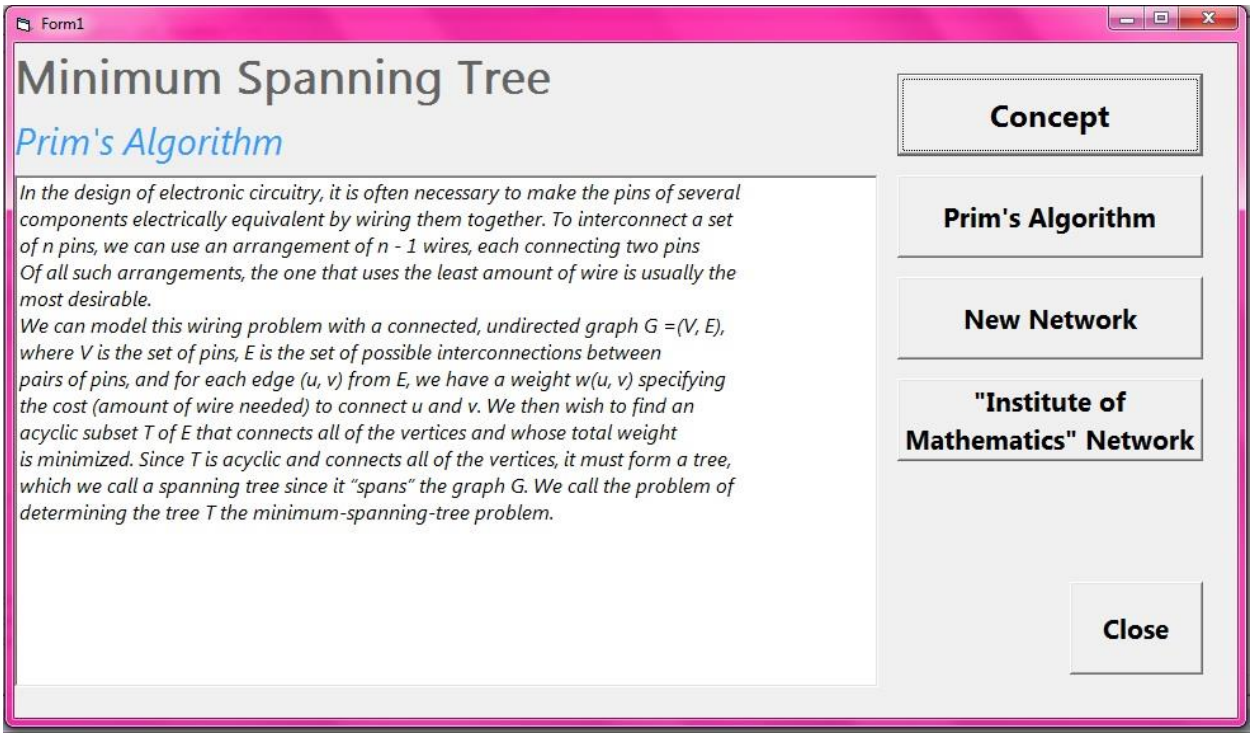
Слика 1. *Развивање на Примовиот алгоритам прикажано на произволна мрежа. Првото теме, или „коренот“ на дрвото во овој случај е a. Задебелените рабови се оние врски кои влегуваат во опфаќачкото дрво кое расте, а сите рабови во мрежата се прикажани со црни линии. Во секој чекор од алгоритамот, темињата во дрвото определуваат еден пресек на мрежата, и еден лесен раб кој го прекршува пресекот се додава на дрвото. Во вториот чекор, на пример, алгоритамот има избор дали да го додаде работ (b, c) или работ (a, h) на дрвото бидејќи и двата се лесни рабови кои го прекршуваат пресекот.*

