

# Графи

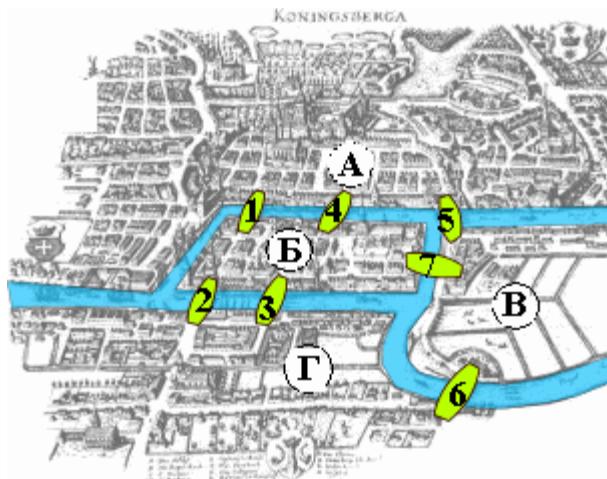
**Теорията на графиките** е клон от дискретната математика. От всички абстрактни математически структури, графиките са предпочитан формален модел за описание на реални процеси, обекти и системи. Графите са едни от най-полезните структури от данни в информатиката. Много задачи от различни области на науката и практиката могат да бъдат моделирани с граф и решени с помощта на съответен алгоритъм върху него.

От академична дисциплина теория на графиките се превръща в инструмент за формализация на широк кръг задачи от най-различни научни области. Освен в традиционните математически направления като топология, теория на групите и теория на вероятностите, графиките намират голямо приложение в теоретичната кибернетика, в теория на автоматите, изследване на операциите, математическо моделиране, теория на кодирането, теория на игрите, както и във физиката, биологията, историята, електротехниката, химията, икономиката, социологията, лингвистиката и др.

## 1. Въведение

Исторически първият проблем, който може да бъде отнесен към теория на графиките, е задачата за Кьонигсбергските мостове на *Ленард Ойлер* (1736 г.) (фигура 1): Гражданите на Кьонигсберг обичали много да се разхождат през

почивните дни из двата острова и свързващите ги 7 моста на река Прегел, а незайно кога и от кого възникнал въпросът *дали е възможно да се тръгне от една точка от сушата на града, да се мине по всичките 7 моста само по веднъж и да се стигне до началната точка*. На фигура 2 е показана опростена схема на мостовете и граф, представляващ модел на задачата. Мостовете са представени с ребрата, а двата бряга и островите в реката са върховете в граф.



*Фигура 1. Старинна карта на Кьонигсберг.*

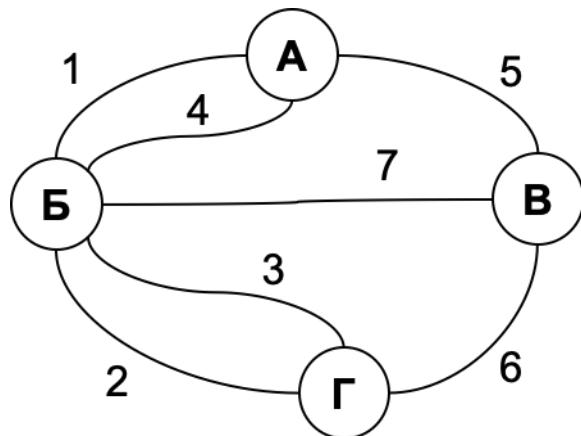
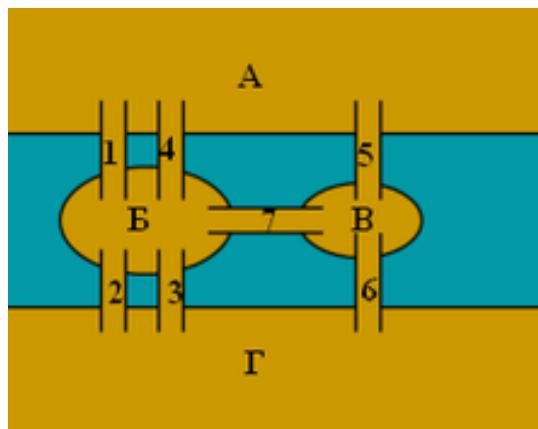
*Части на града: А. Алтищадт, Б. Кнайпхоф, В. Ломзе, Г. Форищат.*

*Мостове (хронологично): 1. Пазарен, 2. Зелен, 3. Работен, 4. Ковашки, 5. Дървен, 6. Висок, 7. Меден*

Леонард Ойлер доказва, че не е възможно, като се тръгне от някоя точка от сушата, да се премине през всичките седем моста само по един път, след което да се достигне до началната точка от сушата.

През 1859 г. Уилям Хамилтън (1805 – 1865) създава игра, при която трябва да се премине през всичките 20 върха на додекаедъра само по един път, след което да се достигне до първия връх. Додекаедърът е един от петте типа правилни изпъкнали многостени: състои се от 12 правилни петоъгълника, 30 ребра и 20 върха, във всеки от които се събират по 3 ребра (фигура 3а). (Други видове правилни изпъкнали многостени: тетраедър, куб, октаедър, додекаедър, икосаедър.) Хамилтън именувал всеки от 20те върха с име на град и формулирал следната забавна задача: *да се намери път, който тръгва от произволен град,*

минава през всички останали 19 града по един път и се връща в началния град.

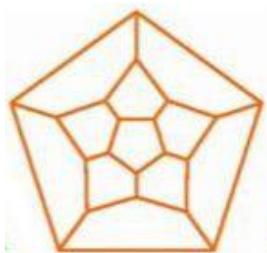


Опростена схема на мостовете

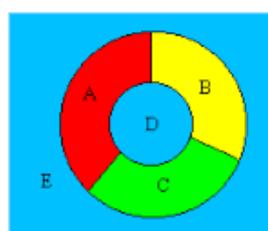
(За значението на буквите и цифрите  
вижте фигура 1)

Граф на Кьонигсбергските мостове

*Фигура 2. Модел на задачата във вид на граф.*

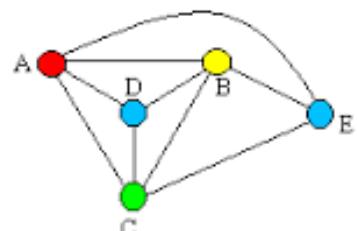


а



б

Додекаедър – равнинен  
вариант



в

Оцветяване на графи

Оцветяване на графи

*Фигура 3. Задачи, решавани с графи.*

Най-известната задача в теорията на графиките е **хипотезата за четирите цвята** (фигура 3б и 3в): всяка равнинна карта може да се оцвети с четири цвята така, че съседните области (имащи обща граница в повече от една точка) да са оцветени в различни цветове. За пръв път тази задача е формулирана от **Франсис Гътри** през 1850 г.

Тя е разпространявана и от учителя на Гътри, Август де Морган (1806 – 1871), за която той пише в писмо с дата 23 октомври 1852 г. до Уилям Хамилтон. Според някои източници задачата е била известна на Август Мьобиус (1790 – 1868) още през 1840 г. Хипотезата за четирите цвята става по-известна едва през 1878 г., когато Артър Кейли (1821 – 1895) я предлага на Лондонското математическо дружество. *Задачата е решена едва през 1976 г. от К.И. Апел и У. Хакен с помощта на цифрови изчислителни машини.* Това е задача от теорията на графиките, тъй като всяка област от картата може да се означи с кръгче, а с линии се съединяват кръгчетата, съответстващи на области, имащи обща граница в повече от една точка. Така, че се стига до задачата за оцветяване на върховете на планарен граф.

През 1936 г. психологът **Левин** е изказал предположението, че “жизненото пространство” на человека може да се представи с помощта на планарна карта, областите от която представляват различните видове дейности, които човек извършва на работа, вкъщи, неговото хоби. Така фактически той е работил с графи. Тази гледна точка е довела до друга психологическа интерпретация на графиките, при която хората се представят с кръгчета, а отношенията между тях – със свързващи ги линии.

Веригите на Марков в теорията на вероятностите се представят чрез ориентирани графи: събитията се означават с кръгчета, а чрез съединяващи ги насочени линии се указват възможните директни преходи между тях.

При проектирането на схеми, конструкции и програми с голяма сложност е по-удобно да се работи със структурните им модели във вид на графи. Преходът от реалния обект към съответния му граф се извършва като елементите на обекта се заменят с точки, а връзките между тях – с линии. Тези модели са много нагледни и са близки до реалния обект. Те позволяват да се отчитат най-съществените елементи и връзки, а също и да се получи оптимално решение на задачата на проектирането.

## 2. Определение и основни видове графи

Графът  $G = (V, U)$  е математически обект, представляващ съвкупност от две не пресичащи се, но свързани помежду си множества: **множество на върховете** и **множество на линиите**. Върховете и линиите или ребрата се наричат **елементи на графа**. Те се означават по следния начин:

$V = \{v_1, v_2, \dots, v_n\}$ ,  $|V| = n$  – множество на върховете (възлите, точките), в което броят на елементите е равен на  $n$ ;

$U = \{u_1, u_2, \dots, u_m\}$ ,  $|U| = m$  – множество на линиите (ребрата, дъгите) с брой елементи в него  $m$ .

Върховете се представят графично или изобразяват с точки или кръгчета и са свързани помежду си с линии или ребра, представени с отсечки.

Ако  $V = \{v_1, v_2, \dots, v_n\}$  и  $U = \{u_1, u_2, \dots, u_m\}$  са крайни множества, то  $G$  се нарича **краен граф**, в противен случай  $G$  се нарича безкраен. **Безкрайният граф** съдържа безкрайно множество върхове или безкрайно множество ребра.

Граф, за който всяка двойка върхове е съединена с ребро, се нарича **пълен граф**.

В общия случай:

$$U = \tilde{U} \cup \bar{U} \cup \overset{\circ}{U}$$

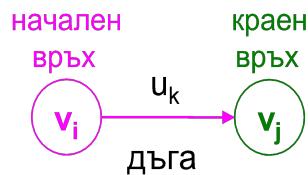
където:

$\tilde{U}$  – подмножество на неориентираните линии (ребрата), за които не е важна посоката на съединение на върховете. Всяко ребро  $u_k \in \tilde{U}$  се определя от ненаредената двойка върхове  $v_i, v_j$ , които то съединява и се записва  $u_k = (v_i, v_j)$  или  $u_k = (v_j, v_i)$  (фигура 4).



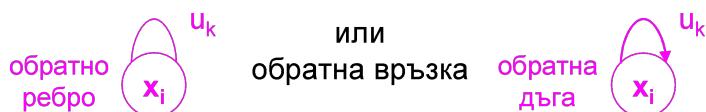
**Фигура 4. Ребра – неориентирани линии.**

$\bar{U}$  – подмножество на ориентираните линии (дъгите), за които е съществена посоката на съединение на върховете. Всяка дъга  $u_k \in \bar{U}$  се определя от наредената двойка върхове  $v_i, v_j$ , които тя съединява и се записва  $u_k = \langle v_i, v_j \rangle$ . При това  $\langle v_i, v_j \rangle \neq \langle v_j, v_i \rangle$  (фигура 5). Върхът  $v_i$  се нарича начален, а върхът  $v_j$  – краен за дъгата  $u_k = \langle v_i, v_j \rangle$ . Дъгата е входяща за крайния връх  $v_j$  и изходяща за началния връх  $v_i$ . Графично дъгите се изобразяват чрез стрелки, следващи посоката начален-краен връх.



**Фигура 5. Ориентирани линии – дъги.**

$\overset{\circ}{U}$  – подмножество на обратните връзки (ребра/дъги), имащи за начало и за край един и същ връх. Всяка линия  $u_k \in \overset{\circ}{U}$  може да се зададе чрез ненаредена или чрез наредена двойка:  $u_k = (v_i, v_i)$  или  $u_k = \langle v_i, v_i \rangle$  (фигура 6).



**Фигура 6. Обратни ребра и дъги.**

Ако  $U = \tilde{U} \cup \bar{U} \cup \overset{\circ}{U}$  и  $\bar{U} \neq \emptyset$ ,  $\tilde{U} \neq \emptyset$  и  $\overset{\circ}{U} \neq \emptyset$ , то графът се нарича **смесен** (фигура 7).

За него:

Мощност, т.е. брой на елементите на съответното множество:

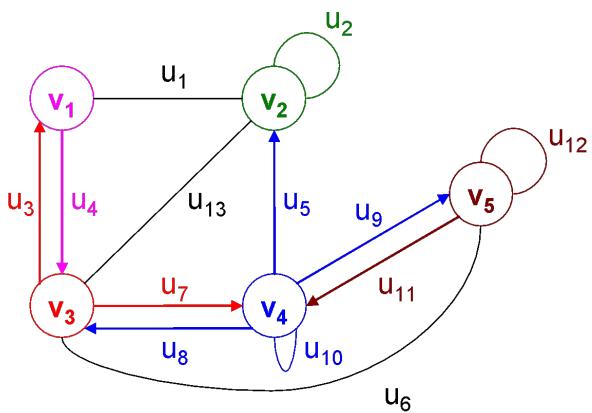
$|V| = 5$  – брой върхове;  $|U| = 13$  – брой линии/ребра

$$\tilde{U} = \{u_1, u_6, u_{13}\}$$

$$\bar{U} = \{u_3, u_4, u_5, u_7, u_8, u_9, u_{11}\}$$

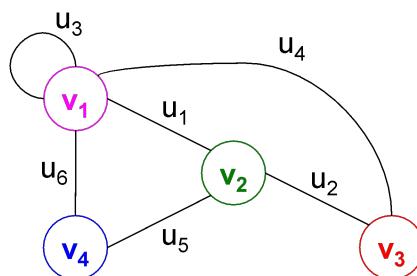
$$\overset{\circ}{U} = \{u_2, u_{10}, u_{12}\}$$

$$U = \tilde{U} \cup \bar{U} \cup \overset{\circ}{U}$$



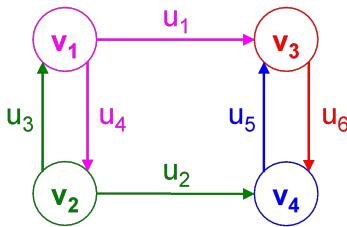
**Фигура 7. Смесен граф.**

Ако  $U = \tilde{U}$ , т.е.  $\tilde{U} \neq \Phi$ ,  $\bar{U} = \Phi$ , то графът се нарича **неориентиран** или **неорграф**, и се бележи с  $G$  (фигура 8).



