МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2003
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2003
Proceedings of the Thirty Second Spring Conference of
the Union of Bulgarian Mathematicians
Sunny Beach, April 5–8, 2003

ПОКРИВАЩИ ДЪРВЕТА В ГРАФИ

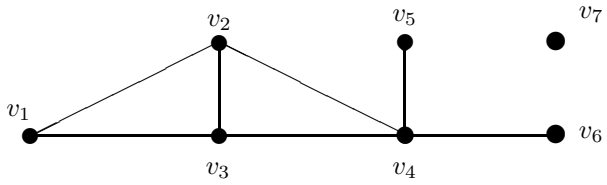
Борислав П. Юруков, Костадин П. Калинов, Иван А. Мирчев

В тази статия авторите продължават да изследват възможността елементи на математическото оптимиране да бъдат преподавани в средното училище (виж [3]). Избран е подход при който се анкцентира върху приложния характер на математиката и информатиката. Много различни проблеми от областта на психологията, химия, индустрията, управлението, планирането, образованието и др., могат нагледно и ефективно да бъдат описани на езика от теория на графите.

Какво е *граф*? Графът е съвкупност от две групи елементи – точки (*върхове*) и линии (*ребра*), съединяващи тези точки. С други думи, ако $V = \{v_1, v_2, ..., v_n\}$ е множество от върхове и $A = \{(v_i, v_j) | v_i \in V,\ v_j \in V\}$ е множество от ребра, съвкупността $G = (V, A)$ ще наричаме *неориентиран граф*.

Всяка последователност от ребра на графа (v_i, v_k), (v_k, v_l), ..., (v_s, v_t), (v_t, v_j) се нарича *път* от v_i до v_j. Когато v_i съвпада с v_j пътят се нарича *затворен* (*цикъл*). *Прост път* е път, който не съдържа цикли.

Пример 1.



Пътят (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_2) не е прост, за разлика от (v_1, v_2), (v_2, v_3), (v_3, v_1), който е прост път (прост цикъл).

Граф, на който между всеки два върха има път, се нарича *свързан*. Графът от пример 1 не е свързан.

Понякога на всяко ребро на графа G се приписва (съпоставя) число, наречено *„тегло“, „дължина“, „стойност“, „цена“* на реброто. Граф, чиито ребра или върхове имат съответни тегла, се нарича *мрежа*. Най-често теглото на дъгата (v_i, v_j) се бележи със $c(v_i, v_j)$ или c_{ij}.

Ако ρ е някакъв път, под *тегло (дължина, стойност, цена) на пътя* ρ се разбира най-често сумата от теглата (дължините, стойностите, цените) на неговите ребра.

Една съвкупност от ребра се нарича *дърво*, ако за нея са изпълнени следните две условия:

1. *ребрата пораждат свързан граф;* 2. *няма цикли.*

Нека $G = (V, A)$ е произволен граф. Всяко дърво, образувано от ребра на G, включващи всички върхове на графа, се нарича *покриващо дърво*. За всеки свързан граф съществува покриващо дърво. Очевидно, дърво състоящо се от едно ребро, включва два върха, от две ребра — три върха и т.н. индуктивно, дърво от $n-1$ ребра включва n върха. От това следва, че всяко покриващо дърво на един свързан граф с n върха се състои от $n-1$ ребра.

Задача 1. *Нека $G = (V, A)$ е свързан граф, на който всички върхове са инцидентни с четен брой ребра. Да се докаже,че съществува цикъл, включващ всички ребра на графа (Ойлеров цикъл).*

Задача 2. *Да се докаже, че във всяко дърво, имащо поне два върха, най-малко два върха са със степен 1.*

Задача 3. *Да се докаже еквивалентността на следните три дефиниции за дърво:*

а) свързан граф без цикли;

б) свързан граф, съдържащ n върха и $n-1$ ребра;

в) граф, в който за всеки два върха съществува единствен прост път (път без повтарящи се върхове), свързващ тези върхове.

