МАТЕРІАЛИ ДЛЯ ПІДГОТОВКИ ДО ЗАЛІКУ

НЕОРІЄНТОВАНІ ТА ОРІЄНТОВАНІ ГРАФИ. ДЕРЕВА

1 Хроматичний поліном графа

Довільна функція $f(x)$ на множині вершин графа називається *розкраскою* графа. Розкраска називається *правильною*, якщо $f(x_1) \neq f(x_2)$ для довільних суміжних вершин x_1 і x_2. Мінімальне число *k*, при якому граф *G* є *k-розфарбованим*, називається *хроматичним числом* графа $\chi(G)$.

Для визначення кількості способів розкраски графа в t кольорів можна скласти *хроматичний поліном* $P(G,t)$. Значення полінома при деякому конкретному $t = t_0$ дорівнює числу правильних розкрасок графа в t_0 кольорів. Хроматичний поліном графа має вигляд:

$$P(G,t) = P(G_1,t) + P(G_2,t), \qquad (1)$$

де G_1 – граф, отриманий з G додаванням нового ребра (x,y), а граф G_2 отриманий з G ототожненням вершин *x* і *y*.

Інакше:

$$P(G,t) = P(G_1,t) - P(G_2,t), \qquad (2)$$

де G_1 – граф, отриманий з G видаленням ребра (x,y), а граф G_2 отриманий з G ототожненням вершин *x* і *y*.

Операцію ототожнення вершин *x* і *y* називають також *стягуванням* ребра (x,y).

Обидва варіанти є основою для хроматичної редукції графа. *Хроматична редукція графа* – представлення графа у вигляді декількох порожніх або повних графів, сума хроматичних поліномів яких дорівнює хроматичному поліному графа. Очевидно, що хроматичний поліном порожнього графа O_n дорівнює t^n (кожна вершина може бути розкрашена незалежно від інших), а для повного графа $P(K_n,t) = t(t-1)(t-2)...(t-n+1)$. Останній вираз називають *факторіальним степенем* змінної t: $P(K_n,t) = t^{(n)}$.

Приклад 1 Знайти хроматичний поліном графа (рис. 1).

Розв'язання.

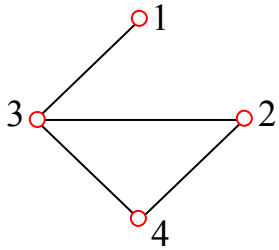


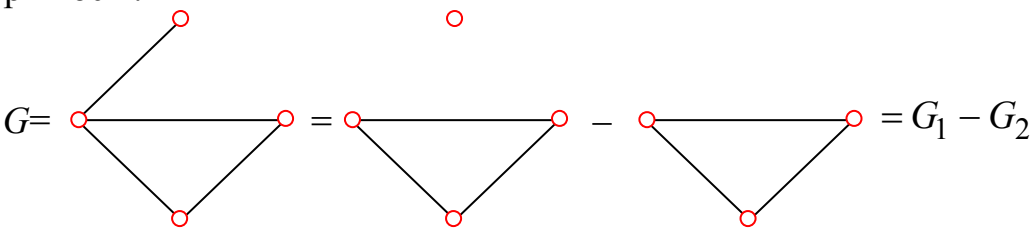
Рисунок 1

В залежності від числа ребер графа можна використати розклад (1) або (2). Якщо граф майже повний, то, додавши декілька ребер за розкладом (1), отримаємо хроматичний поліном у вигляді суми факторіальних степенів.

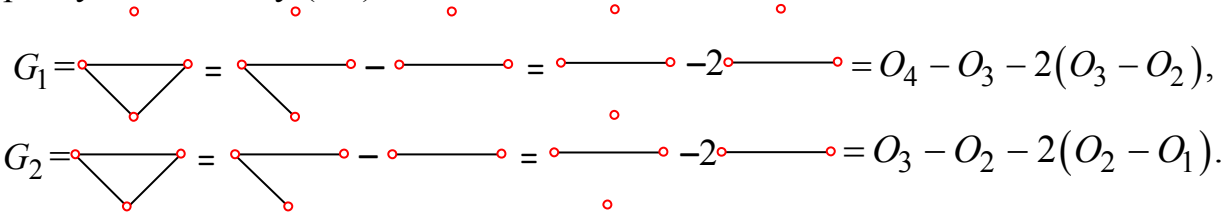
Якщо ж ребер мало й для отримання порожнього графа вимагається видалити тільки декілька ребер, то треба використати розклад (2) з видаленням ребер. Такі дії

називаються хроматичною редукцією.

1. *Хроматична редукція по порожніх графах.* Скористаємось розкладом (5.2). Видаливши ребра і ототожнюючи відповідні вершини (стягуючи ребра), зведемо вихідний граф до порожніх графів. Спочатку розкладемо граф на два, прибравши, а потім стягнувши ребро 1–3. Виконану дію запишемо у вигляді умовної рівності:



Тут операція додавання (або віднімання) відноситься не до самого графу, а до його хроматичного поліному. Таким чином, остання рівність означає, що $P(G)=P(G_1)-P(G_2)$. Далі розкладемо кожний з графів G_1 і G_2, користуючись знову (5.2):



Зведемо подібні члени:

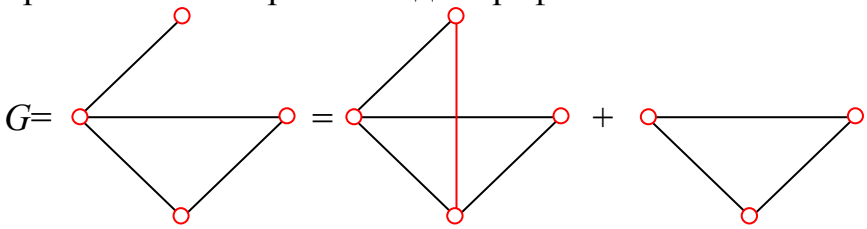
$$G_1-G_2=O_4-O_3-2(O_3-O_2)-(O_3-O_2-2(O_2-O_1))=$$
$$=O_4-4O_3+5O_2-2O_1.$$

Останній розклад називається *хроматичною редукцією по порожніх графах*.

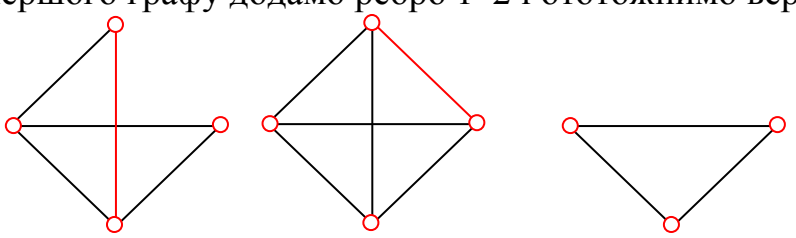
В підсумку отримаємо хроматичний поліном:

$$P(G,x)=x^4-4x^3+5x^2-2x.$$

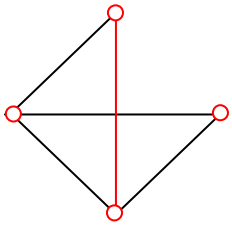
2. *Хроматична редукція по повних графах.* Додавши до графа ребро 1–4, отримаємо граф з більшою кількістю ребер. Потім в цьому ж (вихідному) графі ототожнимо вершини 1 і 4. Отримаємо два графи:



Ототожнення вершин призводить до зменшення порядку й іноді розміру графа. Другий граф – це повний граф K_3, його перетворювати більше не потрібно. До першого графу додамо ребро 1–2 і ототожнимо вершини 1 і 2:



В підсумку отримаємо:



$= K_4 + 2K_3.$

Останній розклад називається *хроматичною редукцією по повних графах*.

Хроматичний поліном буде мати вигляд:

$$P(G,x) = x^{(4)} + 2x^{(3)} = x(x-1)(x-2)(x-3) +$$
$$+2x(x-1)(x-2) = x^4 - 4x^3 + 5x^2 - 2x\,.$$

2 Фундаментальні цикли

Кістяком графа G називають граф, що не містить циклів і складається з ребер графа G і всіх його вершин. Кістяк графа визначається неоднозначно.

Ребра графа, що не входять в кістяк, називаються *хордами*. Цикл, що отримано при додаванні до кістяку графа його хорди, називається *фундаментальним* щодо цієї хорди.

Теорема 1 Число ребер неорграфа, які необхідно видалити для отримання кістяка, не залежить від послідовності їх видалення і дорівнює цикломатичному рангу графа.

Приклад 2 По заданій матриці суміжності

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

визначити число циклів довжини 3 (c_3) і довжини 4 (c_4). Записати матрицю інцидентності та матрицю фундаментальних циклів.