**Фигура 8. Неориентиран граф.**

Ако  $U = \bar{U}$ , т.е.  $\tilde{U} = \Phi$ ,  $\bar{U} \neq \Phi$ , то графът се нарича ориентиран или орграф, и се бележи с  $D$  (фигура 9).



*Фигура 9. Ориентиран граф.*

Подмножеството на обратните връзки може да се разглежда като подмножество от неориентирани или от ориентирани линии. Обикновено графите с обратни връзки се дефинират специално.

Всеки *неориентиран граф*  $G$  може да се трансформира в *ориентиран граф*  $D$  чрез замяна на всяко ребро с две противоположно насочени дъги. Затова много резултати за неориентиранныте графи са валидни и за ориентиранныте.

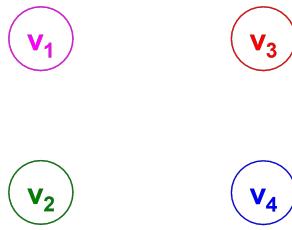
Ако едно ребро  $u_k$  съединява върховете  $v_i$  и  $v_j$  казваме, че реброто  $u_k$  и върховете  $v_i$  и  $v_j$  са *инцидентни*. *Инцидентността* е отношение между реброто или дъгата и неговите крайни върхове. Реброто  $u_k$  е инцидентно на върховете  $v_i$  и  $v_j$  и върховете  $v_i$  и  $v_j$  са инцидентни на реброто  $u_k$ . Ако даден връх няма инцидентни ребра или дъги той се нарича *изолиран*.

Върховете, които са свързани с общо ребро или дъга се наричат *съседни*. Ребрата или дъгите, имащи общ връх също се наричат съседни.

*Равнинен или планарен граф* се нарича този граф, който може да бъде изобразен в равнината така, че никои две ребра да нямат общи точки, освен инцидентния им връх.

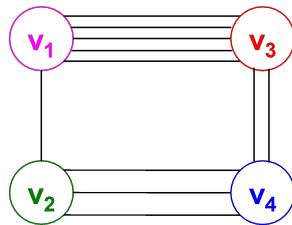
Две ребра, свързващи една и съща двойка върхове се наричат *паралелни или кратни*. В ориентирания граф, дъгите, които имат общ начален и краен връх се наричат кратни дъги.

*Нулев граф*, за който  $V \neq \emptyset$ ,  $U = \emptyset$  (фигура 10). Върховете му се наричат изолирани.



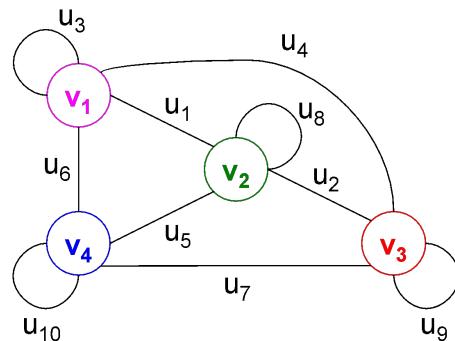
**Фигура 10. Нулев граф.**

Неориентиран граф с кратни ребра се нарича **мултиграф** (фигура 11). Поне една двойка върхове  $v_i, v_j$  е свързана с  $m = 2, 3, \dots$  ребра. Най-голямото  $m$  се нарича **мултичисло** на графа.



**Фигура 11. Мултиграф.**

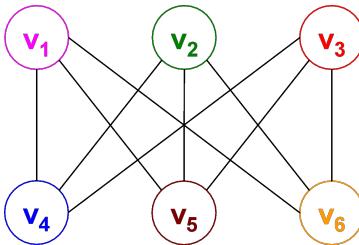
Граф, в който между всяка двойка върхове  $v_i, v_j$ , има ребро  $u_k$  се нарича пълен граф (фигура 12).



**Фигура 12. Пълен граф.**

Ако върховете на даден граф могат да бъдат разделени на две не пресичащи се подмножества:  $V' \cap V'' = \emptyset$ , при което краищата на всяко ребро  $(v_i, v_j)$  принадлежат на различни подмножества:  $v_i \in V', v_j \in V''$ , то този граф се нарича

**двуетажен или двудолен граф.** На фигура 13 е показан пълният двуетажен граф  $K_{3,3}$  (на Куратовски).



**Фигура 13. Пълен двуетажен граф  $K_{3,3}$ .**

### 3. Свързаност на графи

**Маршрут (път)**  $S$  в графа  $G$  се нарича крайна последователност от ребра:

$$S = (v_0, v_1), (v_1, v_2), \dots, (v_{i-1}, v_i),$$

където  $v_0$  е началният, а  $v_i$  е крайният връх на маршрута. Броят на ребрата в маршрута  $S$  се нарича негова дължина. В крайния граф  $G$  (за когото  $V$  и  $U$  са крайни множества) могат да се отбележат само краен брой маршрути.

Маршрут, в който няма повтарящи се ребра, се нарича **верига**. Затворена верига, в която  $v_0 = v_i$ , се нарича цикъл. Веригите и циклите се наричат **прости**, ако не съдържат повтарящи се върхове, освен първия и последния в случай на цикъл. В противен случай се наричат **сложни (съставни)**.

**Пример:** За графът от фигура 14 могат да се определят:

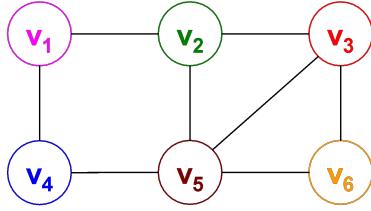
$$S_1 = (v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_2), (v_2, v_1), (v_1, v_4) - \text{маршрут}$$

$$S_2 = (v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_2) - \text{сложна верига}$$

$$S_3 = (v_1, v_2), (v_2, v_3), (v_3, v_5), (v_5, v_6) - \text{проста верига}$$

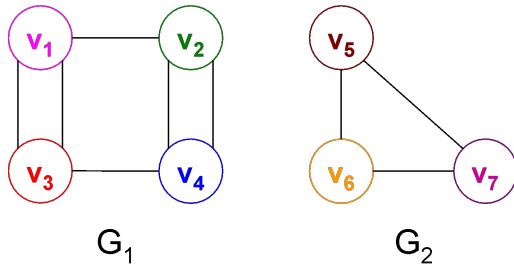
$$S_4 = (v_5, v_2), (v_2, v_3), (v_3, v_6), (v_6, v_5) - \text{прост цикъл}$$

$$S_5 = (v_1, v_4), (v_4, v_5), (v_5, v_6), (v_6, v_3), (v_3, v_5), (v_5, v_2), (v_2, v_1) - \text{сложен цикъл}$$



**Фигура 14.** Примерен граф за определяне на верига, маршрут и цикъл.

Два върха  $v_i, v_j \in V$  на графа  $G$  се наричат **свързани**, ако съществува такъв маршрут  $S$ , за когото върховете  $v_i$  и  $v_j$  са крайни. Графът  $G$  се нарича **свързан**, ако всеки два негови върха са свързани, в противен случай  $G$  е **несвързан** и всеки от съответстващите му подграфи  $G_1, G_2, \dots, G_i$  се нарича **компонента на свързаност**. Показаният на фигура 15 граф е несвързан и  $G_1$  и  $G_2$  са двете му компоненти. Казваме, че това е двукомпонентен несвързан граф.



**Фигура 15.** Пример за двукомпонентен несвързан граф.

В **ориентирания** граф  $D = (V, U)$  маршрутът представлява редуваща се последователност от върхове и дъги от вида:

$$S = (v_0, u_1, v_1), (v_1, u_2, v_2), \dots, (v_{n-1}, u_n, v_n),$$

в която всяка дъга  $u_i = \langle v_{i-1}, v_i \rangle$ .

**Път** в ориентиран  $D$  граф представлява маршрут, в който всичките върхове са различни. **Контур** в  $D$  граф представлява затворен маршрут, в който всичките върхове са различни с изключение на първия и последния. Ако съществува път от  $v_i$  до  $v_j$ , се казва, че  $v_j$  е **достижим** от  $v_i$ . Графът се нарича **силно свързан**, ако всеки два негови върхове са взаимно-достижими.

Броят на ребрата, свързани (инцидентни) на върха  $v_i \in G$ , се нарича валентност или локална степен на този връх и се означава с  $\rho(v_i)$ . Ако  $\rho(v_i)=1$ , то върхът  $v_i$  се нарича висящ. Ако  $\rho(v_i)=0$ , то върхът  $v_i$  се нарича изолиран.

**Регулярен** граф е граф, за който степените на всички върхове са равни. Регулярен граф от степен 0 е празен граф. Регулярен граф от степен n представлява n-мерен куб.

**Лема за ръкостисканията** (Handshake's lemma): Ако G е неориентиран граф и  $V = \{v_1, v_2, \dots, v_n\}$ , то сумата от степените на всички върхове на графа е четно число, равно на удвоения брой на ребрата.

Наименованието на лемата е свързано с аналогията при ръкостисканията, чийто брой е двойно по-голям от броя на участващите хора. В сила е за мулти и псевдографи. Върху нея се базира и доказателството на следната **теорема**:

**Във всеки неориентиран граф броят на върховете с нечетна степен е четен.**

**Доказателство:** Нека S е сумата от степените на върховете с четна степен, а T е сумата от степените на върховете с нечетна степен. Очевидно S е четно, тъй като сумата на произволен брой четни числа също е четно число. Съгласно лемата за ръкостисканията, сумата:

$$S + T = 2|E|$$

Е също е четно, което е изпълнено само при условие, че T е четно число. Т е сума от нечетни числа и може да бъде четно само ако броят им е четен.

Друга **теорема** за графи, която се използва е следната: **Във всеки граф с  $n \geq 2$  върха, съществуват поне два с еднаква степен.**

**Доказателство:** Базира се на принципа на Дирихле за чекмеджетата. В граф с  $n \geq 2$  върха минималната степен на връх е 0, а максималната  $n - 1$ .

Допускаме, че всички върхове

$$\{1, 2, 3, \dots, n\}$$

имат различни степени

$$0, 1, 2, \dots n-1.$$

Но ако има връх със степен 0 (не свързан с никой от останалите върхове), не може да има връх от степен n-1 (свързан чрез ребра с всички останали). Съгласно принципа на Дирихле, поне два върха имат еднаква степен.

За графа  $G = (V, U)$  с брой върхове  $|V| = n$  и брой ребра  $|U| = m$ , е в сила равенството:

$$m = \frac{1}{2} \sum_{i=1}^n \rho(v_i).$$

Ако в G има обратни ребра, то всяко от тях се отчита по два пъти.

За неориентирания граф G от фигура 8. имаме:

$$\rho(v_1) = 5, \rho(v_2) = 3, \rho(v_3) = 2, \rho(v_4) = 2,$$

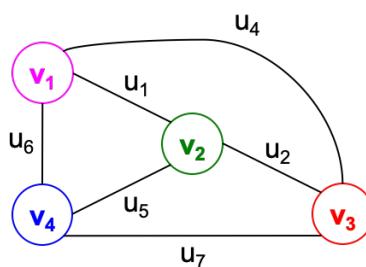
$$m = \frac{5 + 3 + 2 + 2}{2} = 6.$$

За пълния граф  $K_n$ , без обратни ребра, имаме:

$$\rho(v_1) = \rho(v_2) = \dots = \rho(v_n) = n - 1 \text{ и}$$

$$m = \frac{n(n-1)}{2}.$$

За примерния граф от фигура 16 следва, че при  $n = 4$ ,  $m = \frac{4 \cdot 3}{2} = 6$ .



*Фигура 16. Пълен граф.*

В  $D$  графа дъгата  $u_k = \langle v_i, v_j \rangle$  се счита за **положително инцидентна** на нейния краен връх  $v_j$ . Броят на дъгите, положително инцидентни на върха  $v_j$ , се нарича **полустепен на входа** и се означава  $\rho^+(v_j)$  (на практика това е броят на дъгите, влизащи в този връх).

Дъгата  $u_k = \langle v_i, v_j \rangle$  се счита за **отрицателно инцидентна** на нейния начален връх  $v_i$ . Броят на дъгите, отрицателно инцидентни на върха  $v_i$ , се нарича **полустепен на изхода** и се означава  $\rho^-(v_i)$  (това е броят на дъгите, излизащи от този връх).

Очевидно  $\rho(v_j) = \rho^+(v_j) + \rho^-(v_j)$ , ако дъгите се заменят с ребра (от  $D$  се премине към  $G$ ).

Тъй като всяка дъга е положително инцидентна само на един връх ( $v_j$ ) и е отрицателно инцидентна само на един връх ( $v_i$ ), то

$$|U| = m = \sum_{v_j \in V} \rho^+(v_j) = \sum_{v_i \in V} \rho^-(v_i).$$

Подмножеството от върховете, представляващи начала на дъгите към върха  $v_j$ , се означава  $\Gamma^+(v_j)$ . Подмножеството от върховете, представляващи краища на дъгите, имащи за начало върха  $v_j$ , се означава  $\Gamma^-(v_j)$ .

