ЛЕКЦІЯ 11. ДЕРЕВА

11.1. Основні поняття

1. Сформулюємо означення дерева.

Деревом називають зв'язний неорієнтований граф без простих циклів.

Оскільки дерево не має простих циклів, дерево не може мати кратних ребер та петель. Отже, будь-яке дерево є простим графом.

Приміром, на рис. 11.1 графи G_1 та G_2 є деревами. Граф G_3 не є деревом, оскільки він має простий цикл e, b, a, d, e. Граф G_4 не є деревом, оскільки він не є зв'язним.

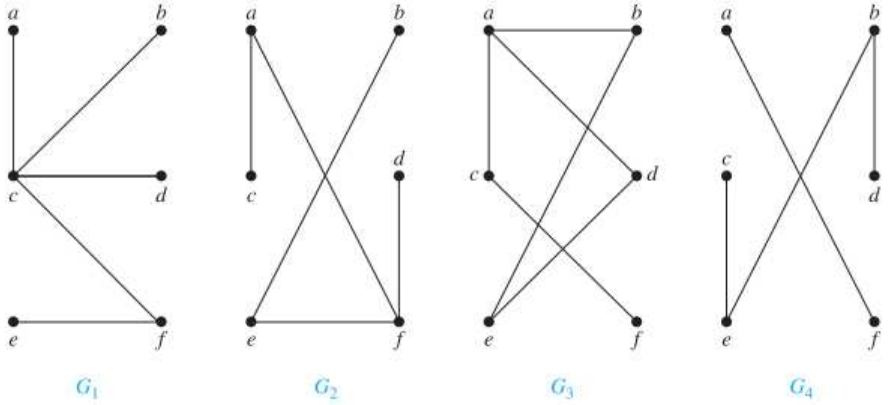

Рис. 11.1

2. Ліси. Граф, який не містить простих циклів, але не є обов'язково зв'язним, називають *лісом*. Будь-яка зв'язна компонента лісу є деревом.

Приміром, на рис. 11.2 подано ліс із трьома зв'язними компонентами.

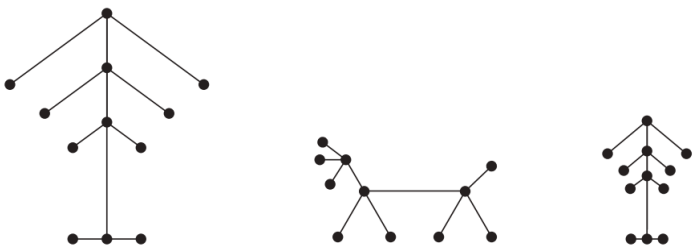

Рис. 11.2

Граф G_4 на рис. 11.1 теж є лісом із двома деревами.

Теорема 11.1.

Неорієнтований граф є деревом тоді й лише тоді, коли існує єдиний простий шлях між будь-якими його вершинами.

3. Орієнтовані дерева. У багатьох застосуваннях у дереві фіксують певну вершини — її називають *коренем*. Оскільки існує єдиний шлях від кореня до будь-якої іншої вершини, то вибір кореня орієнтує кожне ребро, яке виходить з кореня. Тому дерево разом із коренем є *орієнтованим графом*.

Приміром на рис. 11.3 подано орієнтовані дерева, сформовані призначенням коренем вершини a та вершини c неорієнтованого дерева T.

