Міністерство освіти і науки України

ВСП « Ковельський промислово-економічний фаховий коледж

Луцького національного технічного університету»

ДИСКРЕТНА МАТЕМАТИКА

Методичні вказівки до практичних занять

для здобувачів освітньо-професійного ступеня фаховий молодший бакалавр спеціальності 122 Комп'ютерні науки

Практичне заняття № 1

Тема: Способи задання графів.
Мета: Навчити студенті задавати графи різними способами.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Що таке гра? Які способи задання графів ви знаєте?
2. Як задати граф матрицею суміжності?
3. Як задати граф матрицею інцидентності?

Теоретичні відомості

<u>*Матриця інцидентності*</u> $\left\|\varepsilon_{ij}\right\|$ – це матриця розміром $m \times n$, де вертикально вказуються вершини $i = \overline{1,n}$, а горизонтально – ребра $j = \overline{1,m}$. На перетині i-того і j-того рядків число ε_{ij} дорівнює:

а) у випадку неорієнтованого графа:

$$\varepsilon_{ij} = \begin{cases} 1, \text{ якщо ребро } e_i \text{ інцидентно вершині } v_j; \\ 0, \text{ якщо ребро } e_i \text{ не інцидентно вершині } v_j; \\ \alpha, \text{ якщо ребро } e_i - \text{ петля} \left(\alpha \neq 0 \text{ і } \alpha \neq 1\right). \end{cases}$$

б) у випадку орієнтованого графа:

$$\varepsilon_{ij} = \begin{cases} -1, \text{ якщо } v_i - \text{початок ребра } e_j; \\ 1, \text{якщо } v_i - \text{кінець ребра } e_j; \\ 0, \text{ якщо вони не інцидентні;} \\ \alpha, \text{ якщо } e_j - \text{петля, а } v_i - \text{інцидентна їй вершина.} \end{cases}$$

<u>*Матриця суміжності*</u> $\left\|\delta_{ij}\right\|$ – це квадратна матриця розміром $n \times n$, де вертикально і горизонтально вказуються вершини графа $i = \overline{1,n}$ і $j = \overline{1,n}$. На перетинанні i-того і j-того рядків число δ_{ij} дорівнює:

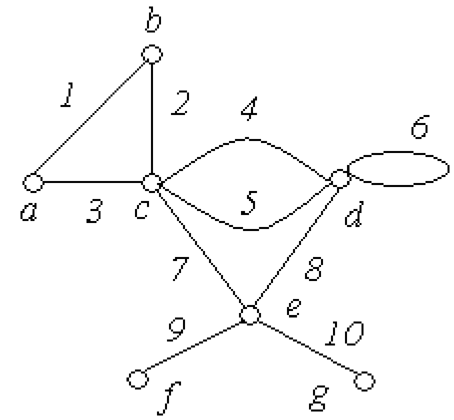
- числу ребер, що з'єднують ці вершини у випадку неорієнтованого графа;

- числу ребер з початком в i-тій вершині і кінцем в j-тій вершині у випадку орієнтованого графа.

<u>Список ребер</u> графа – це таблиця, що складається із трьох рядків. У першому перераховані всі ребра; у другому і третьому – інцидентні їм вершини:

- у випадку неорієнтованого графа порядок вершин у рядку довільний;
- у випадку орієнтованого графа першою записується вершина, де починається ребро (другий рядок); вершина, де закінчується ребро, записується у третій рядок.