Построяване на покриващи дървета

Пример 2. (Задача за клюката). В един университет някои от преподавателите ежедневно се срещат и разговарят, като споделят помежду си всеки интересен слух. Може ли в този университет, т.е. между всички преподаватели, да бъде разпространена някаква клюка? Поставеният проблем само на пръв поглед звучи несериозно, защото най-общо това е проблем, свързан с комуникациите, управлението и разпространението на информация, рекламна дейност и др. Нека всеки преподавател разглеждаме като връх на един неориентиран граф G, на който ребрата показват кои от преподавателите ежедневно се срещат и разговарят. По своята същност въпросът е дали този граф е свързан. Отговорът на този въпрос е положителен, ако може да се построи покриващо дърво за този граф. Невъзможността да се построи покриващо дърво за графа означава и невъзможност за разпространение на слуха в целия университет.

Пример 3. Строителна фирма разглежда проект за строеж на пътища между шест селища, който трябва да осигури комуникациите между тези населени места. Известни са разходите (цените), необходими за прокарването на всеки от възможните пътища между тези селища. Фирмата иска с минимални разходи да осигури пътните комуникации (не е задължително да се прокарва път между всеки две селища). Да разгледаме неориентирания граф $G = (V, A)$, чийто върхове съответстват на градовете, ребрата — на пътищата, които могат да се прокарат между тези градове, а теглата на ребрата са съответните разходи за прокарване на пътя. Ясно е, че в случая стратегията на фирмата се свежда до намиране на покриващо дърво за този граф с минимално тегло (разходи, цени). Теглата на ребрата не е задължително да удовлетворяват условието на триъгълника $c_{ij} \leq c_{ik} + c_{kj}$.

Ясно е, че построяването на покриващо дърво за граф е важен за практиката проблем.

Идея на алгоритъма. В произволен ред се разглеждат ребрата на изходния граф, като на всяка стъпка на алгоритъма се взима решение съответното ребро да бъде включено или не в покриващото дърво. Реброто, което се включва в дървото, се оцветява в *зелено*, а това, което не се включва — в *черно*, т.е. алгоритъмът е процес на *оцветяване на ребрата*.

На всяка стъпка в алгоритъма се прави проверка дали разглежданото ребро в съвкупност със зелените (т.е. включените вече в дървото) ребра образува цикъл. Ако това е така — реброто се оцветява в черно (т.е. не се включва в покриващото дърво), в противен случай се оцветява в зелено (т.е. се включва в покриващото дърво).

Зелените (включените в дървото) ребра образуват една или повече свързани компоненти. Върховете на всяка от компонентите образуват множество от върхове, което ще наричаме *„букет“*. Алгоритъмът приключва своята работа, когато всички n върха на графа попаднат в един букет, т.е. ребрата, включени в строеното дърво, се окажат еднокомпонентен (свързан) граф без цикли, включващ всички върхове на изходния граф.

Описание на алгоритъма. **Стъпка 1.** Избираме произволно ребро. Оцветяваме това ребро в зелено. Сформираме букет от върховете на това ребро.

Стъпка 2. Избираме произволно неоцветено ребро. Ако такова ребро няма, преминаваме към *стъпка 4*.

Стъпка 3. Възможни са следните четири случая:

а) нито един от краищата на реброто не принадлежи на сформиран до момента букет — оцветяваме реброто в зелено и сформираме нов букет от върховете (краищата) на това ребро. Преминаваме към *стъпка 4*;

б) и двата края на реброто принадлежат на един и същ букет върхове, сформиран до момента — оцветяваме реброто в черно, т.е. не го включваме в дървото, което строим. Преминаваме към *стъпка 4*;

в) единият връх на реброто принадлежи на даден букет, а другият не принадлежи на никой от сформираните букети — оцветяваме реброто в зелено и невключения в букет връх включваме в букета, съдържащ другия връх. Преминаваме към *стъпка 4*;

г) върховете на реброто принадлежат на различни букети — оцветяваме реброто в зелено и обединяваме двата букета в един нов букет. Преминаваме към *стъпка 4*.

