МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до розрахункового завдання «Теорія графів»
з курсу «Дискретна математика»

для студентів спеціальностей
122 «Комп'ютерні науки», 124 «Системний аналіз»

Затверджено редакційно-
видавничою радою університету,
протокол № 1 від 15.02.2024 р.

Харків
2024

Методичні вказівки до розрахункового завдання «Теорія графів» з курсу «Дискретна математика» для студентів спеціальностей 122 «Комп'ютерні науки», 124 «Системний аналіз» / Уклад.: Н. А. Марченко, О. С. Мельников. – Харків: НТУ «ХПІ», 2024. – 64 с.

Укладачі: Н. А. Марченко
О. С. Мельников

Рецензент М. А. Гринченко

Кафедра системного аналізу та інформаційно-аналітичних технологій

ВСТУП

Теорія графів є важливим розділом дискретної математики, який широко застосовується при розв'язанні математичних, економічних та логічних задач, а також дозволяє спростити моделювання фізичних та хімічних процесів і явищ.

Термін «граф» вперше з'явився в науковій літературі в 1936 році в роботах угорського математика Д. Кьоніга, хоча родоначальником теорії графів вважають швейцарського математика Леонарда Ейлера. Саме Л. Ейлер у 1736 році сформулював та розв'язав задачу щодо мостів Кеніґсберґа, яка є першою історичною задачею теорії графів.

У теперішній час у вигляді графів зображають електричні, транспортні, інформаційні та комп'ютерні мережі. Графи використовують у моделях кристалів та молекул хімічних речовин. За допомогою графів описують різні математичні об'єкти (відношення, частково впорядковані множини, решітки, автомати, ланцюги Маркова, алгоритми та програми), лабіринти, плани діяльності, плани виконання певних робіт (розклади), генеалогічні дерева тощо.

Метою даних методичних вказівок є вивчення студентами основних положень теорії графів та їх закріплення шляхом виконання розрахункового завдання.

Розрахункове завдання включає задачі:

- математичного опису графів;
- пошуку кількості дерев графа;
- визначення мінімального кістякового дерева;
- знаходження мінімальної відстані між вершинами графа;
- пошуку циклів на графі;
- розрахунку мережевого графа;
- побудові лінійної (стрічкової) діаграми.

1. ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Основні визначення

З поняттям графу зазвичай пов'язують його графічне представлення, при якому він зображується як множина точок, деякі з яких з'єднані лініями. Але граф відрізняється від геометричних конфігурацій (скажімо, фігур, які також складаються з точок та ліній) тим, що *в графі несуттєві відстані між точками, форма з'єднувальних ліній та кути між ними*. Важливо лиш те, чи з'єднана дана пара точок лінією, чи ні.

Тому граф іноді називають *топологічним об'єктом*, тобто об'єктом, властивості якого не змінюються при розтягуванні, стисненні та викривленні. За цією ж причиною (важливим є тільки наявність або відсутність з'єднань) граф – об'єкт дискретний і може бути заданий двома дискретними множинами: множиною точок, які будемо називати *вершинами*, та множиною ліній, які з'єднують деякі вершини. Ці лінії називають *ребрами*.

Графом $G = (V, E)$ називається об'єкт, який заданий парою множин (V, E), де V – множина вершин, $E \subseteq V \times V$ – множина ребер.

Граф називається *скінченним*, якщо множини його вершин і ребер є скінченними.

Множину вершин графу G позначають $V(G)$, а множину ребер – $E(G)$.

Кількість вершин графу позначають через $n(G) = |V(G)|$, а кількість ребер – $m(G) = |E(G)|$.

Кількість вершин $n(G)$ графу прийнято називати його *порядком*.

На рис. 1.1а-д представлено п'ять графів четвертого порядку.

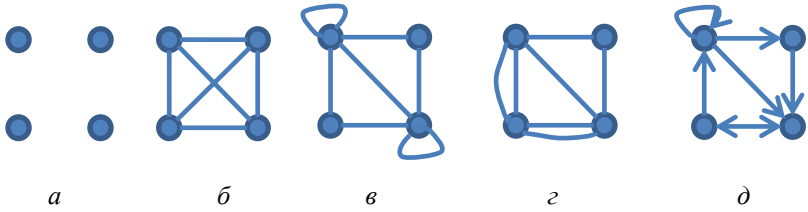


Рис. 1.1. Приклади графів четвертого порядку

Якщо деяке ребро e зв'язує вершини v та w, то кажуть, що:

- вершини v та w *суміжні*;
- вершини v та w *інцидентні ребру* e;
- ребро e *інцидентне вершинам* v і w.

Множина вершин, які суміжні з вершиною v, називається *множиною суміжності* вершини v і позначається $\Gamma^+(v)$:

$$\Gamma^+(v) = \{w \in V: (w, v) \in E\};$$

$$\Gamma^*(v) = \Gamma^+(v) \cup \{v\}.$$

Зазвичай $\Gamma^+(v)$ позначається просто $\Gamma(v)$.

Очевидно, $w \in \Gamma(v)$ тоді й тільки тоді, коли $v \in \Gamma(w)$.

Якщо $A \subset V$ – множина вершин, то $\Gamma(A)$ – множина всіх вершин, суміжних з вершинами з A:

$$\Gamma(A) = \{w \in V: \exists v \in A,\ w \in \Gamma(w)\} = \bigcup_{v \in A} \Gamma(v).$$

Множина ребер E може бути порожньою. Такий граф називається *нуль-графом* і позначається $\emptyset$. Приклад нуль-графа зображено на рис. 1.1*а*.

Якщо ж множина вершин V – порожня, то порожньою є також множина E. Такий граф називається *порожнім*.

Лінії, що зображують ребра графа, можуть перетинатися, але точки перетину не є вершинами, як це показано на рис. 1.1*б*.

Ребро може з'єднувати деяку вершину саму із собою, таке ребро називається *петлею*. Цей випадок відповідає наявності в множині E пар вигляду (v, v). Приклад графа четвертого порядку, що містить петлі, подано на рис. 1.1*в*.

Різні ребра можуть бути інцидентними одній і тій самій парі вершин (тобто одну й ту саму пару вершин з'єднує більше ніж одне ребро), такі ребра називаються *кратними*. Приклад графа четвертого порядку, що містить кратні ребра, подано на рис. 1.1*г*.

Граф називається *простим*, якщо кожну пару вершин з'єднує не більше, ніж одне ребро. Приклад простого графа подано на рис. 1.1*б*.

Граф називається *мультиграфом*, якщо він має кратні ребра.

Граф називається *псевдографом*, якщо він має петлі та кратні ребра.

В розглянутих вище прикладах порядок з'єднання вершин у графі був неважливим. Такі графи називають *неорієнтованими*. Поруч з ними розглядають також орієнтовані графи.

Орієнтованим графом називається граф $G = (V, E)$, де V – множина вершин, $E \subseteq V \times V$ – множина орієнтованих ребер, які називають *дугами*. При зображенні орієнтованих графів напрямки ребер позначаються стрілками. Приклад орієнтовного графа четвертого порядку подано на рис. 1.1*д*.

Орієнтований граф може мати кратні ребра, петлі, а також петлі, що з'єднують одні й ті самі вершини, але у зворотних напрямках.

Кожному неорієнтованому графу можна поставити у відповідність орієнтований граф з тією самою множиною вершин, в якій кожне ребро замінено двома орієнтованими ребрами, що є інцидентними тим самим вершинам і мають зворотні напрямки.

1.2. Математичний опис графів

Задати граф означає задати множини його вершин і ребер, а також відношення інцидентності. Коли граф G – скінченний, для опису його вершин та ребер достатньо їх занумерувати.

Нехай $v_1, v_2, \dots, v_n$ – вершини графу G, а $e_1, e_2, \dots, e_m$ – його ребра. Відношення інцидентності можна означити матрицею $E = (\varepsilon_{ij})$, яка має n рядків та m стовпців. Рядки відповідають вершинам графу, а стовпці – його ребрам. Елементи матриці визначаються наступним чином:

$$\varepsilon_{\text{ij}} = \begin{cases} 1, \text{ якщо ребро } e_j \text{ є інцидентним вершині } v_i \\ 0, \text{ в іншому випадку} \end{cases}.$$

Побудована у такий спосіб матриця називається *матрицею інцидентності неорієнтованого графа* G і є одним із способів його визначення.

Матриця інцидентності орієнтованого графа G задається наступним чином:

$$\varepsilon_{\text{ij}} = \begin{cases} -1, \text{ якщо вершина } \text{v}_\text{i} \text{ – початок дуги } \text{e}_\text{j} \\ 1, \text{ якщо вершина } \text{v}_\text{i} \text{ – кінець дуги } \text{e}_\text{j} \\ 2, \text{ якщо } \text{e}_\text{j} \text{ – петля, а } \text{v}_\text{i} \text{ – інцидентна їй вершина} \\ 0, \text{ в інших випадках} \end{cases}.$$

В кожному рядку матриці інцидентності для неорієнтованого або орієнтованого графу тільки два елементи відмінні від 0 (або один, якщо ребро є петлею). Тому такий спосіб задання графу не економний. Більш компактно відношення інцидентності можна задати ще *списком ребер графа*. Кожний рядок цього списку відповідає одному ребру і містить номери вершин, інцидентних йому. Для неорієнтованого графа порядок цих вершин у рядку довільний. Для орієнтованого графа першим в списку записується номер (або інша позначка) початку ребра, а другим – його кінця.

Матриця суміжності – це квадратна матриця $A = (a_{ij})$, стовпцям і рядкам якої відповідають вершини графу.

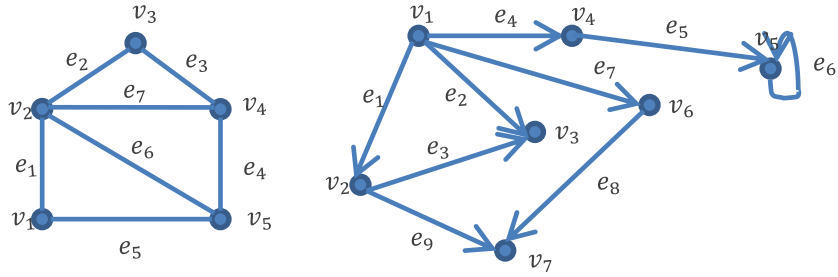
Для неорієнтованого графу a_{ij} дорівнює кількості ребер, інцидентних i- та j-й вершинам, для орієнтованого – цей елемент матриці відповідає кількості ребер з початком в i-й вершині та кінцем у j-й вершині.

Таким чином, матриця суміжності неорієнтованого графу є симетричною ($a_{ij} = a_{ji}$), а орієнтованого – необов'язково. Якщо вона все ж симетрична, то для кожного ребра орієнтованого графу існує ребро, яке з'єднує ті самі вершини, але йде у зворотному напрямку.

Очевидно, орієнтований граф із симетричною матрицею суміжності відповідає неорієнтованому графу, який має ту саму матрицю суміжності.

Якщо графи використовуються для моделювання реальних систем, їх вершинам або/та ребрам приписуються деякі числа. Природа цих чисел може бути найрізноманітніша. Наприклад, якщо граф є моделлю залізничної мережі, то число, приписане ребру, може вказувати відстань між двома станціями. Але що б не означали ці числа, їх прийнято називати *вагами*, а граф із заданими вагами вершин та/або ребер – *зваженим графом*.

Приклад 1.1. Для графів, зображених на рис. 1.2, записати матриці інцидентності, суміжності і задати граф за допомогою списку ребер.



а *б*

Рис. 1.2. Графи для прикладу 1.1

Розв'язання. Граф, зображений на рис. 1.2*а*, є неорієнтовним та має 5 вершин та 7 ребер. Спочатку побудуємо матрицю інцидентності; вона буде мати розмір 5×7. Для зручності також покажемо заголовки рядків і стовпців матриці. В результаті матриця інцидентності матиме вигляд:

$$E = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} \begin{array}{c} \begin{array}{ccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}.$$

Запишемо для графа рис.1.2*а* список ребер. Результат представлено в табл. 1.1.

Таблиця 1.1 – Список ребер графа

Ребро	Вершини
e_1	(v_1, v_2)
e_2	(v_2, v_3)
e_3	(v_3, v_4)
e_4	(v_4, v_5)
e_5	(v_1, v_6)
e_6	(v_2, v_5)
e_7	(v_2, v_4)

Нарешті, для неорієнтованого графа побудуємо матрицю суміжності розміром 5×5 (за кількістю його вершин).

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

Аналогічно розглянемо орієнтовний граф, зображений на рис. 1.2*б*. Він має 7 вершин та 9 дуг, причому дуга e_6 є петлею та входить до вершини v_5.

Матриця інцидентності матиме вигляд:

$$E = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} \begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \begin{pmatrix} -1 & -1 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

Запишемо для графа з рис.1.2*б* список ребер. Результат представлено в табл. 1.2.

Таблиця 1.2 – Список ребер графа

Ребро	Вершини
e_1	(v_1, v_2)
e_2	(v_1, v_3)
e_3	(v_2, v_3)
e_4	(v_1, v_4)
e_5	(v_4, v_5)
e_6	(v_5, v_5)
e_7	(v_1, v_6)
e_8	(v_6, v_7)
e_9	(v_2, v_7)

Матриця суміжності орієнтовного графа має розмір 7×7.

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

1.3. Підграфи

Граф $G' = (V', E')$ називається *підграфом* графа $G = (V, E)$, якщо $V'(G') \subseteq V(G)$ і $E'(G') \subseteq E(G)$.

Всякий підграф може бути отриманий з графа видаленням деяких вершин і ребер.

На рис. 1.3 представлено граф G і його підграфи G_1, G_2, G_3, G_4.

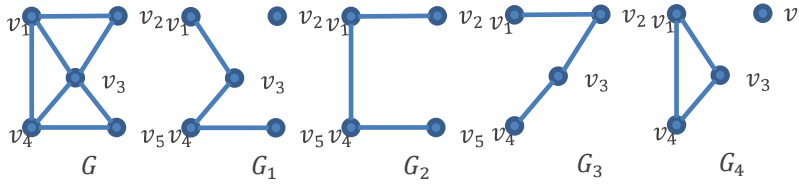


Рис. 1.3. Граф G і його підграфи

Підграф G' графа G називається *кістяковим*, якщо $V'(G') = V(G)$.

Кістяковий підграф може бути отриманий з графа видаленням деяких ребер, вершини ж залишаються незмінними. З представлених на рис. 1.3 кістяковим є тільки підграф G_3.

1.4. Степені вершин графа

Кількість ребер, інцидентних даній вершині називається її *степенем* або *валентністю* та позначають $\deg(v)$.

Якщо задано матрицю суміжності A, то

$$\deg(v_i) = \sum_{j=1}^{n} a_{ij}.$$

Приклад 1.2. Визначити степені вершин графа, поданого на рис.1.2а.

Розв'язання. Розглядаючи матрицю суміжності, отримаємо, що заданий граф має такі степені вершин:

$$\deg(v_1) = 2,$$
$$\deg(v_2) = 4,$$
$$\deg(v_3) = 2,$$
$$\deg(v_4) = 3,$$
$$\deg(v_5) = 3.$$

При визначенні степені вершини ребро, що утворює петлю, враховується двічі.

Якщо граф орієнтований, то число дуг, що входять до вершини, називають *напівстепенем заходу* та позначають $\deg(v)^+$, а число дуг, що виходять з вершини – *напівстепенем виходу* та позначають $\deg(v)^-$.

Тоді степінь вершини орієнтовного графа

$$\deg(v) = \deg(v)^+ + \deg(v)^-.$$

Вершина степені 0 називається *ізольованою*, а вершина степені 1 – *підвішеною* або *кінцевою*.

Введемо позначення для числа вершин і числа ребер в графі: $n = |V|$, $m = |E|$.

Величини n і m є основними числовими характеристиками графа або його *основними інваріантами*.

Для будь-якого графа має місце лема Ейлера про «потискання рук».

Лема Ейлера: Сума степенів усіх вершин графа – парна і дорівнює подвоєній кількості ребер графа:

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

Цю лему можна сформулювати неформально у такому вигляді: якщо кілька осіб потиснуть один одному руки, то загальна кількість рук, які були потиснуті, буде парною, адже в кожному потисканні беруть участь дві руки.

Наслідком наведеної леми є таке **твердження**: в довільному графі кількість вершин непарної степені є парною.

Простий граф називається *повним*, якщо будь-які його дві вершини є суміжними. Повний граф з n вершинами позначається через K_n. Приклад графа K_4 зображено на рис. 1.1*б*.

Якщо степені всіх вершин графа рівні між собою, наприклад дорівнюють n, то граф називається *однорідним* або *регулярним степені* n.

Кожний повний граф є регулярним графом степені $n-1$.

1.5. Маршрути, ланцюги та цикли

Нехай $G = (V, E)$ – неорієнтований граф.

Маршрутом M у графі G називається така скінченна або нескінченна послідовність вершин і ребер, які чергуються так, що кожні два сусідні ребра e_{i-1} та e_i мають спільну інцидентну вершину v_i:

$$(\dots, v_1, e_1, v_2, e_2, \dots, v_{n-1}, e_{n-1}, v_n, \dots).$$

Маршрут M можна задавати послідовністю його вершин $(\dots, v_1, v_2, \dots, v_n, \dots)$ (у звичайному графі), а також послідовністю ребер $(\dots, e_1, e_2, \dots, e_n, \dots)$. Одне і те саме ребро може зустрічатися в маршруті кілька разів.