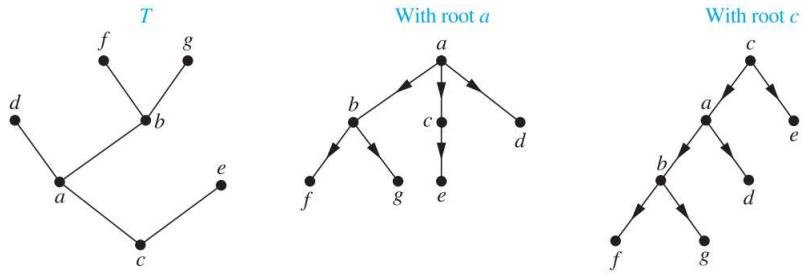

Рис. 11.3

4. Термінологія для дерев має ботанічне та генеалогічне походження. Нехай T є орієнтованим деревом. Якщо v є вершиною в T, відмінною від кореня, *батьком* v є єдина вершина u така, що існує дуга з u до v. Якщо u є батьком v, то v називають *сином* u. Вершини с тим самим батьком називають *братами*. *Предками* вершини, відмінної від кореня, називають вершини називають на шляху від кореня до цієї вершини, виключаючи саму вершину і включаючи корінь. *Нащадками* вершини v називають ті вершини, для яких v є предком. Вершину орієнтованого дерева, що не має синів, називають *листком*. Вершини, які мають синів, називають *внутрішніми*. Корінь є внутрішньою вершиною, якщо вона не єдиною вершиною графа, або листом, якщо єдиною.

Якщо a є вершиною дерева, піддерево з коренем a є підграфом дерева, який містить a та її нащадки та всі ребра, інцидентні цим нащадкам.

Приміром, в орієнтованому дереві T з коренем a (рис. 11.4), b є батьком c. Синами g є вершини h, i та j. Братами h є вершини i та j. Предками e є вершини c, b та a. Внутрішніми вершинами є a, b, c, g, h та j. Листками є d, e, f, i, k, l та m. Піддерево з коренем g подано на рис. 11.5.

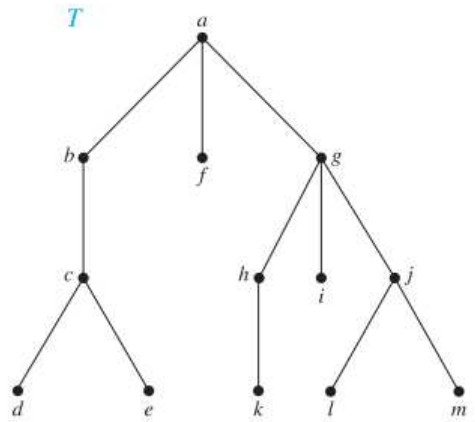

Рис. 11.4

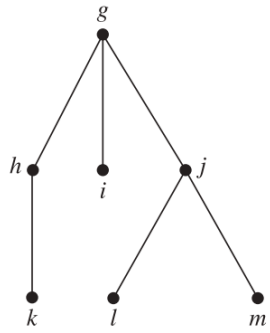

Рис. 11.5

5. m-арні дерева. Орієнтоване дерево називають *m-арним деревом*, якщо всі його внутрішні вершини мають не більше, ніж m синів. Дерево називають *повним* m-арним деревом якщо кожна внутрішня вершина має точно m синів. 2-арне дерево називають *бінарним*.

Приміром, дерево T_1 на рис. 11.6 є повним бінарним деревом, оскільки кожна внутрішня вершина має два сини. T_2 є повним 3-арним деревом, T_3 є повним 5-арним деревом. T_4 не є повним m-арним деревом для будь-якого m, оскільки деякі внутрішні вершини має двох синів, а інші мають трьох синів.

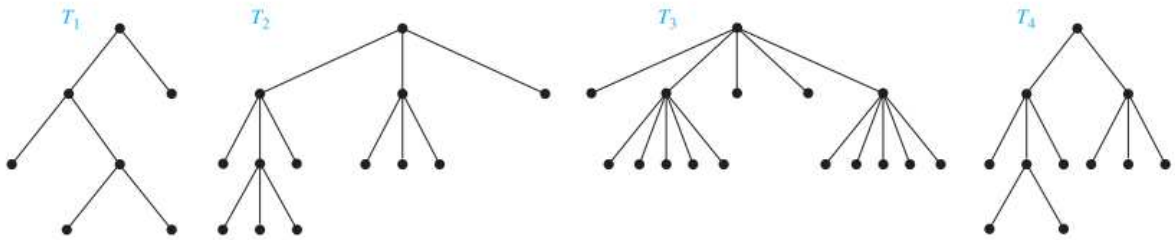

Рис. 11.6

6. Упорядковані орієнтовані дерева. *Упорядкованим* орієнтованим деревом називають орієнтоване дерево, у якому сини кожної внутрішньої вершини упорядковані. Упорядковані орієнтовані дерева зображують так, що сини кожної внутрішньої вершини йдуть за порядком зліва на право.