V ПРИМЕНА НА ПРИМОВИОТ АЛГОРИТАМ СО ПОМОШ НА Visual Basic

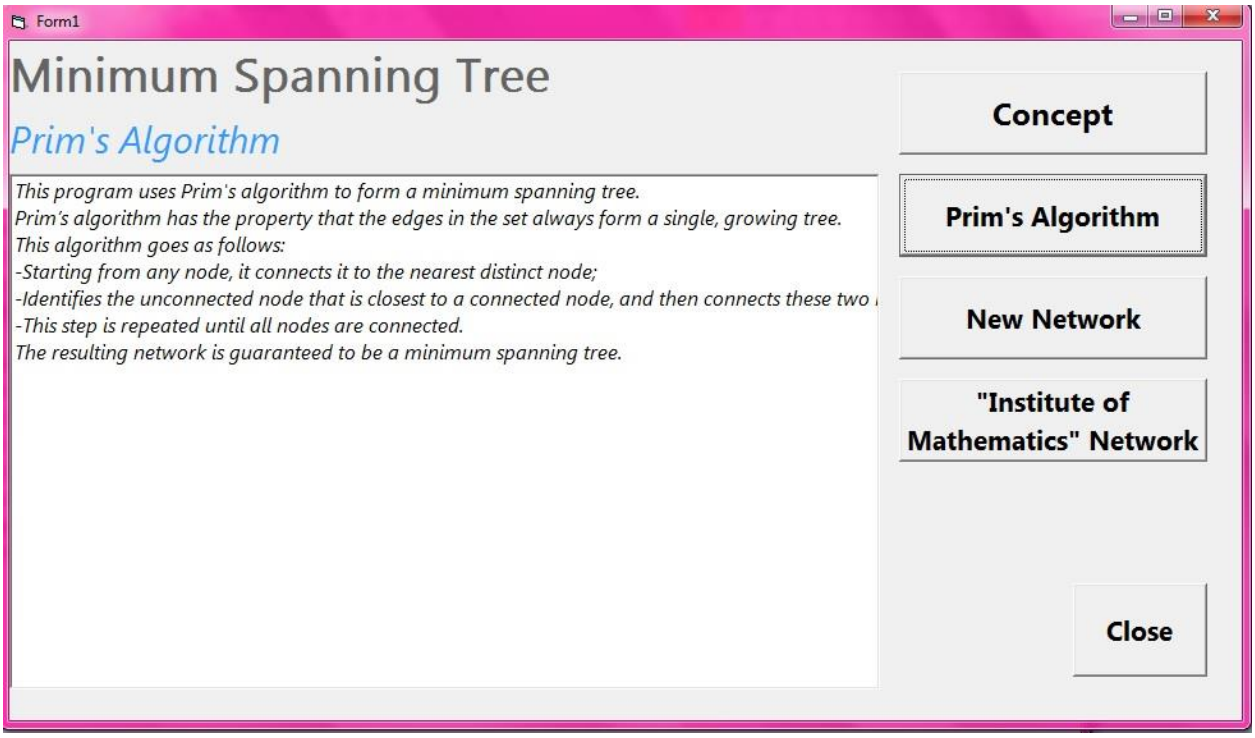
Во ова поглавие ќе го изнесеме претходно опишаниот проблем на минимално опфаќачко дрво со користење на Примовиот алгоритам, и поопсно ќе ја објаснеме неговата работа. За таа цел користиме претходно изготвена програма.

V.1 ПРОГРАМА ЗА ОПТИМИЗАЦИЈА НА ПРОИЗВОЛЕН ПРОБЛЕМ НА МИНИМАЛНО ОПФАЌАЧКО ДРВО

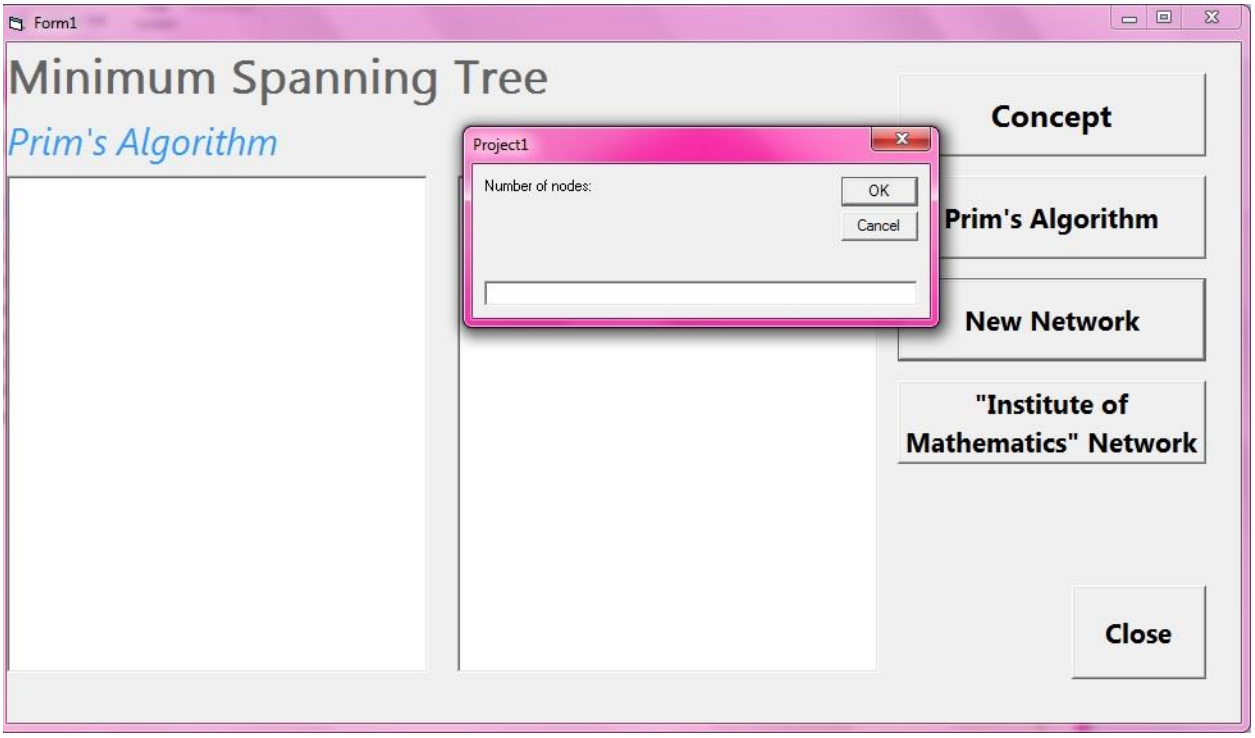
Во продолжение ќе ја објаснеме примената на претходно изнесените принципи и Примовиот алгоритам во општ случај. За таа цел, ќе го објаснеме принципот на работа на ваква програма изработена со програмскиот јазик Visual Basic.