Надалі будуть розглядатися в основному скінченні маршрути. В таких маршрутах існують перше (e_1) та останнє (e_n) ребра.

Вершина v_0, інцидентна ребру e_1, називається *початком маршруту*.

Якщо ребра e_1 та e_2 – кратні, то потрібна спеціальна вказівка, яку з двох інцидентних ним вершин слід вважати початком маршруту.

Аналогічно вводиться *кінець маршруту*.

Вершини, інцидентні ребрам маршруту, крім початкової і кінцевої, називаються *внутрішніми*, або *проміжними*.

Оскільки різні ребра маршруту можуть бути інцидентними одній і тій самій вершині, початок або кінець маршруту може одночасно виявитися внутрішньою вершиною.

Нехай маршрут $M(e_1, e_2, \ldots, e_n)$ має початок v_0 і кінець v_n. Тоді його називають *сполучним*.

Число ребер маршруту є його *довжиною*.

Якщо $v_0 = v_n$, то маршрут називають *замкненим*, або *циклічним*.

Відрізок $(e_i, e_{i+1}, \ldots, e_j)$ скінченного або нескінченного маршруту M є маршрутом. Він називається *ділянкою маршруту* M.

Маршрут M називається *ланцюгом*, якщо кожне ребро зустрічається в ньому не більше одного разу, і *простим ланцюгом*, якщо будь-яка вершина, крім, можливо, початкової, зустрічається в ньому не більш як один раз.

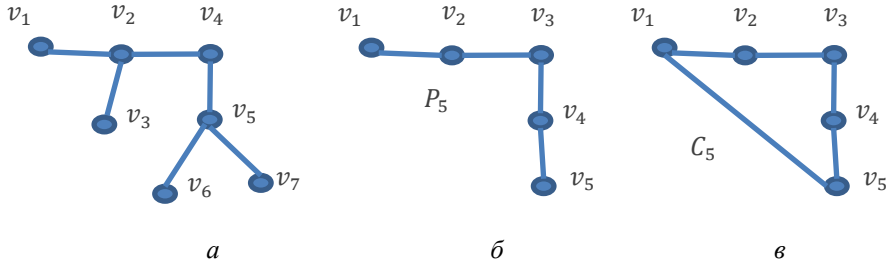
Якщо ланцюг є замкненим, то його називають *циклом*, а якщо простий ланцюг – замкнений, то його називають *простим циклом*.

Граф, якій не містить циклів, називається *ациклічним*. Приклад ациклічного графа зображено на рис. 1.4*а*.

В орієнтованому графі маршрут називається *шляхом*. Відповідно можна перенести також визначення ланцюга, простого ланцюга та циклу. Так, простий цикл в орієнтованому графі ще називається *контуром*.

Граф, якій складається з простого ланцюга з k вершинами, позначається P_k. На рис. 1.4*б* представлено приклад графа P_5.

Граф, якій складається з простого циклу з k вершинами, позначається C_k. На рис. 1.4*в* представлено приклад графа C_5.



а *б* *в*

Рис. 1.4. Приклад ациклічного графа, простого ланцюга і простого циклу

Дві вершини графа v і w називаються *зв'язаними*, якщо існує маршрут M з кінцями v та w.

Граф називається *зв'язним*, якщо будь-яка пара його вершин є зв'язаною, і *незв'язним*, якщо ця умова не виконується.

1.6. Властивості матриці суміжності графа

Матрицю суміжності A графа $G = (V, E)$ можна використовувати для підрахунку кількості різних *маршрутів* (*шляхів* для орієнтовних графів).

Сама матриця A задає ребра графа G, тобто маршрути довжиною 1. Матриця A^k (k-*та степінь* A) задає число маршрутів довжиною k ребер.

Теорема. Елемент $a_{ij}^{(k)}$ матриці A^k графа G дорівнює кількості маршрутів довжини k ребер з вершини v_i у вершину v_j.

Ця теорема справедлива як для неорієнтованих, так і для орієнтованих графів.

Наслідок 1. Елемент b_{ij} матриці $B = A + A^2 + .. + A^k$ графа G дорівнює кількості всіх маршрутів довжини не більше за k ребер з вершини v_i у вершину v_j.

Наслідок 2. В графі $G = (V, E)$ тоді і тільки тоді існує цикл, що містить вершину v_i, коли елемент b_{ii} матриці $B = A + A^2 + .. + A^n$ не дорівнює нулю.

Приклад 1.3. Для графа, зображеного на рис.1.2a, знайти кількість маршрутів довжиною 3 ребра з вершини v_1 до всіх інших вершин та перевірити його на наявність циклів.

Розв'язання. Згідно з теоремою для знаходження кількості маршрутів довжиною в 3 ребра необхідно піднести матрицю суміжності графа до третього степеню. Матриця суміжності графа

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

$$A^2 = \begin{pmatrix} 2 & 1 & 1 & 2 & 1 \\ 1 & 4 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 3 & 1 \\ 1 & 2 & 2 & 1 & 3 \end{pmatrix}, A^3 = \begin{pmatrix} 2 & 6 & 3 & 3 & 5 \\ 6 & 6 & 6 & 7 & 7 \\ 3 & 6 & 2 & 5 & 3 \\ 3 & 7 & 5 & 4 & 7 \\ 5 & 7 & 3 & 7 & 4 \end{pmatrix}.$$

Перший рядок матриці A^3 показує кількість маршрутів від вершини v_1 до всіх інших вершин графа, тобто є 6 маршрутів до вершини v_2, по 3 маршрути до вершин v_3 і v_4, а також 5 маршрутів до вершини v_5.

Так, наприклад, маршрути з вершини v_1 до вершини v_3 містять такі ребра: (e_5, e_4, e_3), (e_5, e_6, e_2), (e_1, e_7, e_3). Із v_1 до вершини v_5 є 5 маршрутів, але тільки один, а саме (e_1, e_4, e_7), містить ребра, що не повторюються. Всі інші

маршрути передбачають проходження по ребрам декілька разів: (e_1, e_1, e_5), (e_5, e_4, e_4), (e_5, e_6, e_6), (e_5, e_5, e_5).

Згідно з наслідком 2, вершина v_i графа входить у деякий цикл, якщо відповідний елемент b_{ii} матриці B, утвореною сумою степеней матриці суміжності, не дорівнює нулю. Для другого ступеня матриці суміжності A^2 всі діагональні елементи є ненульовими, а елементи матриці суміжності – додатні числа. Отже, за властивостями матриць всі діагональні елементи матриці B теж будуть ненульовими. Іншими словами, всі вершини графа, що розглядається, входять до одного чи декількох циклів.

Зауваження. На практиці для зменшення кількості обчислень при перевірці наявності циклів у графі можна обмежитися зведенням у ступінь матриці суміжності і порівнянням з нулем її діагональних елементів. При цьому відсутність циклів у графі необхідно перевіряти за діагональними елементами матриці B.

Ще одним практичним підходом для перевірки наявності циклів в орієнтованому графі є застосування *алгоритму видалення джерел та стоків графа*.

Назвемо *джерелом* вершину графа, до якої не входить жодної дуги, *стоком* – вершину, з якої не виходить жодної дуги. Тоді алгоритм видалення джерел та стоків полягає в наступному.

Крок 1. Знайти вершину-джерело, видалити її і всі інциндентні цій вершині дуги.

Крок 2. Повторювати крок 1 доки в графі є вершини-джерела, в іншому випадку перейти до кроку 3.

Крок 3. Знайти вершину-стік, видалити її і всі інциндентні цій вершині дуги.

Крок 4. Повторювати крок 3 доки в графі є вершини-стоки, в іншому випадку перейти до кроку 5.

Крок 5. Якщо отримано порожній граф, то цикли в вихідному графі відсутні. В іншому випадку отриманий підграф містить цикли вихідного графа.

Приклад 1.4. Перевірити орієнтовний граф G, зображений на рис.1.5, на наявність циклів за допомогою алгоритму видалення джерел та стоків.

Розв'язання. Згідно з алгоритмом шукаємо вершину-джерело. Заданий граф G містить два джерела – це вершини v_1 і v_2. Виберемо вершину v_1, видалимо її та інцидентні цій вершині дуги. В результаті отримаємо підграф G_1, зображений на рис. 1.6.

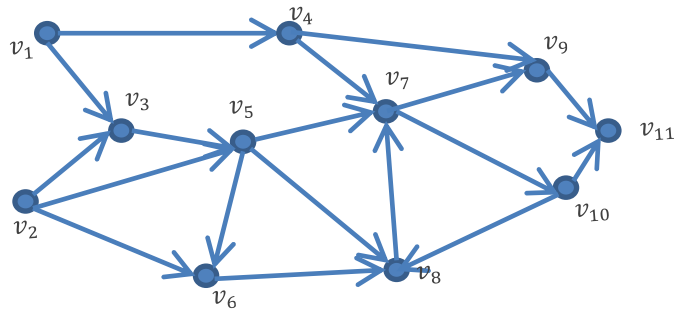


Рис. 1.5. Орієнтовний граф G для прикладу 1.5

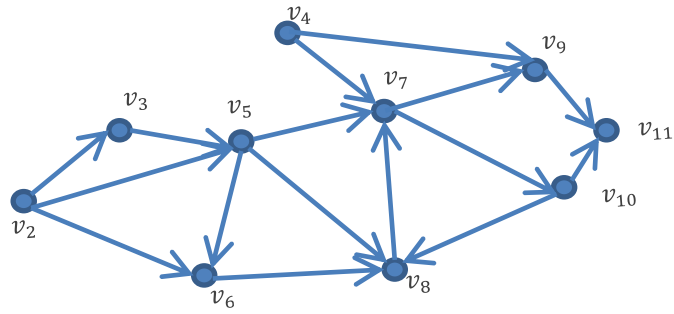


Рис. 1.6. Підграф G_1

Далі процес видалення джерел продовжимо, розглядаючи отриманий підграф як вихідний. Підграф G_1 має дві вершини-джерела – v_2 і v_4. Виберемо вершину v_2, видалимо її та інцидентні цій вершині дуги. В результаті отримаємо підграф G_2, зображений на рис. 1.7.

Підграф G_2 також має дві вершини-джерела – v_3 і v_4. Виберемо вершину v_3, видалимо її та інцидентні цій вершині дуги. В результаті отримаємо підграф G_3, зображений на рис. 1.8.

Отриманий підграф G_3 також має дві вершини-джерела – v_4 і v_5. Аналогічно виберемо вершину v_4, видалимо її та інцидентні цій вершині дуги. В результаті отримаємо підграф G_4, зображений на рис. 1.9.

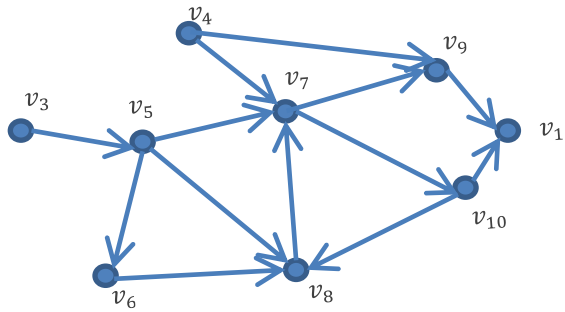


Рис. 1.7. Підграф G_2

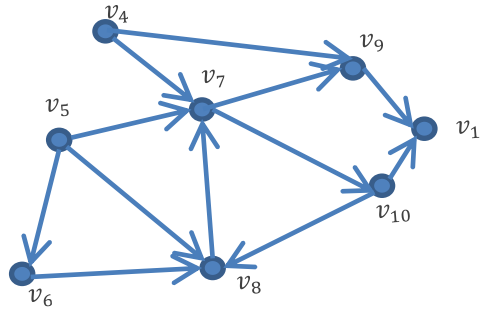


Рис. 1.8. Підграф G_3

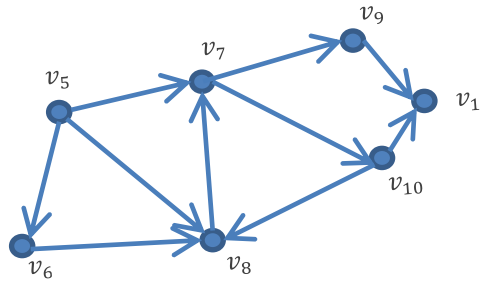


Рис. 1.9. Підграф G_4

Шукаємо в підграфі G_4 вершину-джерело. Це буде вершина v_5. Видаляємо її та інцидентні їй дуги. В результаті отримаємо підграф G_5, зображений на рис. 1.10.

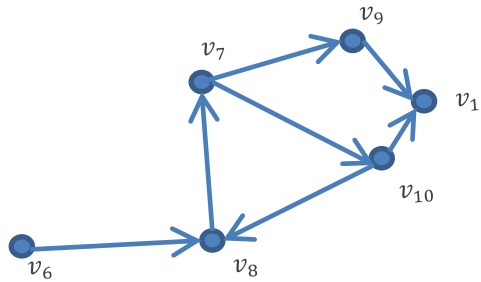


Рис. 1.10. Підграф G_5

Розглядаючи підграф G_5, бачимо, що вершина v_6 є джерелом. Тому видаляємо з підграфа G_5 вершину v_6 та інцидентні їх дуги. В результаті отримаємо підграф G_6, зображений на рис. 1.11.

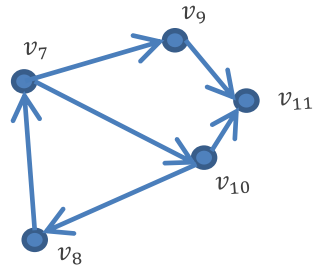


Рис. 1.11. Підграф G_6

Підграф G_6 не має вершини-джерела, тому шукаємо вершину-стік. Цією вершиною буде v_{11}. Видаляємо вершину v_{11} та інцидентні їй ребра. В результаті отримаємо підграф G_7, зображений на рис. 1.12.

Процес продовжуємо, шукаючи в отриманому підграфі стоки. В підграфі G_7, вершина v_9 є стоком. Видаляємо вершину v_9 та інцидентні їй ребра. В результаті отримаємо підграф G_8, зображений на рис. 1.13.

Підграф G_8 не містить вершин-стоків і не є порожнім, тобто підграф G_8 містить цикл вихідного графа G.

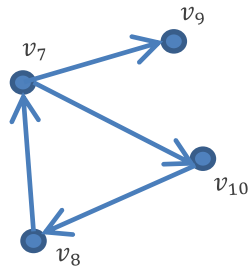


Рис. 1.12. Підграф G_7

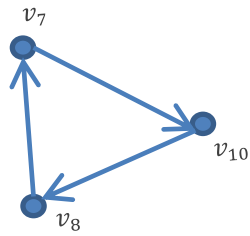


Рис. 1.13. Підграф G_8

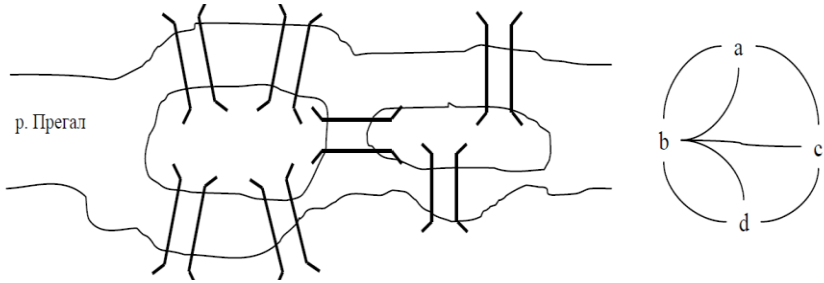
1.7. Ейлерові графи

Одна з перших задач теорії графів опублікована в працях видатного математика XVIII сторіччя Л. Ейлера. Це *задача щодо кенігсберзьких мостів*.

В часи Л. Ейлера річка Прегал розділяла місто Кенігсберг на дві частини, а її рукави утворювали по середині міста два остови. Всі частини міста з'єднувалися між собою за допомогою мостів, що схематично показано на рис. 1.14*а*. Питання задачі: якщо вийти з деякої частини міста, чи можна пройти кожен міст один раз і повернутися в початкову частину міста? На базі схеми міста можна побудувати граф, в якому кожній частині міста відповідає вершина, а кожному мосту – ребро, інцидентне вершинам, тобто частинам міста, що з'єднуються цим мостом (рис. 1.14*б*).

Якщо існує цикл у графі, в якому кожне ребро бере участь тільки один раз, то такий цикл називається *ейлеровим циклом*, а граф, що містить такий цикл, – *ейлеровим графом*. Приклади ейлерових графів зображено на

рис. 1.15. Для графа рис. 1.15*а* ейлеров цикл можна записати наступною послідовністю його вершин $(v_1, v_2, v_3, v_6, v_5, v_4, v_6, v_2, v_5, v_1)$.