У впорядкованому бінарному дереві, якщо внутрішня вершина має двох синів, першого сина називають *лівим* сином, а другого сина називають *правим* сином. Дерево з коренем у лівому сині називають *лівим* піддеревом цієї вершини, а дерево з коренем у правому сині низивають *правим* піддеревом цієї вершини.

Приміром, на рис. 11.7, лівим сином d є вершина f, а правим сином — вершина g. Ліве та праве піддерева вершини c показано на рис. 11.8.

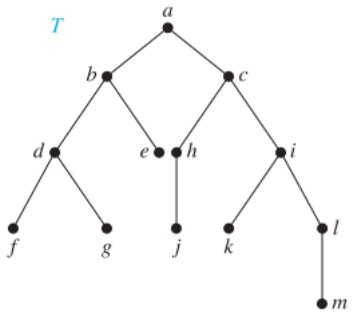

Рис. 11.7

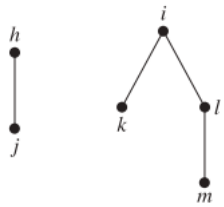

Рис. 11.8

11.2. Властивості дерев

1. У деревах часто потрібно знати співвідношення між кількістю ребер та вершин різного типу.

Теорема 11.2.

Дерево з n вершинами має $n-1$ ребер.

Теорема 11.3.

Повне m-арне дерево з i внутрішніми вершинами містить $n = mi + 1$ вершину.

Теорема 11.4.

Повне m а-рне дерево з:

1) n вершинами має $i = \dfrac{n-1}{m}$ внутрішніх вершин та $l = \dfrac{(m-1)n+1}{n}$ листків;

2) i внутрішніми має $n = mi + 1$ вершин та $l = (m-1)i + 1$ листків;

3) l листками має $n = \dfrac{ml-1}{m-1}$ вершин та $i = \dfrac{l-1}{m-1}$ внутрішніх вершин.

2. Збалансовані m-арні дерева. Часто бажано використовувати орієнтовані дерева, збалансовані так, що піддерева кожної вершини містять шляхи приблизно однакової довжини.

Рівнем вершини v в орієнтованому дереві називають довжину єдиного шляху від кореня до цієї вершини. Вважають, що корінь має нульовий рівень. *Вагою* орієнтованого дерева називають найбільший з рівнів вер-

шин, тобто вага орієнтованого дерева є довжиною найдовшого шляху від кореня до будь-якої вершини.

Приміром, знайдімо рівні всіх вершин дерева на рис. 11.9 та його вагу.

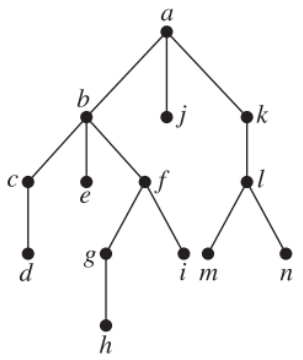

Рис. 11.9

Корінь a має рівень 0. Вершини b, j та k мають рівень 1. Вершини c, e, f та l мають рівень 2. Вершини d, g, i, m та n мають рівень 3. Нарешті, вершина h має рівень 4. Оскільки найбільший рівень вершини дорівнює 4, то дерево має вагу 4.

11.3. Обхід дерева

1. Універсальні системи адресації. Процедури обходу всіх вершин упорядкованого орієнтованого дерева залежать від упорядкування синів. У впорядкованому орієнтованому дереві, сини внутрішньої вершини зображують на рисунку зліва направо.