За ориентирания граф  $D$  от фигура 9 можем да определим полустепените на входа и изхода за всеки връх и съответно множествата  $\Gamma^+(v_j)$  и  $\Gamma^-(v_j)$ :

$$\rho^+(v_1) = 1$$

$$\rho^-(v_1) = 2$$

$$\Gamma^+(v_1) = \{v_2\}$$

$$\Gamma^-(v_1) = \{v_2, v_3\}$$

$$\rho^+(v_2) = 1$$

$$\rho^-(v_2) = 2$$

$$\Gamma^+(v_2) = \{v_1\}$$

$$\Gamma^-(v_2) = \{v_1, v_4\}$$

$$\rho^+(v_3) = 2$$

$$\rho^-(v_3) = 1$$

$$\Gamma^+(v_3) = \{v_1, v_4\}$$

$$\Gamma^-(v_3) = \{v_4\}$$

$$\rho^+(v_4) = 2$$

$$\rho^-(v_4) = 1$$

$$\Gamma^+(v_4) = \{v_2, v_3\}$$

$$\Gamma^-(v_4) = \{v_3\}$$

#### 4. Ойлерови и Хамилтонови графи

Свързаният граф се нарича **Ойлеров**, ако съществува затворена верига (цикъл), минаваща през всяко негово ребро (дъга) само по един път.

**Критерий** за съществуване на Ойлеров цикъл  $C_o$ : Неориентираният граф G има Ойлеров цикъл, ако е свързан и локалните степени на всичките негови върхове са четни числа. Ориентираният граф D има Ойлеров цикъл, ако е свързан и за всеки връх  $v_i$  полу степените на входа и изхода са равни  $\rho^+(v_i) = \rho^-(v_i)$ .

Свързаният граф се нарича **Хамилтонов**, ако съществува цикъл, минаващ през всеки негов връх само по един път. **Общ критерий за съществуването на Хамилтонов** цикъл,  $C_h$ , не е известен.

Пример: За показания на фигура 17 граф определяме степените на върховете:

$$\rho(v_1) = \rho(v_3) = \rho(v_5) = 2,$$

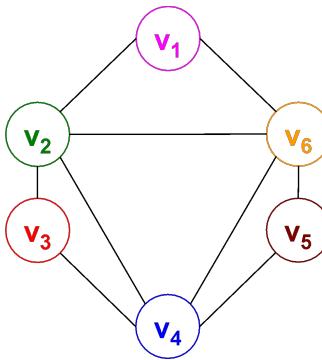
$$\rho(v_2) = \rho(v_4) = \rho(v_6) = 4.$$

Следователно има Ойлеров цикъл  $C_o$ :

$$C_o = (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_2), (v_2, v_6), (v_6, v_4), (v_4, v_5), (v_5, v_6), (v_6, v_1)$$

Има и друг Ойлеров цикъл.

$$C_o = (v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6), (v_6, v_1).$$



**Фигура 17. Примерен граф за намиране на Ойлеров и Хамилтонов цикли.**

**Претеглен граф** е този граф, в който на всяко ребро (дъга) е съпоставено тегло. Теглата се използват за задаване на дължината на пътя между два върха, времето за преход от един връх в друг, цена на билета, разход на гориво и други числови характеристики на реалните задачи. При графично изобразяване на графи, теглата се изписват над съответните им ребра или дъги.

В много случаи, при изследване на структурата на графиките, имената или номерата на върховете нямат значение. Такива графи, които се получават чрез преименуване на върховете, е удобно да се смятат еднакви. За такива графи се отнася и понятието **изоморфизъм**.

**Изоморфизъм** между графи се нарича взаимно еднозначното съответствие (биекция)  $V(G) \rightarrow V(H)$  между множеството от върхове на графа  $G$  и множеството от върхове на графа  $H$ , което запазва отношението на съседство между върховете (образите на съседни върхове са съседни и обратно). Отношението на изоморфизъм между графи е отношение на **еквивалентност** и притежава свойствата симетричност, **транзитивност и рефлексивност**.

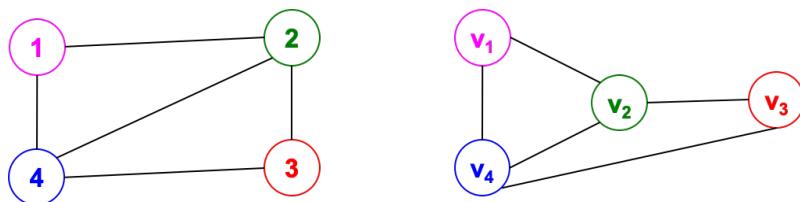
Два графа се наричат изоморфни, ако между тях може да се установи изоморфизъм. Задачата за проверка на изоморфността на два графа е NP-пълна задача.

Изоморфните графи са еквивалентни, различават се само по имената (етикетите) на върховете. Много практически задачи като информационно

търсене, определяне структурата на химически съединения и др. се свеждат до установяване на изоморфизъм между графи.

За двата графа на фигура 18 един възможен изоморфизъм е изображението  $f$ , зададено таблично по следния начин:

u	1	2	3	4
$f(u)$	$v_1$	$v_2$	$v_3$	$v_4$



*Фигура 18. Изоморфизъм между графи.*

## 5. Дървета (ациклични графи)

**Дървата**, като структура за моделиране, са частен случай на графиките. Те са специален вид графи с йерархична организация на свързване на върховете и се използват много често за представяне на реални обекти, в които структурните свойства се изразяват във функционалната връзка от типа 1:n, т.е. всеки връх е свързан с n върха, които от своя страна могат да бъдат свързани с n върха и т.н. Ребрата (дъгите) на графа служат за представяне на връзките между върховете. Ако ребрата са ориентирани (дъги), дървото се нарича ориентирано дърво.

Реални йерархични обекти в нашето ежедневие, които се представлят с дърводидна структура са административните системи, родословното дърво, различни информационни структури, дървата се използват в операционните системи, компилаторите, системите за управление на бази данни, алгоритмите и много други компютърни приложения.

Съществуват множество определения за дефиниране на дърво.

Често се използва следното рекурсивно определение за дърво, дадено от американския математик Доналд Кнут, наричан баща на алгоритмите, заради приносите си към изследването и систематизирането на алгоритмите и тяхната изчислителна сложност.

**Дървото**  $T$  е множество от върхове, което е или празното множество, или:

- Съдържа единствен елемент  $t$ , наречен корен на дървото, който обикновено се означава с  $\text{root}(T)$ .
- Останалите елементи (с изключение на корена) са разпределени в  $m \geq 0$  не пресичащи се множества  $T_1, T_2, \dots, T_m$ , всяко от които е дърво. Множествата  $T_1, T_2, \dots, T_m$  се наричат **под дървета** на  $T$ . Корените  $t_1$  на  $T_1$ ,  $t_2$  на  $T_2, \dots, t_m$  на  $T_m$  се наричат наследници на  $t$ .

Всеки връх на  $T$  е корен на някакво под дърво. Броят на под дърветата с корен върха  $t$  се нарича **степен на върха**  $t$ . Ако степента на върха  $t$  е 0, той се нарича **лист**.

Други определения:

Дърво  $T = (V, U)$  се нарича свързан неориентиран граф без цикли. За  $T$  са еквивалентни твърденията:

- а) произволни два върха в  $T$  са свързани с единствена пристапна верига;
- б) графът е свързан и има  $n - 1$  ребра (при  $n$  върха);
- в) графът не съдържа цикли и има  $n - 1$  ребра
- г) графът не съдържа цикли, но добавянето на ребро между два произволни несъседни върха води до появата на един цикъл;
- д) графът е свързан, но загубва това свойство след отстраняването на произволно ребро.

Всяко от тези твърдения може да служи за определение за дърво.

Клон към върха  $v_i$  на дървото  $T$  се нарича максималното под дърво, съдържащо  $v_i$  като висящ връх. Така, броят на клоните към  $v_i$  е равен на

локалната му степен  $\rho(v_i)$ . Тегло на върха  $v_i$  е най-големият брой ребра в клон към  $v_i$ . Теглото на всеки висящ връх е равно на броя на ребрата на дървото.

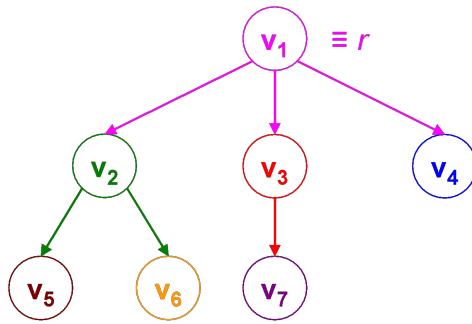
Височина на връх в дървото се задава чрез максималния брой ребра (дъги), свързващи дадения връх с листата. Височина на дървото е височината на неговия корен.

Броят на поддърветата на всеки връх може да бъде различен, в съответствие с което различаваме двоични, троични,  $n$ -арни дървета. Ако степента на всеки връх от дървото е по-малка или равна на две, то дървото се нарича двоично или бинарно дърво. Тъй като поддърветата на всеки връх  $t$  са най-много две, те се наричат ляво поддърво и дясно поддърво на  $t$ , а корените на лявото и дясното поддърво се наричат ляв и десен наследник на  $t$ . Регулярни са дърветата в които степените на върховете съвпадат.

**Ориентирано коренно дърво**  $T(r)$  или ордърво с корен  $r$  се нарича ориентиран граф със специален връх  $r$ , удовлетворяващ следните три условия (фигура 19):

- a)  $T(r)$  е дърво, ако се взема предвид ориентацията на дъгите;
- б) от корена  $r$  е достижим всеки връх (или коренът  $r$  е достижим от всеки връх);
- в) в корена  $r$  не влиза нито една дъга (или от корена  $r$  не излиза нито една дъга).

Във всеки свързан граф  $G$  може да се отдели произволно дърво  $T$ . Дървета, в които броят на върховете е равен на броя на върховете на графа, от който те са отделени, се наричат **покриващи дървета**.



**Фигура 19. Ориентирано дърво.**

За всеки свързан граф може да се получи някакво множество от покриващи дървета. Така, за пълния граф  $K_n$  броят на покриващите дървета е равен на  $n^{n-2}$  (теорема на Кейли). Множеството от дърветата за графа се нарича **гора**. Най-малкият брой ребра, които трябва да се отстраният от графа, за да се превърне той в дърво, се нарича **цикломатично число** на графа. За графа  $G = (V, U)$  с брой върхове  $|V| = n$  и брой ребра  $|U| = m$ , цикломатичното число е:

$$\gamma(G) = m - n + k,$$

където  $k$  е компонентата на свързаност на графа. За свързан граф (имащ една компонента на свързаност, т.е.  $k = 1$ ):

$$\gamma(G) = m - n + 1.$$

## 6. Представяне на графиките в паметта на компютъра

Представянето на графи в оперативната памет на компютрите при разработване на програми предполага съхранение на цялата информация за структурата на графа. Изборът на един или друг начин за представяне на графи зависи от конкретната задача и влияе съществено върху ефективността на използваните алгоритми и алгоритмичен език за програмиране. За целта се използват матрици на съседство, матрици на инцидентност, списъци на съседство, списъци на ребрата и др.

а) **матрица на съседство** – тя е най-използвана:

Това е квадратна матрица  $\mathbf{A} = \left\| a_{ij} \right\|_{n \times n}$ , където:

**За неорграф  $G$ :**  $a_{ij} = 1$ , ако съществува ребро  $(v_i, v_j)$ , т.е.  $v_i$  и  $v_j$  са съседни;  $a_{ij} = 0$  в противен случай.

**За мултиграф:**  $a_{ij} = m$ , ако  $v_i$  е свързан с  $v_j$  чрез  $m$  ребра;  $a_{ij} = 0$  в противен случай.

**За орграф  $D$ :**  $a_{ij} = 1$ , ако съществува дъга  $\langle v_i, v_j \rangle$ ;  $a_{ij} = 0$  в противен случай.

Обратните ребра/дъги в графа  $G/D$  се отразяват с елементите  $a_{ij}$ , разположени по главния диагонал на матрицата  $\mathbf{A}$ .

За неорграфа  $G$  от фигура 8 имаме:

$$\mathbf{A}(G) = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{vmatrix},$$

$$\rho(v_1) = 4 + 1 = 5,$$

$$\rho(v_2) = 3,$$

$$\rho(v_3) = 2,$$

$$\rho(v_4) = 2$$

$$m = \frac{1}{2} \sum_{i=1}^4 \rho(v_i) = \frac{4+1+3+2+2}{2} = 6$$

За неорграф  $G$ ,  $a_{ij} = a_{ji}$ , поради което матрицата  $\mathbf{A}(G)$  е **симетрична** относно главния си диагонал и в паметта е достатъчно да се съхрани само половината от нея. За  $G$  граф броят на единиците в  $i$ -тия ред на  $\mathbf{A}(G)$  определя локалната степен

на върха  $v_i$   $\rho(v_i)$ , като единица, разположена върху главния диагонал, се отчита два пъти.

За орграфа  $D$  от фигура 9 имаме:

$$A(D) = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{vmatrix},$$

$$\rho^-(v_1) = 2 \quad \rho^-(v_2) = 2 \quad \rho^-(v_3) = 1 \quad \rho^-(v_4) = 1$$

$$\rho^+(v_1) = 1 \quad \rho^+(v_2) = 1 \quad \rho^+(v_3) = 2 \quad \rho^+(v_4) = 2$$

$$m = \sum_{i=1}^4 \rho^-(v_i) = \sum_{i=1}^4 \rho^+(v_i) = 2 + 2 + 1 + 1 = 1 + 1 + 2 + 2 = 6.$$

За орграф  $D$ , в общия случай  $a_{ij} \neq a_{ji}$ , поради което матрицата  $A(D)$  не е симетрична относно главния си диагонал. За  $D$  граф, броят на единиците в  $i$ -тия ред на  $A(D)$  е равен на  $\rho^-(v_i)$ , а броят на единиците в  $j$ -тия стълб на  $A(D)$  е равен на  $\rho^+(v_j)$ . Ако от  $v_i$  не излизат дъги, то в  $i$ -тия ред на  $A(D)$  има само нули (върхът  $v_i$  е изход); ако във  $v_j$  не влизат дъги, то в  $j$ -тия стълб на  $A(D)$  има само нули (върхът  $v_j$  е вход). Ако в  $D$  графа има обратна дъга, то съответният елемент  $a_{ii}$  от главния диагонал на  $A(D)$  е равен на единица, която един път се отчита при определянето на  $\rho^-(v_i)$  и втори път се отчита при определянето на  $\rho^+(v_i)$ .