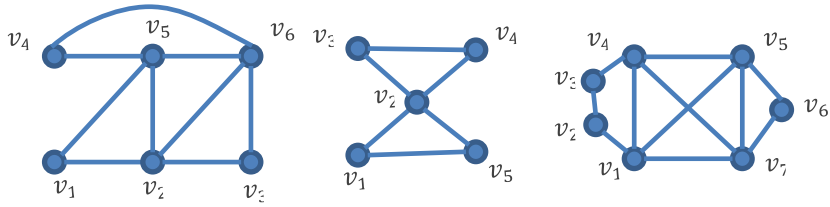


а *б*

Рис. 1.14. Ілюстрація до задачі щодо мостів Кенігсберґа

Теорема (Ейлера). Зв'язний граф є ейлеровим тоді і тільки тоді, коли степені усіх вершин є парними.

Ланцюг, який містить усі ребра графа називається *ланцюгом Ейлера*. Граф, для якого існує ланцюг Ейлера, називається *напівейлеровим графом*. Прикладами напівейлерового графа є граф, зображений на рис. 1.2*а*, а також граф G з рис. 1.3. Для графа з рис. 1.2*а* ейлерів ланцюг $(e_3, e_2, e_1, e_5, e_4, e_7, e_6)$.



а *б* *в*

Рис. 1.15. Приклади ейлерових графів

Теорема. Зв'язний граф є напівейлеровим тоді і тільки тоді, коли тільки дві його вершини мають непарну степінь.

Повертаючись до графа із задачі про кенігсберзькі мости (рис. 1.14*б*), можна сказати що він не є ні ейлеревим, ні напівейлеровим, оскільки степені усіх його чотирьох вершин непарні.

1.8. Гамільтонові графи

Цикл називається *гамільтоновим*, якщо він проходить через кожну вершину графа (за винятком крайньої) точно один раз.

Шлях називається *гамільтоновим*, якщо він проходить через кожну вершину графа точно один раз.

Гамільтонів цикл та гамільтонів шлях завжди є простими. Вони можуть не містити всіх ребер графа.

Граф називається *гамільтоновим*, якщо він має гамільтоновий цикл. Приклад гамільтонова графа подано на рис. 1.16а. Один з можливих гамільтонових циклів можна записати наступною послідовністю вершин $(v_1, v_2, v_3, v_4, v_5)$.

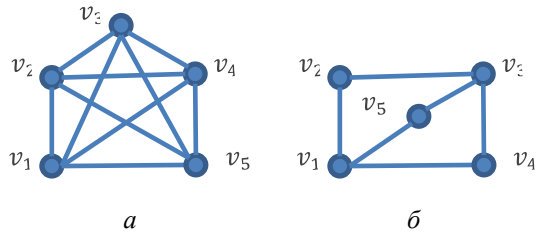


Рис. 1.16. Приклад гамільтонова і квазігамільтонова графів

Граф називається *квазігамільтоновим*, якщо він має гамільтонів шлях. Приклад квізігамільтонова графа подано на рис. 1.16б. Для нього гамільтонів шлях задається послідовністю вершин $(v_2, v_3, v_4, v_1, v_5)$.

Гамільтонів цикл названо іменем ірландського математика Вільямса Гамільтона, який 1859 році почав розглядати *задачу мандрівника:* обійти всі столиці заданих країн, тобто вершин графа, по одному разу і повернутися в першу.

Зауваження. Гамільтонів цикл існує далеко не в кожному графі. Через кожні дві вершини графа може проходити простий цикл, а гамільтонів цикл при цьому може бути відсутній.

Достатні умови гамільтоновості графа сформульовано в вигляді наступної теореми.

Теорема Дірака. Нехай G – неорієнтований граф порядка n і m – мінімальна степінь його вершин. Якщо $n \geq 3$ і $m \geq n/2$, то G – гамільтонів граф.

Приклад 1.5. Використовуючи теорему Дірака, перевірити наявність гамільтонова циклу в графі, поданого на рис.1.2а.

Розв'язання. Як було визначено в прикладі 1.2, мінімальна степінь вершин графа $m = 2$. Заданий граф має 5 вершин, тобто $n = 5$. Оскільки $m \geq 5/2$, то, за теоремою Дірака, граф є гамільтоновим. Гамільтонів цикл можна записати наступною послідовністю вершин $(v_1, v_2, v_3, v_4, v_5)$.

1.9. Дерева

Деревом називають зв'язний граф без простих циклів. З означення випливає, що дерева являють собою прості графи. Приклади дерев подано на рис. 1.4а і 1.4б.

Нехай граф G має n вершин. Тоді такі **твердження** еквівалентні:

- граф G – дерево;
- граф G не містить простих циклів і має $(n-1)$ ребро;
- граф G зв'язний і має $(n-1)$ ребро;
- граф G зв'язний, але вилучення довільного ребра робить його незв'язним;
- довільні дві вершини графу G з'єднані точно одним простим маршрутом;
- граф G не містить простих циклів, але, додавши до нього нове ребро, ми отримаємо точно один простий цикл.

Нехай G – зв'язний граф і $\{Z_i\}$ – сукупність усіх його циклів.

Максимальний зв'язний граф T, що не містить циклів і покриває всі вершини, називається *максимальним деревом* у графі G.

Максимальне дерево є *кістяком*, оскільки з графа вилучено всі цикли. Приклад графа і його кістяка подано на рис. 1.17.

Задача визначення кількості усіх кістяків графа G у загальному випадку розв'язується за допомогою матриці Кірхгофа.

Матрицею Кірхгофа $K(G)$ простого зв'язного графа $G = (E, V)$, $|V| = n$ називається квадратна матриця, елементи якої визначаються за формулою:

$$k_{ij} = \begin{cases} -1, \text{ якщо вершини } v_i \text{ та } v_j \text{ суміжні;} \\ 0, \text{ якщо вершини } v_i \text{ та } v_j \text{ несуміжні;} \\ \deg(v_i), \ i = j. \end{cases}$$

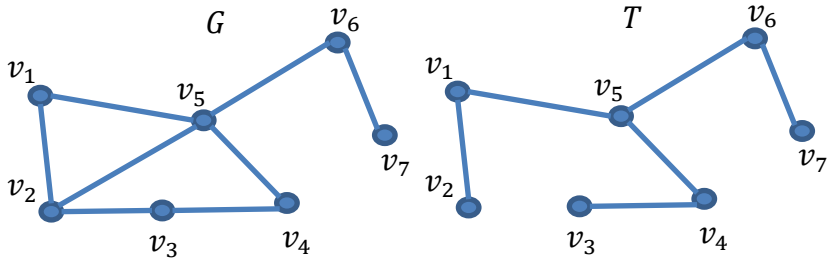


Рис. 1.17. Приклад графа і його кістяка

Твердження. Алгебраїчні доповнення K_{ij} усіх елементів матриці Кірхгофа рівні між собою.

Теорема 1. Кількість кістякових дерев у простому зв'язному графі з n $(n \geq 2)$ вершинами дорівнює алгебраїчному доповненню довільного елемента матриці Кірхгофа.

Теорема 2. Кількість ребер неорієнтованого графа G, які необхідно видалити для отримання кістяка, не залежить від послідовності їх видалення і дорівнює $m - n + 1$, де m – кількість ребер, n – кількість вершин.

Нехай $G = (V, E)$ – n-вершинний неорієнтований зв'язний граф, що містить m ребер. Тоді:

- $v(G) = m - n + 1$ – *цикломатичне число* або *циклічний ранг графа*;
- $v^*(G) = n - 1$ – *коциклічний ранг* або *кораng графа*;
- $v^*(G)$ дорівнює числу ребер, що входять в будь-який кістяк графа;
- $v(G) + v^*(G) = m$.

Приклад 1.6. Для графа, поданого на рис. 1.2а, визначити його цикломатичне число, коциклічний ранг, а також кількість кістякових дерев.

Розв'язання. Граф, зображений на рис. 1.2*а*, має 5 вершин та 7 ребер, тобто $n = 5,\ m = 7$. Тоді цикломатичне число графа

$$v(G) = m - n + 1 = 7 - 5 + 1 = 3,$$

тобто для отримання кістяка необхідно видалити три ребра. Коциклічний ранг графа

$$v^*(G) = n - 1 = 5 - 1 = 4.$$

Для знаходження кількості кістякових дерев графа побудуємо матрицю Кірхгофа. Її можна отримати з матриці суміжності шляхом заміни 1 на -1, а

діагональних елементів – на степені відповідних вершин. Таким чином, матриця Кірхгофа матиме вигляд:

$$K(G) = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{pmatrix}.$$

Для визначення кількісті кістякових дерев візьмемо алгебраїчне доповнення будь-якого елемента матриці Кірхгофа. Нехай це буде K_{22}.

$$K_{22} = \begin{vmatrix} 2 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ -1 & 0 & -1 & 3 \end{vmatrix} = 21,$$

тобто граф має 21 кістякове дерево. Зобразимо сам граф та три з його кістякових дерев. Результат представлено на рис. 1.18.

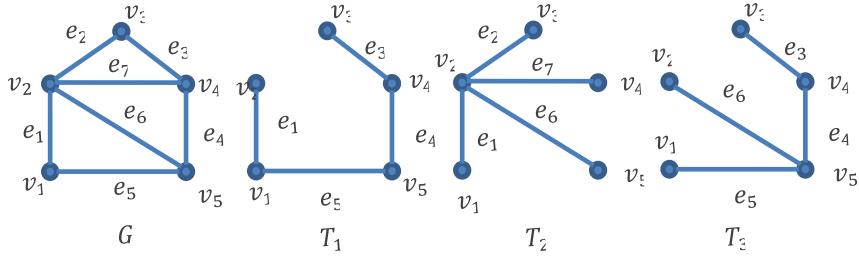


Рис. 1.18. Граф і приклади його кістякових дерев

1.10. Задача про кістяк екстремальної ваги

Задача про знаходження мінімального кістякового дерева виникає при проектуванні ліній електропередачі, трубопроводів, доріг тощо, коли потрібно задані центри з'єднати між собою деякою системою каналів зв'язку. При цьому будь-які два центри мають бути пов'язаними або безпосередньо, або через інші центри і канали так, що загальна довжина (чи, наприклад, вартість) каналів зв'язку буде мінімальною. Задані центри можна вважати вершинами графа, а ваги ребер встановити рівними довжинам (вартостям) каналів, що їх з'єднують. Тоді шукана мережа буде кістяковим деревом найменшої довжини (вартості).

Розв'язання цієї задачі «сліпим» перебором варіантів вимагає надзвичайно великих обчислень навіть при відносно малих розмірах графа. Однак для її розв'язання існують більш ефективні алгоритми (Борувки, Прима, Крускала та ін.).

В алгоритмі Дж. Крускала (1956 р.) на кожному кроці будується ациклічний граф, тобто паралельно може будуватися декілька дерев, які об'єднуються в єдине дерево в процесі роботи алгоритму.

Практична реалізація алгоритму Крускала полягає в наступному.

Крок 1. Ребра відсортувати за зростанням ваги.

Крок 2. Обрати ребро з найменшою вагою, додати обране ребро в дерево та покрасити це ребро в перший колір.

Крок 3. Зі списку ребер вибрати наступне ребро з найменшою вагою. Тоді можливі випадки, де вершини обраного ребра будуть:

- обидві без кольору – покрасити вершини і ребро в наступний колір;
- одна кольорова, інша ні – покрасити безбарвну вершину і ребро в колір розфарбованої вершини;
- різнокольорові – перекрасити всі ребра і вершини кольору, що введений пізніше, в колір, що був введений раніше;
- однокольорові – пропустити це ребро та обрати наступне.

Крок 4. Додати ребро до дерева. Якщо всі вершини графа додані до дерева, то мінімальне кістякове дерево знайдено і алгоритм завершено. В іншому випадку – перейти до кроку 3.

Роботу алгоритму розглянемо на прикладі.

Приклад 1.7. Для графа, поданого на рис.1.19, знайти мінімальне кістякове дерево.

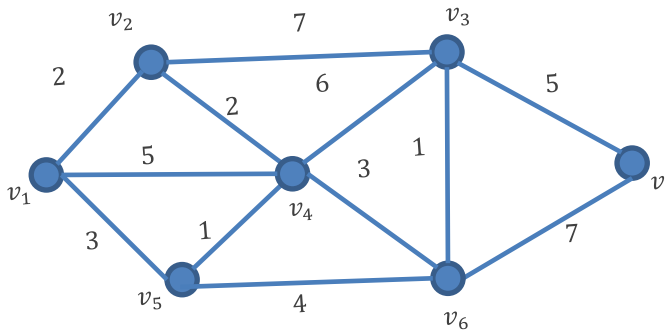


Рис. 1.19. Граф для прикладу 1.5

Розв'язання. Представимо граф у вигляді списку ребер і відсортуємо ребра за зростанням ваги. Результат подано в табл. 1.3.

Таблиця 1.3 – Впорядкований список ребер

Вага ребра	Вершини
1	$(v_3, v_6), (v_4, v_5)$
2	$(v_1, v_2), (v_2, v_4)$
3	$(v_1, v_5), (v_4, v_6)$
4	(v_5, v_6)
5	$(v_1, v_4), (v_3, v_7)$
6	(v_3, v_4)
7	$(v_2, v_3), (v_6, v_7)$

Введемо множину C, в якій будемо зберігати кольори кожної вершини графа. Спочатку жодна вершина не має кольору, тобто

$$C = \{0, 0, 0, 0, 0, 0, 0\}.$$

Обираємо перше ребро з мінімальною вагою, фарбуємо вершини в перший колір, наприклад червоний, та додаємо ребро до кістякового дерева. В нас це ребро (v_3, v_6), тоді множина кольорів вершин прийме вигляд

$$C = \{0, 0, 1, 0, 0, 1, 0\}.$$

Далі обираємо наступне ребро з мінімальною вагою. Це ребро (v_4, v_5). Вершини v_4 і v_5 нефарбовані, тому ребро (v_4, v_5) додаємо до кістяка, а вершини фарбуємо в другий колір. В результаті множина кольорів вершин матиме вигляд

$$C = \{0, 0, 1, 2, 2, 1, 0\}.$$

Процес продовжуємо. Наступне ребро (v_1, v_2), відповідні вершини нефарбовані, тому фарбуємо їх в наступний колір, а ребро додаємо до дерева, в результаті отримаємо

$$C = \{3, 3, 1, 2, 2, 1, 0\}.$$

Наступне ребро (v_2, v_4). Вершини пофарбовані в різні кольори: v_2 має третій колір, а v_4 – другий. Тому для вершини v_2 замінюємо колір на другий і перефарбовуємо в другий колір всі вершини, що пофарбовані в третій. Ребро (v_2, v_4) додаємо до кістякового дерева. Множина кольорів вершин прийме вигляд

$$C = \{2, 2, 1, 2, 2, 1, 0\}.$$

Обираємо наступне ребро (v_1, v_5). Вершини v_1 і v_5 пофабровані однаково (в другий колір), тому ребро (v_1, v_5) пропускаємо.

Наступне ребро (v_4, v_6). Вершини пофабровані в різні кольори: v_4 має другий колір, а v_6 – перший. Тому для вершини v_4 замінюємо колір на перший і перефарбовуємо в другий колір всі вершини, що раніше були пофарбовані в другий колір. Ребро (v_4, v_6) додаємо до кістякового дерева. Множина кольорів вершин прийме вигляд

$$C = \{1, 1, 1, 1, 1, 1, 0\}.$$

Наступне ребро (v_5, v_6). Вершини v_5 і v_6 пофабровані в перший колір, тому ребро (v_5, v_6) пропускаємо.

Обираємо наступне ребро – (v_1, v_4). Аналогічно попередньому кроку, вершини v_1 і v_4 пофабровані в перший колір, тому ребро (v_1, v_4) пропускаємо.

За алгоритмом обираємо наступне ребро – (v_3, v_7). Вершина v_3 пофарбована в перший колір, вершина v_7 – нефарбована, тобто вершину v_7 фарбуємо в перший колір, а ребро (v_3, v_7) додаємо до кістякового дерева. Множина кольорів вершин прийме вигляд

$$C = \{1, 1, 1, 1, 1, 1, 1\},$$

тобто всі вершини пофарбовані в один колір. Таким чином, сформовано мінімальне кістякове дерево графа, що виділено кольором на графі, поданому на рис. 1.20. Вага мінімального кістякового дерева – це сума ваг ребер, що входять до нього, тобто

$$L = 1 + 1 + 2 + 2 + 3 + 5 = 14.$$

На цьому робота алгоритму припиняється.

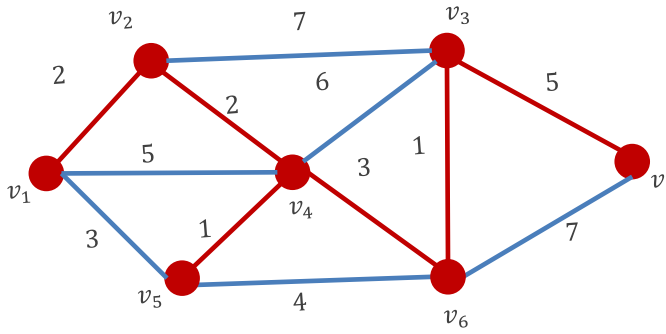


Рис. 1.20. Мінімальне кістякове дерево вихідного графа

1.11. Алгоритм Дейкстри знаходження найкоротшого шляху

Загальний підхід до розв'язання задачі знаходження найкоротшого шляху запропонував американський математик Річард Беллман, який ввів термін *динамічне програмування*. Задача про найкоротший шлях – це окремий випадок наступної задачі: знайти на графі шляхи, що з'єднують дві задані вершини і дають максимум або мінімум деякої адитивної функції, визначеної на шляхах. Найчастіше ця функція трактується як довжина шляху, і задача називається *задачею про найкоротший шлях*.