Опишімо один спосіб цілком упорядкувати вершини упорядкованого орієнтованого дерева. Щоб упорядкувати так вершини, треба спершу розмітити всі вершини. Діємо рекурсивно:

1. Позначаємо корінь числом 0. Позначаємо його k синів (рівня 1) зліва направо числами $1, 2, \ldots, k$.

2. Для кожної вершини v рівня n з міткою A, позначаємо його k_v синів зліва направо мітками $A.1, A.2, \ldots, A.k_v$.

За цієї процедурою, вершина v рівня $n \geq 1$, позначена $x_1.x_2.\ldots.x_n$, де єдиний шлях від кореня до v проходить через x_1-ту вершину 1-го рівня, x_2-ту вершину 2-го рівня і так далі. Така розмітку називають *універсальною системою адресування* впорядкованого орієнтованого дерева.

Можна цілком упорядкувати вершини використовуючи лексикографічний порядок їхніх міток в універсальній системі адресування. Вершина з міткою $x_1.x_2.....x_n$ менше, ніж вершина з міткою $y_1.y_2.....y_m$, якщо існує таке $i, 0 \le i \le n$, що $x_1 = y_1, x_2 = y_2, ..., x_{i-1} = y_{i-1}$ та $x_i < y_i$; або якщо $n < m$ та $x_i = y_i$ для $i = 1,2,...,n$.

Приміром, розгляньмо упорядковане орієнтоване дерево з мітками на рис. 11.10.

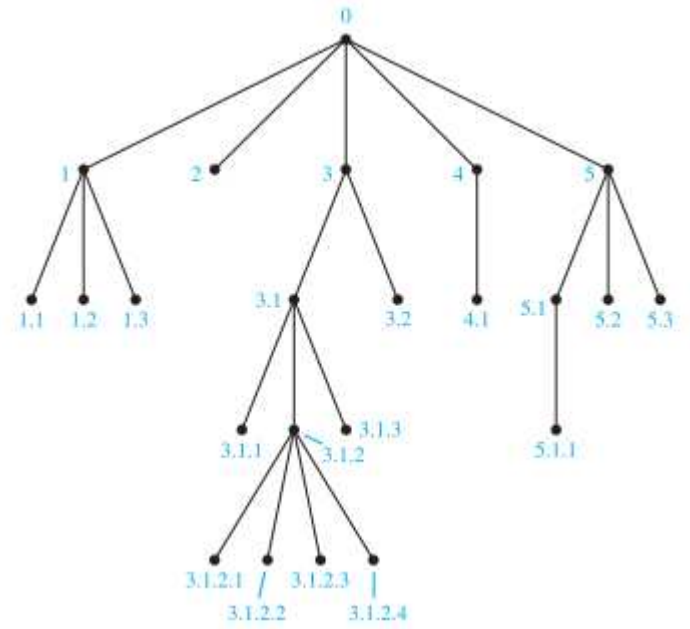

Рис. 11.10

Лексикографічним порядком міток є:

$0 < 1 < 1.1 < 1.2 < 1.3 < 2 < 3 < 3.1 < 3.1.1 < 3.1.2 < 3.1.2.1 < 3.1.2.2 <$
$< 3.1.2.3 << 3.1.2.4 < 3.1.3 < 3.2 < 4 < 4.1 < 5 < 5.1 < 5.1.1 < 5.2 < 5.3$

2. Алгоритми обходу. Процедури систематичного відвідування кожної вершини упорядкованого орієнтованого дерева називають *алгоритмами обходу*. Розгляньмо алгоритми прямого обходу, центрованого обходу та зворотного обходу.

3. Прямий обхід. Розгляньмо впорядковане орієнтоване дерево T з коренем r. Якщо T складається лише з r, то r є прямим обходом дерева T. Інакше, припускаємо, що $T_1, T_2, ..., T_n$ є піддеревами кореня r зліва направо в T. Прямий обхід починають з відвідування r. Його продовжують, обходячи прямо T_1, потім прямо T_2, і так далі до T_n.