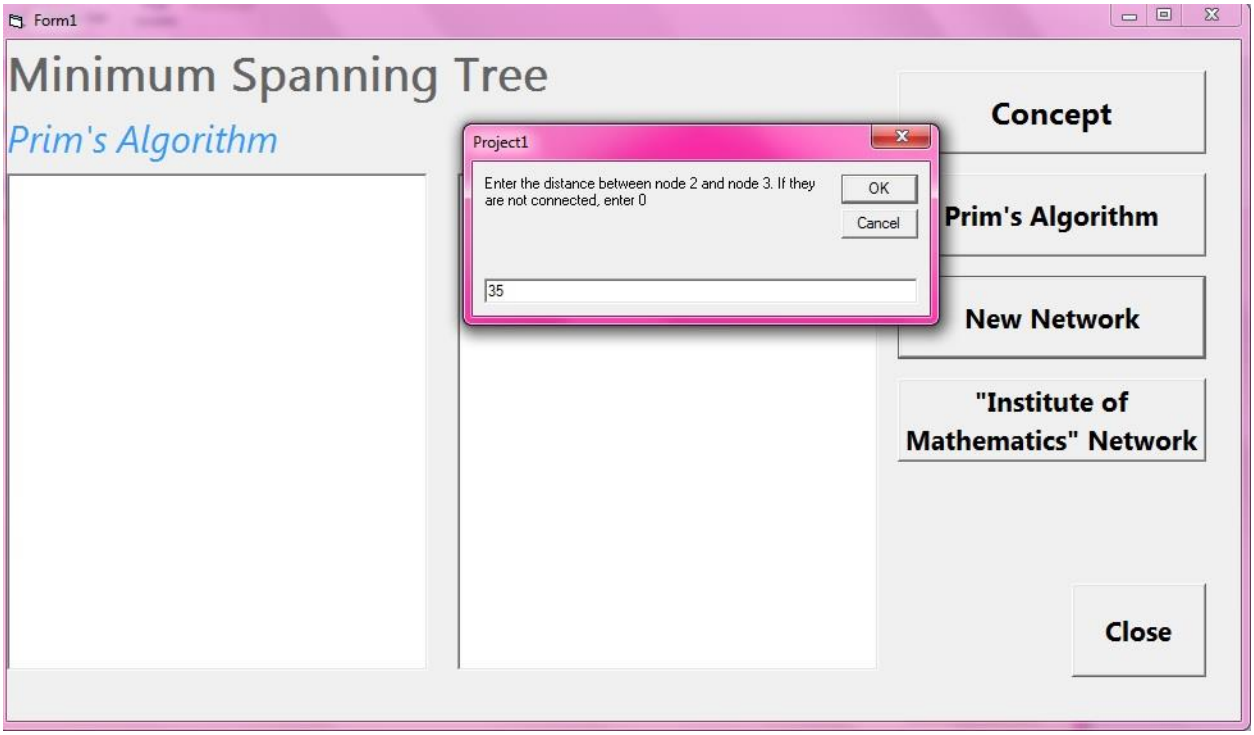
Слика 2. *На почетокот го гледаме првичниот прозорец кој на корисникот му овозможува запознавање со концептот и примената на оваа програма. Првото копче, насловено како Концепт, ни ја дава точно идејата на проблемот на минимално опфаќачко дрво, всушност дел од воведот во овој текст.*