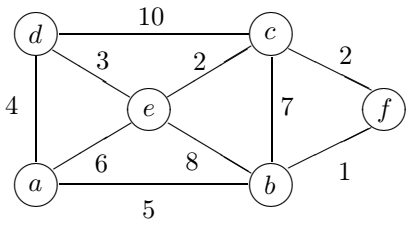
Стъпка 4. Ако всички върхове на графа са в един букет (броят на оцветените ребра е с единица по-малък от броя на върховете на графа), ребрата, оцветени в зелено, образуват покриващо дърво за графа. Край.

Стъпка 5. Преминаваме към стъпка 2.

Очевидно, така формулираният алгоритъм „работи“ добре, когато съществува покриващо дърво и „зацикля“, ако такова не съществува. Как може да се премахне

„зацикляне“, така че алгоритъмът да установява несъществуването на покриващо дърво и да спира своята работа?

Пример 4. Даден е граф $G = (V, A)$



Да се построи покриващо дърво за този граф. Резултатите от прилагането на алгоритъма ще изложим за прегледност в таблица (което не е задължително).

No	Ребро	Цвят	Букет 1 (от върхове)	Букет 2 (от върхове)	Букет 3 (от върхове)	...
1	(a, e)	зелен	a, e	$\emptyset$	$\emptyset$	
2	(b, f)	зелен	a, e	b, f	$\emptyset$	
3	(d, c)	зелен	a, e	b, f	d, c	
4	(e, c)	зелен	обединяваме букети 1 и 3 a, e, d, c	b, f	$\emptyset$	
5	(a, d)	черен	a, e, d, c	b, f	$\emptyset$	
6	(e, b)	зелен	a, e, d, c, b, f	$\emptyset$	$\emptyset$	

Лесно се проверява, че ако разглеждаме ребрата на графа в друг ред, ще получим други покриващи дървета (проверете!). Намереното покриващо дърво може да се разглежда като една от възможните стратегии на строителната фирма, цитирана в пример 3. Ако теглата на ребрата са разходите в млн. лева, за построяването на съответните участъци от пътната мрежа лесно се установява, че това не е най-доброто решение за фирмата. Теглото на намереното покриващо дърво е (общите разходи в млн. лв.)

$$c(a, e) + c(b, f) + c(d, c) + c(e, c) + c(e, b) = 6 + 1 + 10 + 2 + 8 = 27 \text{ млн. лв.}$$

Покриващото дърво, състоящо се от ребрата (f, b), (b, c), (b, a), (a, d), (d, e), има тегло $1 + 7 + 5 + 4 + 3 = 20$, но и то не е *минимално покриващо дърво* — покриващо дърво с минимално тегло.

Ако теглата на ребрата в графа от пример 4 изразяваха не разходи, а печалба в млн. лв., тогава естествено е да се търси *максимално покриващо дърво* — покриващо дърво, чието тегло е не по-малко от теглото на което и да е покриващо дърво.

Интуицията в случая е добър съветник. Постъпва се така:

1. *Алгоритъм за търсене на минимално (максимално) покриващо дърво.*
Прилага се алгоритъмът за търсене на покриващо дърво, като ребрата се разглеждат по реда на нарастване (намаляване) на теглата им. Ако има ребра с еднакви

тегла, те се разглеждат в произволен ред.

Задача 4. *Да се намерят максималното и минималното покриващо дърво за графа от пример 4. Как алгоритъмът за търсене на минимално покриващо дърво може да се използва за търсене на максимално покриващо дърво и обратно?*

Упътване: Разгледайте граф с противоположни тегла на ребрата.

Обосновка на алгоритъма за намиране на максимално покриващо дърво. (Алгоритъмът за търсене на минимално покриващо дърво се обосновава напълно аналогично или като се използва задача 4.)