Постановка задачі знаходження найкоротшого шляху полягає в наступному. Нехай задано:

- орієнтований граф $G = (V, E)$,
- початкова вершина v_1,
- кінцева вершина v_k, причому $v_1,\ v_k\ \in V$,
- ваги дуг задано в матриці суміжності (*ваговій матриці*) A. Якщо вершини i і j несуміжні, то $a_{ij} = \infty$.

Шлях між вершинами v_0 і v_k задається послідовністю ребер $v_1,\ v_2,\ \cdots, v_{k-1}, v_k$ і оцінюється величиною

$$p = \sum_{i=2}^{k} a_{v_{i-1}, v_i}.$$

Необхідно знайти шлях мінімальної довжини. Задача поділяється на *дві підзадачі:*

- знаходження мінімальної оцінки довжини шляху;
- знаходження відповідного шляху.

Алгоритм Дейкстри – це одна з можливих реалізацій задачі знаходження найкоротшого шляху. Його також часто називають *алгоритмом розташування міток*.

В процесі роботи алгоритму Дейкстри вузлам мережі $v_i\ \in V$ приписуються числа (*мітки*) $d(v_i)$, що є оцінками довжини (ваги) найкоротшого шляху від вершини v_1 до вершини v_i. Якщо вершина v_i на деякому кроці отримала мітку $d(v_i)$, це означає, що в графі G існує шлях з від вершини v_1 до вершини v_i з вагою $d(v_i)$. Мітки можуть знаходитеся в двох станах – бути *тимчасовими* або *постійними*. Перетворення мітки в постійну означає, що найкоротший шлях від вершини v_1 до відповідної вершин знайдено.

Алгоритм Дейкстри має одне *обмеження* – ваги дуг графа повинні бути додатними величинами.

Практична реалізація алгоритму Дейкстри полягає в наступному.

Етап 1. Знаходження довжини найкоротшого шляху.

Крок 1. Присвоєння вершинам початкових міток.

Присвоюємо $d(v_1) = 0^*$ і вважаємо цю мітку постійною (постійні мітки будемо помічати символом «*»). Для всіх інших вершин $v_i \in V$ присвоюємо $d(v_1) = \infty$ і вважаємо ці мітки тимчасовими. Позначимо поточну вершину через $\tilde{v}$, тобто $\tilde{v} = v_1$.

Крок 2. Зміна міток.

Для кожної вершини v_i з тимчасовою міткою, що суміжна з вершиною $\tilde{v}$, змінюємо її мітку на нову за правилом

$$d_{\text{н}}(v_i) = \min\{d_c(v_i), d(\tilde{v}) + a_{\tilde{v},v_i}\},$$

де $d_c(v_i)$ – стара мітка вершини v_i;

$d(\tilde{v})$ – постійна мітка поточної вершини $\tilde{v}$;

$a_{\tilde{v},v_i}$ – відповідний елемент вагової матриці A.

Крок 3. Перетворення мітки з тимчасової в постійну.

З усіх вершин з тимчасовими мітками вибираємо вершину v_j^* з найменшим значенням мітки

$$d(v_j^*) = \min\{d(v_j), d(v_j) - \text{тимчасова}\}.$$

Перетворюємо цю мітку в постійну та присвоюємо $\tilde{v} = v_j^*$.

Крок 4. Перевірка на закінчення першого етапу.

Якщо $\tilde{v} = v_k$, то $d(\tilde{v})$ – довжина найкоротшого шляху між вершинами v_1 і v_k, в іншому випадку – перейти до кроку 2.

Етап 2. Побудова найкоротшого шляху.

Крок 5. Послідовний пошук дуг найкоротшого шляху.

Серед вершин, що суміжні з вершиною $\tilde{v}$, безпосередньо передують їй та мають постійні мітки, знаходимо вершину v_i, для якої виконується співвідношення

$$d(\tilde{v}) = d(v_i) + a_{\tilde{v},v_i}.$$

Додаємо дугу $(v_i, \tilde{v})$ в шуканий шлях та присвоюємо $\tilde{v} = v_i$.

Крок 6. Перевірка на закінчення другого етапу.

Якщо $\tilde{v} = v_1$, то найкоротший шлях між вершинами v_1 і v_k знайдено. Його утворює послідовність дуг, отриманих на п'ятому кроці в зворотному порядку. В протилежному випадку – повернутися до кроку 5.

Роботу алгоритму Дейкстри розглянемо на прикладі.

Приклад 1.8. Для графа, поданого на рис.1.19, знайти довжину найкоротших шляхів між вершиною v_1 і всіма іншими та вивести найкоротший шлях із v_1 до v_7.

Розв'язання. Складемо вагову матрицю для заданого графа

$$A = \begin{pmatrix} - & 2 & \infty & 5 & 3 & \infty & \infty \\ 2 & - & 7 & 2 & \infty & \infty & \infty \\ \infty & 7 & - & 6 & \infty & 1 & 5 \\ 5 & 2 & 6 & - & 1 & 3 & \infty \\ 3 & \infty & \infty & 1 & - & 4 & \infty \\ \infty & \infty & 1 & 3 & 4 & - & 7 \\ \infty & \infty & 5 & \infty & \infty & 7 & - \end{pmatrix}.$$

Етап 1.

Крок 1. Присвоюємо $d(v_1) = 0^*$, $\tilde{v} = v_1$.

$$d(v_2) = d(v_3) = \cdots = d(v_7) = \infty.$$

Ітерація 1.

Крок 2. Позначимо множину вершин з тимчасовими мітками, суміжних з $\tilde{v} = v_1$, через $S = \{v_2, v_4, v_5\}$. Оновлюємо тимчасові мітки цих вершин:

$$d(v_2) = \min\{\infty, 0 + 2\} = 2,$$
$$d(v_4) = \min\{\infty, 0 + 5\} = 5,$$
$$d(v_5) = \min\{\infty, 0 + 3\} = 3.$$

Крок 3. Одна з тимчасових міток змінюється на постійну:

$$\min\{0^*, 2, \infty, 5, 3, \infty, \infty\} = 2^* = d(v_2), \qquad \tilde{v} = v_2.$$

Крок 4. Всі вершини графа не мають постійних міток, тобто відстані з вершини v_1 до всіх вершин не знайдено, тому повертаємося до кроку 2.

Ітерація 2.

Крок 2. $S = \{v_3, v_4\}$.

$$d(v_3) = \min\{\infty, 2 + 7\} = 9,$$
$$d(v_4) = \min\{5, 2 + 2\} = 4.$$

Крок 3.

$$\min\{0^*, 2^*, 9, 4, 3, \infty, \infty\} = 3^* = d(v_5), \qquad \tilde{v} = v_5.$$

Крок 4. Всі вершини графа не мають постійних міток.

Ітерація 3.

Крок 2. $S = \{v_4, v_6\}$.

$$d(v_4) = \min\{4, 3 + 1\} = 4,$$
$$d(v_6) = \min\{\infty, 3 + 4\} = 7.$$

Крок 3.

$$\min\{0^*, 2^*, 9, 4^*, 3^*, 7, \infty\} = 4^* = d(v_4), \qquad \tilde{v} = v_4.$$

Крок 4. Всі вершини графа не мають постійних міток.

Ітерація 4.

Крок 2. $S = \{v_3, v_6\}$.

$$d(v_3) = \min\{9, 4 + 6\} = 9,$$
$$d(v_6) = \min\{7, 4 + 7\} = 7.$$

Крок 3.

$$\min\{0^*, 2^*, 9, 4^*, 3^*, 7, \infty\} = 7^* = d(v_6), \qquad \tilde{v} = v_6.$$

Крок 4. Всі вершини графа не мають постійних міток.

Ітерація 5.

Крок 2. $S = \{v_3, v_7\}$.

$$d(v_3) = \min\{9, 7 + 1\} = 8,$$
$$d(v_7) = \min\{\infty, 7 + 7\} = 14.$$

Крок 3.

$$\min\{0^*, 2^*, 8, 4^*, 3^*, 7^*, 14\} = 8^* = d(v_3), \qquad \tilde{v} = v_3.$$

Крок 4. Всі вершини графа не мають постійних міток.

Ітерація 6.

Крок 2. $S = \{v_7\}$.

$$d(v_7) = \min\{14, 8 + 5\} = 13.$$

Крок 3.

$$\min\{0^*, 2^*, 8^*, 4^*, 3^*, 7^*, 13\} = 13^* = d(v_7), \qquad \tilde{v} = v_7.$$

Крок 4. Нарешті, всі вершини графа мають постійні мітки, тобто знайдено довжини найкоротших шляхів між вершиною v_1 та всіма іншими вершинами. Тому переходимо до другого етапу – побудови найкоротшого шляху між вершинами v_1 і v_7.

Граф з постійними мітками, які характеризують довжини найкоротших шляхів між вершиною v_1 та всіма іншими вершинами, подано на рис. 1.21.

Етап 2.

Ітерація 1.

Крок 5. Записуємо множину вершин, суміжних з вершиною $\tilde{v} = v_7$ з постійними мітками, що безпосередньо їх передують, $S = \{v_3, v_6\}$. Перевіряємо для цих вершин виконання умови

$$d(\tilde{v}) = 13 = 8 + 5 = d(v_3) + a_{\tilde{v},v_3},$$
$$d(\tilde{v}) = 13 \neq 7 + 7 = d(v_6) + a_{\tilde{v},v_6}.$$

Включаємо ребро (v_3, v_7) до найкоротшого шляху, присвоюємо $\tilde{v} = v_3$.

Крок 6. $\tilde{v} \neq v_1$, тому процес побудови найкоротшого шляху продовжуємо, повторюючи крок 5.

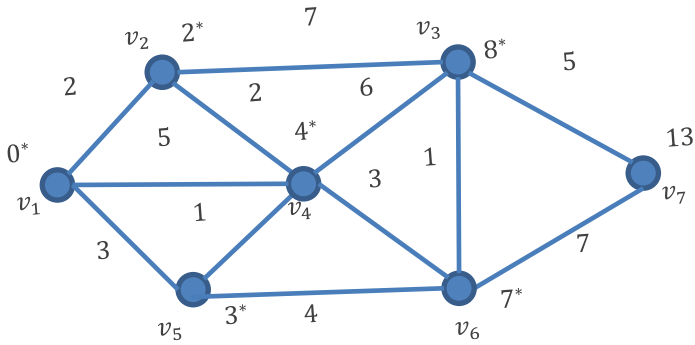


Рис. 1.21. Граф задачі після першого етапу

Ітерація 2.

Крок 5. $S = \{v_2, v_4, v_6\}$.

$$d(\tilde{v}) = 8 \neq 2 + 7 = d(v_2) + a_{\tilde{v},v_2},$$
$$d(\tilde{v}) = 8 \neq 4 + 6 = d(v_4) + a_{\tilde{v},v_4},$$
$$d(\tilde{v}) = 8 = 7 + 1 = d(v_6) + a_{\tilde{v},v_6}.$$

Включаємо ребро (v_3, v_6) до найкоротшого шляху, $\tilde{v} = v_6$.

Крок 6. $\tilde{v} \neq v_1$.

Ітерація 3.

Крок 5. $S = \{v_4, v_5\}$.

$$d(\tilde{v}) = 7 = 4 + 3 = d(v_4) + a_{\tilde{v},v_4},$$
$$d(\tilde{v}) = 7 = 3 + 4 = d(v_5) + a_{\tilde{v},v_5}.$$

Оскільки рівність виконується для двох ребер, то в графі існує декілька найкоротших шляхів між вершинами v_1 і v_7. За умовою задачі необхідно вивести один найкоротший шліх, тому виберемо перше ребро (v_3, v_4) та включаємо його до найкоротшого шляху, покладемо $\tilde{v} = v_4$.

Крок 6. $\tilde{v} \neq v_1$.

Ітерація 4.

Крок 5. $S = \{v_1, v_2, v_5\}$.

$$d(\tilde{v}) = 4 \neq 4 + 5 = d(v_1) + a_{\tilde{v},v_1},$$
$$d(\tilde{v}) = 4 = 2 + 2 = d(v_2) + a_{\tilde{v},v_2},$$
$$d(\tilde{v}) = 4 = 3 + 1 = d(v_5) + a_{\tilde{v},v_5}.$$

Як і на попередній ітерації, виберемо перше ребро, для якого виконується рівність, та включимо його до найкоротшого шляху. Це ребро (v_2, v_4), тому $\tilde{v} = v_2$.

Крок 6. $\tilde{v} \neq v_1$.

Ітерація 5.

Крок 5. $S = \{v_1\}$.

$$d(\tilde{v}) = 2 = 0 + 2 = d(v_1) + a_{\tilde{v},v_1}.$$

Включаємо ребро (v_1, v_2) до найкоротшого шляху, $\tilde{v} = v_1$.

Крок 6. $\tilde{v} = v_1$, тобто побудовано найкоротший шлях між вершинами v_1 і v_7, що складається з послідовності вибраних на другому етапі ребер. Його довжина становить 13. Знайдений найкоротший шлях на графі виділено кольором на рис. 1.22.

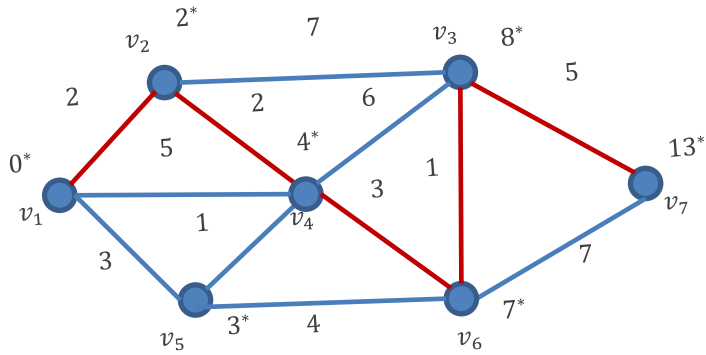


Рис. 1.22. Найкоротший шлях між вершинами v_1 і v_7

1.12. Елементи мережевого планування

1.12.1. Лінійні діаграми Ганта

Мережеве планування (англ. network planning) являє собою окремий випадок застосування теорії графів. Мережеве планування – це одна з форм управління проектами, що ґрунтується на графічному відображенні послідовності і тривалості виконання робіт, які необхідні для успішної реалізації проекту. Воно дозволяє синхронізувати виконання різних робіт у часі і оцінити загальну тривалість проекту.

Для планування робіт історично першими з'явилися *лінійні графіки*, запропоновані в 1910 році Генрі Л. Гантом. В літературі їх також називають *лінійними, або стрічковими діаграмами Ганта*.

Діаграма Ганта складається із *смуг*, орієнтованих уздовж осі часу. Кожна смуга на діаграмі представляє окреме завдання в складі проекту (вид роботи), її кінці відповідають моментам початку і завершення роботи, а протяжність – тривалості роботи.

Вертикальна вісь діаграми відбиває *перелік завдань*. Крім того, на діаграмі можуть бути відзначені сукупні завдання, відсотки завершення, покажчики послідовності і залежності робіт, мітки ключових моментів (віхи), мітка поточного моменту часу «Сьогодні» тощо.

Ключовим поняттям діаграми Ганта є *віха* – мітка значимого моменту в ході виконання робіт, спільна межа двох або більше завдань. Віхи дозволяють наочно відобразити необхідність дотримання послідовності у виконанні різних робіт. Віхи, як і інші межі на діаграмі, не є календарними датами. Зрушення віхи призводить до зрушення всього проекту. Тому діаграма Ганта не є, строго кажучи, графіком робіт.

Крім того, діаграма Ганта не відображує значущості та ресурсоємності робіт, а також їх сутності, керівників, виконавців та інших потенційно важливих з точки зору управління факторів.

Для великих проектів діаграма Ганта стає надмірно великою і втрачає будь-яку наочність. Зазначені вище недоліки і обмеження серйозно обмежують сферу застосування таких діаграм.

Тим не менше, діаграма Ганта є де-факто стандартом в теорії і практиці управління проектами, зокрема, для відображення структури і послідовності виконання робіт за проектом.

В якості **прикладу** розглянемо процес підготовки деякого спеціалізованого видання. Він являє собою ланцюг взаємопов'язаних і взаємообумовлених робіт, що можна представити за допомогою діаграми Ганта, зображеної на рис. 1.23.

На рис. 1.23 великими латинським літерами позначені такі роботи:

- A – наукове (спеціальне) редагування;
- B – смислове редагування;
- C – стилістичне редагування;
- D – художнє оформлення та редагування;
- E – технічне редагування;
- F – оформлення обкладинки, суперобкладинки, форзацу та ін.;
- G – здача в друкарню,;
- H – підготовка матеріалів до реєстрації видання;
- I – друк.