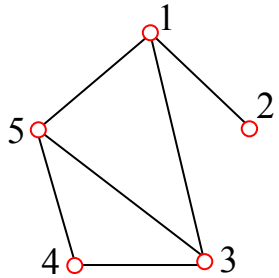


Рисунок 2

Розв'язання.

Матриця суміжності даного графа симетрична, тому їй відповідає неорієнтовані граф. Сума елементів матриці дорівнює 12, отже, по лемі про рукостискання в графі 6 ребер. Побудуємо цей граф (рис. 2). Очевидно, в ньому два цикли (3–4–5 і 1–3–5) довжиною 3 і один цикл (1–3–4–5) довжиною 4.

В цій задачі розв'язок отримано прямим підрахунком по зображенню графа. Для більш складних випадків існує алгоритм розв'язання задачі по матриці суміжності.

Відомо, що слід (trace) матриці суміжності, яка була зведена в k-у степінь, дорівнює числу циклічних маршрутів довжини k. Це число включає в себе й

шукане число циклів. Цикл відрізняється від циклічного маршруту тим, що в ньому не повторюються ребра. Крім того, вважаємо, що шукані цикли не помічені, а в слід матриці входять саме помічені маршрути. Непомічених циклів довжиною 3 в 6 разів менше, ніж помічених, так як кожен позначений цикл може відрізнятися початком, а їх в даному випадку три, і двома напрямками обходу (за і проти годинниковою стрілкою). Зведемо задану матрицю суміжності в третій степінь:

$$A^3 = \begin{pmatrix} 2 & 3 & 6 & 2 & 6 \\ 3 & 0 & 1 & 2 & 1 \\ 6 & 1 & 4 & 5 & 5 \\ 2 & 2 & 5 & 2 & 5 \\ 6 & 1 & 5 & 5 & 4 \end{pmatrix}, \; c_3 = \frac{1}{6}\operatorname{trace} A^3 = 2 .$$

Оскільки циклічних маршрутів довжиною 3, відмінних від циклів довжиною 3, не існує, знайдене число і є відповідь в поставленій задачі.

З циклами довжиною 4 трохи складніше. В слід четвертого степеня матриці суміжності графа:

$$A^4 = \begin{pmatrix} 15 & 2 & 10 & 12 & 10 \\ 2 & 3 & 6 & 2 & 6 \\ 10 & 6 & 16 & 9 & 15 \\ 12 & 2 & 9 & 10 & 9 \\ 10 & 6 & 15 & 9 & 16 \end{pmatrix}$$

входять не тільки цикли, а й циклічні маршрути з подвійним і чотириразовим проходженням ребер. Позначимо кількості таких маршрутів через k_2 і k_4 відповідно. Очевидно, число маршрутів з чотириразовим проходженням одного ребра для вершини x_i дорівнює степеню цієї вершини: $k_4 = \sum_{i=1}^{n} \deg(x_i)$. Число маршрутів з дворазовим проходженням ребра складається з числа k_{2x} маршрутів з висячої вершиною x_i та числа маршрутів з вершиною x_i в центрі. Легко помітити, що $k_{2c} = \sum_{i=1}^{n} \deg(x_i)\big(\deg(x_i) - 1\big)$. Число k_{2x} залежить від степенів вершин, сусідніх з x_i:

$$k_{2c} = \sum_{i=1}^{n} \sum_{\{m,i\} \subseteq G} \big(\deg(x_m) - 1\big)$$

де $\{m, i\}$ – ребро, інцидентне вершинам m та i. Для графа на рис. 2 отримаємо

$$\operatorname{trace} A^4 = 60,$$
$$k_{2x} = 4 + 2 + 5 + 4 + 5 = 20,$$
$$k_{2c} = 6 + 0 + 6 + 2 + 6 = 20,$$
$$k_4 = 3 + 1 + 3 + 2 + 3 = 12.$$

З урахуванням того, що непомічених циклів довжиною 4 в 8 разів менше, отримаємо

$$c_4 = \frac{1}{8}\left(\text{trace}A^4 - k_{2c} - k_{2x} - k_4\right) = \frac{1}{8}(60 - 20 - 20 - 12) = 1.$$

Після перетворень формула набуде вигляду

$$c_4 = \frac{1}{8}\left(\text{trace}A^4 - \sum_{i=1}^{n}\sum_{\{m,i\}\subseteq G}\left(\deg(x_m) - 1\right) - \sum_{i=1}^{n}\deg^2(x_i)\right) = 1.$$

Матрицю інцидентності будуємо за рис. 2, ребра нумеруємо за рис. 3:

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

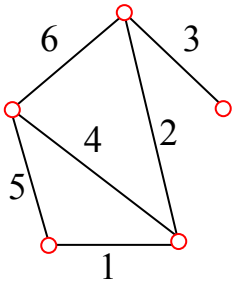


Рисунок 3

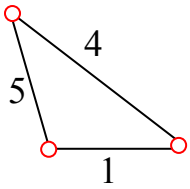


Рисунок 4

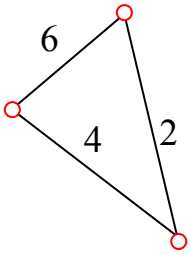


Рисунок 5

Для побудови матриці фундаментальних циклів пронумеруємо ребра графа, починаючи нумерацію з хорд (рис. 3).

Двом хордам, 1 і 2, відповідають два фундаментальних цикли: 1–4–5 і 2–4–6 (рис. 5.4, 5.5). Матриця фундаментальних циклів має два рядки (число циклів) і шість стовпців (число ребер):

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

3 Дерева

Дерево – зв'язний граф без циклів. Неорієнтований граф без циклів називається *ациклічним* (або *лісом*), в іншому разі граф називається *циклічним*. Компонентами ліса є дерева.

Висяча вершина в дереві – вершина степеня 1. Висячі вершини називаються *листям*, всі інші – *внутрішніми* вершинами.

Якщо в дереві особо виділена одна вершина – *корінь*, то таке дерево називається *кореневим,* інакше – *вільним*.

Кореневе дерево можна вважати орграфом з орієнтацією дуг з кореня або в корінь. Очевидно, для довільної вершини кореневого дерева, окрім кореня, $\deg^{in} = 1$. Для кореня $\deg^{in} = 0$, для листів $\deg^{out} = 0$.

Вершини дерева, віддалені на відстані k (в числі дуг) від кореня, утворюють k-й *ярус* (*рівень*) дерева. Найбільше значення k називається *висотою* дерева.

Якщо з вершини x кореневого дерева виходять дуги, то вершини на кінцях цих дуг називаються *синами*.

Вітка до вершини x дерева – це максимальний підграф, який містить x в якості висячої вершини. *Вага* c_k вершини k – найбільший розмір її віток. Центроїд дерева C – множина вершин з найменшою вагою: $C = \{x \mid c(x) = c_{\min}\}$.

Вага довільного листа дерева дорівнює розміру дерева.

Висота дерева з коренем, розташованим в центроїді, не перевищує найменшої ваги його вершин.

Вільне дерево порядку n з двома центроїдами має парну кількість вершин, а вага кожного центроїда дорівнює $\frac{n}{2}$.

Теорема 2 (Жордана) Кожне дерево має центроїд, який складається з однієї або двох суміжних вершин.

Приклад 3 Знайти найменшу вагу вершин дерева (рис. 6) та його центроїд.

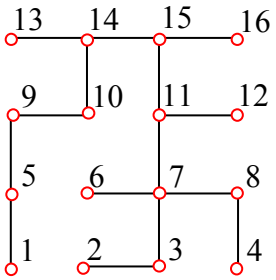


Рисунок 6

Розв'язання.

Очевидно, вага кожної висячої вершини дерева порядку n дорівнює $n-1$. Висячі вершини не можуть скласти центроїд дерева, тому виключимо з розгляду вершини 1, 2, 4, 6, 12, 13 і 16. Для всіх інших вершин знайдемо їх вагу, обчислюючи довжину (розмір) їх віток.

Число віток вершини дорівнює її степені. Розміри віток вершин 3, 5 і 8 дорівнюють 1 і 14 (максимальний підграф, який містить вершину в якості висячої). Отже, вага цих вершин дорівнює 14. До вершини 7 підходять чотири вітки розміром 1, 2, 2 і 10. Таким чином, її вага $c_7 = 10$. Аналогічно обчислюються ваги інших вершин: $c_9 = 13$, $c_{10} = 12$, $c_{14} = 10$, $c_{11} = c_{15} = 8$. Мінімальна вага вершин дорівнює 8, отже, центроїд дерева утворюють дві вершини з такою вагою: 11 і 15.

Кістяком графа G називають граф, що не містить циклів і складається з ребер графа G і всіх його вершин. Таким чином, кістяк графа є деревом. Число ребер кістяка дорівнює рангу графа. Число кістяків графа можна обчислити по матриці Кірхгофа.