<u>Приклад</u> Задати матрицями інцидентності і суміжності, а також списком ребер, неорієнтований граф



Матриця інцидентності

	1	2	3	4	5	6	7	8	9	10
a	1	0	1	0	0	0	0	0	0	0
b	1	1	0	0	0	0	0	0	0	0
c	0	1	1	1	1	0	1	0	0	0
d	0	0	0	1	1	2	0	1	0	0
e	0	0	0	0	0	0	1	1	1	1
f	0	0	0	0	0	0	0	0	1	0
g	0	0	0	0	0	0	0	0	0	1

Матриця суміжності

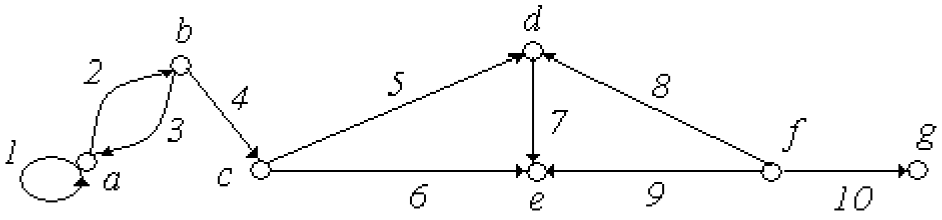
	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	1	0	0	0	0
c	1	1	0	2	1	0	0
d	0	0	2	1	1	0	0
e	0	0	1	1	0	1	1
f	0	0	0	0	1	0	0
g	0	0	0	0	1	0	0

Список ребер

Ребро		1	2	3	4	5	6	7	8	9	10
Вершини	початок	*a*	*b*	*a*	*c*	*c*	*d*	*c*	*d*	*e*	*e*
	кінець	*b*	*c*	*c*	*d*	*d*	*d*	*e*	*e*	*f*	*g*

Матриця суміжності симетрична щодо головної діагоналі.

Список ребер є найбільш компактним способом завдання графів. ***Приклад*** Задати матрицями інцидентності, суміжності, списком ребер орієнтований граф



Матриця інцидентності

	1	2	3	4	5	6	7	8	9	10
a	2	-1	1	0	0	0	0	0	0	0
b	0	1	-1	-1	0	0	0	0	0	0
c	0	0	0	1	-1	-1	0	0	0	0
d	0	0	0	0	1	0	-1	1	0	0
e	0	0	0	0	0	1	1	0	1	0
f	0	0	0	0	0	0	0	-1	-1	-1
g	0	0	0	0	0	0	0	0	0	1

Матриця суміжності

	a	b	c	d	e	f	g
a	1	1	0	0	0	0	0
b	1	0	1	0	0	0	0
c	0	0	0	1	1	0	0
d	0	0	0	0	1	0	0
e	0	0	0	0	0	0	0
f	0	0	0	1	1	0	1
g	0	0	0	0	0	0	0

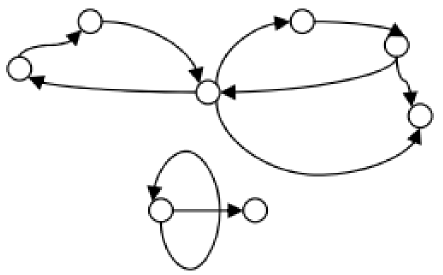
Список ребер

Ребро		1	2	3	4	5	6	7	8	9	10
Вершини	початок	*a*	*a*	*b*	*b*	*c*	*c*	*d*	*f*	*f*	*f*
	кінець	*a*	*b*	*a*	*c*	*d*	*e*	*e*	*d*	*e*	*g*

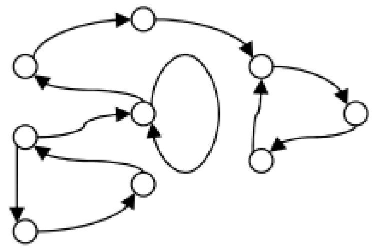
Індивідуальні завдання

1. Задати неорієнтований граф матрицею суміжності, інцидентності та списком ребер.
2. Задати орієнтований граф матрицею суміжності, інцидентності та списком ребер. Визначити степені вершин в орієнтованому графі.

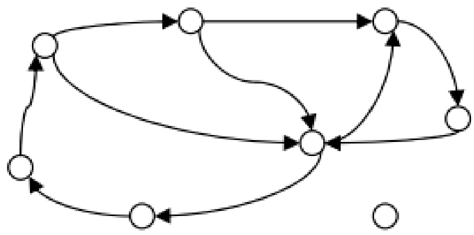
3.