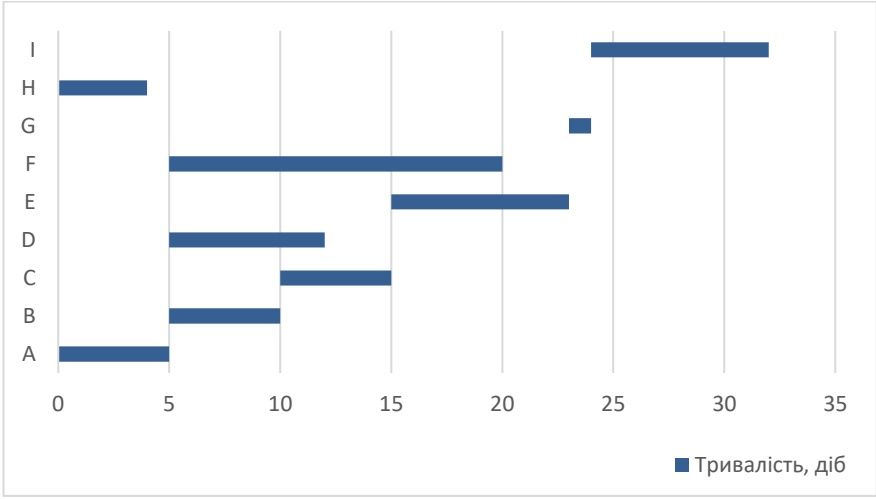


Рис. 1.23. Діаграма Ганта для підготовки видання

З рис. 1.23 видно, що між роботами існують зв'язки:

- роботи B, D і F можуть початися не раніше, ніж буде виконана перша робота A;
- після виконання роботи B можна починати роботу C;
- закінчити роботи C і D необхідно до початку роботи E;
- аналогічно необхідно закінчити роботи E і F до початку роботи G;
- робота I може початися тільки після завершення всіх робіт.

Таким чином, стрічковий графік у сукупності зі словесним описом зв'язків дає повне уявлення про процес будь-якої розробки. Але словесний опис зв'язків навіть для невеликих проектів може бути громіздким. В 1990-х роках було запропоновано зображувати описані зв'язки графічно, доповнивши стрічковий графік стрілками. Відповідний графік Ганта для підготовки видання подано на рис. 1.24.

1.12.2. Основні поняття мережевого плавання

Мережеве планування вперше було запропоновано консалтинговим підрозділом фірми Lockheed Martin Corporation для реалізації проекту розробки ракетної системи «Поларіс» в 1956 році. Проект складався з 60 тис. операцій. В ньому було задіяно близько 3800 основних підрядників.

Основу мережевої моделі складає $граф$ – наочне представлення плану робіт, а основними термінами є $подія$, $робота$ і $шлях$.

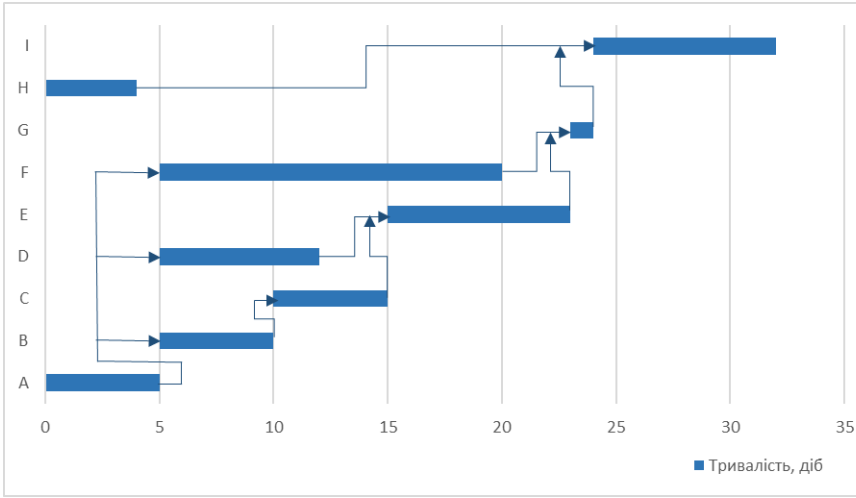


Рис. 1.24. Діаграма Ганта зі зв’язками

Подія – це стан, момент досягнення проміжної або кінцевої мети розробки (факт закінчення однієї або декількох робіт, необхідних для початку наступних). Події бувають початковими, кінцевими і проміжними.

Початкова подія – подія, якій не передує жодна робота, а *кінцева подія* – подія, за якою відсутня будь-яка інша робота. Всі інші події є *проміжними*.

Робота – це процес, що протікає в часі і вимагає для свого здійснення матеріальних та трудових витрат. Кожна робота має попередню події і певною подією закінчується. Крім того, серед робіт виділяють:

• *очікування* – робота, що вимагає витрат часу, але не вимагає витрат ресурсів;

• *фіктивна робота* – логічний зв'язок між певними видами робіт, який не вимагає витрат ані матеріальних ресурсів, ані часу.

Шлях – це ланцюжок подій, які відбуваються одна за одною, з’єднуючи послідовно початкову та кінцеву вершини графа, що відображає мережу.

Якщо позначити роботи колами, а зв’язки між ними стрілками, то отримаємо граф, який в даному випадку прийнято називати *мережевим графіком*. Для компенсації втрати часового масштабу в мережевому графіку прийнято в вершинах позначати тривалості робіт.

Мережевий графік для підготовки видання подано на рис. 1.25.

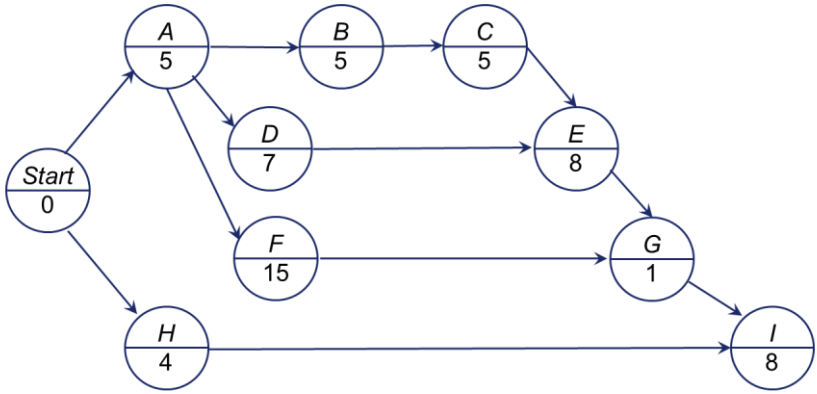


Рис. 1.25. Мережевий графік для підготовки видання

В загальному випадку мережевий графік – це динамічна модель виробничого процесу, яка відображає технологічну залежність і послідовність виконання комплексу робіт, погоджує їх здійснення в часі з урахуванням витрат ресурсів і вартості, а також дозволяє знаходити вузькі (критичні) місця.

1.12.3. Перетворення ребернозважених графів у вершиннозважані

Мережеві графіки можуть будуватися у *двох варіантах*:

• вершини графу відображають стан деякого проекту (наприклад, розробки програмного продукту, підготовки до видання поліграфічної продукції, будівництва тощо), а дуги – роботи, що мають бути виконані;

• вершини графу відбивають роботи, а зв'язки між ними – залежності між роботами.

З точки зору теорії графів перший варіант мережевого графіку – це орієнтовані *ребернозважені графи*, в другий – орієнтовані *вершиннозважені*. Для переходу від ребернозваженого графа до вершиннозваженого можна скористатися наступним *алгоритмом*.

Крок 1. Зображаємо початкову вершину S і розглядаємо всі дуги, що з неї виходять. За кількістю дуг отримуємо наступний набір вершин, який з'єднується дугами з вихідною вершиною S. Кожна нова вершина отримає вагу, що дорівнює вазі відповідної дуги.

Крок 2. Кожну нову вершину розглядаємо як вихідну вершину і повторюємо крок 1, поки не дійдемо до кінцевої вершини T.

Роботу алгоритму розглянемо на прикладі.

Приклад 1.9. Перетворити граф, наведений на рис.1.26, з реберно-зваженого на вершинно-зважений.

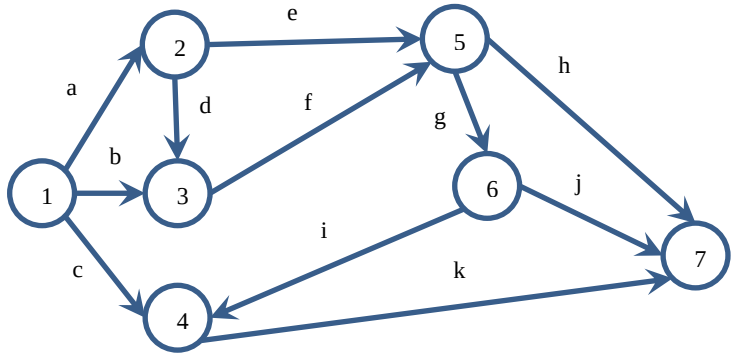


Рис. 1.26. Ребернозважений граф прикладу 1.9

Розв'язання. Вихідний граф за умовою має 7 вершин та 11 ребер, позначених буквами латинського алфавіту. Вважатимемо, що кожне ребро відповідає певній роботі, тривалість та інші характеристики якої задані окремо. Шуканий вершинно-зважений граф повинен мати 13 вершин (11 за кількістю ребер вихідного графа, а також початкова вершина S та кінцева T).

За алгоритмом зображуємо початкову вершину S. Розглядаючи граф рис. 1.26, бачимо, що з вихідної вершини 1 виходить 3 дуги a, b, c. За кількістю дуг отримаємо три відповідні нові вершини a, b, c. З'єднуємо їх ребрами з вершиною S. Далі розглядаємо отримані нові вершини як вихідні.

Робота a дозволяє виконання робіт d та e, якім ставимо у відповідність однойменні вершини на графі, що будується. З'єднуємо їх вхідними ребрами з вершиною a. Роботи b та d дозволяють виконання роботи f. Отже, створюємо вершину f і проводимо до неї ребра з вершин b та d. Роботи h та g вимагають для свого виконання завершення двох робіт, e та f. Створюємо вершини h та g та поєднуємо їх вхідними ребрами з вершинами e та f. Робота g відкриває шлях до виконання робот i та j. Створюємо для них однойменні вершини і приєднуємо до вершини g. Нарешті, роботи c та i дозволяють виконати роботу k. Додаємо вершину k і створюємо ребра з вершин c та i до цієї вершини.

Фінальна вершина 7 на вихідному графі може бути досягнута тільки після завершення робіт h, i, k. Отже, ці вершини на створеному графі треба приєднати ребрами до кінцевої вершини T.

Отриманий в результаті вершиннозважений граф подано на рис. 1.27.

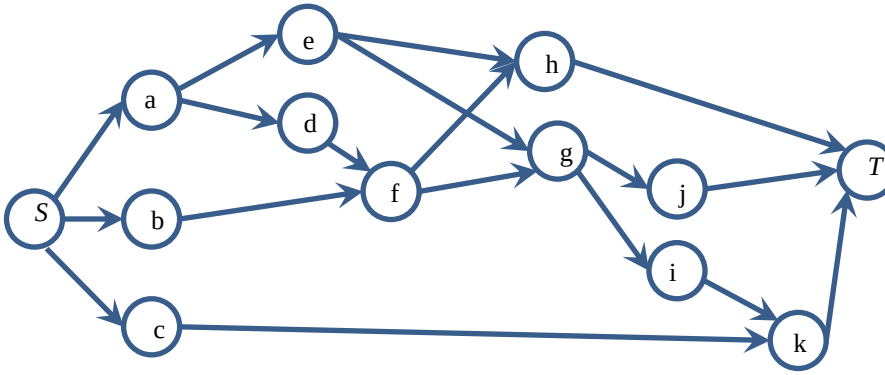


Рис. 1.27. Вершиннозважений граф

1.12.4. Правила побудови мережевого графіка

Коректно побудований мережевий графік повинен задовольняти наступним *вимогам*:

• початкова подія поміщається в лівій, а кінцева – в правій частині мережі;

• кожна робота має попередню подію і певною подією закінчується;

• тільки початкова подія не має вхідних дуг, і тільки кінцева подія не має дуг, що виходять з неї;

• нумерувати події необхідно за зростанням, тобто кожна наступна повинна мати зростаючий номер по відношенню до попередньої. При невиконанні цієї умови необхідно перенумеровати за *алгоритмом*:

– нумерація починається з початкової події, якій привласнюється номер 1;
– з початкової події викреслюють всі роботи, що виходять з неї, і серед них знаходять подію, в яку не входить жодна робота, їй привласнюють номер 2;
– процес продовжують до завершальної події, номер якої має дорівнювати кількості подій в мережевому графіку;
– якщо при черговому викреслюванні робіт одночасно кілька подій не мають робіт, що входять в них, то їх нумерують по черзі у довільному порядку;

• кожна робота на графі повинна закінчуватися подією, а за кожною подією (крім кінцевої) має слідувати робота. Наявність тупиків свідчить про помилку;

• наявність замкнутого циклу також свідчить про помилку;

• будь-які дві події повинні бути пов'язані безпосередньо не більше ніж однією роботою;

• при виявленні паралельних робіт вводиться фіктивна подія і фіктивна робота;

• якщо для початку деякої роботи достатньо виконати не всю попередню роботу, а лише її частину, то ця попередня робота розбивається на дві або кілька самостійних робіт, які йдуть одна за одною;

• якщо для якоїсь події не виявилося попередньої роботи, то вона з'єднується з початковою подією, а при відсутності подальшої роботи – з кінцевою точкою.

Правила побудови мережевого графіка в термінах робіт аналогічні розглянутим вище.

Зазвичай побудова мережевого графа починається зі складання спеціальної *табличної форми опису комплексу робіт*. В таблицю заносяться код або шифр робіт. Крім того, для кожної роботи вказується тривалість її виконання і список попередніх робіт. Приклад опису комплекса робіт для процесу підготовки видання подано в табл. 1.4.

Таблиця 1.4 – Опис комплексу робіт

№ з/п	Параметр	Робота								
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
1	Номер роботи	1	2	3	4	5	6	7	8	9
2	Попередня робота	*Start*	*A*	*B*	*A*	*C,D*	*A*	*E,F*	*Start*	*G,H*
3	Наступна робота	*B, D, F*	*C*	*E*	*E*	*G*	*G*	*I*	*I*	
4	Тривалість роботи d_i	5	5	5	7	8	15	1	4	8

Мережевий граф повинен мати одну початкову роботу та одну кінцеву. Якщо ця умова не виконується, в мережевий граф можуть бути введені *фіктивні роботи*.

У нашому випадку дві роботи A і H не мають попередніх робіт. Тому додаємо одну фіктивну роботу $Start$ з номером 0 і тривалістю 0. Нова робота матиме в списку наступних роботи A і H.

Якщо в комплексі робіт є декілька робіт, для яких немає подальших, необхідно додати ще одну фіктивну роботу. Ця робота матиме тривалість, що дорівнює 0, і вона буде завершувати комплекс робіт.

У нашому випадку тільки одна робота (робота з номером 9) є роботою, для якої немає подальших. Тому в мережевий граф додавати ще одну фіктивну роботу немає потреби.

1.12.5. Характеристики мережевого графіка

Розрахунок мережевого графа зводиться до обчислення таких параметрів:

- ранній час початку роботи $t_{\text{рп}}$;
- ранній час закінчення роботи $t_{\text{рз}}$;
- пізній час закінчення роботи $t_{\text{пз}}$;
- пізній час початку роботи $t_{\text{пп}}$;
- повний резерв часу роботи $R_{\text{п}}$;
- вільний резерв часу роботи $R_{\text{в}}$;
- коефіцієнт напруженості $K_{\text{н}}$.

Найдовший шлях від початкової до кінцевої роботи, який не має резервів часу, називається *критичним шляхом*.

Кожний мережевий граф має хоча б один критичний шлях, який визначає мінімальний термін виконання проекту. Зменшення терміну виконання операцій, розташованих на критичному шляху, забезпечує зменшення тривалості виконання усього комплексу операцій.

Ранній час початку роботи $t_{\text{рп}}$ – це ранній час, необхідний для виконання усіх робіт, які передують цій роботі.

$$t_{\text{рп}}(i) = \max_{k}[\, t_{\text{рп}}(k) \, + \, d_k],$$

де k – номер попередньої роботи;

$t_{\text{рп}}(k)$ – ранній початок попередньої роботи;

d_k – тривалість попередньої роботи.

Раннім часом закінчення роботи $t_{\text{рз}}$ називається самий ранній момент часу, в який може закінчитися ця робота:

$$t_{\text{рз}}(i) = t_{\text{рп}}(i) \, + \, d_i,$$

де i – номер роботи;

$t_{\text{рп}}(i)$ – ранній початок роботи;

d_i – тривалість роботи.

Пізнім часом закінчення роботи $t_{\text{пз}}$ називається пізній термін, в який повинна закінчитися робота, щоб загальний час виконання комплексу робіт не збільшився:

$$t_{\text{пз}}(i) = \min_j [\, t_{\text{пз}}(j) - \, d_j] \,,$$

де j – номер наступної роботи;

$t_{\text{пз}}(j)$ – пізнє закінчення наступної роботи;

d_j – тривалість наступної роботи.