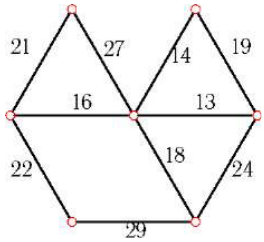


Рисунок 7

Приклад 4 Дано зважений граф (рис. 7). Знайти кількість кістяків графа.

Розв'язання.

Число кістяків графа можна обчислити за його матричними характеристикам. Відома наступна теорема.

Теорема (Кірхгофа) Число кістяків графа дорівнює алгебраїчному доповненню будь-якого елемента матриці Кірхгофа.

Елементи матриці Кірхгофа B графа G визначаються наступним чином. Якщо вершини i та j графа G суміжні, то $b_{ij} = -1$, а якщо вершини i та j не суміжні, то $b_{ij} = 0$. Діагональні елементи матриці Кірхгофа B дорівнюють степеням вершин: $b_{ii} = \deg(i)$. Очевидна властивість: сума елементів в кожному рядку і кожному стовпці матриці Кірхгофа дорівнює 0. Відомо також, що алгебраїчні доповнення всіх елементів матриці Кірхгофа рівні, а ранг матриці Кірхгофа неорграфа порядку n можна обчислити за формулою $\operatorname{rang} B = n - k$, де k – число компонент графа.

Пронумеруємо вершини графа (рис. 8).

Відповідно до визначення запишемо матрицю Кірхгофа:

$$B = \begin{pmatrix} 3 & -1 & 0 & 0 & 0 & -1 & -1 \\ -1 & 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 3 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 3 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 5 \end{pmatrix}.$$

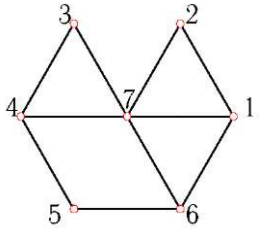


Рисунок 8

Розглянемо мінор елемента b_{11}:

$$M_{11} = \begin{vmatrix} 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 & 0 & -1 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 \\ -1 & -1 & -1 & 0 & -1 & 5 \end{vmatrix} = 79.$$

Таким чином, граф має 79 кістяків.

Алгоритм Краскала, так само, як і алгоритм Прима призначений для пошуку дерева мінімальної вартості неорієнтованого графа. Основна відмінність між даними алгоритмами полягає в тому, що пошук дерева мінімальної довжини за алгоритмом Краскала починається з n дерев, кожне з яких складається з однієї вершини. І на кожному кроці виконуємо операцію об'єднання двох дерев, використовуючи для цього ребро найкоротшої довжини. Процес продовжується поки не отримаємо єдине дерево, яке охоплює всі n вершин вхідного графа, і не містить циклів.

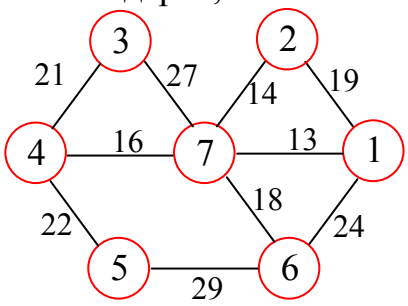


Рисунок 9

Приклад 5 Побудувати для графа (рис. 9) дерево мінімальної вартості за алгоритмом Краскала.

Крок 1: вибираємо ребро найменшої вартості: 1−7=13. В результаті отримаємо одне дерево, яке складається з двох вершин і одного ребра (на рисунку 10 позначено синім кольором) та 5 дерев, які містять по одній вершині.

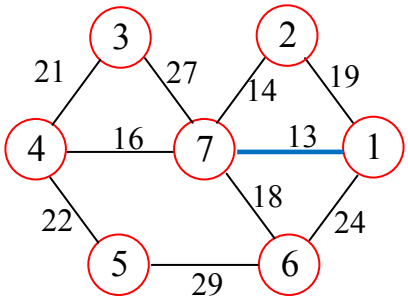


Рисунок 10

Крок 2: на другому кроці об'єднуємо два дерева ребром 2−7=14 (є найменшої довжини). Отримаємо одне дерево з трьома вершинами та двома ребрами, і 4 дерева, які складаються з однієї вершини (рис. 11).

Крок 3: на третьому кроці об'єднуємо два дерева ребром 4−7=16. Результатом даного кроку буде одне дерево, яке складається з чотирьох вершин і трьох ребер, та трьох дерев, які складаються з однієї вершини (рис. 12).

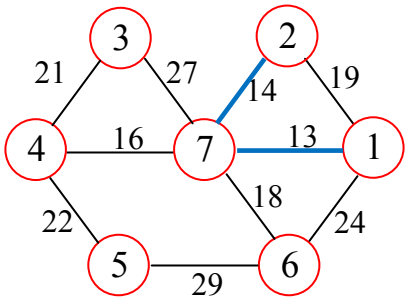


Рисунок 11

Крок 4: знову об'єднаємо два дерева ребром 6−7=18. Отримаємо: одне дерево, яке складається з п'яти вершин і чотирьох ребер, та двох дерев, які складаються з однієї вершини (рис. 13).

Ребро вагою 19 не включається в кістяк, так як воно утворює цикл з ребрами вагою 14 і 13.

Крок 5: знову об'єднаємо два дерева ребром 3−4=21. Отримаємо: одне дерево, яке складається з шести вершин і п'яти ребер, та одного дерева, яке складається з однієї вершини (рис. 14).

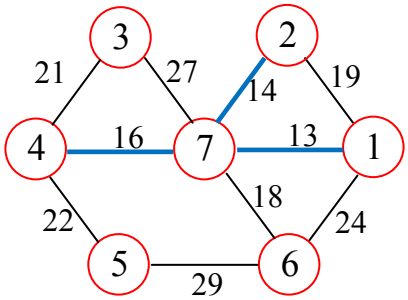


Рисунок 12

Крок 6: на останньому кроці використаємо ребро 4−5=22 і з його допомогою об'єднаємо два дерева. В результаті отримаємо дерево мінімальної вартості 104, яке зображено на рисунку 15.

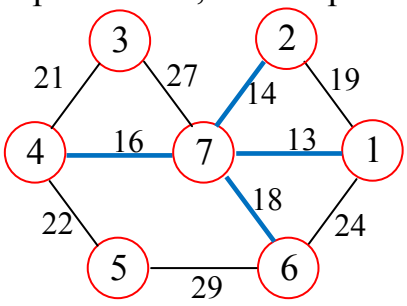


Рисунок 13

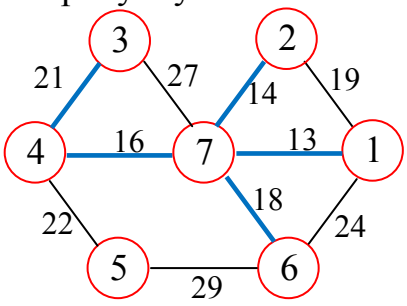


Рисунок 14

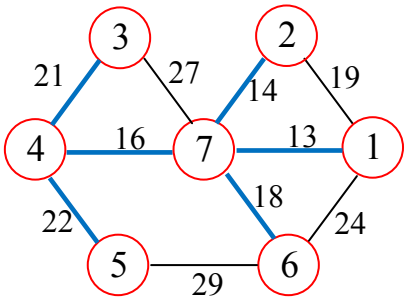


Рисунок 15

Алгоритм Прима полягає в побудові дерева, яке на кожному кроці збільшується на одне ребро. Тобто, побудова починається з графа, який складається з однієї вершини до якої додаємо n–1-у вершину. При цьому, кожен раз, вибираємо мінімальне ребро, яке з'єднує вершину, що вже належить дереву мінімальної вартості, з вершиною, яка ще не міститься в даному дереві.

Приклад 6 Побудувати для графа (рис. 9) дерево мінімальної вартості за алгоритмом Прима.

В якості початкової вибираємо будь-яку вершину графа, нехай вона буде під номером 1. Вершина 1 з'єднана з вершинами 2, 6 і 7. Вибираємо серед них ту, яка має з вершиною 1 ребро найменшої довжини. В нашому прикладі такою є вершина 7 (довжина ребра дорівнює 13). Додаємо її до дерева (на рис. 16 синім кольором).

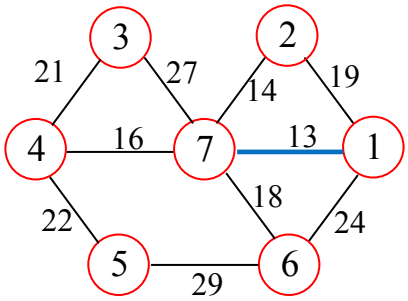


Рисунок 16

На наступному кроці, шукаємо вершину, яка має спільне ребро найменшої довжини з вершинами 1 або 7. Очевидно, що це буде вершина 2, з'єднана з вершиною 7 ребром, довжина якого становить 14 (рис. 17).