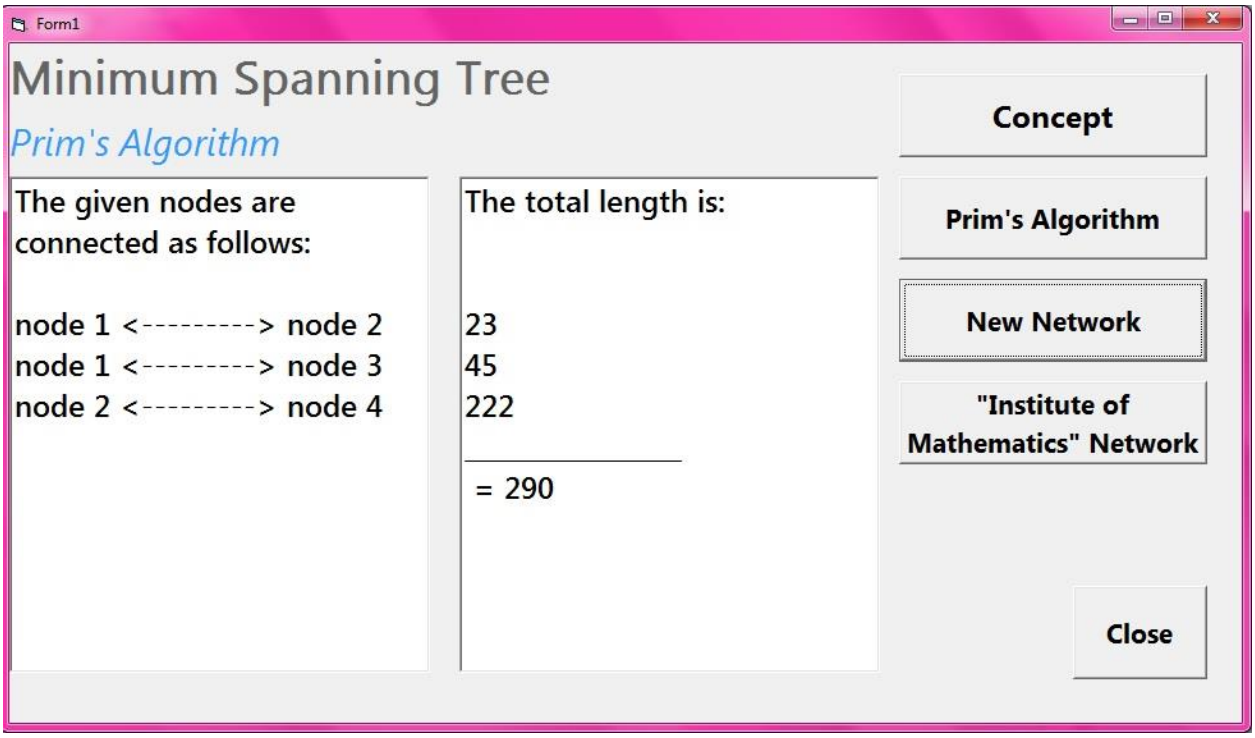
Слика 3. *Второто копче, со име Примов алгоритам, дава описен пристап кон начинот на работа на Примовиот алгоритам. Овој дел поопширно е образложен во поглавијата III и IV.*



Слика 4. *Третото копче, со име Нова мрежа, дава можност алгоритамот да се примени на нова, произволна мрежа. Начинот на кој се внесуваат информациите за оваа функција е прикажан на сликата. Во дијалогниот прозорец, прво се внесува бројот на темиња во мрежата (во зависност од конкретниот проблем, темињата може да претставуваат различни простории/објекти/предмети).*



Слика 5. *На оваа слика го гледаме прозорецот кој се појавува после притискање на копчето OK во прозорецот од Слика 4. Редоследно се внесуваат тежините на сите рабови помеѓу секој пар темиња. Текстот во дијалогниот прозорец гласи „Внеси ја далечината помеѓу темињата 2 и 3. Доколку помеѓу нив не постои директна врска, внеси 0.“ (Забелешка: иако прозорецот бара внесување на „далечина“, овде се работи за тежината на работ, како што видовме во поглавието III. Тежината може да ја претставува било која особина на работ, во зависност од конкретниот проблем.) После секое следно внесување, се притиска копчето OK.*