Пізній час початку роботи $t_{\text{пп}}$ – це найпізніший момент часу початку роботи, який не призводить до збільшення загального часу виконання комплексу робіт:

$$t_{\text{пп}}(i) = t_{\text{пз}}(i) - \, d_i,$$

де i – номер роботи;

$t_{\text{пз}}(i)$ – пізнє закінчення роботи;

d_i – тривалість роботи.

Повний резерв часу роботи $R_{\text{п}}$ – максимальний час, на який можна відстрочити початок роботи, не змінюючи терміну виконання всього комплексу робіт.

$$R_{\text{п}}(i) = t_{\text{пп}}(i) - \, t_{\text{рп}}(i)$$

Вільний резерв часу роботи $R_{\text{в}}$ – це максимальний час, на який можна відстрочити початок або збільшити тривалість роботи за умови, що всі попередні і всі наступні для даної роботи події мережі настануть в свої ранні терміни.

$$R_{\text{в}}(i) = \min_j \big[\, t_{\text{рп}}(j) - \, d_i\big] - t_{\text{рп}}(i)$$

де i – номер роботи;

$t_{\text{рп}}(j)$ – ранній початок наступної роботи;

d_i – тривалість поточної роботи;

$t_{\text{рп}}(i)$ – ранній початок роботи.

Для оптимізації мережевої моделі, яка полягає в перерозподілі ресурсів з ненапружених робіт на критичні для прискорення їх виконання, необхідно якомога точніше оцінити ступінь важкості своєчасного виконання всіх робіт, а також “ланцюгів” шляху. Більш точним інструментом розв’язування цієї

задачі в порівнянні з повним резервом часу є *коефіцієнт напруженості*, який може бути обчислений за формулою

$$K_{\text{н}}(i) = 1 - \frac{R_{\text{п}}(i)}{L_{\text{кр}}},$$

де $L_{\text{кр}}$ – довжина критичного шляху.

Коефіцієнт напруженості змінюється від нуля до одиниці, причому чим він ближчий до одиниці, тим складніше виконати дану роботу у встановлений термін. За коефіцієнтом напруженості всі роботи можуть бути поділені на кілька *груп*:

- критичні – $K_{\text{н}} = 1$;
- напружені – $K_{\text{н}} > 0{,}8$;
- підкритичні – $0{,}6 < K_{\text{н}} \leq 0{,}8$;
- резервні – $K_{\text{н}} \leq 0{,}6$.

Самими напруженими є роботи критичного шляху, для яких він дорівнює 1. В результаті перерозподілу ресурсів прагнуть максимально зменшити загальну тривалість робіт, що можливе при переведенні всіх робіт в першу групу.

1.12.6. Розрахунок мережевого графіка

Алгоритм розрахунку мережевого графа наступний.

Крок 1. Від початкової вершини до кінцевої розраховується ранній час початку роботи з урахуванням «довжин» всіх дуг, що входять у відповідну вершину (береться максимум з отриманих величин).

Крок 2. Від кінцевої вершини до початкової розраховується пізній час закінчення з урахуванням «довжин» всіх дуг, що виходять із відповідної вершини (береться мінімум з отриманих величин).

Крок 3. Для кожної вершини розраховується резерв та інші характеристики.

Крок 4. Визначається критичний шлях, що проходить через вершини з нульовим резервом, та коефіцієнт напруженості для кожної роботи.

Шлях – будь-яка безперервна послідовність робіт у мережевому графі між подіями. Виділяють:

- *повний шлях* – шлях, початок якого збігається з вихідною подією мережі, а кінець із завершальною;
- *шлях між подіями* – шлях, що з'єднує будь-які дві події в мережі, крім початкової та завершальної;

• *критичний шлях* – найдовший шлях між початковою та кінцевою подіями;

• *некритичний шлях* – шлях між початковою та кінцевою подіями мережі, що має меншу тривалість складових робіт порівняно з критичним шляхом.

Критичні роботи – це роботи, що лежать на критичному шляху. Роботи на некритичних шляхах мають запас часу, ресурсів, якими можна маневрувати.

Справедливі наступні **твердження**:

1. Якщо $R_{\text{в}}(i) = 0$, то робота i лежить на критичному шляху; якщо $R_{\text{в}}(i) > 0$, то робота i не лежить на критичному шляху.

2. Якщо час початку роботи i, що не лежить на критичному шляху, відкласти на строк менший, ніж $R_{\text{в}}(i)$, то час настання наступної події не зміниться.

3. Якщо час початку роботи i, що не лежить на критичному шляху, відкласти на термін менший, ніж $R_{\text{в}}(i)$, то час, необхідний на виконання всього проєкту, не збільшиться.

Способи скорочення критичного шляху:

• перерозподіл ресурсів;

• зміна організації робіт;

• перекидання ресурсів на критичний шлях під час очікування, якщо це дозволяє технологія.

На мережевому графіку часові параметри відображаються, як показано на рис. 1.28. На рис. 1.28 позначено через i – номер роботи, $t_{\text{р}}(i)$ – ранній час початку роботи, $t_{\text{п}}(i)$ – пізній час закінчення роботи, d_i – тривалість роботи.

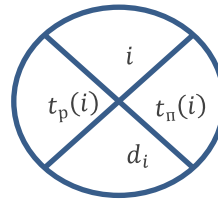


Рис. 1.28. Параметри мережевого графіка

Приклад 1.10. Провести розрахунок мережевого графіка для підготовки видання (рис. 1.25).

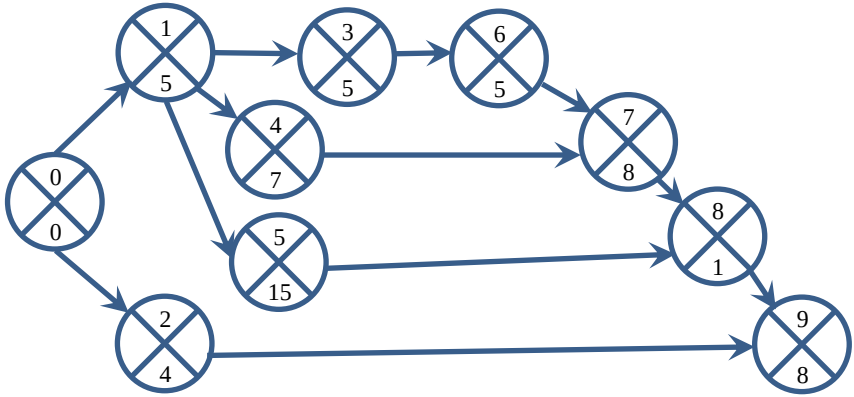


Рис. 1.29. Мережевий графік для розрахунку

Розв'язання. Спочатку зобразимо мережевий граф з урахуванням чотирисекційної моделі (результат подано рис. 1.29), а потім проведемо розрахунок його числових характеристик.

Крок 1. Визначення раннього початку робіт.

Робота з номером 0 є початковою роботою. Тому вона має ранній час початку виконання $t_{рп}(0) = 0$.

Для робіт з номерами 1 та 2 робота з номером 0 є попередньою. Тому ранній початок робіт 1 та 2:

$$t_{\text{рп}}(1) = \text{t}_{\text{рп}}(0) + \ d_0 = 0 + 0 = 0,$$
$$\text{t}_{\text{рп}}(2) = \text{t}_{\text{рп}}(0) + \ d_0 = 0 + 0 = 0.$$

Для робіт з номерами 3, 4, 5 попередньою є робота з номером 1. Тому ранній початок робіт 3, 4, 5:

$$t_{\text{рп}}(3) = t_{\text{рп}}(1) + \ d_1 = 0 + 5 = 5,$$
$$t_{\text{рп}}(4) = t_{\text{рп}}(1) + \ d_1 = 0 + 5 = 5,$$
$$t_{\text{рп}}(5) = t_{\text{рп}}(1) + \ d_1 = 0 + 5 = 5.$$

Для роботи з номером 6 робота з номером 3 є попередньою. Тому ранній початок роботи 6:

$$t_{\text{рп}}(6) = t_{\text{рп}}(3) + \ d_3 = 5 + 5 = 10.$$

Роботи з номером 7 має дві попередні роботи – 4 та 6. Тому ранній початок роботи 7 буде визначатися як максимальне значення з двох величин

$$t_{\text{рп}}(7) = max\left[t_{\text{рп}}(4) + \ d_4\ ,\ t_{\text{рп}}(6) + \ d_6\right] = max[5 + 7\ ,10 + 5] = 15.$$

Для роботи з номером 8 попередніх робіт також дві – 5 та 7. Тому аналогічно ранній початок роботи 8

$$t_{\text{рп}}(8) = max\left[t_{\text{рп}}(5) + \ d_5\ ,\ t_{\text{рп}}(7) + \ d_7\right] = max[5 + 15\ ,15 + 8] = 23.$$

Робота з номером 9 також має дві попередні робіт – 2 та 8. Тому ранній початок роботи 9 :

$$t_{\text{рп}}(9) = max\left[t_{\text{рп}}(2) + \ d_2\ ,\ t_{\text{рп}}(8) + \ d_8\right] = max[0 + 4\ ,23 + 1] = 24.$$

На цьому завершується перший крок розрахунку. Мережевий графік після першого кроку подано на рис. 1.30.

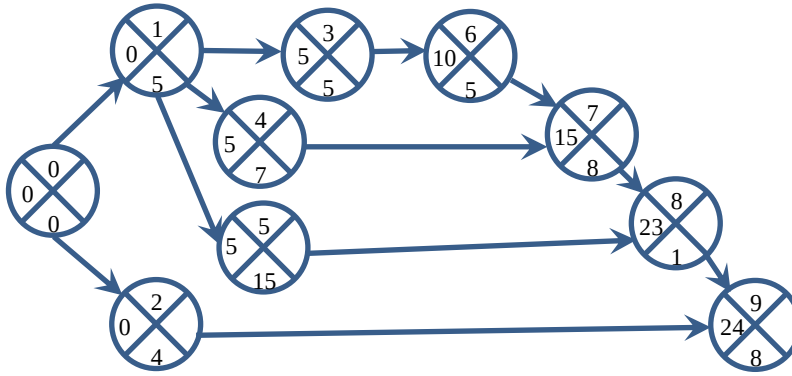


Рис. 1.30. Мережевий графік після першого кроку розрахунку

Крок 2. Визначення пізнього часу закінчення робіт.

Робота з номером 9 є останньою, тому пізній час закінчення роботи 9

$$t_{\text{пз}}(9) = t_{\text{рп}}(9) + \ d_9 = 24 + 8 = 32.$$

Визначаємо пізній час закінчення робіт 2, 8, 7, 5, 4, 6 і 3 за формулою

$$t_{\text{пз}}(i) = \min_j[\ t_{\text{пз}}(j) - \ d_j]$$

$$t_{\text{пз}}(2) = t_{\text{пз}}(9) - \ d_9 = 32 - 8 = 24,$$
$$t_{\text{пз}}(8) = t_{\text{пз}}(9) - \ d_9 = 32 - 8 = 24,$$
$$t_{\text{пз}}(7) = t_{\text{по}}(8) - \ d_8 = 24 - 1 = 23,$$
$$t_{\text{пз}}(5) = t_{\text{пз}}(8) - \ d_8 = 24 - 1 = 23,$$
$$t_{\text{пз}}(4) = t_{\text{пз}}(7) - \ d_7 = 23 - 8 = 15,$$
$$t_{\text{пз}}(6) = t_{\text{пз}}(7) - \ d_7 = 23 - 8 = 15,$$
$$t_{\text{пз}}(3) = t_{\text{пз}}(6) - \ d_6 = 15 - 5 = 10.$$

Робота з номером 1 має три наступні роботи – 3, 4, 5. Тому пізній час закінчення роботи 1 буде визначатися як мінімальне значення

$$t_{\text{пз}}(1) = min[t_{\text{пз}}(3) - d_3\,,\ t_{\text{пз}}(4) - d_4\,,\ t_{\text{пз}}(5) - d_5] = \\ = min[10-5\,,15-7\,,\ 23-15] = min[5{,}8{,}8] = 5.$$

Робота з номером 0 має дві наступні роботи – з номерами 1 і 2. Тому аналогічно попередній роботі 1 пізній час закінчення роботи 0 буде визначатися як мінімальне значення

$$t_{\text{пз}}(0) = min[t_{\text{пз}}(1) - d_1\,,\ t_{\text{пз}}(2) - d_2\,] = \\ = min[5-5\,,\ 24-4] = min[0,\ 20] = 0.$$

На цьому завершується другий крок розрахунку. Мережевий графік після другого кроку подано на рис. 1.31.

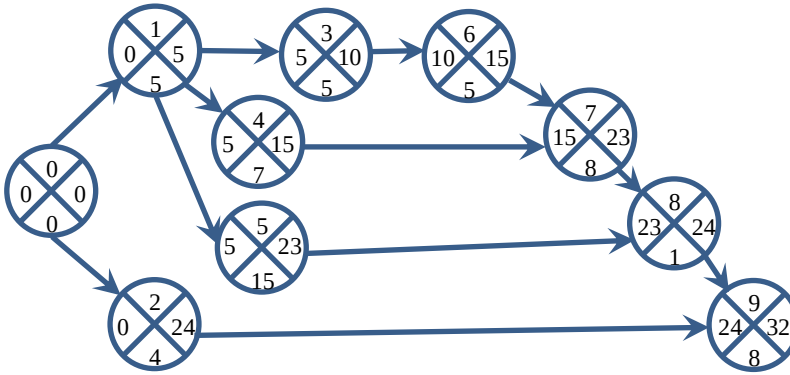


Рис. 1.31. Мережевий графік після другого кроку розрахунку

Крок 3. Обчислення резерв та інших характеристик.

Почнемо з розрахунку раннього часу закінчення кожної роботи

$$t_{\text{рз}}(0) = t_{\text{рп}}(0) + d_0 = 0 + 0 = 0,$$
$$t_{\text{рз}}(1) = t_{\text{рп}}(1) + d_1 = 0 + 5 = 5,$$
$$t_{\text{рз}}(2) = t_{\text{рп}}(2) + d_2 = 0 + 4 = 4,$$
$$t_{\text{рз}}(3) = t_{\text{рп}}(3) + d_3 = 5 + 5 = 10,$$
$$t_{\text{рз}}(4) = t_{\text{рп}}(4) + d_4 = 5 + 7 = 12,$$
$$t_{\text{рз}}(5) = t_{\text{рп}}(5) + d_5 = 5 + 15 = 20,$$
$$t_{\text{рз}}(6) = t_{\text{рп}}(6) + d_6 = 10 + 5 = 15,$$
$$t_{\text{рз}}(7) = t_{\text{рп}}(7) + d_7 = 15 + 8 = 23,$$
$$t_{\text{рз}}(8) = t_{\text{рп}}(8) + d_8 = 23 + 1 = 24,$$

$$t_{\text{рз}}(9) = t_{\text{рп}}(9) + d_9 = 24 + 8 = 32.$$

Далі визначимо пізній час початку кожної роботи

$$t_{\text{пп}}(0) = t_{\text{пз}}(0) - d_0 = 0 - 0 = 0,$$
$$t_{\text{пп}}(1) = t_{\text{пз}}(1) - d_1 = 5 - 5 = 0,$$
$$t_{\text{пп}}(2) = t_{\text{пз}}(2) - d_2 = 24 - 4 = 20,$$
$$t_{\text{пп}}(3) = t_{\text{пз}}(3) - d_3 = 10 - 5 = 5,$$
$$t_{\text{пп}}(4) = t_{\text{пз}}(4) - d_4 = 15 - 7 = 8,$$
$$t_{\text{пп}}(5) = t_{\text{пз}}(5) - d_5 = 23 - 8 = 15,$$
$$t_{\text{пп}}(6) = t_{\text{пз}}(6) - d_6 = 15 - 5 = 10.$$
$$t_{\text{пп}}(7) = t_{\text{пз}}(7) - d_7 = 23 - 8 = 15,$$
$$t_{\text{пп}}(8) = t_{\text{пз}}(8) - d_8 = 24 - 1 = 23,$$
$$t_{\text{пп}}(9) = t_{\text{пз}}(9) - d_9 = 32 - 8 = 24.$$

Знайдемо повний резерв часу для кожної роботи

$$R_{\text{п}}(0) = t_{\text{пп}}(0) - t_{\text{рп}}(0) = 0 - 0 = 0,$$
$$R_{\text{п}}(1) = t_{\text{пп}}(1) - t_{\text{рп}}(0) = 0 - 0 = 0,$$
$$R_{\text{п}}(2) = t_{\text{пп}}(2) - t_{\text{рп}}(2) = 20 - 0 = 20,$$
$$R_{\text{п}}(3) = t_{\text{пп}}(3) - t_{\text{рп}}(3) = 5 - 5 = 0,$$
$$R_{\text{п}}(4) = t_{\text{пп}}(4) - t_{\text{рп}}(4) = 8 - 5 = 3,$$
$$R_{\text{п}}(5) = t_{\text{пп}}(5) - t_{\text{рп}}(5) = 15 - 5 = 10,$$
$$R_{\text{п}}(6) = t_{\text{пп}}(6) - t_{\text{рп}}(6) = 10 - 10 = 0,$$
$$R_{\text{п}}(7) = t_{\text{пп}}(7) - t_{\text{рп}}(7) = 15 - 15 = 0,$$
$$R_{\text{п}}(8) = t_{\text{пп}}(8) - t_{\text{рп}}(8) = 23 - 23 = 0,$$
$$R_{\text{п}}(9) = t_{\text{пп}}(9) - t_{\text{рп}}(9) = 24 - 24 = 0.$$

Обчислимо вільний резерв часу для кожної роботи. У роботи 0 наступними є роботи 1 та 2, тому вільний резерв буде визначатися як мінімальне значення з двох величин

$$R_{\text{в}}(0) = min\left[t_{\text{рп}}(1) - d_0, t_{\text{рп}}(2) - d_0 \right] - t_{\text{рп}}(0) = min[\,0,0] - 0 = 0.$$

У роботи 1 наступними є роботи 3, 4 та 5, тому вільний резерв

$$R_{\text{в}}(1) = min\left[t_{\text{рп}}(3) - d_1, t_{\text{рп}}(4) - d_1, t_{\text{рп}}(5) - d_1 \right] - t_{\text{рп}}(0)$$
$$= min[\,5 - 5,\ 5 - 5,\ 5 - 5] - 0 = 0.$$

Для всіх інших робіт

$$R_{\text{в}}(2) = t_{\text{рп}}(9) - d_2 - t_{\text{рп}}(2) = 24 - 4 - 0 = 20,$$
$$R_{\text{в}}(3) = t_{\text{рп}}(6) - d_3 - t_{\text{рп}}(3) = 10 - 5 - 5 = 0,$$

$$R_{\text{в}}(4) = t_{\text{рп}}(7) - d_4 - t_{\text{рп}}(4) = 15 - 7 - 5 = 3,$$
$$R_{\text{в}}(5) = t_{\text{рп}}(8) - d_5 - t_{\text{рп}}(5) = 23 - 15 - 5 = 3,$$
$$R_{\text{в}}(6) = t_{\text{рп}}(7) - d_6 - t_{\text{рп}}(6) = 15 - 5 - 10 = 0,$$
$$R_{\text{в}}(7) = t_{\text{рп}}(8) - d_7 - t_{\text{рп}}(7) = 23 - 8 - 15 = 0,$$
$$R_{\text{в}}(8) = t_{\text{рп}}(9) - d_8 - t_{\text{рп}}(8) = 24 - 1 - 23 = 0.$$

У роботи 9 наступних немає, тому д я неї вільний резерв дорівнює нулю, тобто $R_{\text{в}}(9) = 0$.