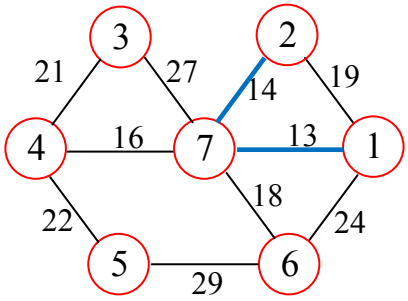


Рисунок 17

Наступною вершиною, яку ми добавимо до дерева буде вершина під номером 4, бо саме вона з'єднана ребром найменшої довжини з вершинами, що вже належать дереву мінімальної вартості, а саме з вершиною 7 (рис. 18).

І так далі, проводимо аналогічні дії, поки всі вершини вхідного графа не будуть включені до дерева мінімальної вартості. У підсумку отримаємо граф мінімальної вартості 104 (рис. 19).

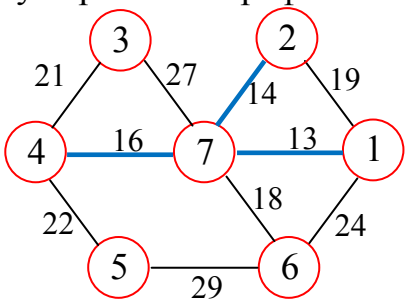


Рисунок 18

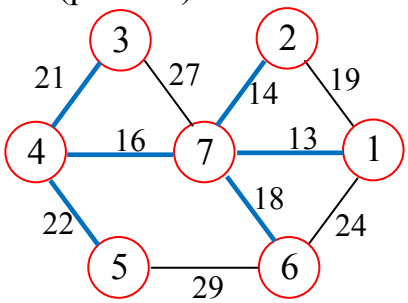


Рисунок 19

4 Маршрути на орграфах

Орієнтовані маршрути – початкова вершина дуги маршруту має збігатися з кінцевою вершиною попередньої дуги, тобто рух за маршрутом припускається лише в напрямках, зазначених стрілками.

Маршрут, який не містить повторних дуг, називається *шляхом*, а той, що не містить повторних вершин, – *простим шляхом*. Замкнений шлях називається *контуром*, а простий замкнений шлях – *простим контуром*.

Орграф без циклів називається *безконтурним*, інакше – *контурним*.

В орграфі дві локальні степені вершини x: $\deg^{in}(x)$ – число дуг, що входять в x, і $\deg^{out}(x)$ – число дуг, що виходять з x. Лема про рукостискання для орграфа має вигляд $\sum \deg^{in}(x) = \sum \deg^{out}(x) = m$, де підсумовування проводиться по всіх вершин графа.

Задачі, пов'язані з маршрутами в орграфі, мають велике практичне значення, що й дає стимул до розвитку та вдосконалення методів їх розв'язання. Найбільш часто постає питання про мінімальні та максимальні відстані, про кількість маршрутів певної довжини.

Приклад 7 Дано орграф (рис. 20). Скільки в ньому маршрутів довжиною 3?

Розв'язання.

Використовуємо алгебраїчний метод розв'язання задачі. Запишемо матрицю суміжності:

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

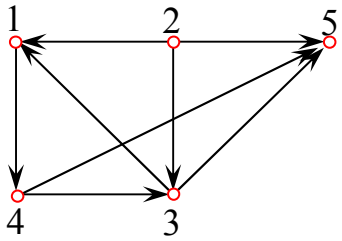


Рисунок 20

Піднесемо цю матрицю в степінь 3:

$$A^3=\begin{pmatrix}1&0&0&0&1\\0&0&1&1&1\\0&0&1&0&1\\0&0&0&1&0\\0&0&0&0&0\end{pmatrix}.$$

Додаючи всі елементи отриманої матриці, знаходимо, що число маршрутів довжиною 3 дорівнює восьми.

Три одиниці, які стоять по головній діагоналі, показують, що сюди входить 3 помічених контури. Очевидно, це контури 1–4–3, 4–3–1 і 3–1–4.

Алгоритм Дейкстра пошуку найкоротшого шляху.

Алгоритм Дейкстра вимагає, щоб довжини всіх дуг були додатними. Основна його ідея полягає в тому, щоб відшукувати найкоротші шляхи в порядку зростання довжини шляху. Найкоротшим серед всіх найкоротших шляхів від вершини-входу є шлях, що складається з однієї дуги, що з'єднує вершину-вхід з найближчою сусідньою вершиною, так як будь-який шлях, що складається з декількох дуг, буде завжди довше першої дуги внаслідок припущення про додатності всіх дугових довжин. Наступним найкоротшим серед найкоротших шляхів повинен бути або шлях з однієї дуги до наступного найближчого з сусіда вершини-входу, або найкоротший шлях з двох дуг, що проходить через вершину, обраний на першому кроці і т.д. Алгоритм Дейкстра складається у виконанні наступних операцій:

Крок 0. Позначаємо нульову вершину індексом $\lambda_0=0$.

Крок k. Нехай вже позначено кілька вершин. Позначимо Q безліч непомічених вершин, суміжних з поміченими. Для кожної вершини k, що належить Q, обчислюємо величину $d_k=\min(\lambda_k+l_{ki})$, де мінімум береться по всім позначеним вершинам i, суміжним з вершиною k. Помічаємо вершину k, для якої величина d_k мінімальна, індексом $\lambda_k=d_k$. Подібну процедуру

повторюємо до тих пір, поки не буде помічена вершина n. Довжина найкоротшого шляху дорівнює λ_n.

Приклад 8 Дано орграф (рис. 21). Знайти найкоротший шлях з вершини A в B.

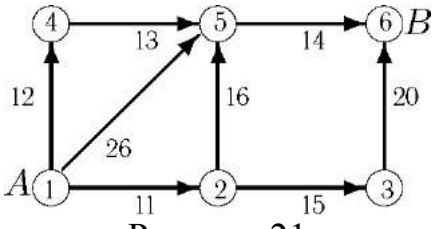


Рисунок 21

Розв'язання.

Помітимо вершину 1 індексом λ_1.

Суміжними з вершиною 1 є вершини 2, 4 і 5. Величини $d_2 = 0+11=11$, $d_4 = 0+12=12$, $d_5 = 0+26=26$, з яких d_2 – мінімальна. Помічаємо вершину 2: $\lambda_2 = 11$. Помічені вершини 1 та 2.

Суміжними з поміченими вершинами 1 і 2 є вершини 3, 4 і 5. Величини $d_3 = 11+15=26$, $d_4 = 0+12=12$, $d_5 = 0+26=26$, $d_5 = 11+16=27$. Помічаємо вершину 4: $\lambda_4 = 12$. Помічені вершини 1, 2 та 4.

Суміжними з поміченими вершинами 1, 2 і 4 є вершини 3 і 5. Величини $d_3 = 11+15=26$, $d_5 = 12+13=25$, $d_5 = 0+26=26$, $d_5 = 11+16=27$. Помічаємо вершину 5: $\lambda_5 = 25$. Помічені вершини 1, 2, 4 та 5.

Суміжними з поміченими вершинами 1, 2, 4 і 5 є вершина 6. Величина $d_6 = 25+14=29$, $d_6 = 26+14=30$, $d_6 = 27+14=31$, $d_6 = 26+20=46$. Помічаємо вершину 6: $\lambda_6 = 29$.

За допомогою алгоритму Дейкстра ми отримали довжину найкоротшого шляху з вершини 1 у вершину 6, яка складе 29. Найкоротшим шляхом є шлях 1–4–5–6 (рис. 8).

5 Кістяковий орграф, його зв'язність

Кістяк орграфа – неорграф з тими ж вершинами, але ребрами замість відповідних дуг.

Орграф називається *зв'язним*, якщо є зв'язним його кістяк.

Вершина x *досяжна* з вершини y, якщо існує маршрут з y в x.

Орграф називається *сильно зв'язним* (або *орзв'язним*), якщо будь-яка його вершина досяжна з будь-якої вершини.

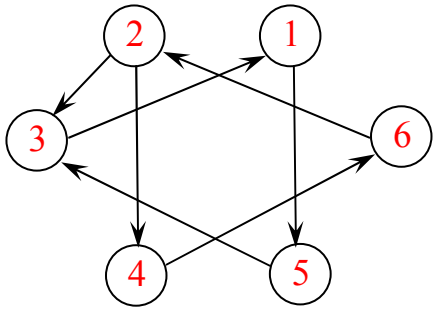


Рисунок 22

Граф називається *орієнтованим*, якщо він є кістяком сильно зв'язного графа.

Приклад 9 Знайти компоненти сильної зв'язності графа (рис. 22).

Розв'язання.