Да означим с $T_{алг.}$ дървото, което алгоритъмът построява, а с T_{max} — максималното покриващо дърво. Да допуснем, че тези две дървета се различават поне по едно ребро и нека $e_1 = (u, v)$ е първото от разглежданите в алгоритъма ребра, което е от $T_{алг.}$, но не принадлежи на T_{max}. Тъй като T_{max} е покриващо дърво, ще съществува единствен прост път $\rho(u, v)$, свързващ върховете u и v. Да добавим реброто e_1 към T_{max}. Тогава очевидно в T_{max} ще се появи цикъл, при това поне едно ребро на този цикъл няма да е от $T_{алг.}$ (в противен случай ще излезе, че в $T_{алг.}$ съществува цикъл, което е невъзможно). Да означим това ребро с e_2 и да изключим това ребро от T_{max}.

И така, в T_{max} включваме реброто e_1 и изключваме реброто e_2. Да означим с T^1_{max} полученото по този начин покриващо дърво. Тъй като T_{max} е максимално дърво, то

$$T^1_{max} \leq T_{max} \quad \text{(по определение)},$$

откъдето пък следва

$$\text{теглото } (e_1) \leq \text{теглото } (e_2). \tag{1}$$

От друга страна, реброто e_2 не може да е разглеждано в алгоритъма преди реброто e_1. Да допуснем противното, т.е. реброто e_2 е разглеждано в алгоритъма преди e_1. Тъй като e_2 не е включено в $T_{алг.}$, следва, че реброто e_2 е образувало цикъл с ребрата преди него, включени в $T_{алг.}$. Но тези ребра са и ребра на T_{max} (такива са всички ребра, разглеждани до e_1), следователно в T_{max} ще съществува цикъл, което е невъзможно. Отхвърляме допускането, т.е. e_2 е разглеждано в алгоритъма след реброто e_1. Тогава

$$\text{теглото } (e_1) \geq \text{теглото } (e_2). \tag{2}$$

От (1) и (2) следва

$$\text{теглото } (e_1) = \text{теглото } (e_2),$$

което означава, че

$$\text{теглото } (T^1_{max}) = \text{теглото } (T_{max}). \tag{3}$$

Освен това, общите ребра на T^1_{max} и $T_{алг.}$ са с единица повече от общите ребра между T_{max} и $T_{алг.}$.

Ако в качеството на максимално покриващо дърво сега разгледаме T^1_{max} (възможно е поради (3)) и повторим горните разсъждения, очевидно ще получим дърво T^2_{max}, общите ребра на което с $T_{алг.}$ ще са с единица повече, отколкото тези на T^1_{max} и $T_{алг.}$. Така след краен брой стъпки ще стигнем до максимално покриващо дърво T^k_{max}, което напълно съвпада с $T_{алг.}$.

Задача 5. *Формулирайте реални стопански и управленски проблеми, които се интерпретират и решават с помощта на покриващи дървета.*

ЛИТЕРАТУРА

[1] E. Minieka. Optimization Algorithms for Networks and Graphs. Marcel Deker, Inc., New York and Basel, 1978 (Майника, Э., 1983 Алгоритмы оптимизации на сетях и графах, М., „Мир“, 1981).
[2] Ив. Мирчев. Оптимиране. Университетско издателство „Неофит Рилски“, Благоевград, 1999.
[3] Б. Юруков. Технология за обучение по математическо оптимиране в средното училище, дисертация, София, 2001.

Борислав Петров Юруков
Югозападен университет
„Неофит Рилски“
бул. „Иван Михайлов“ 66
2700 Благоевград
e-mail: yurukov@aix.swu.bg

Иван Асенов Мирчев
Югозападен университет
„Неофит Рилски“
бул. „Иван Михайлов“ 66
2700 Благоевград
e-mail: mirchev@aix.swu.bg

SPANNING TREE ALGORITHM IN GRAPHS

Borislav P. Yurukov, Kostadin P. Kalinov, Ivan A. Mirchev

The authors continue to investigate the possibility for teaching some elements of mathematical optimization at upper secondary school level given in [3]. The approach accents on the applied character of Mathematics and Informatics. Many problems arising in such diverse fields as Psychology, Chemistry, Industrial and Electricial Engineering, Marketing and Education can be posed clearly and effectively in terms of the Graph theory.