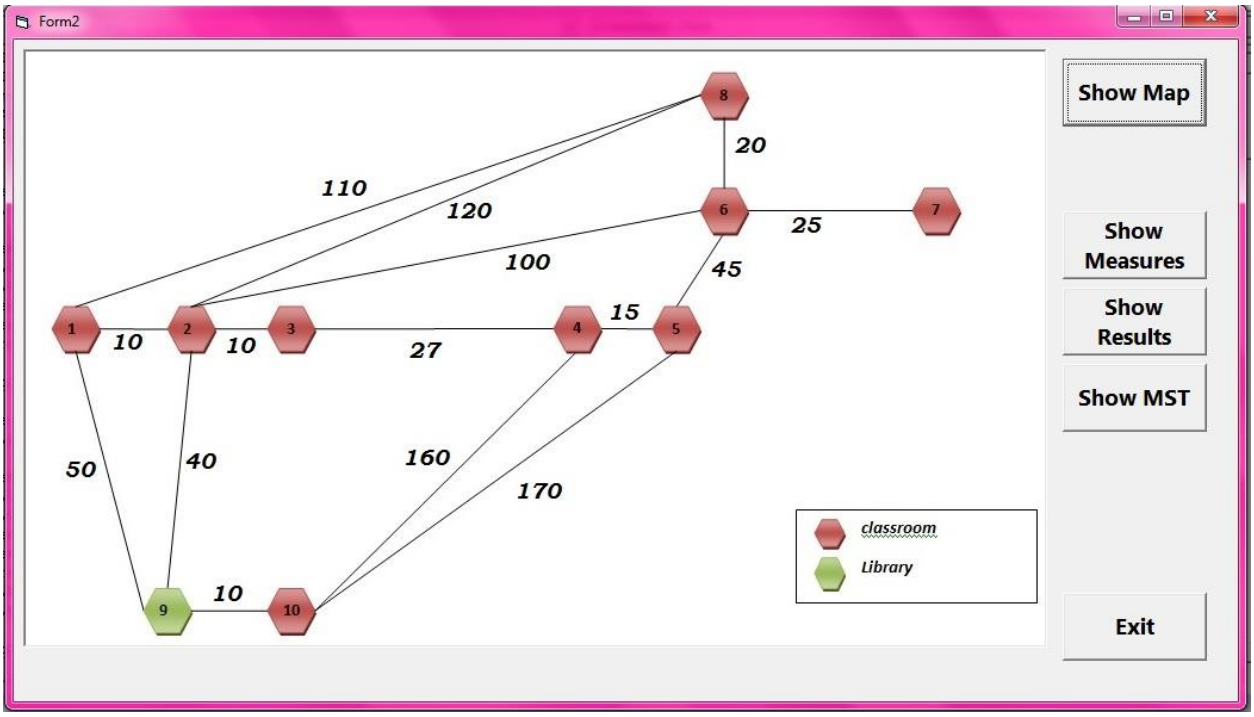


Слика 6. *После последната внесена информација, програмата го испишува пресметаниот резултат. Во рамката од левата страна, „Дадените темиња се поврзани на следниов начин“, дадени ни се гранките на минималното опфаќачко дрво. Во овој конкретен случај сме добиле дрво А кое ги содржи рабовите (1,2), (1,3) и (2,4). Во рамката од десната страна, „Вкупната должина е:“, излистани ни се тежините на соодветните рабови испишани на левата страна, и под линијата програмата ни ја пресметува вкупната тежина на минималното опфаќачко дрво, што всушност ни го претставува оптималното решение на зададениот проблем.*

Во продолжение ќе го разгледаме вториот дел од прикажаната програма.