Матриця суміжності графа має вигляд

$$A_1 = A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

У графі 7 дуг, тому найбільший шлях буде не довше семи. Побудуємо матрицю досяжності:

$$A_7 = \sum_{k=1}^{7} A^k = \begin{pmatrix} 2 & 0 & 2 & 0 & 3 & 0 \\ 3 & 2 & 6 & 3 & 3 & 2 \\ 3 & 0 & 2 & 0 & 2 & 0 \\ 3 & 2 & 3 & 2 & 1 & 3 \\ 2 & 0 & 3 & 0 & 2 & 0 \\ 3 & 3 & 3 & 2 & 3 & 2 \end{pmatrix}.$$

Виділимо з цієї матриці головні мінори максимального порядку, що не містять нулі. Якщо граф зв'язний, то в матриці будуть рядки, що не містять нулів. Це рядки 2, 4, 6:

$$\begin{pmatrix} . & . & . & . & . & . \\ 3 & 2 & 6 & 3 & 3 & 2 \\ . & . & . & . & . & . \\ 3 & 2 & 3 & 2 & 1 & 3 \\ . & . & . & . & . & . \\ 3 & 3 & 3 & 2 & 3 & 2 \end{pmatrix}.$$

Мінор з рядками і стовпцями з цими номерами відповідає одній компоненті зв'язності:

$$A_{(2,4,6)} = \begin{pmatrix} . & . & . & . & . & . \\ . & 2 & . & 3 & . & 2 \\ . & . & . & . & . & . \\ . & 2 & . & 2 & . & 3 \\ . & . & . & . & . & . \\ . & 3 & . & 2 & . & 2 \end{pmatrix}.$$

Видалимо з матриці рядки і стовпці з цими номерами. Отримаємо мінор, який відповідає другій компоненті зв'язності:

$$A_{(1,3,5)} = \begin{pmatrix} 2 & . & 2 & . & 3 & . \\ . & . & . & . & 3 & . \\ 3 & . & 2 & . & 2 & . \\ . & . & . & . & . & . \\ 2 & . & 3 & . & 2 & . \\ . & . & . & . & . & . \end{pmatrix}.$$

Отже, в графі дві компоненти сильної зв'язності: підграф з вершинами 1, 3, 5 і підграф з вершинами 2, 4, 6. Зобразимо обидві компоненти сильної зв'язності у вигляді окремих графів (рис. 23, 24). Загальна кількість ребер в компонентах менше розміру вихідного графа. Дуга [2, 3] не увійшла в одну компоненту.

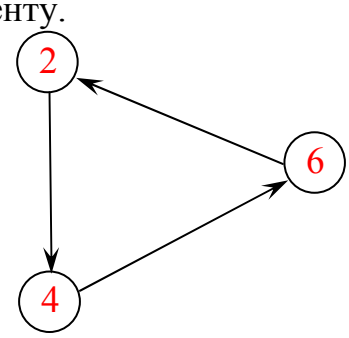


Рисунок 23

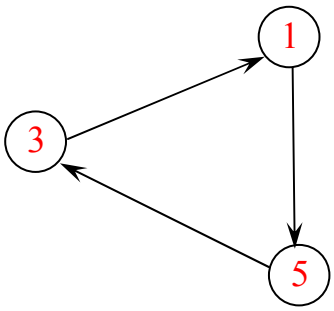


Рисунок 24

6 Транспортна мережа

Мережею називається зв'язний орграф без петель, тобто зважений орграф з двома виділеними вершинами: *витоком* і *стоком*. *Потоком* в мережі називається деяка функція, яка ставить у відповідність дузі деяке число – вагу дуги. Вага дуги означає її пропускну здатність. Потік дуги не перевищує її пропускної здатності та може змінюватися. Потік виходить з витоку й без втрат, в тому ж обсязі заходить до стоку. Якщо потік дорівнює вазі дуги, то ця дуга є *насиченою*, тобто через неї можна пройти при розгляді ланцюгів в графі. Потік називається *насиченим*, якщо будь-який ланцюг з витоку в стік містить насичену дугу. Умова рівноваги (за обсягом входу і виходу) виконується для кожної вершини мережі.

Завдання про найбільший потік в мережі – не єдина, але, ймовірно, основна задача для потоків в мережі. Очевидна можливість практичного застосування цього завдання для вирішення транспортних проблем (пробки на дорогах можна умовно пов'язувати з насиченням мережі або окремої її дуги), проблем транспортування.

Алгоритм Форда-Фалкерсона пошуку максимального потоку в мережі складається з двох частин – насичення потоку і його перерозподілу.

1. *Насичення потоку*. Розшукуються ланцюги з витоку в стік й всі дуги ланцюга насичуються однаковим найбільшим потоком, що визначаються пропускною здатністю найбільш «тонкої» дуги або найменшою різницею між

пропускною спроможністю й потоком в дузі. Різні ланцюги можуть мати спільні дуги. Отриманий потік узгоджений з умовою збереження в вузлах (вершинах). Потік, що входить в вершину, дорівнює потоку, що виходить з неї. Потік в мережі проходить по ланцюгах з витоку в стік, тобто неприпустимий багаторазовий прохід по окремій дузі. Насичені дуги викреслюються. Перша частина завдання вважається розв'язаною, якщо немає ненасичених ланцюгів з витоку в стік. Перша частина задачі не має однозначної відповіді.

2. *Перерозподіл потоку*. Перерозподіл потоку виконується виходячи з умови досягнення загального за мережею максимуму потоку. Для цього в графі, в якому знята орієнтація дуг розшукуються маршрути з витоку в стік, що складаються з ребер, які відповідають ненасиченим дугам, спрямованим вперед, і непорожнім дугам, спрямованим назад. Потоки в дугах прямого напрямку збільшуються на величину, на яку зменшуються потоки в зворотних дугах обраного маршруту. При цьому, очевидно, не можна перевищувати пропускну здатність дуг, спрямованих вперед, і допускати негативних потоків в зворотних дугах. У деяких випадках при вдалому виборі ланцюгів в першій частині алгоритму перерозподілу потоку буде не потрібно.

Приклад 10 Задана пропускна здатність дуг транспортної мережі (рис. 25) з початком в вершині 1 й кінцем в вершині 8. Використовуючи алгоритм Форда-Фалкерсона, знайти максимальний потік в мережі.

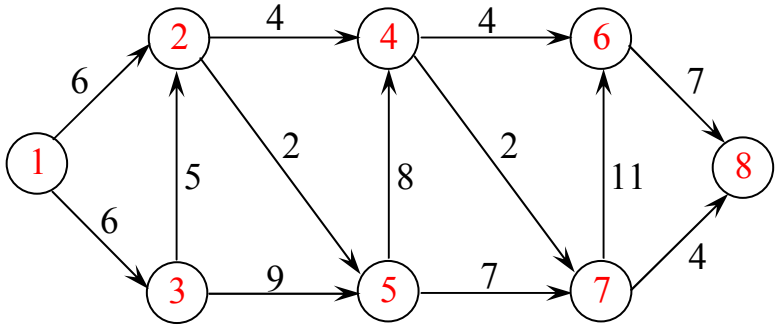


Рисунок 25

Розв'язання.

1. *Насичення потоку*. Розглянемо шлях 1–2–4–6–8. Припустимо через цей шлях потік, рівний 4 (найменша вага). При цьому дуги [2, 4] і [4, 6] будуть насиченими. Аналогічно, шлях 1–3–5–7–8 наситимо потоком 4. Розподіл потоку відзначимо на графі (рис. 26). У чисельнику ставимо пропускну здатність, в знаменнику – потік. Чисельник завжди більше знаменника, а знаменник може бути і нулем. Насичені дуги викреслюємо.

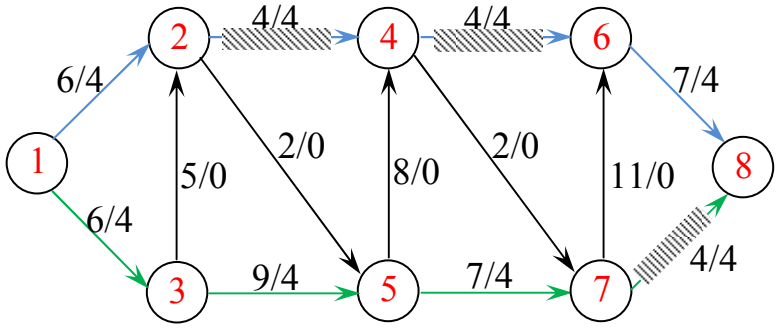


Рисунок 26

Зауважимо, що з 1 в 8 є ще ненасичений шлях, 1–3–2–5–4–7–6–8, потік в якому можна збільшити на 2. При цьому наситяться дуги [1, 3], [2, 5] і [4, 7], їх викреслюємо (рис. 27).