Когато елементите  $a_{ij}$  са равни на теглата на съответните ребра/дъги (представляващи обикновено дълчините на тези линии), то  $A$  се нарича **претеглена матрица на съседство**.

Матрицата на съседство е удобна за ръчни изчисления; някои стандартни алгоритми работят директно с  $A$ ; тя представя компактно графи с голям брой ребра/дъги (особено ако се работи с двоични битове в машинната дума, както допуска, например, езикът ПЛ/1); за граф  $G$  е достатъчно да се съхранява само една триъгълна матрица; удобна е за представяне на мултиграфи и на претеглени

графи. Основният недостатък при работа с  $A$  е големият разход на памет (използваният обем памет е пропорционален на  $|V|^2 = n^2$ ) дори и при графи с малък брой ребра ( $A$  се получава силно разредена, т.е. имаща голям брой нули); освен това не може да се намали трудоемкостта на алгоритмите, когато тя е пропорционална на броя на ребрата, а не на броя на върховете.

**б) матрица на инцидентност** – това е правоъгълна матрица

$$\mathbf{I} = \left\| i_{kl} \right\|_{n \times m}, \text{ където:}$$

за **неорграф**  $G$ :

$i_{kl} = 1$ , ако върхът  $v_k$  е инцидентен на реброто  $u_l$ ;

$i_{kl} = 0$  в противен случай.

за **орграф**  $D$ :

$i_{kl} = 1$ , ако върхът  $v_k$  е начален за дъгата  $u_l$ ;

$i_{kl} = -1$ , ако върхът  $v_k$  е краен за дъгата  $u_l$ ;

$i_{kl} = 0$ , ако върхът  $v_k$  не е инцидентен на реброто  $u_l$ .

При наличие на обратни дъги в  $D$ , последното от горните три условия може да се до дефинира:  $i_{kl} = 0$ , ако върхът  $v_k$  не е инцидентен на реброто  $u_l$  или  $u_l$  е обратна дъга. Освен това матрицата на инцидентност може да се раздели на две под матрици: положителна и отрицателна.

За хиперграф  $H$ :

$i_{kl} = 1$ , ако  $v_k \in e_l$  (върхът  $v_k$  принадлежи на реброто  $e_l$ );

$i_{kl} = 0$  в противен случай.

За **неорграфа**  $G$  от фигура 8 имаме:

$$I(G) = \begin{vmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{vmatrix}$$

Редовете на матрицата  $I(G)$  съответстват на върховете на графа  $G$ , стълбовете – на ребрата, а елементът  $u_{kl}$  указва инцидентността на върха  $v_k$  и реброто  $u_l$ .

Във всеки стълб на  $I(G)$  се разположени по две единици, тъй като всяко ребро съединява точно по два върха. Ако в графа  $G$  има обратно ребро, то в съответния стълб на матрицата  $I(G)$  има само една единица.

За органа  $D$  от фигура 9 имаме:

$$I(G) = \begin{vmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

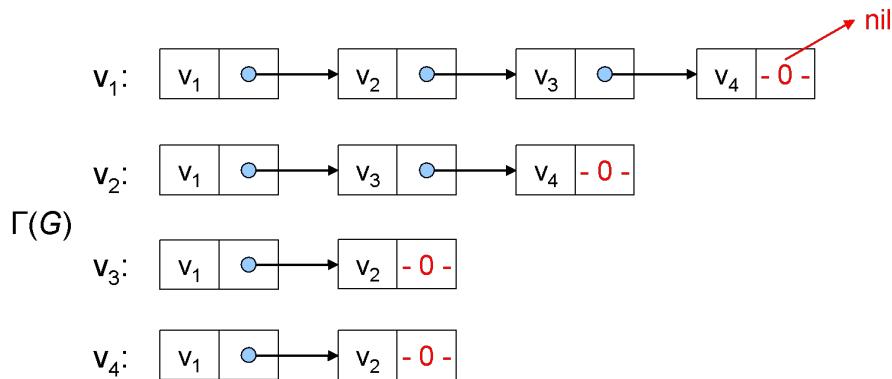
В случая това е само положителната под матрица. Поради това само по един от елементите във всеки стълб е 1. Аналогично, само по един от елементите във всеки стълб на отрицателната под матрица е – 1.

Матрицата на инцидентност предоставя най-удобния класически начин за задаване на хиперграфи. Тя, обаче, се използва много рядко поради голямата си размерност (използваният обем памет е пропорционален на  $|V||U| = n.m$ ) и поради практическото отсъствие на алгоритми, работещи директно с  $I$ .

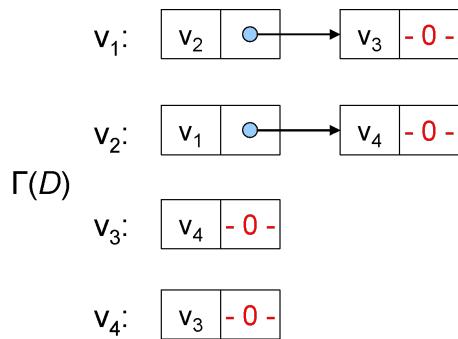
**в) списъци на съседство** – те са главната алтернатива на представянето чрез матрица на съседство. Графът се представя с помощта на  $|V|=n$  списъка на съседство, т.е. за всеки връх се съставя по един списък. В случай на  $G$  граф списъкът за върха  $v_i$  е списък на съседните за  $v_i$  върхове. В случай на  $D$  граф списъкът за върха  $v_i$  е списък на върховете от множеството  $\Gamma^-(v_i)$ , т.е. списък на върховете, които са краища на дъгите, излизящи от върха  $v_i$ . Ако броят на дъгите

в D граф е съществено по-малък в сравнение с пълния граф, то представянето чрез списъци на съседство е много ефективно. В случай на D граф използваният обем памет е пропорционален на  $|V| + |U| = n + m$ , а в случай на G граф – на  $|V| + 2|U| = n + 2m$ . Списъците на съседство лесно се реализират чрез динамични списъчни структури. Списъците на съседство не са много удобни за представяне на претеглени графи, понеже възниква необходимостта някъде другаде да се съхраняват теглата на дъгите и да се установи съответствието между ребрата/дъгите и техните тегла.

За неорграфа G от фигура 8 множеството от списъци е:



За неорграфа D от фигура 9 множеството от списъци е:



## 7. Задачи, решавани с графи

### Множество и матрица на достижимост и обратна достижимост

**Матрица на достижимост (за ориентиран граф)** се определя съгласно следното правило:

$$R = \|r_{ij}\|_{n \times n}$$

$r_{ij} = 1$  ако върхът  $v_i$  е достижен от върха  $v_j$

$r_{ij} = 0$  в противен случай

### Матрица на обратна достиженост

$$Q = \|q_{ij}\|_{n \times n}$$

$q_{ij} = 1$  ако върхът  $v_j$  е достижен от върха  $v_i$

$q_{ij} = 0$  в противен случай

От определението за матрица на обратна достиженост е очевидно, че стълбът  $v_i$  в матрицата  $Q$  съвпада с реда  $v_i$  в матрицата  $R$ . Т.е. матрицата на обратна достиженост е транспонираната матрица на достиженост.

### Основни стъпки в алгоритъма:

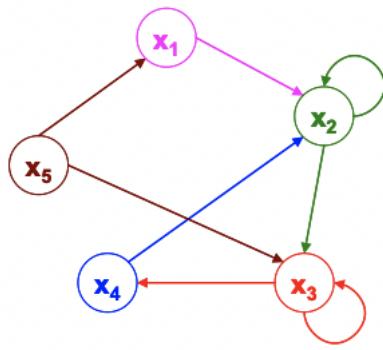
- 1) Определя се матрицата на съседство за графа.
- 2) Определят се  $\Gamma^{-1}(v_i)$  – множеството от върхове, непосредствено достижили от върха  $v_i$ .
- 3) Определят се множествата на достиженост  $R(v_i)$ :

$$R(v_i) = \{v_i\} \cup \Gamma^{-1}(v_i) \cup \Gamma^{-2}(v_i) \cup \dots \cup \Gamma^{-p}(v_i)$$

където  $\Gamma^{-p}(v_i)$  е множеството от върхове, които са достижели от  $v_i$  на разстояние  $p$  на брой дъги.

- 4) Матрицата на достиженост се определя, като за всеки връх се определя множеството от достижели върхове. В съответния ред за този връх поставяме единици в колоните, съответстващи на достижели от този връх върхове и нули в останалите.

Например: За показания граф на фигура 20 да се определи матрицата на достиженост и матрицата на обратна достиженост



**Фигура 20.** Ориентиран граф за илюстрация на алгоритмите за определяне на матрица на достижимост и обратна достижимост.

1. Определяме матрицата на съседство:

$$A = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{vmatrix}$$

2. Определяме  $\Gamma^{-1}(x_i)$  – множеството от върхове, непосредствено

достижими от върха  $x_i$ .

$$\Gamma^{-1}(x_1) = \{x_2\},$$

$$\Gamma^{-1}(x_2) = \{x_2, x_3\},$$

$$\Gamma^{-1}(x_3) = \{x_3, x_4\},$$

$$\Gamma^{-1}(x_4) = \{x_2\},$$

$$\Gamma^{-1}(x_5) = \{x_1, x_3\}.$$

3. Определят се множествата на достижимост  $R(x_i)$ :

$$R(x_i) = \{x_i\} \cup \Gamma^{-1}(x_i) \cup \Gamma^{-2}(x_i) \cup \dots \cup \Gamma^{-p}(x_i).$$

$$R(x_1) = \{x_1\} \cup \{x_2\} \cup \{x_2, x_3\} \cup \{x_2, x_3, x_4\} \cup \{x_2, x_3, x_4\} = \{x_1, x_2, x_3, x_4\}$$

$$R(x_2) = \{x_2\} \cup \{x_2, x_3\} \cup \{x_2, x_3, x_4\} \cup \{x_2, x_3, x_4\} = \{x_2, x_3, x_4\}$$

$$R(x_3) = \{x_3\} \cup \{x_3, x_4\} \cup \{x_2, x_3, x_4\} \cup \{x_2, x_3, x_4\} = \{x_2, x_3, x_4\}$$

$$R(x_4) = \{x_4\} \cup \{x_2\} \cup \{x_2, x_3\} \cup \{x_2, x_3, x_4\} \cup \{x_2, x_3, x_4\} = \{x_2, x_3, x_4\}$$

$$R(x_5) = \{x_5\} \cup \{x_1, x_3\} \cup \{x_2, x_3, x_4\} \cup \{x_2, x_3, x_4\} = \{x_1, x_2, x_3, x_4, x_5\}$$

4. Матрица на достиженост:

$$R = \begin{vmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

А от нея лесно получаваме матрицата на обратна достиженост:

$$Q = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

## Определяне на нареждаща функция на ориентиран граф

Тази функция съпоставя на всеки връх от графа ниво, към което той принадлежи. Тази задача е известна още и като задача за топологично сортиране на граф. Какъв е смисълът от нейното решаване и кога може да се използва в практиката?

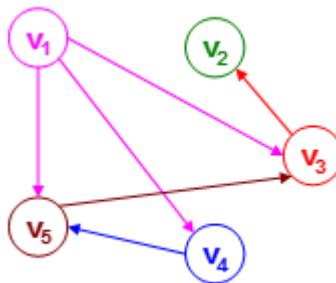
**Топологично сортиране** на ориентиран ацикличен граф се нарича линейно нареден списък от върховете му, такъв, че за всеки два върха  $i$  и  $j$ , за които има път от  $i$  до  $j$ , върхът  $i$  е преди  $j$ . За повечето ориентирани ациклични графи съществуват повече от един такива списъци.

Решаването на подобна задача има смисъл, когато за системата или процесът, който сме моделирали с граф е важна последователността от обхождане на върховете например.

Известни са различни алгоритми за топологично сортиране, един от най-лесните и ясни е следният:

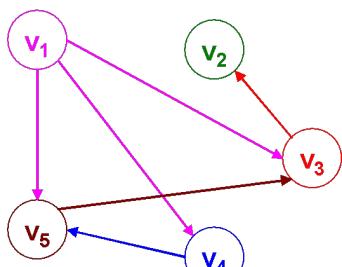
1. Определяме матрицата на съседство;
2. Определяме върховете без входни дъги – в колоната за тях в матрицата на съседство има само 0. Записваме ги в съответното ниво, според итерацията.
3. Премахваме тези върхове и изходящите за тях дъги и правим стъпки 1 и 2 за получения подграф. Това се повтаря до изчерпване на върховете в графа.

Пример: За показания на фигура 21 граф да се определи наредждащата функция.