Крок 4. Визначається критичний шлях, що проходить через вершини з нульовим резервом, та коефіцієнт напруженості для кожної роботи.

Роботи з номерами 0, 1, 3, 6 – 9 не мають резерву часу, тобто є критичними та утворюють критичний шлях. Довжина критичного шляху $L_{\text{кр}}$– це самий довгий шлях від початкової роботи до кінцевої:

$$L_{\text{кр}} = t_{\text{пз}}(9) = 32.$$

Критичний шлях на рис. 1.32 виділено кольором.

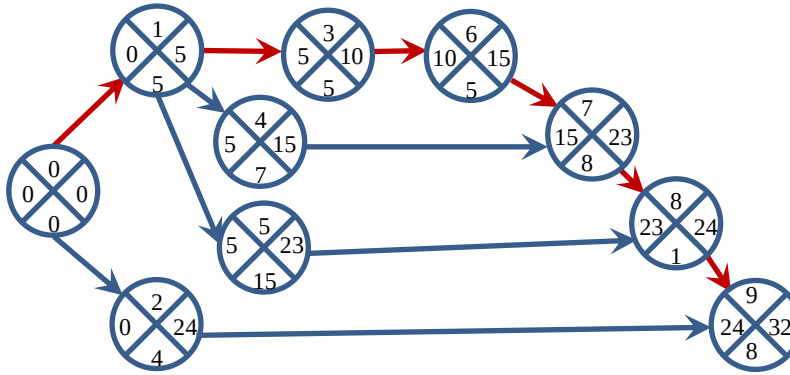


Рис. 1.32. Критичний шлях мережевого графіка

Розрахуємо коефіцієнти напруженості для кожної роботи.

$$K_{\text{н}}(0) = 1 - \frac{R_{\text{п}}(0)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$
$$K_{\text{н}}(1) = 1 - \frac{R_{\text{п}}(1)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$
$$K_{\text{н}}(2) = 1 - \frac{R_{\text{п}}(2)}{L_{\text{кр}}} = 1 - \frac{20}{32} = 0{,}375,$$

$$K_{\text{н}}(3) = 1 - \frac{R_{\text{п}}(3)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$

$$K_{\text{н}}(4) = 1 - \frac{R_{\text{п}}(4)}{L_{\text{кр}}} = 1 - \frac{3}{32} = 0{,}906,$$

$$K_{\text{н}}(5) = 1 - \frac{R_{\text{п}}(5)}{L_{\text{кр}}} = 1 - \frac{10}{32} = 0{,}688,$$

$$K_{\text{н}}(6) = 1 - \frac{R_{\text{п}}(6)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$

$$K_{\text{н}}(7) = 1 - \frac{R_{\text{п}}(7)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$

$$K_{\text{н}}(8) = 1 - \frac{R_{\text{п}}(8)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1,$$

$$K_{\text{н}}(9) = 1 - \frac{R_{\text{п}}(9)}{L_{\text{кр}}} = 1 - \frac{0}{32} = 1.$$

Як показано вище, критичними є роботи 0, 1, 3, 6 – 9. Вони мають нульовий резерв часу і коефіцієнт напруженості, що дорівнює 1.

Аналізуючи інші роботи за коефіцієнтом напруженості, можна сказати, що роботи 2 і 5 мають великий резерв часу (відповідно 20 та 10), відтак, можна на даному етапі з них зняти ресурси і перекинути їх на ті, що перебувають на критичному шляху. Аналогічно, робота 4 має резерв часу 3. Роботу 2 вважаємо резервною, 4 – напруженою, а роботу 5 – підкритичною.

2. ПОСТАНОВКА ЗАДАЧІ ДЛЯ РОЗРАХУНКОВОГО ЗАВДАННЯ

Для заданого орієнтовного графа G (згідно з варіантом, що подано в табл. 2.1) розв’язати наступні задачі.

1. Пронумеровати вершини і дуги та записати матриці суміжності та інцидентності.

2. Розглядаючи заданий граф як неорієнтований, записати його матриці суміжності та інцидентності. Обчислити валентність вершин. Відповісти на наступні питання.

- Чи є граф ейлеровим або напівейлеровим? Якщо граф є ейлеровим, то побудувати ейлеров цикл, напівейлеровим – ейлеров маршрут.
- Чи є граф однорідним?
- Перевірити за теоремою Дірака, чи є граф гамільтоновим.

3. Розглядаючи заданий граф як неорієнтований, визначити його цикломатичне число, коциклічний ранг, а також кількість кістякових дерев. Зобразити будь-яких 5 його кістякових дерев.

Зауваження. Для обчислення загальної кількості кістякових дерев через алгебраїчне доповнення матриці Кірхгофа можна скористатися будь-яким математичним пакетом або Excel.

4. Розглядаючи заданий граф як неорієнтований, знайти мінімальне кістякове дерево за алгоритмом Крускала.

5. Розглядаючи заданий граф як неорієнтований, знайти довжину найкоротших шляхів між вершиною v_1 і всіма іншими вершинами за алгоритмом Дейкстри. Привести найкоротший шлях із вершини v_1 до вершини v_{13}.

6. Для заданого орієнтованого графа G провести перевірку на наявність циклів за алгоритмом відкидання джерел та стоків. Якщо в графі буде виявлено цикл, узгодити з керівником зміну орієнтації однієї з дуг. Далі розглядати модифікований орієнтований граф.

7. Перетворити граф з реберно-зваженого на вершинно-зважений та пронумерувати вершини отриманого графа.

8. Скласти таблицю опису комплексу робіт.

9. Побудувати лінійний графік Ганта.

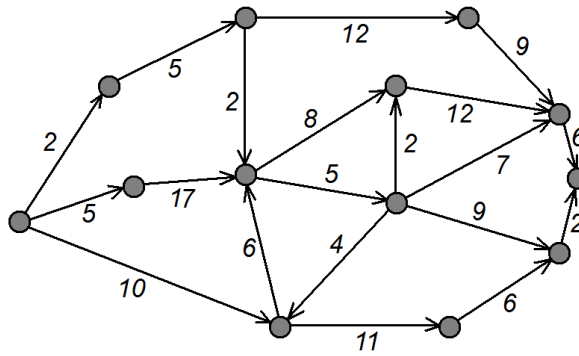
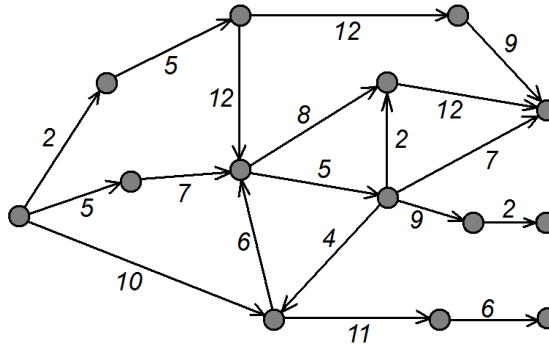
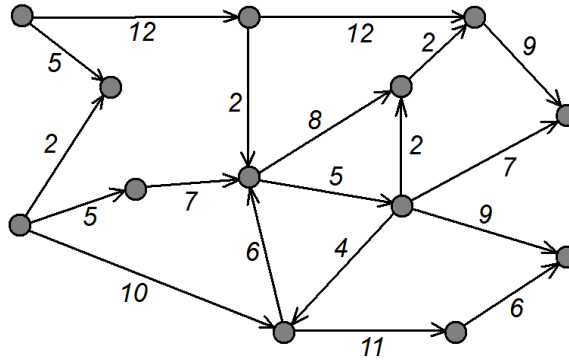
10. Скласти мережевий графік, а також розрахувати всі його числові характеристики:

- ранній час початку роботи $t_{\text{рп}}$;
- ранній час закінчення роботи $t_{\text{рз}}$;
- пізній час закінчення роботи $t_{\text{пз}}$;
- пізній час початку роботи $t_{\text{пп}}$;
- повний резерв часу роботи $R_{\text{п}}$;
- вільний резерв часу роботи $R_{\text{в}}$;
- коефіцієнт напруженості $K_{\text{н}}$.

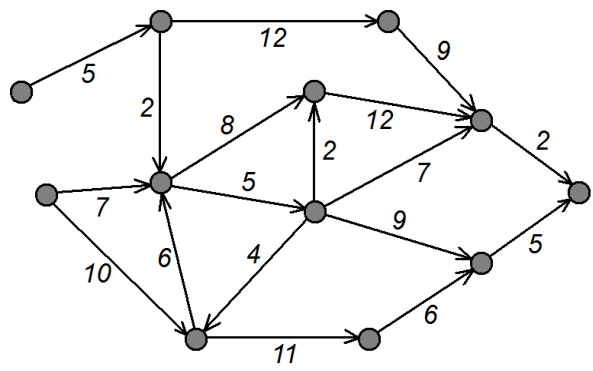
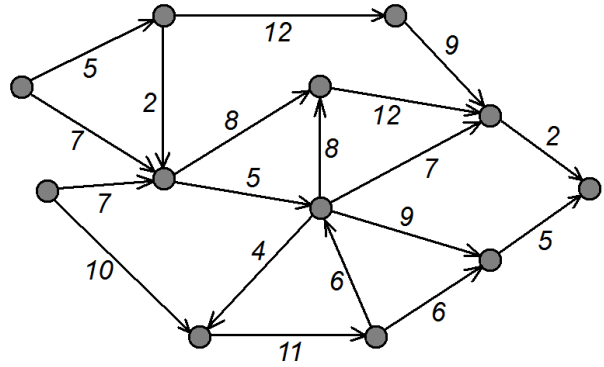
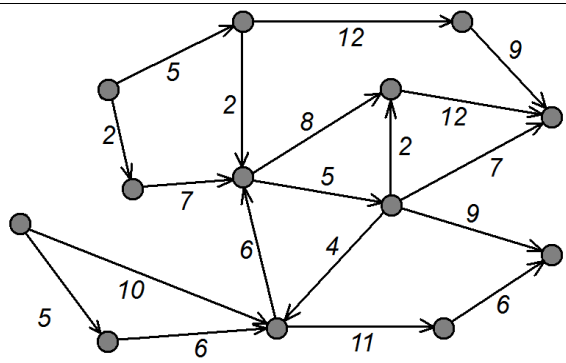
Знайти на графі критичний маршрут, проаналізувати роботи за коефіцієнтом напруженості.

Зауваження. Для обчислення числових характеристик можна скористатися Excel.

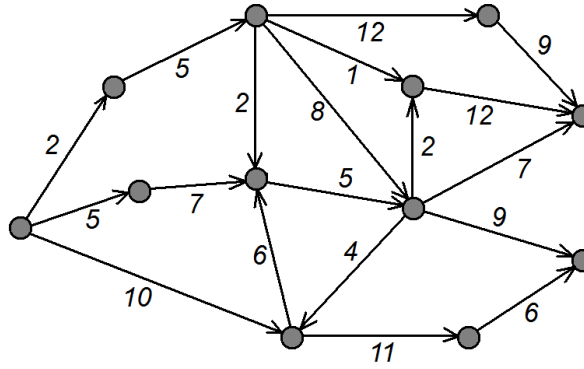
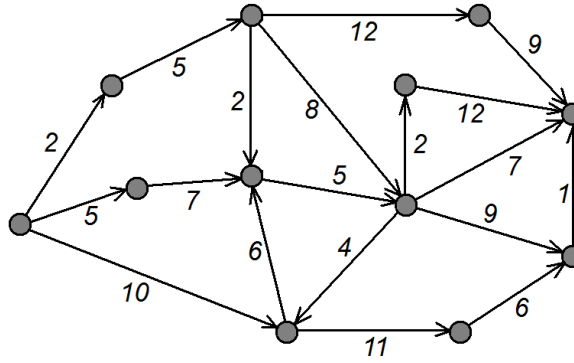
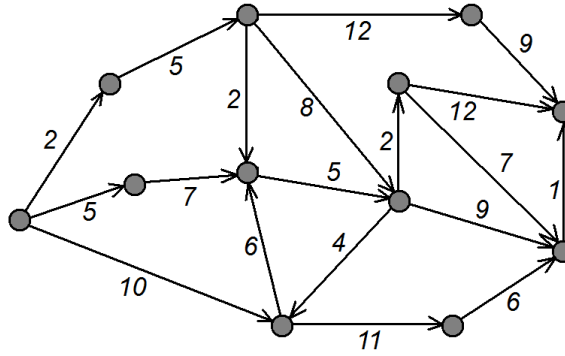
Таблиця 2.1 – Варіанти графів для розрахункового задання

Варіант	Орієнтований граф G
1.	
2.	
3.	

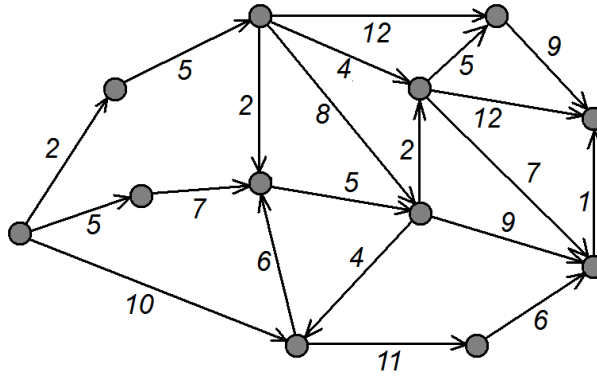
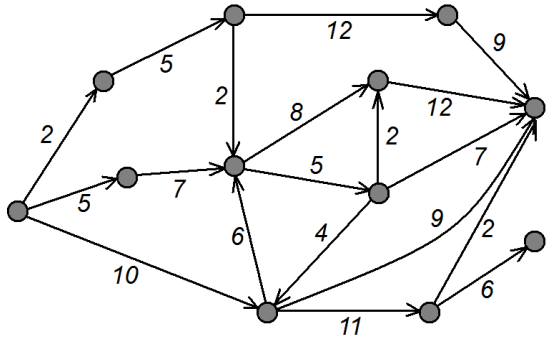
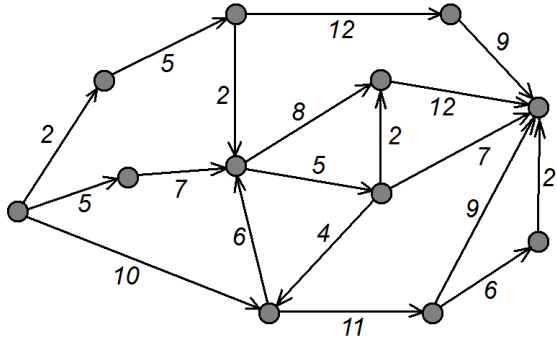
Продовження табл. 2.1

Варіант	Орієнтований граф G
4.	
5.	
6.	