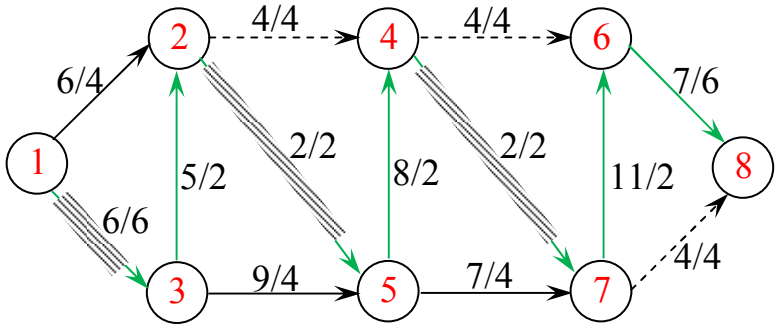


Рисунок 27

З 1 в 8 більше немає ненасичених шляхів. По дузі [1, 3] рухатися не можна (вона вже насичена), а рух по дузі [1, 2] закінчується в вершині 2, так як обидві дуги, які виходять з неї, насичені.

2. *Перерозподіл потоку*. Знайдемо послідовність вершин від 1 до 8, таку, що дуги, які з'єднують сусідні вершини, спрямовані з 1 в 8, не насичені, а дуги, спрямовані в зворотному напрямку, не порожні. Маємо єдину послідовність: 1–2–3–5–7–6–8. Перерозподіляємо потік. Потік в дугах прямого напрямку збільшуємо на 1 (бо найменша різниця 1=7–6), а потік в дугах зворотного напрямку зменшуємо на 1. Процес продовжуємо до тих пір, поки одна з прямих дуг не буде насичена або якась зворотна дуга не буде порожня.

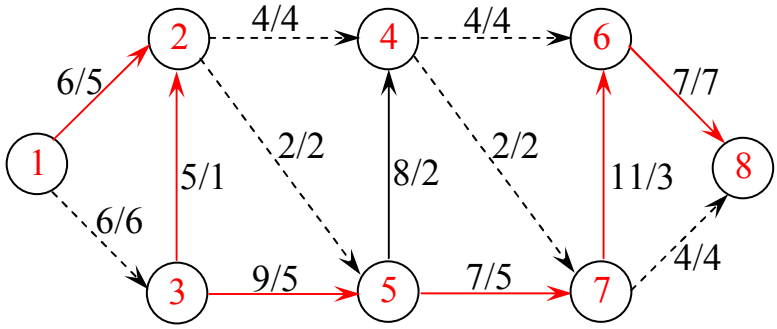


Рисунок 28

Потік в насиченій мережі можна порахувати по потоку, що виходить з витоку 1 або входить в стік 8. Очевидно, ці числа повинні бути рівні. Крім того, для перевірки розв'язку слід перевірити умову збереження потоку по вузлах. Для кожного вузла сумарний вхідний потік має дорівнювати потоку, який виходить. У розглянутому прикладі потік дорівнює 11. Розподіл потоку по дугах при одному і тому ж сумарному мінімальному потоці в мережі не єдиний.

Топологічне сортування мережі. Нехай в мережі в загальному випадку є кілька витоків і стоків. Стоки і витоки займають в мережі крайні положення. Всі інші вершини можуть бути далі або ближче до них. Розташування вершини в мережі визначається її *рівнем*. Визначимо це поняття. Вершини рівня 0 – це

витоки; вони утворюють множину N_0. Якщо N_i – множина вершин рівня $i \le k$, то N_{k+1} – множина вершин рівня $k+1$ складається з тих і тільки тих вершин, попередники яких належать будь-якої з множин N_0, ..., N_k, причому серед цих множин немає порожніх.

Порядковою функцією мережі називають відображення, яке зіставляє кожній вершині мережі її рівень.

Приклад 11 Відсортувати топологічно мережу на рис. 29.

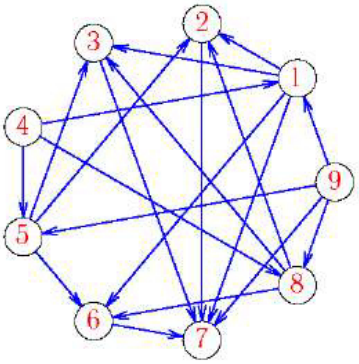


Рисунок 29

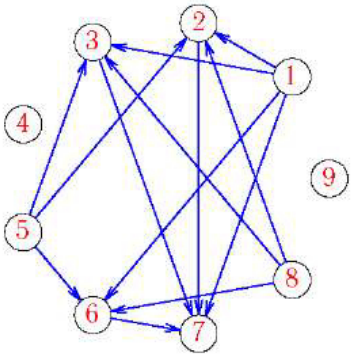


Рисунок 30

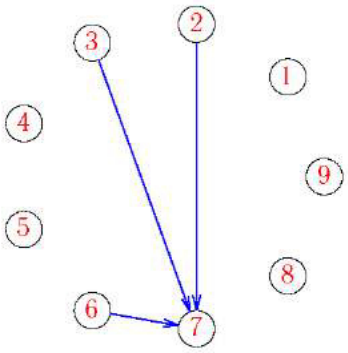


Рисунок 31

Розв'язання.

Знаходимо вершини (4 і 9), півстепені входу яких дорівнює 0. Видаляємо ці вершини і дуги, що виходять з них. Видалені вершини утворюють перший рівень впорядкованої мережі. Отримуємо нову мережу (рис. 30).У новій мережі в вершини 1, 5 і 8 не входять дуги. Видаляємо ці вершини (вони утворюють другий рівень) і отримуємо мережу, зображену на рис. 31. Очевидно, третій рівень складається з вершин 2, 3 і 6, а останній (стік мережі) – з єдиної вершини 7. Зображаємо ту ж мережу за рівнями (рис. 32).

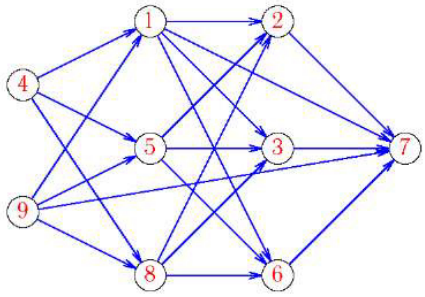


Рисунок 32

7 Десяткове кодування

Наведемо одну з найпростіших кодувань помічених дерев з виділеним коренем – десяткову.

Кодуючи дерево, дотримуємося наступних правил:

1) Кодування починається з кореня і закінчується в корені. Спочатку з вершини, поміченої як корінь дерева, здійснюється пошук в глибину аж до листа, у якого степінь дорівнює 1 (висяча вершина). З цієї вершини починається хід назад з позначкою нулями і видаленням всіх пройдених ребер, що попереджає можливість повторного проходження будь-якого ребра.

2) Кожен крок на одну дугу від кореня кодується одиницею.

3) У вузлі вибираємо напрямок на вершину з меншим номером.

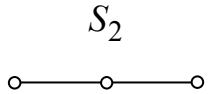
4) Досягнувши листа, йдемо назад, кодуючи кожен крок нулем. Рух назад можливо тільки по тих вершин, які вже були пройдені в першій частині.

5) При русі назад у вузлі завжди вибираємо напрямок на не пройдену вершину з меншим номером.

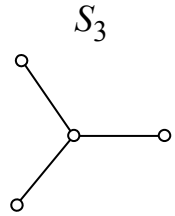
Кодування в такій формі виходить досить компактним, проте воно не несе в собі інформації про номери вершин дерева. Існують аналогічні кодування, де замість одиниць в такому ж порядку проставляються номери або назви вершин.

Є дерева, для яких нескладно вивести формулу десяткового кодування. Розглянемо, наприклад, графи-зірки, які є повними дводольними графами, одна з часток яких складається з однієї вершини. Ця ж вершина є центром і центроїдом дерева. Найменша вага вершин зірки дорівнює 1.

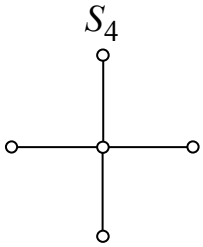
На рисунках 33 – 36 наведені зірки та їх двійкові та десяткові кодування. Корінь дерева розташовується в центральній вершині зірки. Легко отримати загальну формулу: $Z(S_n) = 2\left(4^n - 1\right)\big/3$.



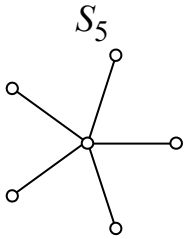
1010
10
Рисунок 33



101010
42
Рисунок 34



10101010
170
Рисунок 35



1010101010
682
Рисунок 36

Якщо корінь помістити в будь-яку з висячих вершин, то код Z' такого дерева буде виражатися більшим числом. Більш того, існує залежність $Z(S_n) - Z'(S_n) = Z(S_{n-1})$. Аналогічно розглядаються ланцюги (C_n; рис. 37). У ланцюгів C_{2n} і C_{2n+1} найменша вага вершин дорівнює n. Центр і центроїд ланцюгів співпадають.