**Фигура 21. Примерен ориентиран граф за илюстрация на алгоритъма за намиране на наредждаща функция**

1. За показания граф определяме матрицата на съседство:

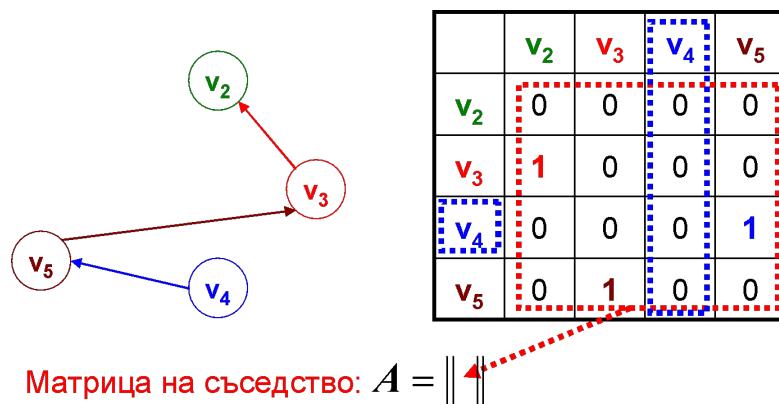


	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$
$v_1$	0	0	1	1	1
$v_2$	0	0	0	0	0
$v_3$	0	1	0	0	0
$v_4$	0	0	0	0	1
$v_5$	0	0	1	0	0

Матрица на съседство:  $A = \begin{pmatrix} & \\ & \end{pmatrix}$

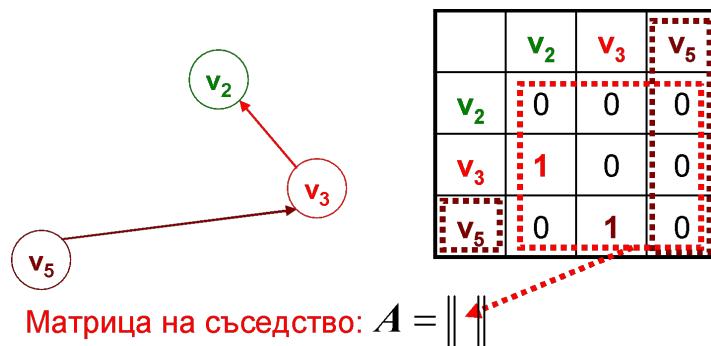
$$A = \begin{vmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{vmatrix}$$

2. Определяме върховете без входни дъги – в колоната за тях в матрицата на съседство има само 0. В случая това е  $v_1$ .  $\Rightarrow N_1 = \{v_1\}$
3. Премахваме тези върхове и излизашите им дъги и правим стъпки 1 и 2 за подграфа.



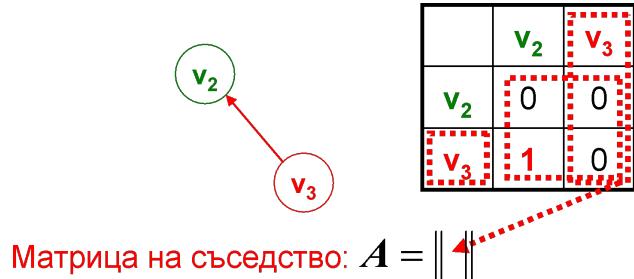
$$A = \begin{vmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix}$$

Върхове без входни дъги:  $v_4$ .  $\Rightarrow N_2 = \{v_4\}$ .



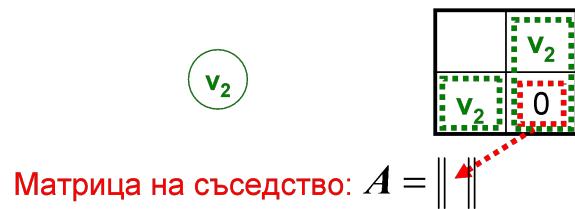
$$A = \begin{vmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{vmatrix}$$

Върхове без входни дъги:  $v_5$ .  $\Rightarrow N_3 = \{v_5\}$ .



$$A = \begin{vmatrix} 0 & 0 \\ 1 & 0 \end{vmatrix}$$

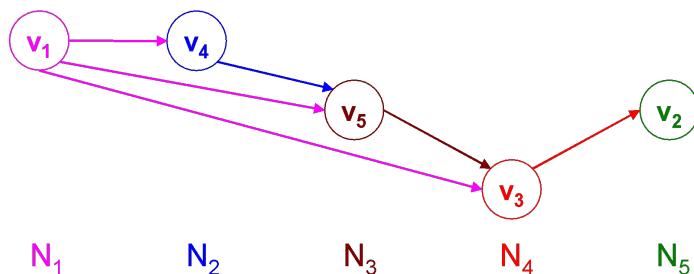
Върхове без входни дъги:  $v_3$ .  $\Rightarrow N_4 = \{v_3\}$ .



$$A = \begin{vmatrix} 0 \end{vmatrix}$$

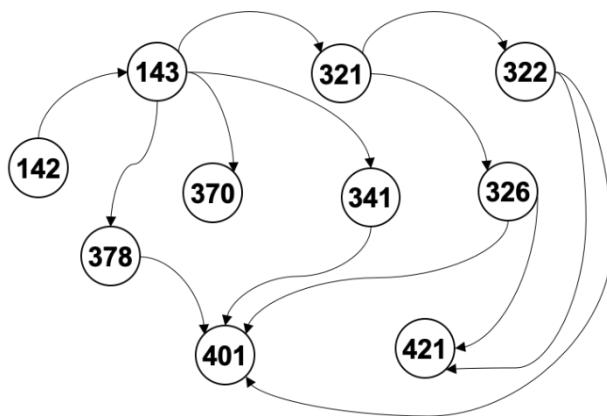
Върхове без входни дъги:  $v_2$ .  $\Rightarrow N_5 = \{v_2\}$ .

Нареждащата функция на примерния граф от фигура 21 има следния вид, показан на фигура 22.



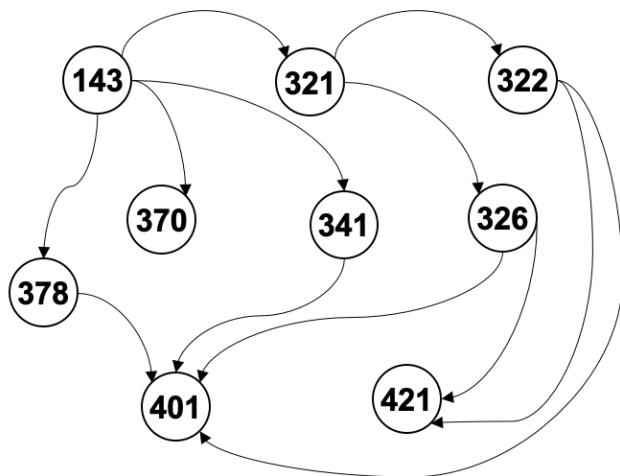
*Фигура 22. Нареждаща функция на граф, пример.*

Можем, например, учебния план за дадена специалност в университета да представим във вид на ориентиран граф. Дисциплините в плана са върховете на графа, а входните и изходни връзки между тях са дъгите в графа. Представете си, че предварително не е зададена последователността на изучаване на дисциплините, а всеки студент може да си избира кои дисциплини кога да изучава. За да има смисъл от такова обучение и за да се спазват изискванията за необходими знания за всяка дисциплина е необходимо да се реши именно задачата за определяне на нареджаща функция или топологично сортиране. По този начин ще се определи в каква последователност да се избират дисциплините и кои от тях могат да бъдат изучавани паралелно. На фигура 23 е показан примерен опростен модел на учебния план, като номерата, с които са означени върховете, са шифрите на дисциплините.

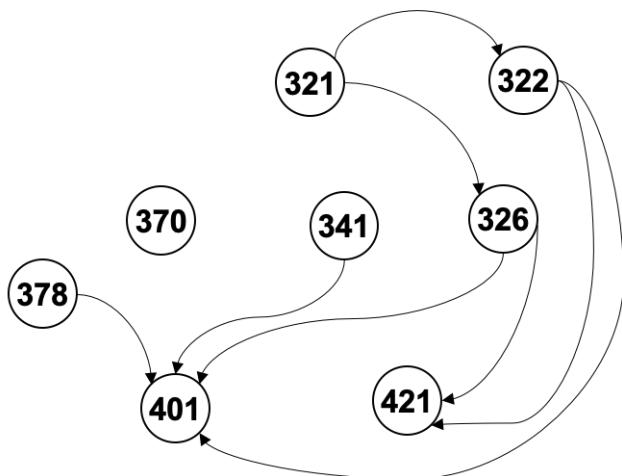


**Фигура 23. Модел на учебен план във вид на граф.**

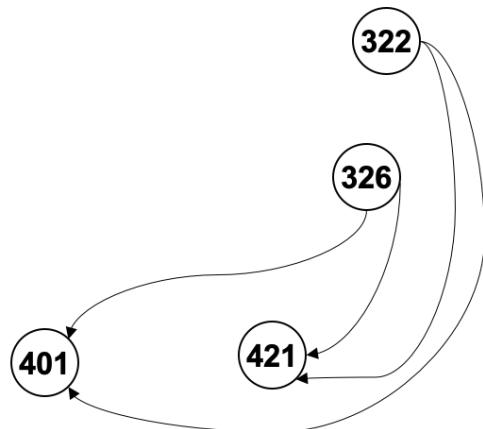
142 е върхът, който няма входящи дъги, следователно дисциплина с шифър 142 може да бъде първата избрана. След като отстраним този връх в графа и се следва даденият алгоритъм последователно се получават показаните на фигура 24 подграфи и списъкът от дисциплини.



отделя се дисциплина 142



дисциплина 143 се добавя в списъка



на тази стъпка - 321, 341, 370 и 378

паралелно 322, 326



и последно възможни за избор

остават 401 и 421, които също могат да се избират паралелно.

*Фигура 24. Стъпки за определяне на наредждащата функция на граф – пример.*

Като резултат, подредбата на дисциплините би трябвало да е:

142

143

321, 341, 370, 378

322, 326

401, 421

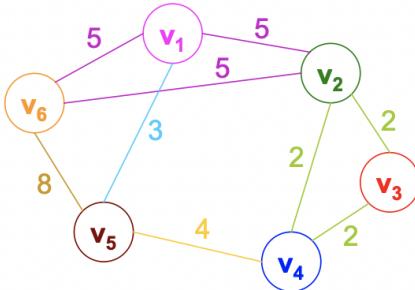
### **Определяне на минимално покриващо дърво за претеглен неориентиран граф (Алгоритъм на Краскал)**

Решаването на този тип задача няма нищо общо със задачите за намиране на най-къси пътища от зададен връх до всички останали върхове. Идеята в този случай е подходяща за практически задачи, свързани например с полагане на канализация в населено място и стремеж към минимизиране на дължините на тръбите или комуникационна задача, при която се целят минимални разходи за свързване на отделните възли в системата.

Стъпки от алгоритъма:

1. Започва се с несвързан граф, който съдържа всички върхове на изходния, за който ще търсим минималното покриващо дърво. Потенциално всеки от върховете в графа участва в дървото.
2. Построяваме претеглената матрица на съседство за графа и в нея търсим най-късото ребро.
3. Добавяме това ребро в графа, ако добавянето му не води до появата на цикъл в графа.
4. Търсим следващото най-късо ребро и повтаряме стъпка 3 толкова пъти, колкото е необходимо, докато броя на ребрата в дървото, което строим, не достигне броя на ребрата в графа – 1.

Например, да намерим минималното покриващо дърво за показания на фигура 25 претеглен неориентиран граф:

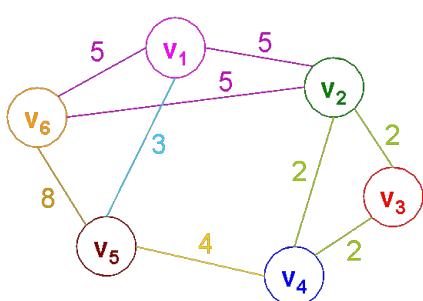


**Фигура 25. Примерен претеглен неориентиран граф.**

- Всеки от върховете участва в дървото и на ниво 0 имаме:



- Определяме претеглената матрица на съседство за този граф



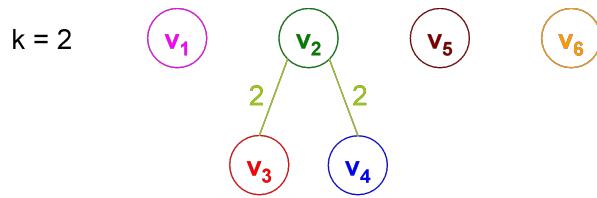
	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_6$
$v_1$	0	5	0	0	3	5
$v_2$	5	0	2	2	0	5
$v_3$	0	2	0	2	0	0
$v_4$	0	2	2	0	4	0
$v_5$	3	0	0	4	0	8
$v_6$	5	5	0	0	8	0

Претеглена матрица на съседство:  $A = \boxed{\quad}$

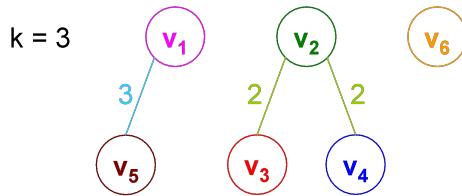
$$A = \begin{vmatrix} 0 & 5 & 0 & 0 & 3 & 5 \\ 5 & 0 & 2 & 2 & 0 & 5 \\ 0 & 2 & 0 & 2 & 0 & 0 \\ 0 & 2 & 2 & 0 & 4 & 0 \\ 3 & 0 & 0 & 4 & 0 & 8 \\ 5 & 5 & 0 & 0 & 8 & 0 \end{vmatrix}$$

Търсим най-късите ребра в матрицата. Това са тези с дължина 2 в случая. Те свързват  $v_2$  с  $v_3$ ,  $v_2$  с  $v_4$  и  $v_3$  с  $v_4$ .