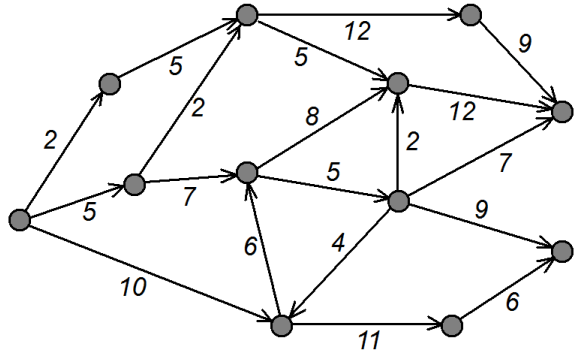
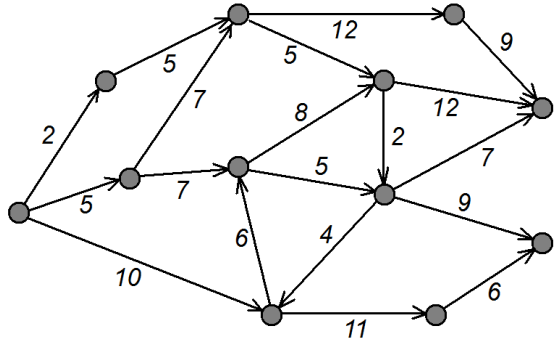
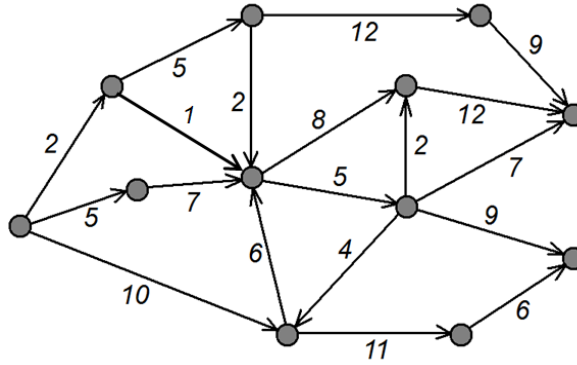
Продовження табл. 2.1

Варіант	Орієнтований граф G
7.	
8.	
9.	

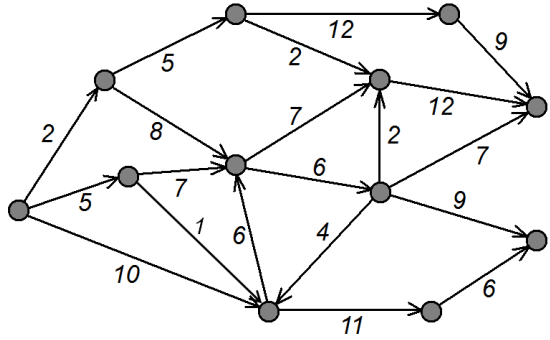
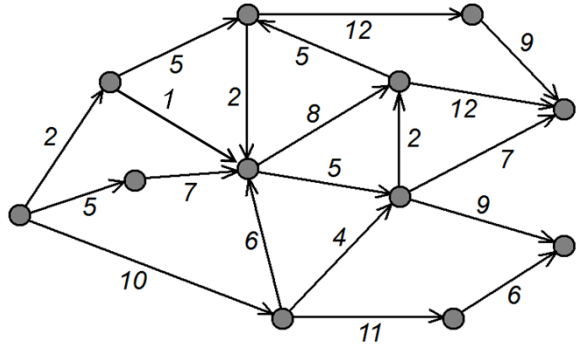
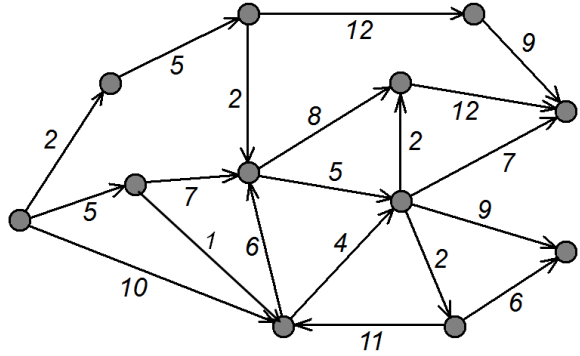
Продовження табл. 2.1

Варіант	Орієнтований граф G
10.	
11.	
12.	

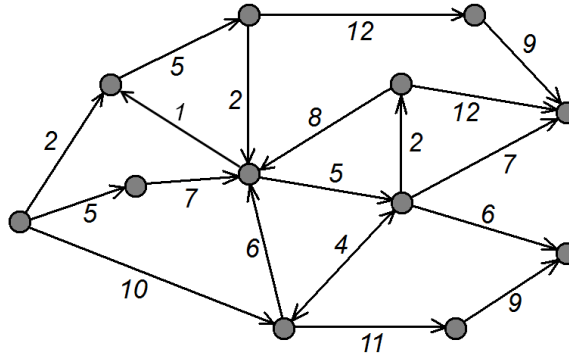
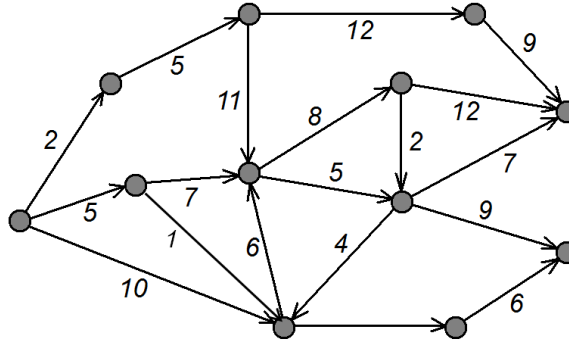
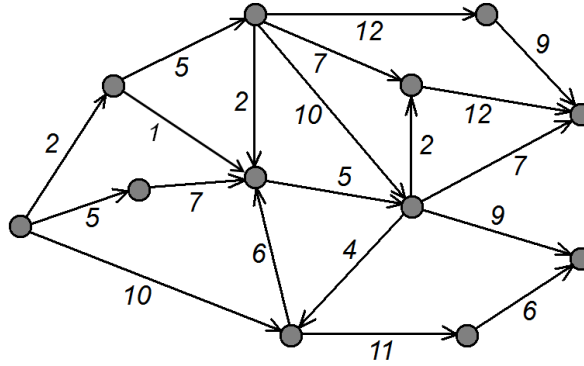
Продовження табл. 2.1

Варіант	Орієнтований граф G
13.	
14.	
15.	

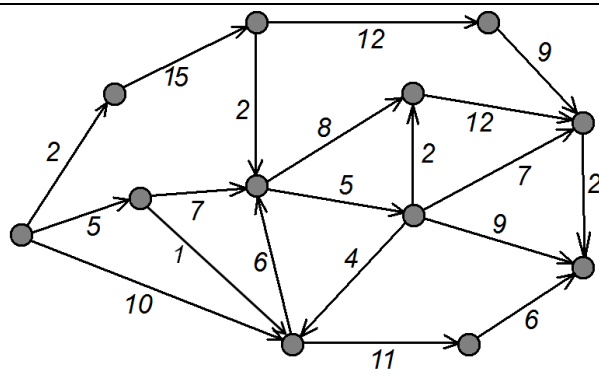
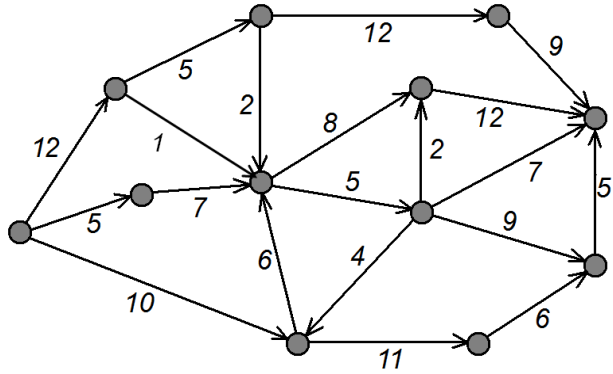
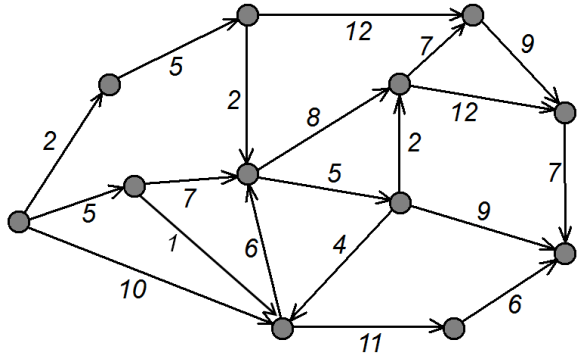
Продовження табл. 2.1

Варіант	Орієнтований граф G
16.	
17.	
18.	

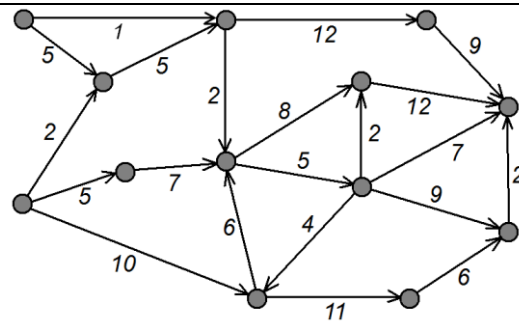
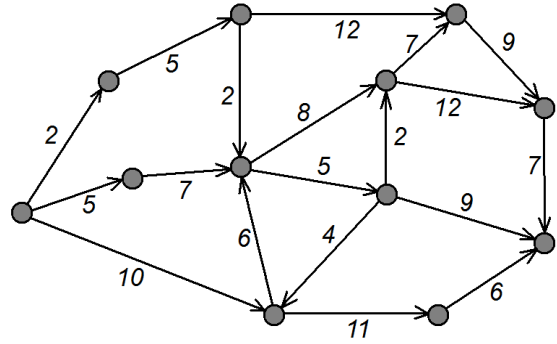
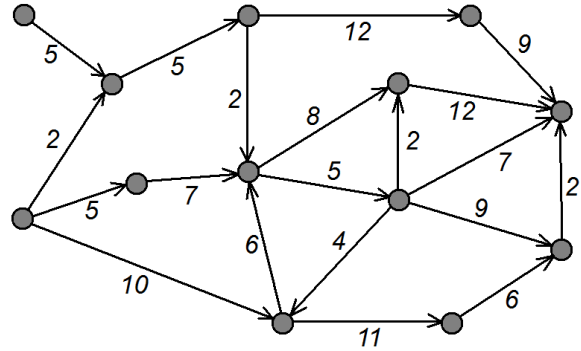
Продовження табл. 2.1

Варіант	Орієнтований граф G
19.	
20.	
21.	

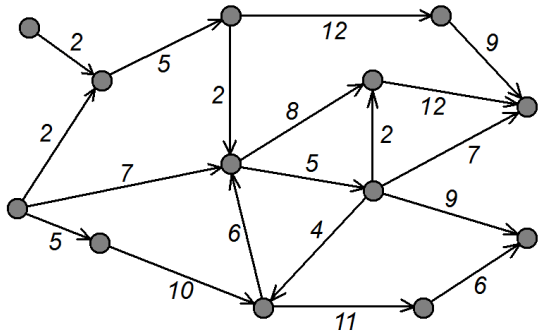
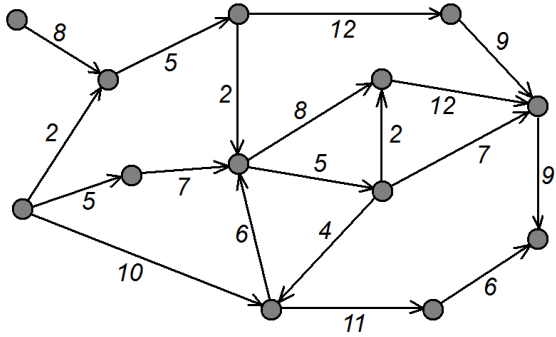
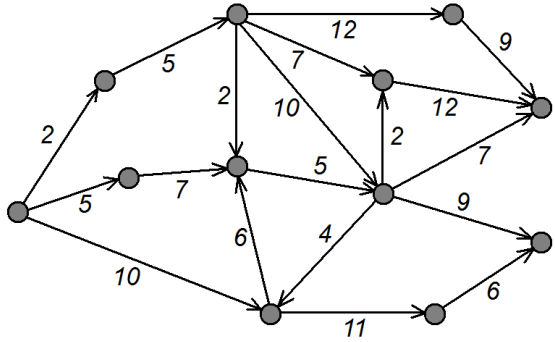
Продовження табл. 2.1

Варіант	Орієнтований граф G
22.	
23.	
24.	

Продовження табл. 2.1

Варіант	Орієнтований граф G
25.	
26.	
27.	

Закінчення табл. 2.1

Варіант	Орієнтований граф G
28.	
29.	
30.	

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Дайте визначення неорієнтованого і орієнтованого графа.
2. Наведіть приклади практичних задач, які можна розв'язувати за допомогою теорії графів.
3. Якими способами можна задати граф?
4. Який граф називають порожнім, нуль-графом, повним? Наведіть відповідні геометричні зображення.
5. Дайте визначення підграфа.
6. Який граф називають зваженим?
7. Що називають ступенем вершини графа?
8. Яку умову щодо степенів вершин має задовольняти однорідний граф?
9. Дайте визначення матриці суміжності графа.
10. В якому випадку матриця суміжності графа буде мати на своїй головній діагоналі відмінні від нуля елементи?
11. Чим визначається розмір матриці інцидентності графа?
12. Дайте визначення маршруту, ланцюга та циклу на графі.
13. Яким чином можна перевірити граф на наявність циклів?
14. Який граф називають ейлеровим, напівейлеровим?
15. Яку умову щодо степенів вершин має задовольняти ейлеров граф?
16. Який граф називають гамільтоновим, квазігамільтоновим?
17. Яку умову щодо степенів вершин має задовольняти гамільтонов граф?
18. Який граф називають деревом?
19. Дайте визначення кістяка графа.
20. Яким чином будується матриця Кірхгофа графа?
21. Як визначити загальну кількість кістякових дерев графа?
22. Дайте визначення цикломатичного числа та коциклічного рангу графа. Що ці величини показують?
23. Сформулюйте задачу про кістяк екстремальної ваги.
24. Наведіть практичні приклади застосування задачі про кістяк екстремальної ваги.
25. Які алгоритми можна використовувати для розв'язання задачі про кістяк екстремальної ваги? Поясніть роботу одного з них більш докладно.
26. Наведіть постановку задачі знаходження найкоротшого шляху.
27. На які підзадачі поділяється задача знаходження найкоротшого шляху?

28. Які обмеження має алгоритм знаходження найкоротшого шляху, запропонований Е. Дейкстрою?

29. Поясніть роботу алгоритму Дейкстри.

30. Як здійснити перехід від ребернозваженого графа до вершиннозваженого?

31. Для чого використовують мережеве планування?

32. Назвіть основні поняття мережевого планування.

33. Що називають діаграмою Ганта? Назвіть сфери їх використання.

34. Які дані містить таблиця опису робіт?

35. Сформулюйте основні правила побудови мережевого графіка.

36. Дайте визначення таких понять: критичний шлях, ранній час початку роботи, пізній час початку роботи, пізній час закінчення роботи; ранній час початку роботи; повний резерв часу; вільний резерв часу; тривалість шляху.

37. Що називають коефіцієнтом напруженості роботи?

38. Як можна класифікувати роботи за коефіцієнтом напруженості?

39. В якій послідовності виконується розрахунок мережевого графіка?

40. Назвіть способи скорочення критичного шляху.

СПИСОК ЛІТЕРАТУРИ

1. Слесарєв В.В. Дискретна математика: навч. посібник / В.В. Слєсарєв, І.В. Новицький, С.А. Ус. – М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2023. – 183 с.

2. Теорія графів. [Електронний ресурс]: навч. посіб. для здобувачів ступеня бакалавра за освітньою програмою «Комп'ютерний моніторинг та геометричне моделювання процесів і систем» спеціальності 122 «Комп'ютерні науки»/ І.М. Кузьменко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,7 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 71 с.

3. Курс лекцій з дисципліни «Дискретна математика» розділ «Теорія графів» для студентів факультету «Комп'ютерно-інформаційних систем і програмної інженерії» / Упоряд.: Н.Р. Крива, Н.І. Блащак. – Тернопіль: ТНТУ, 2023. – 40 с.

4. Балога С.І. Дискретна математика. Навчальний посібник. / С.І. Балога. – Ужгород: ПП «АУТДОР-ШАРК», 2021. – 124 с.

5. Дискретна математика [Електронний ресурс] : методичні рекомендації до самостійної роботи з теми "Теорія графів" для студентів галузі знань 12 "Інформаційні технології" першого (бакалаврського) рівня / уклад. Т. В. Денисова, В. Ф. Сенчуков. – Харків : ХНЕУ ім. С. Кузнеця, 2020. – 100 с.

ЗМІСТ

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до розрахункового завдання «Теорія графів»
з курсу «Дискретна математика» для студентів спеціальностей
122 «Комп'ютерні науки», 124 «Системний аналіз»

Укладачі: МАРЧЕНКО Наталя Андріївна
МЕЛЬНИКОВ Олег Станіславович

Відповідальний за випуск Ю.І. Дорофєєв
Роботу до видання рекомендував М.І. Безменов

У авторській редакції

План 2024р., поз. 182

Підписано до друку 29.03.2024 р. Гарнітура Times. Ум. друк. арк. 2,5.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Електронне видання