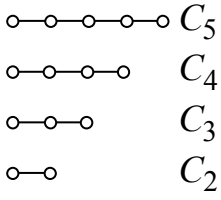


Рисунок 37

У зірках тільки два варіанти розташування кореня з різними десятковими кодуваннями. У ланцюзі ж число варіантів кодувань в залежності від положення кореня зростає зі збільшенням n. Розглянемо найпростіший варіант, розташувавши корінь в кінцевий вершині (листі). Для C_2 отримаємо десяткове кодування 10 і двійкове 2.

Аналогічно для інших ланцюгів: 1100 і 12, 111000 і 56, 11110000 і 240.

Загальна формула для десяткового кодування ланцюга з коренем в кінцевий вершині має вигляд: $Z(C_n) = 2^{n-1}\left(2^{n-1} - 1\right)$.

Приклад 12 Записати десятковий код дерева (рис. 38) з коренем у вершині 3.

Розв'язання.

На основі правила кодування, рухаючись по дереву, проставимо в код одиниці і нулі.

При русі з кореня 3 до вершини 7 проходимо чотири ребра. У код записуємо чотири одиниці: 1111. Повертаючись від вершини 7 до вершини 2 (до найближчої розвилки), проходимо три ребра. Записуємо в код три нулі: 000.

Від вершини 2 до 5 і далі до 8 (менший номер): 11; від 8 назад до 5 і від 5 до 9: 01; від 9 до кореня 3: 000. І нарешті, від 3 до 6 і назад: 10. У результаті, збираючи все разом, отримаємо двійковий код дерева:

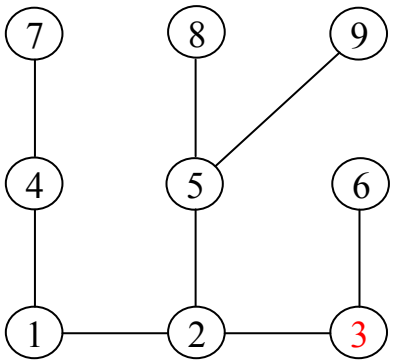


Рисунок 38

$$1\ 111\ 000\ 110\ 100\ 010.$$

Двійкове число з n цифр можна перевести в десяткове число за формулою $\sum_{i=1}^{n} k_i 2^{n-i}$, де k_i – i-а цифра (0 або 1) в двійковому числі. Для числа з прикладу будемо мати:

$$1\cdot 2^{15}+1\cdot 2^{14}+1\cdot 2^{13}+1\cdot 2^{12}+0\cdot 2^{11}+0\cdot 2^{10}+0\cdot 2^{9}+1\cdot 2^{8}+$$
$$+1\cdot 2^{7}+0\cdot 2^{6}+1\cdot 2^{5}+0\cdot 2^{4}+0\cdot 2^{3}+0\cdot 2^{2}+0\cdot 2^{1}=61858.$$

8 Кодування Прюфера

Вибір кодування дерева залежить від розв'язуваної теоретичної або технічної задачі. Серед усіх можливих кодувань природно відшукати оптимальні по якійсь якості розв'язки. Кодування Прюфера застосовується до вільних дерев (неорієнтованим деревам, тобто деревам, в яких немає виділеного кореня).

Наведемо алгоритм кодування поміченого дерева по Прюферу:

1) Знайти висячу вершину з мінімальним номером i.
2) Записати в код Прюфера вершину, суміжну з i.
3) Видалити вершину i з дерева. Якщо дерево не порожнє, то перейти до п. 1.

Приклад 13 Записати код Прюфера для дерева (рис. 39).

Розв'язання.

Знаходимо висячу вершину з мінімальним номером, записуємо в код Прюфера вершину, суміжну з нею, і видаляємо її з дерева. Послідовність визначення коду Прюфера для розглянутого дерева показана на рисунках 40 – 45. Для наочності зображення віддаленої вершини залишається на рисунку, а стирається тільки ребро.

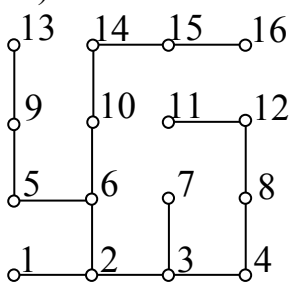
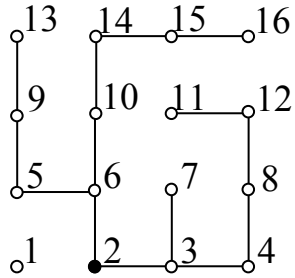
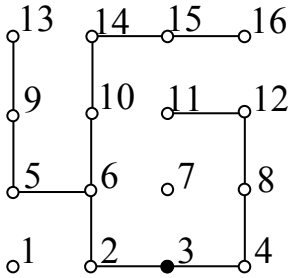


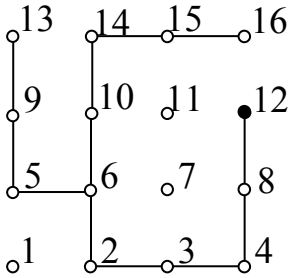
Рисунок 39



$P=[2]$

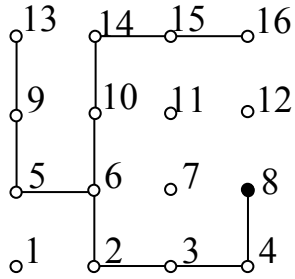


$P=[2,3]$

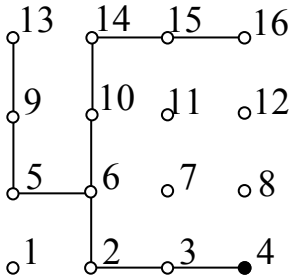


$P=[2,3,12]$

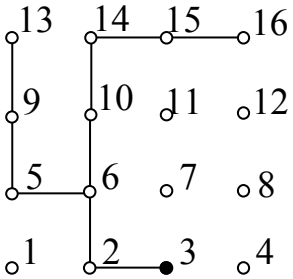
Рисунок 40



$P=[2,3,12,8]$

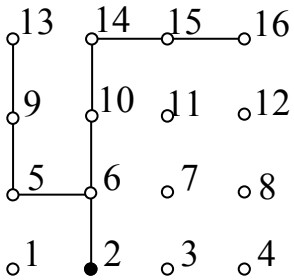


$P=[2,3,12,8,4]$

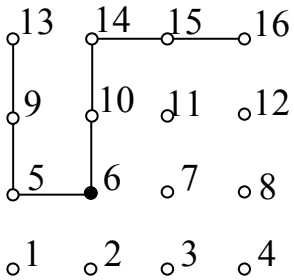


$P=[2,3,12,8,4,3]$

Рисунок 41

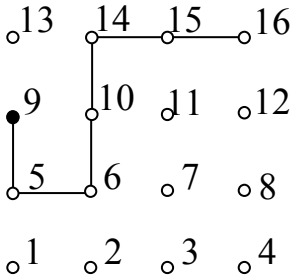


$P=[2,3,12,8,4,3,2]$

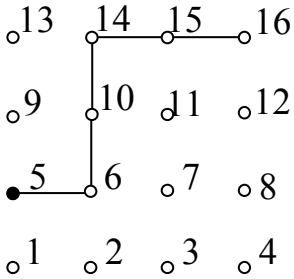


$P=[2,3,12,8,4,3,2,6]$

Рисунок 42

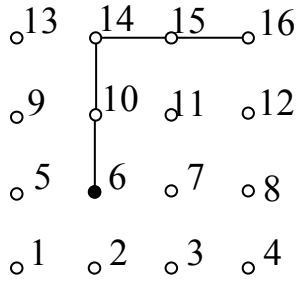


$P=[2,3,12,8,4,3,2,6,9]$

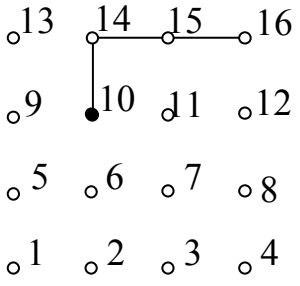


$P=[2,3,12,8,4,3,2,6,9,5]$

Рисунок 43

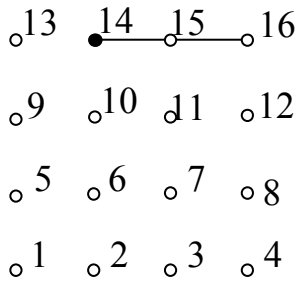


$P=[2,3,12,8,4,3,2,6,9,5,6]$

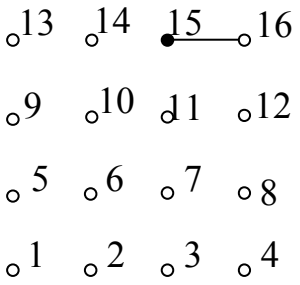


$P=[2,3,12,8,4,3,2,6,9,5,6,10]$

Рисунок 44



$P=[2,3,12,8,4,3,2,6,9,5,6,10,14]$



$P=[2,3,12,8,4,3,2,6,9,5,6,10,14,15]$

Рисунок 45

В результаті код Прюфера має вигляд

$$P=[2,3,12,8,4,3,2,6,9,5,6,10,14,15].$$

Вершини в коді Прюфера можуть повторюватися, більш того, в коді Прюфера може бути тільки одна вершина. Так, якщо номер центральної вершини зірки на рис. 36 дорівнює 5, то код складається з трьох однакових цифр – 555.