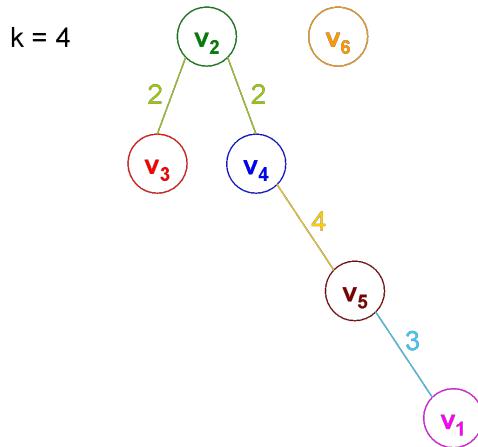
3. Добавяме първите две ребра, защото ако добавим и третото ще се получи цикъл, а в дърветата не може да има цикли.



4. Следващото по дължина ребро е с дължина 3 и е между  $v_1$  и  $v_5$ .



5. Следващото по дължина ребро е с дължина 4 и е между  $v_4$  и  $v_5$ . По този начин се свързват двете под дървета.

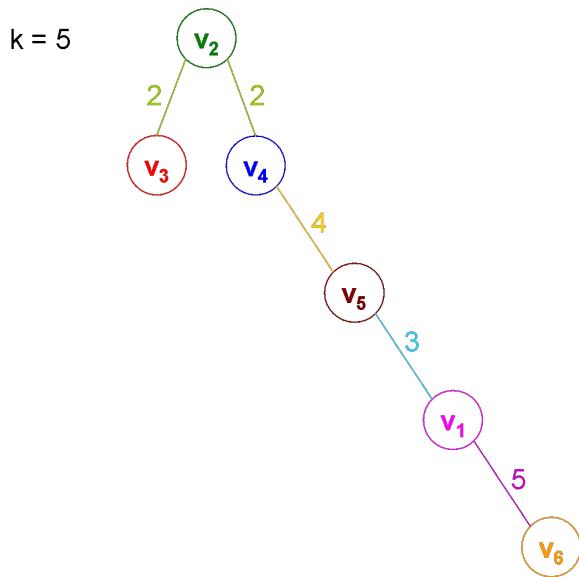


1. Следващите по дължина ребра са с дължина 5 и са между:

$v_1$  и  $v_2$  – води до цикъл;

$v_1$  и  $v_6$  – може да се включи;

$v_2$  и  $v_6$  – вече сме добавили горното и това ще доведе до цикъл.



Полученото дърво има дължина:  $2 + 2 + 4 + 3 + 5 = 16$ .

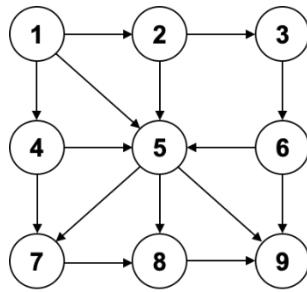
### Задачи за екстремални пътища в граф

Интересен клас задачи са свързани с търсене на път, отговарящ на някакви предварително зададени критерии, например най-къс път, най-дълъг път, път между два фиксираны върха и др. Разглежданите графи могат да бъдат претеглени или не претеглени ориентирани или неориентирани.

Един от известните алгоритми е свързан с построяване на дърво на решението (дърво на възможните пътища). Приложим е както за неориентирани, така и за ориентирани графи. Дължината на пътя се определя от броя на участващите в него ребра или дъги. Може да се използва за намиране на броя на възможните пътища, за търсене на път с минимална или максимална дължина.

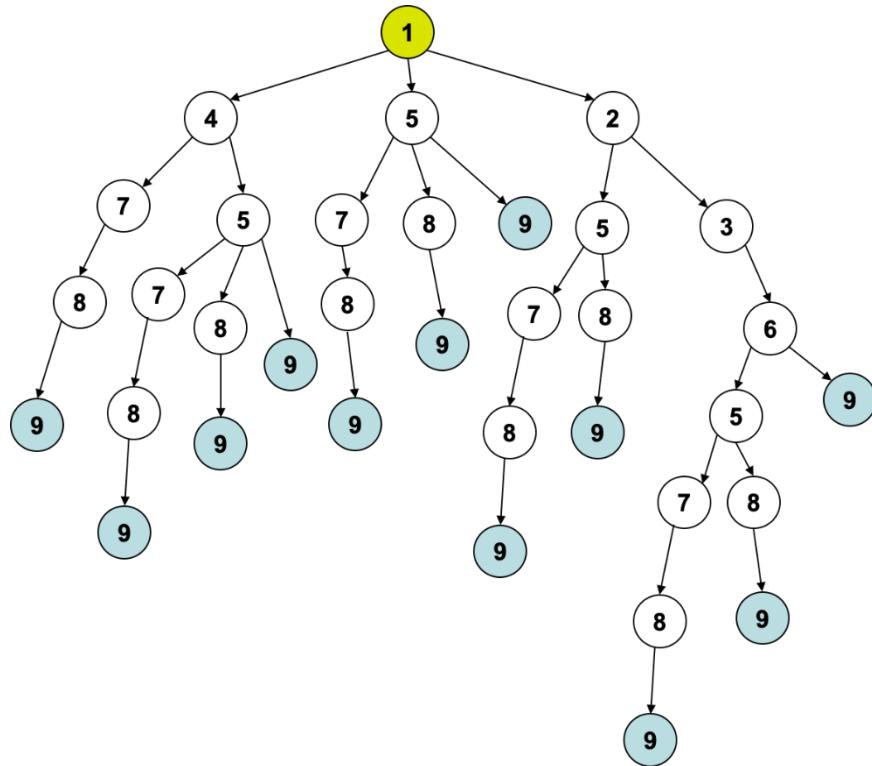
Дървото на решението се построява чрез обхождане на графа по широчина. За демонстрация на алгоритъма ще разгледаме графа на фигура 26.

Търсим най-краткия път от връх 1 до връх 9, в който всеки връх участва само по един път. Движението се осъществява само по посока на стрелките. На графа съпоставяме дърво, построено в съответствие със стратегията за обхождане на граф в ширина по следния начин: за корен на дървото определяме връха 1.



**Фигура 26.** Търсене на екстремален път в ориентиран (неориентиран) граф от връх 1 до връх 9.

От връх 1, следвайки дъгите, можем да попаднем във върховете 4, 5 и 2, които приемаме за наследници. Следваме същата логика за определяне на наследниците на 4, 5 и 2 връх и т.н. до достигане на връх 9, в който пътя приключва. Получаваме дървото, показано на фигура 27.



**Фигура 27.** Дърво на възможните пътища от връх 1 до връх 9 за графа от фигура 26.

Търсенето на път следва стратегията за обхождане в дълбочина, но не в самия граф, а в дървото на възможните пътища. Броят на възможните пътища съвпада с броя на листата на дървото и е 12. Нивата на листата определят

дължината на съответния път. Най-късият път 1, 5, 9 е с дължина 2, а най-дългия 1, 2, 3, 6, 5, 7, 8, 9 е с дължина 7.

Алгоритъмът може да се модифицира и за претеглени графи. В такъв случай, след построяване на дървото на решениета, намираме дължината на всички пътища от корена до листата, сравняваме дълчините им и избираме минималния (максималния).

### **Задачи за определяне на най-къс път**

Обикновено това са задачи за определяне на най-късия път между два зададени върха от претеглен граф, като крайният връх задължително е елемент на множеството на достиженост на началния. Много често се задава началния връх и се търсят най-късите пътища от него до всички останали върхове в графа или пък се търсят най-късите пътища между всички двойки върхове в графа.

За намиране на най-къс път между два зададени върха се използва алгоритъма на Дийкстра:

- Алгоритъмът следи текущо известното най-късо разстояние от всеки възел до началния възел и актуализира тези стойности, ако открие по-къс път.
- След като алгоритъмът намери най-краткия път между началния възел и друг възел, този възел се маркира като „посетен“ и се добавя към пътя.
- Процесът продължава, докато всички възли в графа не бъдат добавени към пътя. По този начин имаме път, който свързва изходния възел с всички други възли, следвайки възможно най-краткия път за достигане до всеки възел.
- На финала на практика имаме най-късия път от началния възел до всички останали възли, ако сме запазили междинните резултати.

Алгоритъмът на Дейкстра може да работи само с претеглени графи, които имат положителни тегла. Това е така, защото по време на процеса трябва да се добавят теглата на ребрата, за да се намери най-краткият път.

След като даден възел бъде маркиран като „посетен“, текущият път до този възел се маркира като най-краткия път за достигане до този възел. И отрицателните тегла могат да променят това, ако общото тегло може да бъде намалено след извършването на тази стъпка.

На всеки възел на графа поставяме в съответствие запис с 3 полета:

$V$  — признак за обработен (посетен) връх ( $0$  — необработен,  $1$  — обработен);

$D$  — дължина на пътя до този връх;

$P$  — връх, предшественик на дадения връх в пътя.

### **Стъпка 1:**

За всеки възел изпълняваме следното:

- на полетата  $V$  за всеки връх присвояваме стойности  $0$ ;
- на полетата  $D$  за всеки връх присвояваме стойности БЕЗКРАЙНОСТ (само началният е със стойност  $0$ );
- на полетата  $P$  за всеки връх присвояваме стойности  $-1$ .

### **Стъпка 2:**

Повтаряме следните стъпки толкова пъти, колкото са върховете:

- избираме връх с най-малка досегашна дължина на пътя (минималната стойност в колоната  $D$ ), измежду не посетените все още върхове (обозначаваме го като  $x$ ) - в началото това е началният връх, тъй като изкуствено сме направили дълчината на пътя до него  $0$ .
- маркираме обработения връх като посетен (променяме  $V$  стойността му на  $1$ )
- за всеки не посетен (т.е. за съседите на  $x$ , за които  $V = 0$ ) съсед у на  $x$  изпълняваме следното:
  - ако

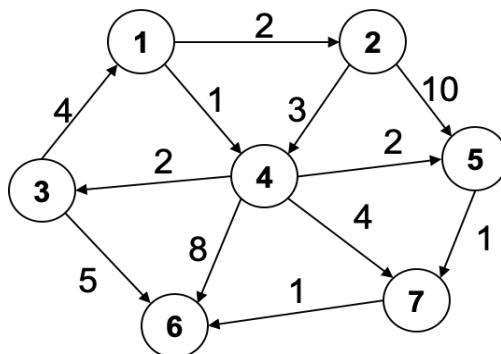
(стойността на полето D за върха x) + дължината на дъгата (x,y) <= (стойността на полето D за y)

- тогава

в полето D за върха y записваме (стойността на полето D за върха x) + дължината на дъгата (x,y)

- В полето P за връх y записваме x.

**Пример:**



*Фигура 28. Примерен граф за илюстриране на алгоритъма на Дийкстра*

Претеглената матрица на съседство:

$$A = \begin{vmatrix} 0 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 10 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 2 & 0 & 2 & 8 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{vmatrix}$$

**Стъпка 1:**

**Началния връх е 1** и съгласно стъпките в алгоритъма за V, всички върхове в началото имат стойност 0 (не са “обработени” все още), включително и началният.

За D поставяме стойността 0 за началния връх (тъй като пътят от началния връх до него самия винаги е 0), но за другите поставяме стойността “ $\infty$ ”

(безкрайност), защото тази стойност ще участва в сравненията в стъпка 2 (разбира се, ако имплементираме алгоритъма на език за програмиране, бихме използвали конкретна стойност).

За стойностите на  $P$ , решението е просто — поставяме стойността  $-1$  (тъй като все още нямат предшественик и няма връх  $-1$ ). Ако съществува връх  $-1$ , ще се наложи да изберем друга стойност, за да е коректно изпълнен алгоритъмът.

Таблично това изглежда по следния начин:

връх	V	D	P
1	0	0	-1
2	0	$\infty$	-1
3	0	$\infty$	-1
4	0	$\infty$	-1
5	0	$\infty$	-1
6	0	$\infty$	-1
7	0	$\infty$	-1

## Стъпка 2:

Сега трябва да изберем връх с най-малка досегашна дължина (стойност на  $D$ ). Това ще е началният връх 1, тъй като на него изкуствено сме му поставили дължина 0.

Приемаме, че с  $x$  бележим текущия връх. Следователно, на тази стъпка,  $x = 1$ .

След това маркираме този връх като посетен (променяме  $V$  стойността му на 1).

Трябва да разберем кои са всички не посетени съседи на този връх. Поглеждаме графа и виждаме, че това са върховете 2 и 4 (не забравяйте, че това е ориентиран граф и затова гледаме само стрелките, които излизат от върха 1).

Правим съответните изчисления за всеки от тези не посетени съседи на връх 1:

- Във връх 1 сме, т.e.  $x = 1$ , и анализираме неговия съседен връх 2, т.e.  $y = 2$  (x - текущ връх, y - не посетен съсед на текущия връх)

За текущия връх:

$$D = 0 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$0 + 2 \leq \infty \quad (0 \text{ е } D \text{ за } x, \text{ а } 2 \text{ е дължината на дъгата от } x(1) \text{ до } y(2))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 2 = 0 + 2 = 2$$

$$P \text{ за } y = x = 1$$

- Във връх 1 сме, т.e.  $x = 1$ , и анализираме неговия съседен връх 4, т.e.  $y = 4$  (x - текущ връх, y - не посетен съсед на текущия връх)

За текущия връх:

$$D = 0 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$0 + 1 \leq \infty \quad (0 \text{ е } D \text{ за } x, \text{ а } 1 \text{ е дължината на дъгата от } x(1) \text{ до } y(4))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 1 = 0 + 1 = 1$$

$$P \text{ за } y = x = 1$$

Добавяме стойностите в обновена таблица:

връх	V	D	P
1	1	0	-1
2	0	2	1
3	0	$\infty$	-1
4	0	1	1
5	0	$\infty$	-1
6	0	$\infty$	-1
7	0	$\infty$	-1

Продължаваме по същия алгоритъм със стъпка 1. Виждаме, че върхът с най-малка досегашна дължина (от не посетените) е връх 4 (с досегашна дължина 1).  $x = 4$

**Маркираме го като посетен**, т.е.  $V$  за връх 4 = 1.

Този връх има доста не посетени съседи — 3, 6, 7 и 5. За всеки от тях правим същите изчисления за стъпка 2, както за предишната итерация.