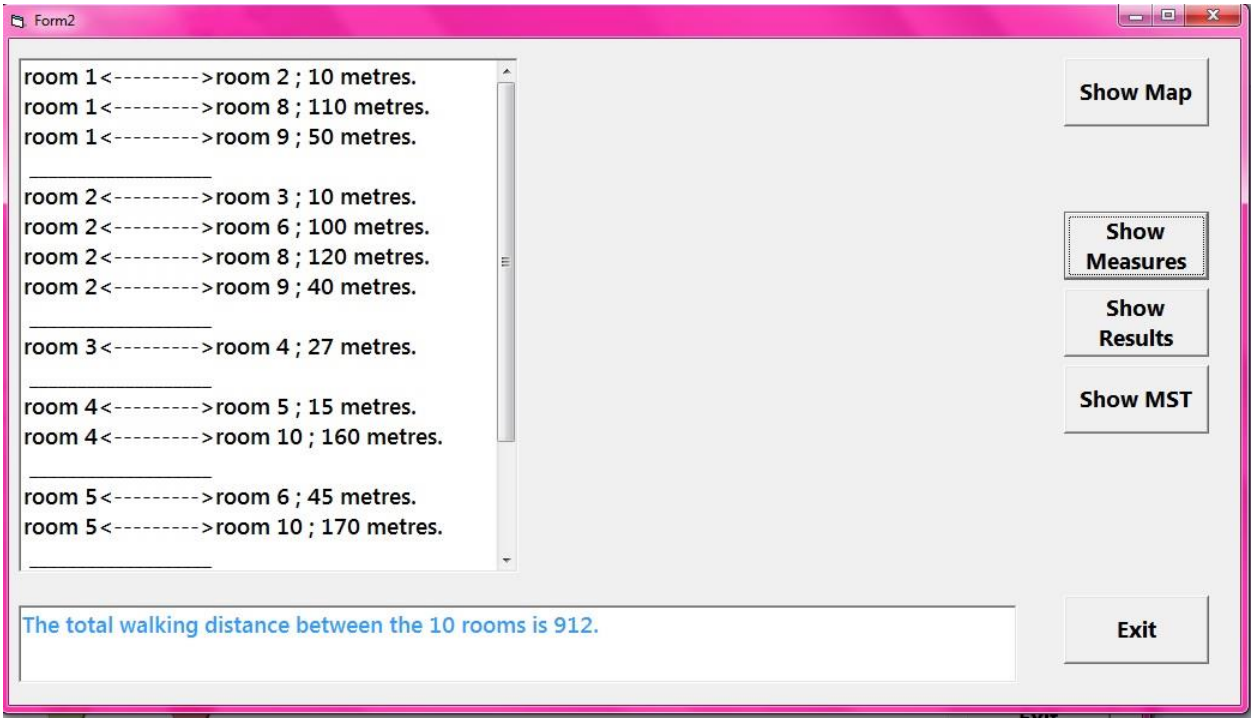
V.2. ОПТИМАЛЕН ПЛАН ЗА МРЕЖНО ПОВРЗУВАЊЕ НА ПРОСТОРИИТЕ ОД ИНСТИТУТ ЗА МАТЕМАТИКА, ПМФ, СКОПЈЕ

За посликовит приказ на Примовиот алгоритам и подобро разбирање на проблемот на минимално опфаќачко дрво, сé што беше до сега изнесено ќе покажеме како да се примени на еден конкретен, реален проблем. За оваа цел, го избравме проблемот на оптимален план за мрежно поврзување на сите простории во рамките на Институтот за математика на Природно-математичкиот факултет во Скопје. За барање на оптималниот план го искористивме принципот прикажан во V.1. Во продолжение ќе го опишеме текот на програмата, после притискање на четвртото копче на прозорецот од Слика 6, „Мрежа во ‘Институт за математика’“.