Приміром, прямий обхід для дерева на рис. 11.11

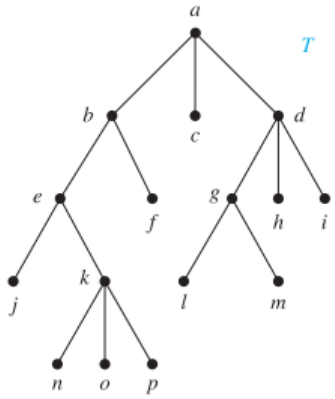

Рис. 11.11

подано на рис. 11.12

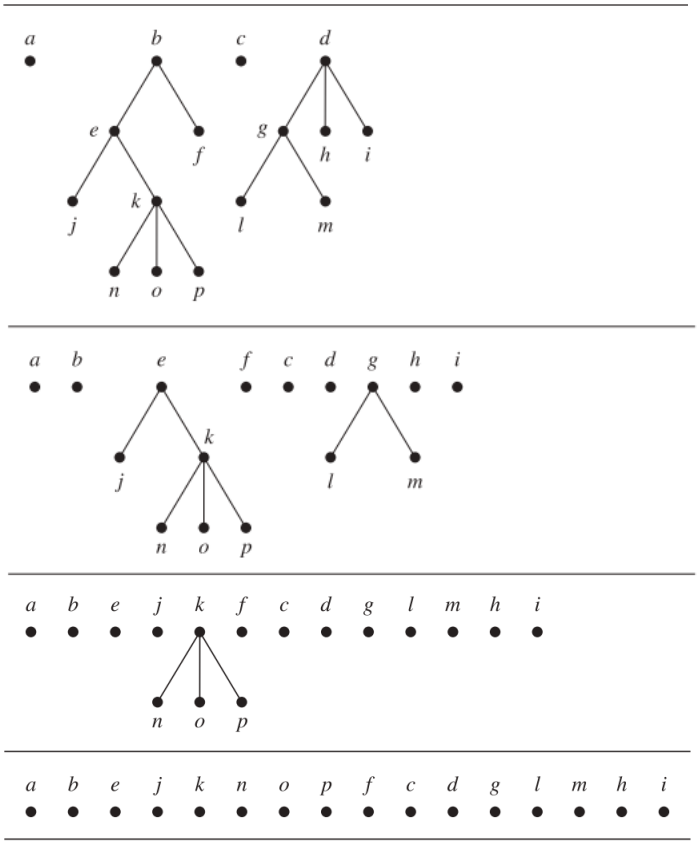

Рис. 11.12

4. Центрований обхід. Розгляньмо впорядковане орієнтоване дерево T з коренем r. Якщо T складається лише з r, то r є центрованим обходом дерева T. Інакше, припускаємо, що $T_1, T_2, ..., T_n$ є піддеревами кореня r зліва направо в T. Центрований обхід починають з відвідування T_1 центровано, потім відвідування r. Його продовжують, обходячи центровано T_1, потім центровано T_2, і так далі до T_n.