9 Розпакування коду Прюфера

Розпакування коду Прюфера P проводиться за наступним алгоритмом.

Введемо список (вектор) деяких елементів $N=[a_1,...,a_n]$ і операцію укорочення списку на один перший елемент, позначивши її зірочкою: $N^*=[a_2,a_3,...,a_n]$. Тоді

1) $A=P$, $N=[1,...,n]$;

2) $x=\min N$, $x\notin A$, $y=a_1$;

3) вершини y та x з'єднати ребром;

4) $N=N\setminus x$, $A=A^*$;

5) якщо $|N|=2$, то з'єднати дві останні вершини, n_1 і n_2, і завершити процедуру; в іншому випадку повернутися до п. 2.

Основна вимога до алгоритмів кодування – однозначність відновлення інформації.

Приклад 13 Побудувати дерево, що відповідає коду Прюфера

$$P=[5,6,7,8,6,10,14,15,11,10,6,7,8,12].$$

Розв'язання.

Для коду, даного в умові завдання, послідовно отримуємо:

- $A=[5,6,7,8,6,10,14,15,11,10,6,7,8,12]$,
  $B=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]$,
  ребро (5, 1);
- $A=[6,7,8,6,10,14,15,11,10,6,7,8,12]$,
  $N=[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]$,
  ребро (6, 2);
- $A=[7,8,6,10,14,15,11,10,6,7,8,12]$,
  $N=[3,4,5,6,7,8,9,10,11,12,13,14,15,16]$,
  ребро (7, 3);
- $A=[8,6,10,14,15,11,10,6,7,8,12]$,
  $N=[4,5,6,7,8,9,10,11,12,13,14,15,16]$,
  ребро (8, 4);
- $A=[6,10,14,15,11,10,6,7,8,12]$,
  $N=[5,6,7,8,9,10,11,12,13,14,15,16]$,
  ребро (6, 5);
- $A=[10,14,15,11,10,6,7,8,12]$,
  $N=[\underline{6,7,8},9,10,11,12,13,14,15,16]$.

  Вершини 6, 7, 8 є в *A*, тому вершину 10 (перша зі списку *A*) з'єднуємо з 9; отримуємо ребро (10, 9). Укорочуємо *A* і видаляємо вершину 9 з *N*. Курсивом виділені вершини списку *N*, присутні в *A*. Далі
- $A=[14,15,11,10,6,7,8,12]$,
  $N=[\underline{6,7,8,10,11,12},13,14,15,16]$.

  Вершини 6, 7, 8, 10, 11, 12 є в *A*, тому вершину 14 (перша з *A*) з'єднуємо з 13; отримуємо ребро (14, 13). Укорочуємо спереду *A*, відрізаючи від нього 14, видаляємо вершину 13 з *N*. Далі
- $A=[15,11,10,6,7,8,12]$,
  $N=[\underline{6,7,8,10,11,12},14,15,16]$.

  Аналогічно отримуємо ребро (15, 14). Вершину 15 беремо з *A*, вершину 14 – з *N*.
- $A=[11,10,6,7,8,12]$,
  $N=[\underline{6,7,8,10,11,12},15,16]$,
  ребро (11, 15);

– $A=[10,6,7,8,12]$,
$N=[\underline{6,7,8,10},11,12,15,16]$,
ребро (10, 11);
– $A=[6,7,8,12]$,
$N=[\underline{6,7,8},10,11,12,15,16]$,
ребро (6, 10);
– $A=[7,8,12]$,
$N=[6,7,8,12,16]$,
ребро (7, 6);
– $A=[8,12]$,
$N=[7,8,12,16]$,
ребро (8, 7);
– $A=[12]$,
$N=[8,12,16]$,
ребро (12, 8).
– На останньому етапі отримуємо ребро, утворене двома вершинами з N:

$A=[\ ]$, $N=[12,16]$, ребро (12, 16).

Дерево, закодоване за Прюфером, – вільне, тобто воно не має кореня. Воно може бути зображено, наприклад, у вигляді на рис. 46 або на рис. 47. В останньому випадку в дереві штучно видалено корінь 6. Отримане дерево має 6 ярусів.

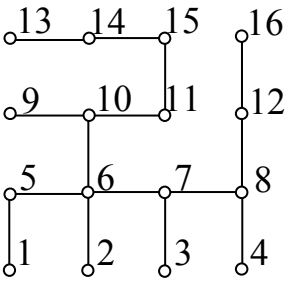


Рисунок 46

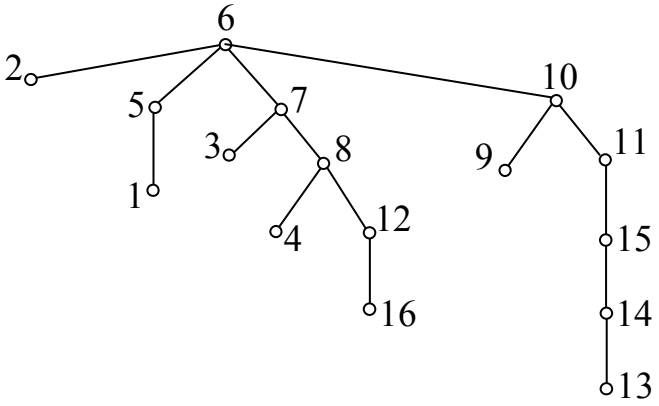


Рисунок 47

10 Кодування Гапта

Для дерев типу 2–3, тобто дерев, кожна не кінцева вершина яких має по 2 або 3 сина, застосовується код Гапта. Без будь-якої зміни алгоритму цей код узагальнюється і на більш складні випадки. Дерево не обов’язково має бути

позначено. Кодування Гапта (на відміну від кодування Прюфера) не зберігає інформацію про імена вершин.

Приклад 14 Знайти код Гапта дерева (рис. 48).

Розв'язання.

Виберемо напрямок обходу дерева. Нехай код складається з числа синів кожної вершини дерева при обході дерева зліва направо, знизу вгору. Висячі вершини (їх степінь дорівнює 1) не мають синів, тому в код дерева порядку n, що має n_0 висячих вершин, увійде $n - n_0$ чисел.

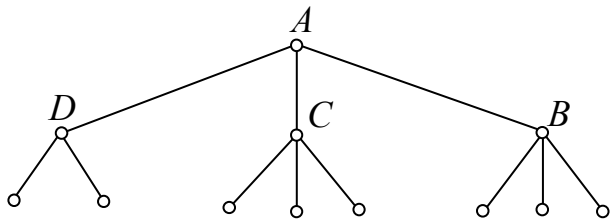


Рисунок 48

Для дерева на рис. 16 кодування повинне містити 12–8=4 числа. Почнемо кодування з самої верхньої вершини – A. Вона має три сина – B, C, D. Отже, заносимо в код число 3:

$$[-\ \ -\ \ -\ \ 3].$$

Переходимо на наступний ярус. Сама права вершина, B, має трьох синів. Заносимо в код число 3:

$$[-\ \ -\ \ 3\ \ 3].$$

Продовжуючи далі, остаточно отримуємо код $[2\ \ 3\ \ 3\ \ 3]$.

11 Розпакування коду Гапта

Приклад 15 За заданим кодом Гапта [3,1,1,1,1,4,2,1,2,3,3,3] побудувати дерево.

Розв'язання.

Побудова починається з кореня. Від кореня, згідно з останнім числу коду, йде три дуги до трьох вершин-синам 2-го ярусу. Розглядаючи наступні з кінця три числа коду, з'ясовуємо, що від цих вершин йде 3, 3 і 2 дуги відповідно. Зобразимо цю частину дерева і продовжуємо будувати наступний ярус. В результаті отримаємо рис. 49.

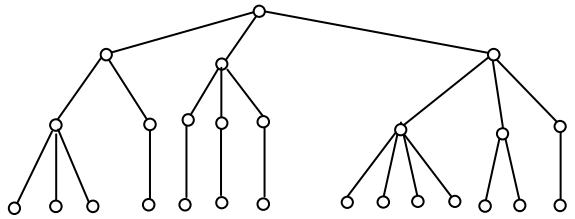


Рисунок 49