- Във връх 4 сме, т.е.  $x = 4$ , и **анализираме неговия съседен връх 3**, т.е.  $y = 3$  ( $x$  - текущ връх,  $y$  - не посетен съсед на текущия връх).

За текущия връх:

$$D = 1 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$1 + 2 \leq \infty \quad (1 \text{ е } D \text{ за } x, \text{ а } 2 \text{ е дължината на дъгата от } x(4) \text{ до } y(3))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 2 = 1 + 2 = 3$$

$$P \text{ за } y = x = 4$$

- Във връх 4 сме, т.е.  $x = 4$ , и *анализираме неговия съседен връх 6*, т.е.  $y = 6$  (x - текущ връх, y - не посетен съсед на текущия връх).

За текущия връх:

$$D = 1 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$1 + 8 \leq \infty \text{ (1 е } D \text{ за } x, \text{ а } 8 \text{ е дължината на дъгата от } x(4) \text{ до } y(6))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 8 = 1 + 8 = 9$$

$$P \text{ за } y = x = 4$$

- Във връх 4 сме, т.е.  $x = 4$ , и *анализираме неговия съседен връх 7*, т.е.  $y = 7$  (x - текущ връх, y - не посетен съсед на текущия връх).

За текущия връх:

$$D = 1 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$1 + 4 \leq \infty \text{ (1 е } D \text{ за } x, \text{ а } 4 \text{ е дължината на дъгата от } x(4) \text{ до } y(7))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 4 = 1 + 4 = 5$$

$$P \text{ за } y = x = 4$$

- Във връх 4 сме, т.е.  $x = 4$ , и *анализираме неговия съседен връх 5*, т.е.  $y = 5$  (x - текущ връх, y - не посетен съсед на текущия връх).

За текущия връх:

$$D = 1 \text{ за } x$$

$$D = \infty \text{ за } y$$

$$1 + 2 \leq \infty (1 \text{ е } D \text{ за } x, \text{ а } 2 \text{ е дължината на дъгата от } x(4) \text{ до } y(5))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 2 = 1 + 2 = 3$$

$$P \text{ за } y = x = 4$$

връх	V	D	P
1	1	0	-1
2	0	2	1
3	0	3	4
4	1	1	1
5	0	3	4
6	0	9	4
7	0	5	4

Продължаваме по същия алгоритъм със стъпка 1. Виждаме, че върхът с най-малка досегашна дължина (от не посетените) е връх 2 (с досегашна дължина 2).  $x = 2$

**Маркираме го като посетен**, т.е.  $V$  за връх 2 = 1.

Този връх има два съседни върха — 4 и 5. Единият вече е посетен — 4, така че стъпка 2 изпълняваме само за връх 5.

- Във връх 2 сме, т.е.  $x = 2$ , и **анализираме неговия съседен връх 5**, т.е.  $y = 5$  ( $x$  - текущ връх,  $y$  - не посетен съсед на текущия връх).

За текущия връх:

$$D = 2 \text{ за } x$$

$$D = 3 \text{ за } y$$

$$2 + 10 \leq 3 \text{ (2 е } D \text{ за } x, \text{ а } 10 \text{ е дълчината на дъгата от } x(2) \text{ до } y(5))$$

Не е изпълнено горното и ***в този случай не променяме никакви стойности, но маркираме върха като посетен!***

От таблицата виждаме, че има два върха от не посетените, които имат минимална дължина – това са 3 и 5 с дължина 3. Няма значение кой от двата ще изберем първо. Ако избираме първия по ред на номериране – избираме 3 т.е.

$$x = 3.$$

връх	V	D	P
1	1	0	-1
2	1	2	1
3	0	3	4
4	1	1	1
5	0	3	4
6	0	9	4
7	0	5	4

***Маркираме го като посетен***, т.е.  $V$  за връх 3 = 1.

Този връх има 2 съседни — 1 и 6, но 1 е вече посетен. За 6 правим изчисленията за стъпка 2, както за предишната итерация.

- Във връх 3 сме, т.е.  $x = 3$ , и ***анализираме неговия съседен връх 6***, т.е.  $y = 6$  ( $x$  - текущ връх,  $y$  - не посетен съсед на текущия връх).

За текущия връх:

$$D = 3 \text{ за } x$$

$$D = 9 \text{ за } y$$

$$3 + 5 \leq 9 \text{ (3 е } D \text{ за } x, \text{ а } 5 \text{ е дълчината на дъгата от } x(3) \text{ до } y(6))$$

Щом е изпълнено горното то тогава:

$$D \text{ за } y = D \text{ за } x + 5 = 3 + 5 = 8$$

$$P \text{ за } y = x = 3$$

Нанасяме новите стойности в таблицата:

връх	V	D	P
1	1	0	-1
2	1	2	1
3	1	3	4
4	1	1	1
5	0	3	4
6	0	8	3
7	0	5	4

Следващият не посетен връх е 5 с дължина 3 или  $x = 5$

**Маркираме го като посетен**, т.е.  $V$  за връх 5 = 1.

Този връх има 1 съседен — 7. За 7 правим изчисленията за стъпка 2, както за предишната итерация.

- Във връх 5 сме, т.е.  $x = 5$ , и **анализираме неговия съседен връх 7**, т.е.  $y = 7$  ( $x$  - текущ връх,  $y$  - не посетен съсед на текущия връх).

За текущия връх:

$$D = 3 \text{ за } x$$

$$D = 5 \text{ за } y$$

$$3 + 1 \leq 5 \quad (3 \text{ е } D \text{ за } x, \text{ а } 1 \text{ е дължината на дъгата от } x(5) \text{ до } y(7))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 1 = 3 + 1 = 4$$

$$P \text{ за } y = x = 5$$

Нанасяме новите стойности в таблицата:

връх	V	D	P
1	1	0	-1
2	1	2	1
3	1	3	4
4	1	1	1
5	1	3	4
6	0	8	3
7	0	4	5

Останаха ни само два върха. 7 има по-малка стойност за D, затова избираме него, т.е.  $V$  за връх  $7 = 1$ .

Този връх има 1 не посетен съседен — 6. За него правим изчисленията за стъпка 2, както за предишната итерация.

- Във връх 7 сме, т.е.  $x = 7$ , и *анализираме неговия съседен връх 6*, т.е.  $y = 6$  ( $x$  - текущ връх,  $y$  - не посетен съсед на текущия връх).

За текущия връх:

$$D = 4 \text{ за } x$$

$$D = 8 \text{ за } y$$

$$4 + 1 \leq 8 \quad (4 \text{ е } D \text{ за } x, \text{ а } 1 \text{ е дължината на дъгата от } x(7) \text{ до } y(6))$$

Щом е изпълнено горното то тогава:

$$D(\text{за } y) = D \text{ за } x + 1 = 4 + 1 = 5$$

$$P \text{ за } y = x = 7$$

Нанасяме новите стойности в таблицата:

връх	V	D	P
1	1	0	-1
2	1	2	1
3	1	3	4
4	1	1	1
5	1	3	4
6	0	5	7
7	1	4	5

Остана ни само един връх — 6. Той очевидно няма не посетени съседи, така че просто го маркираме като посетен и обновяваме таблицата:

връх	V	D	P
1	1	0	-1
2	1	2	1
3	1	3	4
4	1	1	1
5	1	3	4
6	1	5	7
7	1	4	5

**Това е финалната версия на нашата таблица** и алгоритъмът завършва.

В тази таблица гледаме крайния връх, който ни интересува. Да кажем 6. Виждаме, че в P колоната му е 7. Това означава, че 7 е негов предшественик. В

колоната Р на 7 пък виждаме стойността 5. Значи 5 е предшественик на 7. 4 пък е предшественик на 5. Най-накрая виждаме, че 1 е предшественик на 4.

От това следва, че най-краткият път от 1 до 6 е  $1 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 6$ . Ако сметнем общата дължина, тя е  $1+2+1+1 = 5$  (каквато е и D стойността на крайния връх 6).

По този начин можем да намерим най-краткия път от връх 1 до който и да е друг връх, ако искаме да използваме друг за начален, трябва да повторим алгоритъма отново.

### **Задача за търговския пътник (Traveling salesman problem)**

Търговски пътник трябва да посети всеки от дадени  $n$  града по един път, тръгвайки от един начален град и връщайки се в него. Ако са известни разстоянията между всеки два града, да се намери най-краткия път.

Така формулирана, задачата е оптимизационна, тъй като от всички възможни пътища търсим този с минимална дължина. Математическата постановка е свързана с намиране на Хамилтонов цикъл (или контур) в пълен претеглен граф с минимално тегло. В зависимост от смисъла на теглата, съпоставени на ребрата (дъгите), можем да търсим път с минимална дължина, с минимален разход на гориво, с най-ниска цена и др.

Известни са и други формулировки на задачата, свързани с проверка за съществуване на път, започващ от даден град и завършващ в него или съществуване на път с дължина по-малка или равна на предварително зададена дължина.

Задачата за търговския пътник е NP-пълна задача. За решението и не съществуват полиномиални алгоритми, единствения метод за решение е изчерпващото търсене в пространството на допустимите решения с временна сложност  $O(n!)$ . При голям брой градове задачата е практически неразрешима, затова се използват евристики за намаляване на броя на анализираните маршрути или приближени методи, които дават близко до оптималното решение като алчни

алгоритми (метод на най-близкия съсед), генетични алгоритми, вероятностни методи и др. Задачата е еталон за сложност и има много голямо теоретично и практическо значение. Именно тя се използва при анализ на различни евристични подходи за съкращаване на изчерпващото търсене.

При изчерпващо търсене номерираме подлежащите на обхождане градове последователно  $\{1, 2, 3, \dots, n\}$ . Всеки маршрут, започващ и завършващ в град 1, може да се разглежда като перmutация на числата 2, 3, ...n. Задачата за генериране на всички маршрути е еквивалентна на задачата за получаване на всички  $(n-1)!$  на брой перmutации на числата от 2 до n. Алгоритъмът за решение на задачата за търговски пътник, чрез изчерпващо търсене, включва следните стъпки:

1. Полагаме дължината на минималния път  $S_{MIN} = \infty$ , инициализираме търсения път  $P_{MIN} = " "$ ; номера на поредната перmutация  $I = 0$ ;
2.  $I = I + 1$ ; генерираме I-тата перmutация; пресмятаме дължината на текущия път  $SI$ ;
3. Ако  $S_{MIN} > SI$ , полагаме  $S_{MIN} = SI$ ; запомняме I - тата перmutация в  $P_{MIN}$ ;
4. Ако  $I = (n-1)!$  – край на алгоритъма; в противен случай се връщаме към стъпка 2.

След приключване на алгоритъма  $S_{MIN}$  ще съдържа дължината на минималния път, а  $P_{MIN}$  – реда на обхождане на градовете.

При метода на клоните и границите на всяка стъпка се поражда един от възможните варианти на обхождане, като се отчита оптималния до момента път, който се използва за сравнение със следващите кандидати. Ако добавянето на нов, все още не посетен връх към текущото решение увеличава дължината на пътя над оптималната за момента, върхът заедно със всички възможни продължения се отсича. В резултат на отсичането на неперспективни решения броят на вариантите съществено се намалява, но въпреки това остава експоненциален  $1.26^n$ . В най – лошия случай, когато оптималният път е последен

от анализираните, сложността на метода ще съвпадне със сложността на изчерпващото търсене.

Алчните алгоритми следват стратегия, при която не се анализират всички възможни варианти. Решението се конструира чрез последователно добавяне на следващ не посетен връх с най- малко тегло на реброто (дъгата). Методът е високо ефективен, тъй като на всяка стъпка се избира само един вариант за продължение. Въпреки че няма гаранция за оптималност на полученото решение, евристиката на „най-близкия съсед“ е възможен компромис в случаите, когато изчерпващото търсене е практически неосъществимо.

Вероятностните алгоритми на всяка стъпка генерираят по случаен начин един от възможните пътища и пресмятат дължината му, която се сравнява с оптималната до момента. Ако текущо генеририаният път има по-малка дължина, то се съхранява този път и неговата дължина. Многократното повторение на тези действия ще доведе до получаване на път с минимална дължина от всички генериирани.

### **Задача за китайския пощалъон**

Chinese postman's problem – в претеглен граф да се намери цикъл, минаващ през всяко ребро поне по един път, за който сумата от теглата на ребрата да бъде минимална. Дължината на всяко ребро се включва в сумата толкова пъти, колкото реброто се среща в цикъла. Ако графът е Ойлеров, всеки Ойлеров цикъл е решение на задачата.

### **Задача за сватбите. Теорема на Хол**

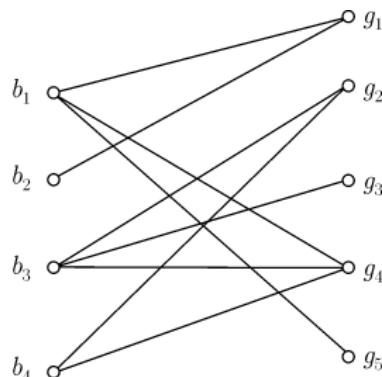
За илюстрация на двуделните графи често се използва следната Задача за сватбите.

Дадени са две крайни множества: на юношите и на девойките. Всеки юноша познава няколко девойки. При какви условия можем да оженим юношите така, че всеки юноша да се ожени за позната девойка?

Нека юношите са четири  $\{b_1, b_2, b_3, b_4\}$ , а девойките пет  $\{g_1, g_2, g_3, g_4, g_5\}$ . Отношението «юношата  $x$  познава девойката  $y$ » задаваме таблично:

юноша	девойки, които познава		
$b_1$	$g_1$	$g_4$	$g_5$
$b_2$	$g_1$		
$b_3$	$g_2$	$g_3$	$g_4$
$b_4$	$g_2$	$g_4$	

Графично можем да представим юношите и девойките чрез върхове, а отношението на познанство - чрез ребра по начина, показан на фигура 29.