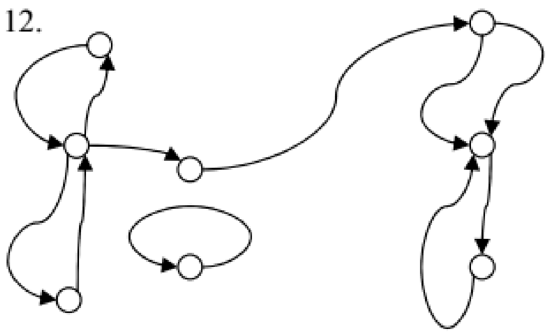
4.



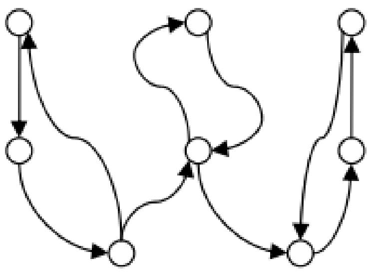
11.



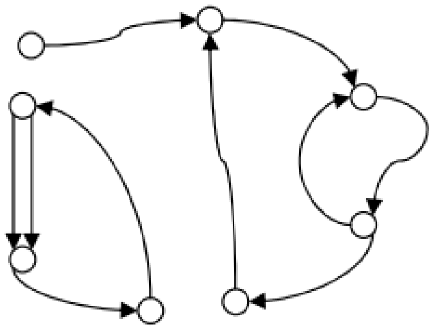
12.



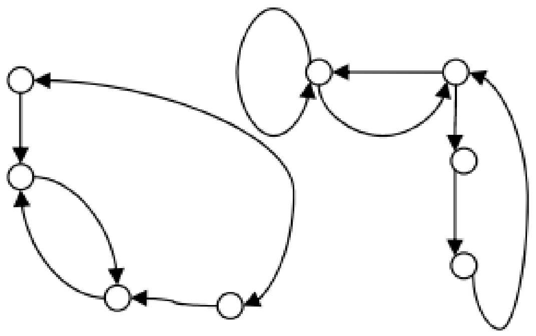
13.



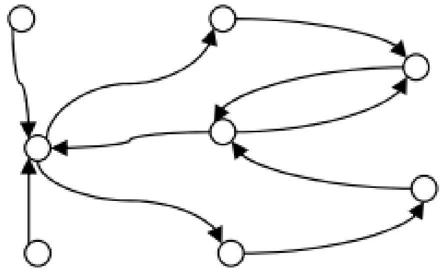
14.



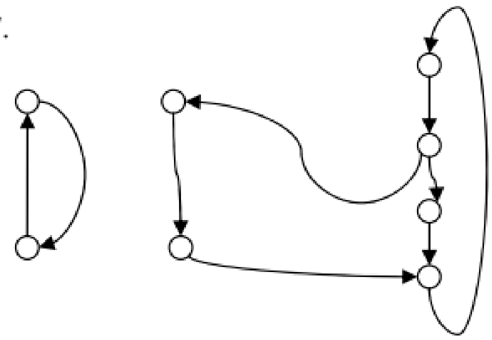
15.

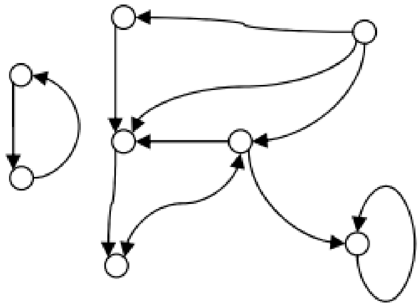


16.