Приміром, центрований обхід для дерева на рис. 11.13

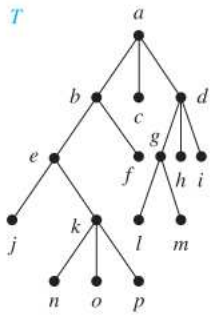

Рис. 11.13

подано на рис. 11.14

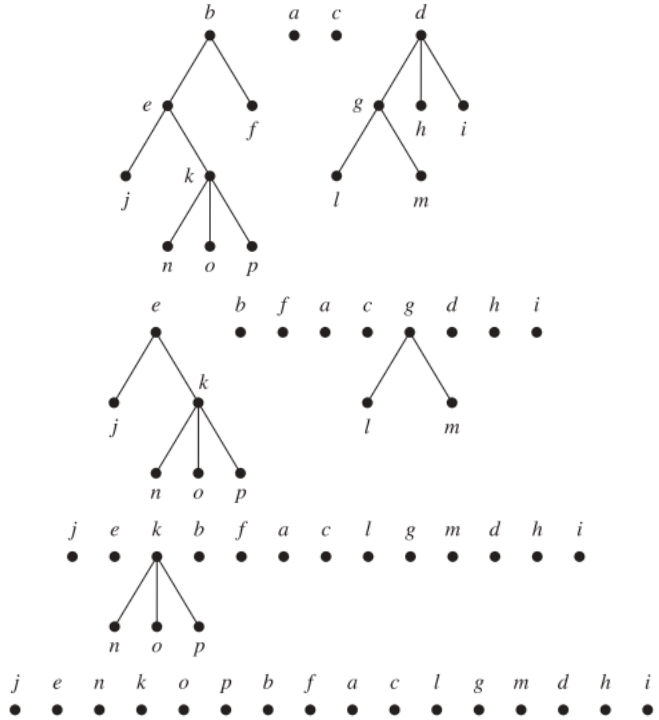

Рис. 11.14

5. Зворотний обхід. Розгляньмо впорядковане орієнтоване дерево T з коренем r. Якщо T складається лише з r, то r є зворотним обходом дерева T. Інакше, припускаємо, що $T_1, T_2, ..., T_n$ є піддеревами кореня r зліва направо в T. Зворотний обхід починають з обходу T_1 зворотно, потім обходу T_2 зворотно і так далі до T_n. Нарешті, відвідуємо r.

Приміром, зворотний обхід для дерева на рис. 11.15

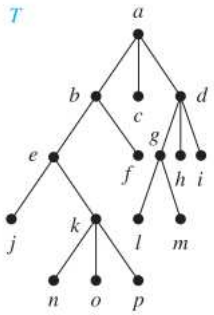

Рис. 11.15

подано на рис. 11.16

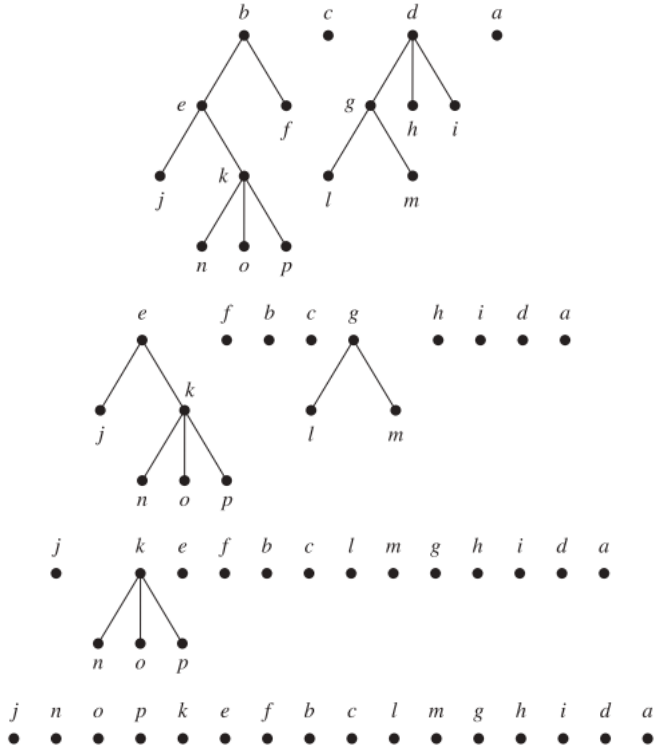

Рис. 11.16

6. Існують прості способи виписи список вершин у прямому, центрованому та зворотному порядку. Спершу, обводимо кривою впорядковане орієнтоване дерево, починаючи з кореня, рухаючись уздовж ребер як на рис. 11.17.