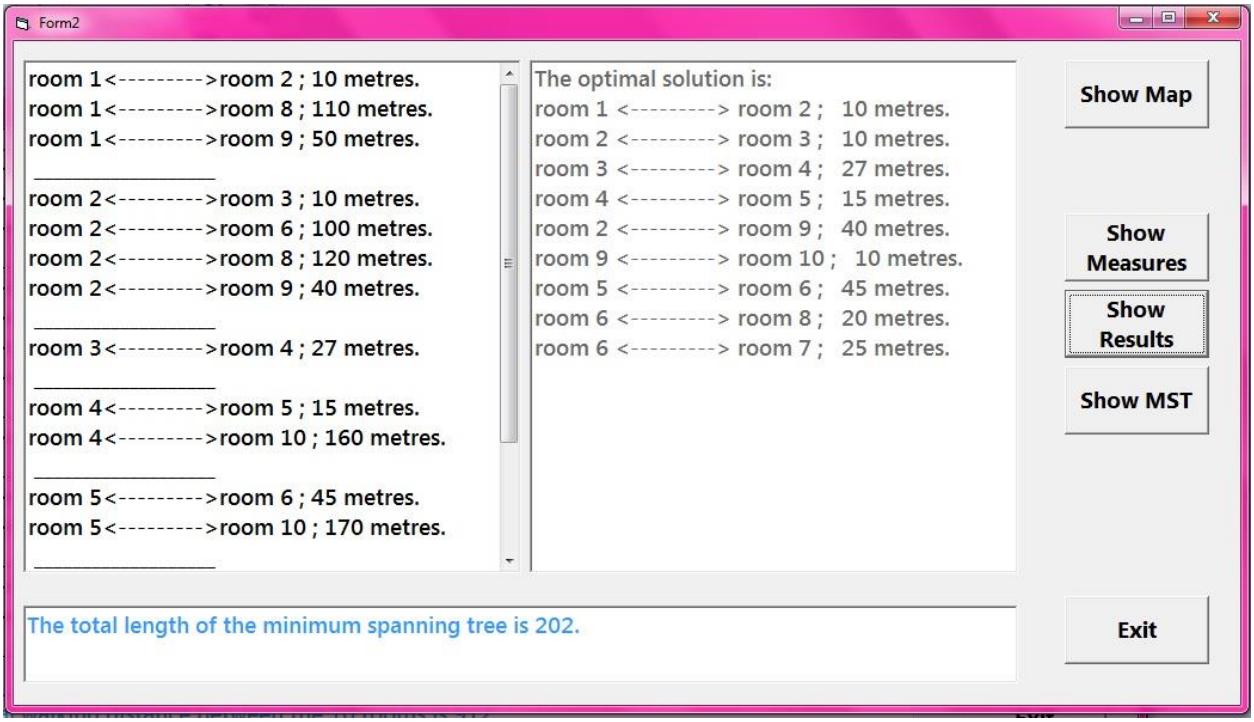


Слика 7. *На новиот прозорец, со притискање на првото копче „Прикажи ја мапата“, програмата ни јавува слика од мрежа, каде темињата претставуваат простории: со црвени шестаголници претставени се сите училници, а со зелен шестаголник е претставена математичката библиотека. До секој раб запишан е по еден број, рабовите се всушност најлогичните патишта (ходници и скали), а броевите ги претставуваат растојанијата помеѓу соодветните простории во метри (Забелешка: на сликата се претставени приближните должини помеѓу просториите).*

Претходната слика ја претставува првичната мрежа G = (V, E), каде бројот на елементи во множеството темиња |V| = 10, што значи очекуваме 9 елементи во множеството A од рабови кои влегуваат во минималното опфаќачко дрво.

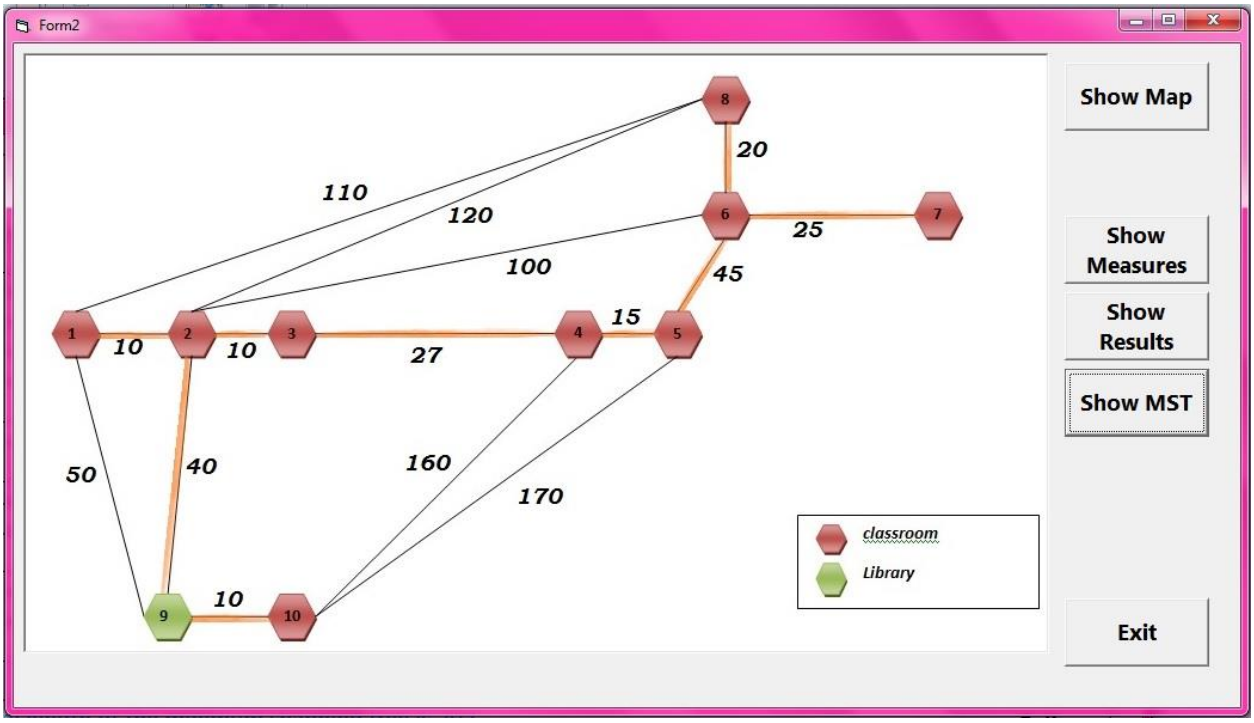


Слика 8. *Следното копче „Прикажи ги мерките“, општо кажано, ни ги испишува тежините на секој од рабовите кои претходно беа внесени во програмата. На прикажаната рамка од левата страна на прозорецот, гледаме листа од растојанијата помеѓу секои две простории во метри. Во рамката во долниот дел на прозорецот, со сини букви ни испишува вкупниот збир на сите патишта од мрежата, „Вкупното растојание помеѓу сите 10 простории е 912 метри.“*