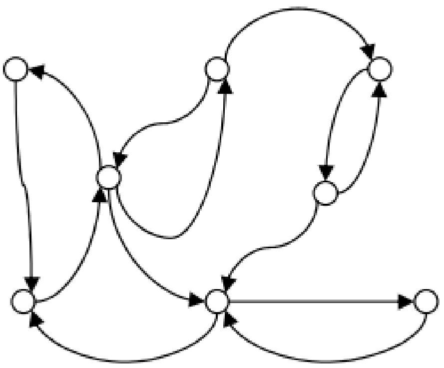


17.

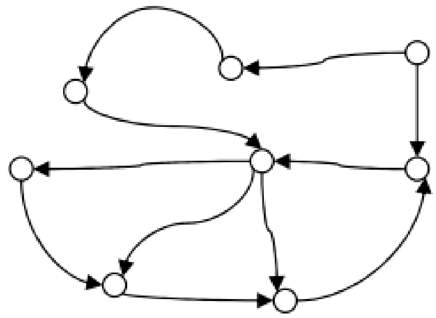




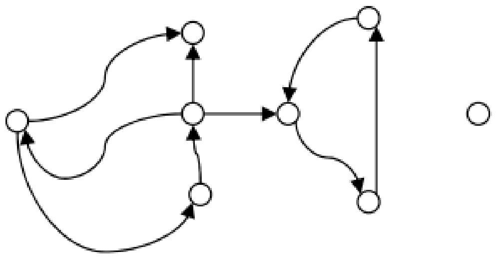
27.



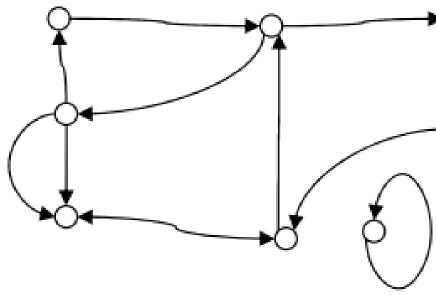
28.



29.



30.



Практичне заняття № 2

Тема: Ейлеровий цикл . Ейлеровий граф. Гамільтоновий цикл.

Мета: Навчити студенті знаходити у графах ейлерів та гамільтонів цикли.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Що таке ейлеровий цикл, шлях? Умови їх існування.
2. Що таке гамільтоновий цикл, шлях? Умови їх існування.

Теоретичні відомості

Ейлеровим циклом називається цикл, що містить всі ребра графа.

Ейлеровим графом називається граф, що містить ейлерів цикл.

Теорема Ейлера. Кінцевий неорієнтований граф ейлеровий тоді і тільки тоді, коли він зв'язний, і степінь всіх його вершин парна (при підрахунку степеня вершини, будь-яку інцидентну їй петлю вважати двічі).

Шлях, що включає кожне ребро графа G тільки один раз, називається *ейлеровим шляхом*. У тому випадку, якщо ейлеров шлях не є ейлеровим циклом, він називається *власним ейлеровим шляхом*.

Теорема Граф G має власний ейлеровий шлях тоді і тільки тоді, коли він зв'язний і рівно дві його вершини мають непарний степінь.

Шляхом Гамільтона (або *гамільтоновим ланцюгом*) називається простий ланцюг, що проходить через всі вершини графа, з початком і кінцем у різних вершинах $v', v'' \in G$.

Цикл Гамільтона - це простий цикл, що проходить через всі вершини графа.

Теорема Для будь-якої вершини із циклу Гамільтона існує рівно два ребра із цього циклу, інцидентні даній вершині.

Граф, що має цикл Гамільтона, називається *гамільтонів*.

Будь-який граф, що має вершину степені 1, не є гамільтонів. Для того, щоб граф мав цикл Гамільтона, необхідно, щоб він був зв'язним.

<u>Теорема</u> Якщо G - зв'язний граф з n вершинами $(n \geq 3)$ і для кожної пари несуміжних вершин $v', v'' \in V$, сума степенів вершин задовольняє умові $\deg v' + \deg v'' \geq n$, тоді граф G має цикл Гамільтона.