*Фигура 29. Граф, представящ отношението „познанство“ между 4 юноши и 5 девойки*

**Теорема на Хол за сватбите** : Задачата за сватбите има решение тогава и само тогава, когато всеки юноша познават поне  $k$  девойки за всяко  $0 \leq k \leq n$ , където  $n$  е броят на юношите.

**Доказателство:**

Необходимост: Предполагаме, че задачата има решение. Ще докажем, че условието на теоремата е вярно.

Нека допуснем, че условието на теоремата не е изпълнено, т.е съществуват  $k$  юноши, които могат да бъдат оженени най-много за  $k-1$  девойки. От принципа на Дирихле за чекмеджетата следва, че поне един от тези  $k$  юноши не може да бъде оженен за нито една девойка, следователно и всичките  $n$  юноши не могат да бъдат оженени. Но тогава задачата няма решение, което противоречи на допускането, че задачата има решение. Следователно допускането не е вярно и условието на теоремата е изпълнено.

Достатъчност: Предполагаме, че условието на теоремата е вярно. Ще докажем, че задачата има решение.

Доказателството ще извършим чрез индукция по  $n$ .

Нека  $n = 1$ . Тогава трябва да оженим точно един юноша. Но условието “всеки  $k$  юноши познават поне  $k$  девойки за всяко  $k$ ” е еквивалентно на условието “единственият юноша познава поне една девойка”, за която можем да го оженим и задачата има решение.

Допускаме, че задачата има решение, ако броят на юношите е по-малък от  $n$ .

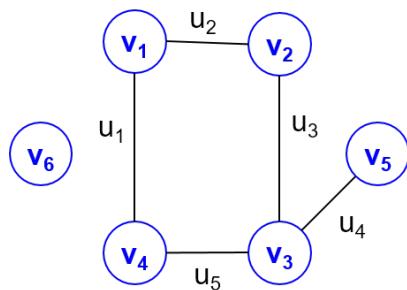
Ще докажем, че задачата има решение и ако броят на юношите е точно  $n$ . Нека първо считаме, че всеки  $k$  юноши познават поне  $k+1$  девойки ( $1 \leq k \leq n$ ). Тогава, ако оженим един от юношите за позната девойка, ще останат още  $n-1$  юноши. Но от  $n$ -те юноши всеки  $k$  познават поне  $k+1$  девойки ( $1 \leq k \leq n$ ) и понеже един вече е оженен, то сега всеки  $k$  юноши познават поне  $k$  девойки. Така получихме група от  $n-1$  юноши, всеки  $k$  от които познават поне  $k$  девойки и от индукционното предположение задачата има решение ( $1 \leq k \leq n$ ). Нека сега съществуват  $k$  юноши, които познават точно  $k$  девойки. По индукционното предположение за тези  $k$  юноши задачата има решение. Остават още  $n-k$  юноши. Ако допуснем, че съществуват  $h$  юноши ( $1 \leq h \leq n-k$ ), които могат да бъдат оженени за най-много  $h-1$  девойки, то тези юноши, заедно с предните  $k$  образуват група от  $k+h$  юноши, които могат да бъдат оженени за най-много  $k+h-1$  девойки ( $k+h \leq n$ ), което е в противоречие с условието. Следователно всеки  $h$  юноши

познават поне  $h$  девойки ( $h \leq n-k$ ) и от индукционното предположение следва, че задачата има решение и за  $(n-k)$ -те юноши. Съгласно метода на пълната математическа индукция, задачата има решение за всичките  $n$  юноши ( $n-k \leq n$ ).

Теоремата на Хол има теоретично значение, поради трудността за практическа проверка на формулирания в нея критерий.

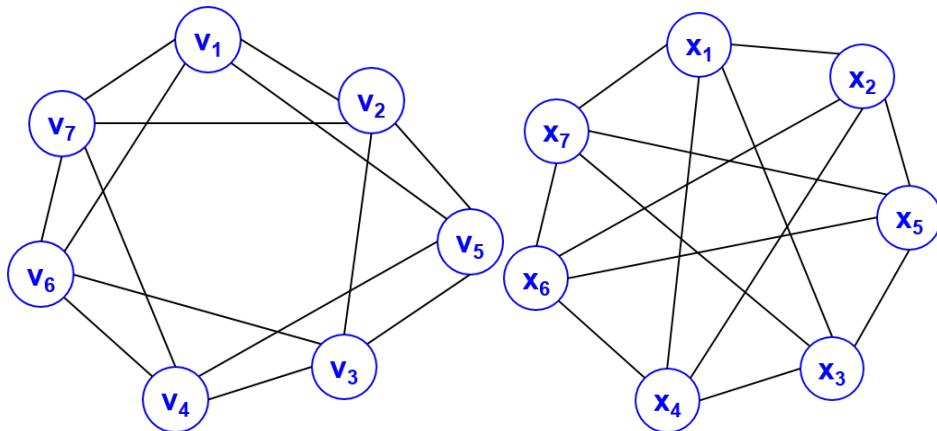
### Задачи:

**Задача 1:** Задайте аналитично (чрез множество от върхове и множество от ребра) показания неориентиран граф на фигура 30.



*Фигура 30. Примерен неориентиран граф.*

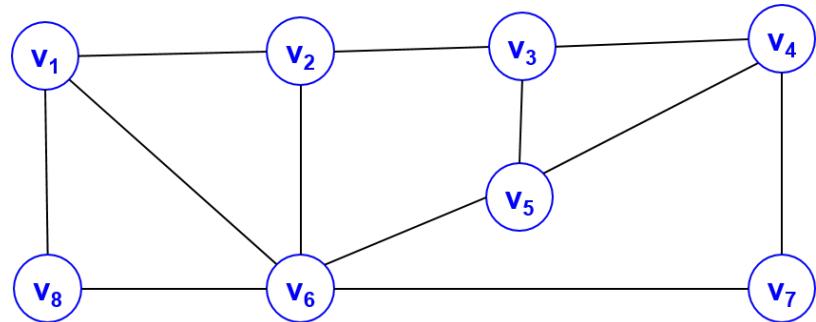
**Задача 2:** Изоморфни ли са двойките графи, показани на фигура 31.



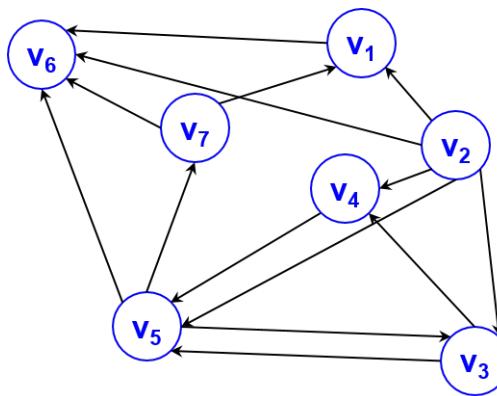
*Фигура 31. Примерни графи за изследване на изоморфизъм.*

**Задача 3:** Представете зададените графи на фигура 32 и 33 чрез матрица на съседство.

За ориентирания граф (фигура 33) определете матриците на достижимост и обратна достижимост.

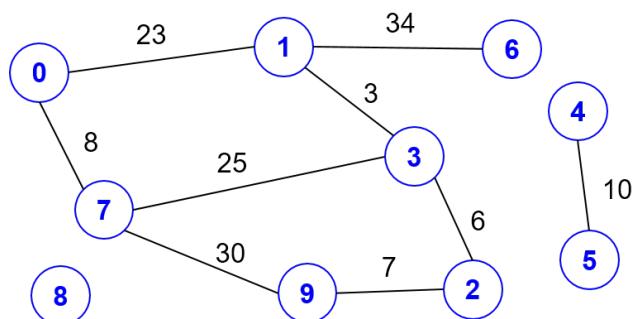


*Фигура 32. Неориентиран граф*



*Фигура 33. Ориентиран граф.*

**Задача 4:** Приложете алгоритъма на Дейкстра за намиране на пътища с минимални дължини от върха 0 до всички останали върхове на графа от фигура 34

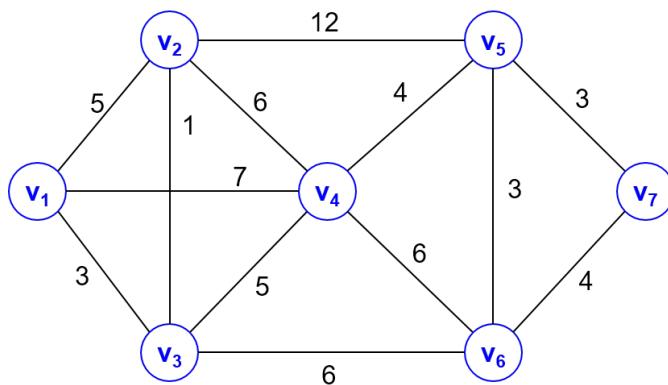


*Фигура 34. Примерен граф.*

Отговор: 1-23; 2-32; 3-26; 4 – няма път; 5- няма път; 6 – 57; 7- 8; 8 – няма път; 9 – 38.

**Задача 5:** За графа от фигура 35 определете:

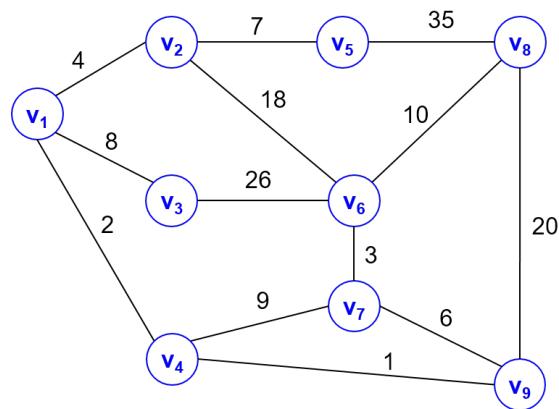
- a) Матрицата на съседство;
- b) Претеглената матрица на съседство;
- c) Локалните степени на върховете и цикломатичното число;
- d) Докажете, че има или няма Ойлеров цикъл (ако има – опишете един от възможните);
- e) Минималното покриващо дърво, като последователно опишете всички стъпки на алгоритъма на Крускал;
- f) Намерете най-късите пътища от връх  $V_1$  до всички останали;
- g) Хамилтонов цикъл



*Фигура 35. Примерен граф.*

**Задача 6:** За показания на фигура 36 претеглен неориентиран граф определете.

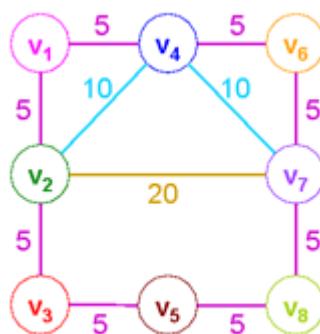
- a) Матрицата на съседство;
- b) Претеглената матрица на съседство;
- c) Локалните степени на върховете и цикломатичното число;
- d) Докажете, че има или няма Ойлеров цикъл (ако има – опишете един от възможните);
- e) Минималното покриващо дърво, като последователно опишете всички стъпки на алгоритъма на Крускал;
- f) Намерете най-късите пътища от връх  $V_1$  до всички останали върхове в графа;
- g) Хамилтонов цикъл



**Фигура 36.** Претеглен неориентиран граф.

**Задача 7:** За графа от фигура 37 определете:

1. Претеглена матрица на съседство
2. Локални степени на върховете
3. Брой ребра
4. Цикломатично число
5. Ойлеров цикъл
6. Хамилтонов цикъл

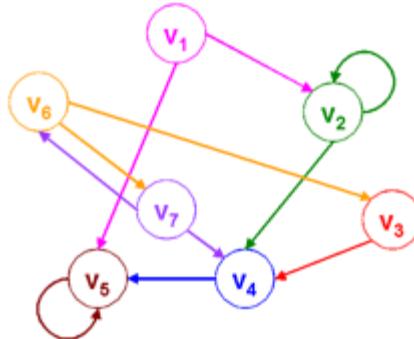


**Фигура 37.** Примерен граф.

**Задача 8:** За графа от фигура 38 да се определят:

1. Матрица на съседство;

- 2.** Матрица на достиженост;
- 3.** Матрица на обратна достиженост –  $\mathbf{Q}$  (получава се от матрицата  $\mathbf{R}$ , като стълбът  $v_i$  от  $\mathbf{R}$  става ред  $v_i$  в  $\mathbf{Q}$ ).



**Фигура 38. Примерен граф.**

**Задача 9:** За зададения с показаната матрица на съседство ориентиран граф  $D$  да се определи нареждащата функция.

$$\mathbf{A} = \begin{vmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{vmatrix}$$

**Задача 10:** Система за предаване на информация има 10 станции  $v_1 \div v_{10}$ .

Всяко предаване на съобщение от станция  $i$  към станция  $j$  става с определена вероятност  $P_{ij}$  за прихващане на съобщението. Да се определи начинът, по който трябва да се предаде информацията до всички станции, така че рискът от прихващане на съобщение да е минимален, ако вероятностите са следните:

$$P_{14} = 0,11 \quad P_{25} = 0,8 \quad P_{46} = 0,1 \quad P_{79} = 0,4 \quad P_{24} = 0,18$$

$$P_{15} = 0,5 \quad P_{27} = 0,11 \quad P_{47} = 0,7 \quad P_{710} = 0,13 \quad P_{39} = 0,6$$

$$P_{12} = 0,6 \quad P_{34} = 0,8 \quad P_{56} = 0,15 \quad P_{89} = 0,14 \quad P_{78} = 0,9$$

$$P_{910} = 0,19 \quad P_{23} = 0,15 \quad P_{38} = 0,18 \quad P_{57} = 0,9$$