Слика 9. *Со копчето „Прикажи го резултатот“, на десната страна од прозорецот ни се појавува нова рамка која ни го испишува оптималното решение. Всушност, ова се елементите од множеството A, т.е. рабовите кои влегуваат во минималното опфаќачко дрво, додека рамката од левата страна ни ги прикажува елементите од множеството E. Во рамката во долниот дел на прозорецот, со сини букви, овој пат ни се испишува вкупната должина на сите врски од минималното опфаќачко дрво, „Вкупната должина на минималното опфаќачко дрво е 202“. Ова значи дека оптималното решение е 202, т.е. во овој случај, кабел со должина од 202 метри е доволен за мрежно поврзување на сите 10 простории.*

Во споредба со текстот со сини букви од Слика 8, кој кажува дека вкупната должина е 912 метри, гледаме дека оптималното решение добиено со овој принцип, всушност значително ни ја намалува вкупната должина на каблите. Исто така, како што и претпоставивме, минималното опфаќачко дрво користи точно 9 рабови од првичната мрежа (за 1 помалку од вкупниот број на простории, кои се темиња во мрежата).



Слика 10. *После пресметувањето на оптималното решение, за сликовит приказ на минималното опфаќачко дрво ја прикажуваме и првичната мрежа со на неа означени елементите од множеството A, т.е. рабовите кои се искористени во растењето на минималното опфаќачко дрво. Ова се всушност деветте рабови кои беа испишани во Силка 9.*

Понатаму, со последното копче „Крај“ се враќаме во претходниот прозорец, каде повторно може да се пресмета оптимално решение со минимално опфаќачко дрво за некој друг проблем.

Со оваа програма целосно се опфаќа примената и улогата на она што беше изложено во претходните поглавија.

Се поставува прашањето: Како работи оваа програма? За таа цел ќе го дадеме псевдокодот на алгоритамот искористен во реализирањето на програмава со програмскиот јазик Visual Basic.

V.3. МИНИМАЛНО ОПФАЌАЧКО ДРВО СО ПОМОШ НА ПРИМОВИОТ АЛГОРИТАМ: ПСЕВДОКОД

За крај, ќе го објаснеме кодот искористен во делот V.1., земајќи во превид дека во делот V.2. програмата работи на истиот начин, со тоа што сите податоци се претходно внесени.

Во следново излагање ќе го прикажеме псевдокодот на оваа програма:

```
ВЛЕЗ 1: n = број на темиња во избраната мрежа
Димензионирање на матрица A  (1 до n, 1 до n) како Double
За i од 1 до n
        За j од 1 до n
        Ако i<j, тогаш:
                ВЛЕЗ 2: Внеси вредност за A(i, j)
                ИЗЛЕЗ: Грешка, ако внесената вредност A(i, j) < 0
                A(i, j) = A(j, i)
        Инаку  i = j, тогаш A(i, j) = 0
Димензионирање на низата node (1 до n) како Integer
node(1) = 1
suma = 0
За T = 1 до n – 1

min = 0
За p = 1 до n + 1
        За q = 1 до n
                Ако p < q тогаш
                        Ако A(p, q) > min тогаш
                        min = A(p, q)
        Следно q
Следно p
brojac = 0
За br = 1 до T
        За i = 1 до n
                За j = 1 до T
                        Ако i е различно од node(j) тогаш:
                                brojac = brojac + 1
                Следно j
                        Ако brojac = T и ако A(node(br), i) е различно од 0 и ако A(node(br), i) < M
                                min = A(node(br), i)
                                nextnode = i
                                firstnode = node(br)
        brojac = 0
        Следно i
Следно br
node(T + 1) =  nextnode
suma = suma + min
ИЗЛЕЗ: Следна гранка која се додава на дрвото е (firstnode, nextnode), и вкупната должина е suma.
```

VI КОРИСТЕНА ЛИТЕРАТУРА

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, INTRODUCTION TO ALGORITHMS, Second Edition, The MIT Press, 2001
[2] F. S. Hillier, G. J. Lieberman, INTRODUCTION TO OPERATIONS RESEARCH, Seventh Edition, The McGraw-Hill Book Company, 2001
[3] James Aspnes, NOTES ON GRAPH THEORY, Yale University, December 2010