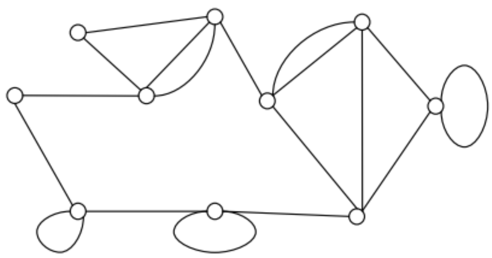
<u>Наслідок</u>. Якщо G - зв'язний граф з n вершинами $(n \geq 3)$ і для кожної вершини $v \in V$ виконується умова $\deg v \geq \dfrac{n}{2}$, то граф G має цикл Гамільтона.

Індивідуальні завдання

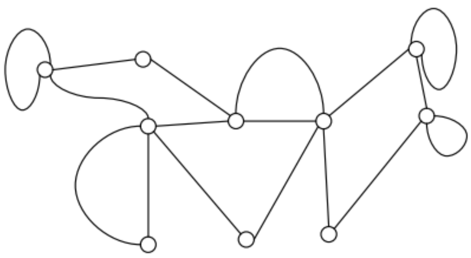
Визначити чи має гра ейлеровий та гамільтоновий цикли (шляхи)

Завдання 1.

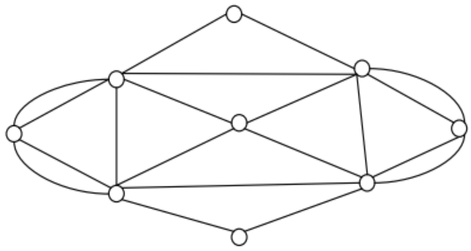
1.



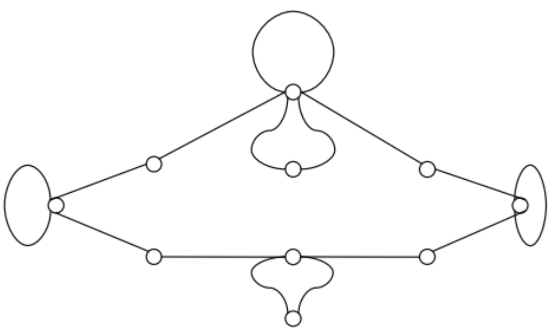
3.



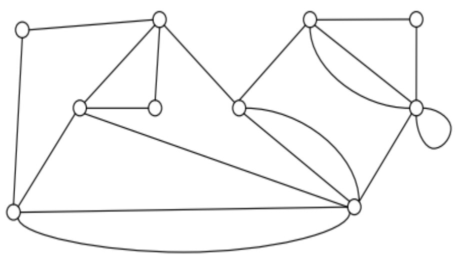
4.



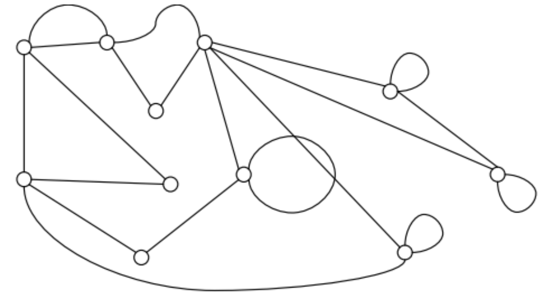
5.



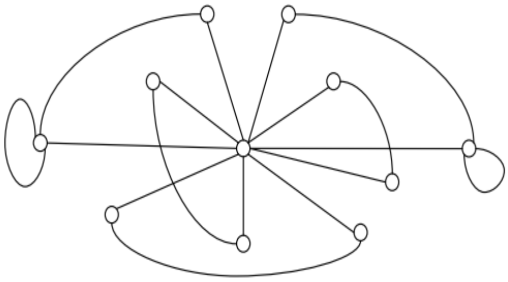
6.