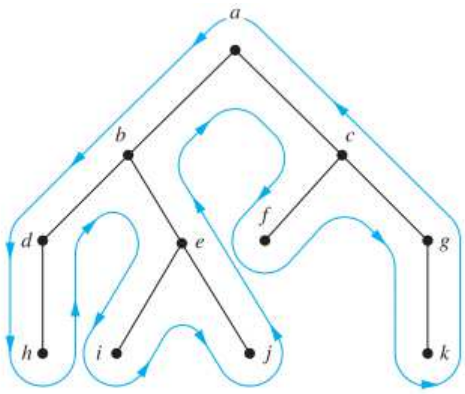

Рис. 11.17

Список вершин у прямому порядку можна одержати виписуючи кожну вершину вперше, коли крива обмине її.

Список вершин у центрованому порядку можна одержати виписуючи листок уперше, коли криве обмине його, а потім кожну внутрішню вершину, коли крива обмине її вдруге.

Список вершин у зворотному порядку можна одержати виписуючи вершину в останній раз, коли крива обмине її, вертаючись до її батька.

А саме, для дерева на рис. 11.17 маємо:

— прямий обхід $a, b, d, h, e, i, j, c, f, g, k$;

— центрований обхід $h, d, b, i, e, j, a, f, c, k, g$;

— зворотний обхід $h, d, i, j, e, b, f, k, g, c, a$.

11.4. Каркасні дерева

1. Розгляньмо простий граф G. *Каркасним деревом* графа G називають підграф G такий, що є деревом, яке містить усі вершини G.

Простий граф із каркасним деревом повинен бути зв'язним, оскільки в каркасному дереві існує шлях між будь-яким двома вершинами. Зворотне твердження також правильне. Будь-який зв'язний простий граф має каркасне дерево.

Приміром, побудуємо каркасне дерево для простого графа на рис. 11.18

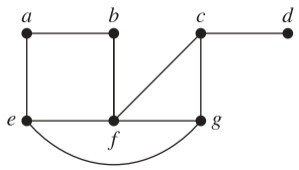

Рис. 11.18

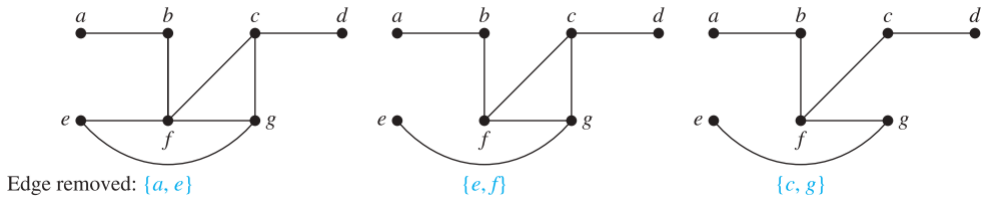

Рис. 11.19

Простий граф G є зв'язним, але не є деревом, оскільки містить прості цикли. Видаляємо ребро $\{a,e\}$. Це вилучити один простий цикл, одержаний підграф залишається ще зв'язним і все ще містить усі вершини G. Потім видаляємо ребро $\{e,f\}$ для того щоб вилучити другий простий цикл. Нарешті, видаляємо ребро $\{c,g\}$ для того щоб одержати простий граф без простих циклів. Цей підграф є каркасним деревом, тому що є деревом, яке містить усі вершини графа G.

Каркасне дерева на рис. 11.19 не є єдиним для графа G. Будь-яке дерево на рис. 11.20 є каркасним для графа G.

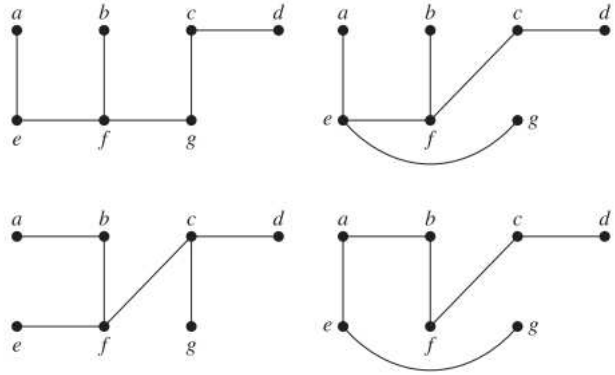

Рис. 11.20

Теорема 11.5.

Простий граф є зв'язним тоді й лише тоді, коли він має каркасне дерево.

2. Алгоритм пошуку в глибину. Замість побудови каркасного дерева видаленням ребер, каркасне дерево можна побудувати послідовним додаванням ребер.

Побудуємо каркасне дерево для зв'язного простого графа використовуючи алгоритм пошуку в глибину. Ми побудуємо орієнтоване дерево, а потім каракасне деревом, заміною в орієнтованому дереві дуг на ребра. Довільним чином вибираємо вершину графа за корінь. Формуємо шлях, що починається в цій вершині послідовно додаючи вершини і ребра, де кожне нове ребро інцидентне останній вершині шляху, а вершина ще не

була включена. Продовжуємо додавання вершин та ребер до цього шляху, до поки можливо. Якщо шлях проходить через усі вершини графа, дерево, яке містить цей шлях є каркасним. Однак, якщо шлях не проходить через усі вершини, до нього додати ще вершини й ребра. Вертаємось до передостанньої вершини шляхи та, якщо можливо, формуємо новий шлях, починаючи з цієї вершини, обходячи вершини, що ще не було включено. Якщо це неможливо, вертаємось назад до наступної вершини, і пробуємо додати нове ребро та вершину знову.

Повторюючи цю процедуру, починаючи з останньої відвіданої вершини, вертаючись на шляху на одну вершину, формуємо нові шляхи якомога довші, до тих пір поки жодного ребра вже не можна додати. Оскільки граф має скінченну кількість ребер і є зв'язним, цей процес завершиться побудовою каркасного дерева. Кожна вершина, що завершує шлях на кожному етапі алгоритму буде листком орієнтованого дерева, і кожна вершина, де починається шлях, буде внутрішньою вершиною.

Приміром, побудуємо каркасне дерево для графа на рис. 11.21 за алгоритмом пошуку в глибину (рис. 11.22).

1. Починаємо з довільно вибраної вершини f.

2. Будуємо шлях послідовно додаючи ребра інцидентні вершинам, які ще не включено до шляху, так довго як це можливо, приміром f, g, h, k, j.

3. Далі, вертаємось до k. Не існує шляху, який починається в k, і містить невідвіданні вершини.

4. Далі, вертаємось до h. Формуємо шлях h, i.

5. Вертаємось до h.

6. Вертаємось до f і формуємо шлях f, d, e, c, a.

7. Вертаємось до c і формуємо шлях c, b.

Каркасне дерево побудувано.

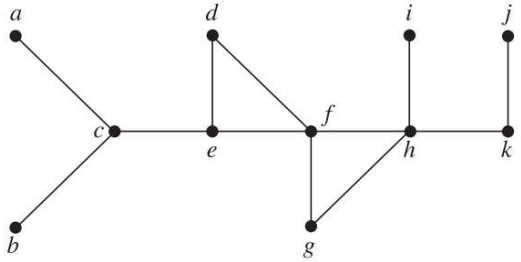

Рис. 11.21

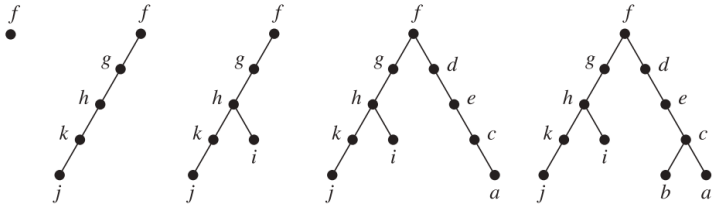

Рис. 11.22