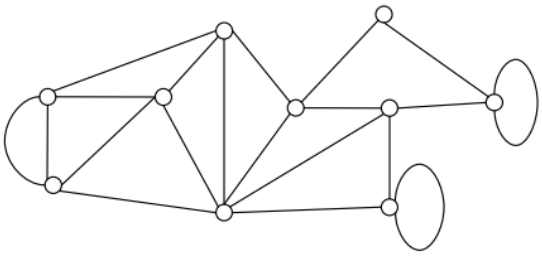
7.



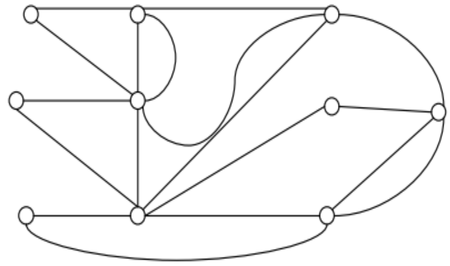
8.



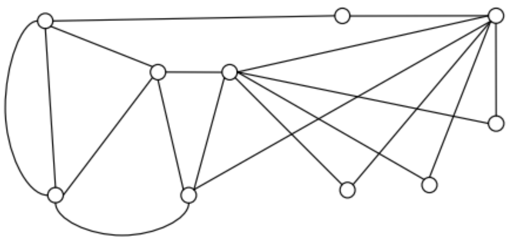
9.



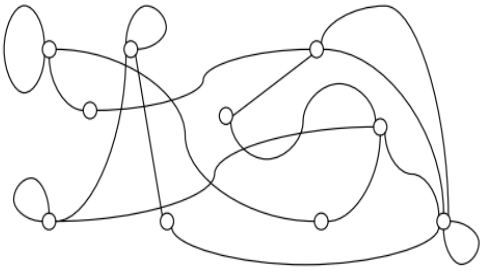
10.



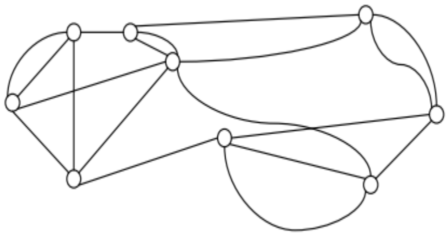
11.



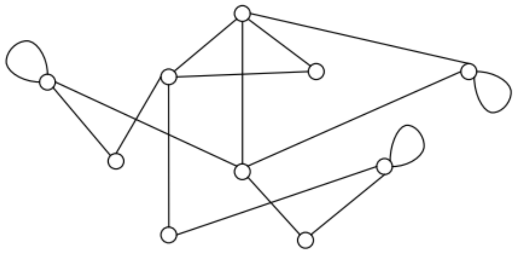
12.



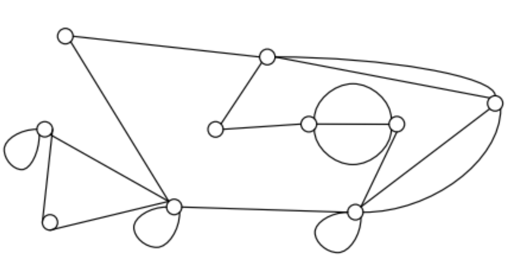
13.



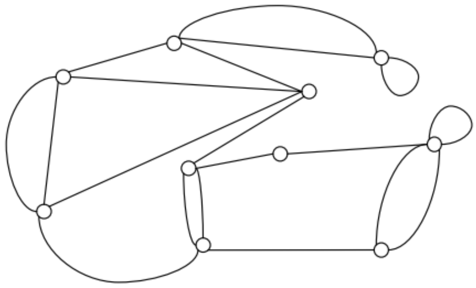
14.



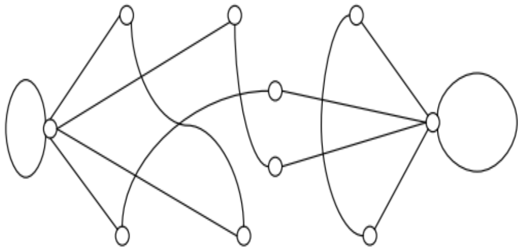
15.



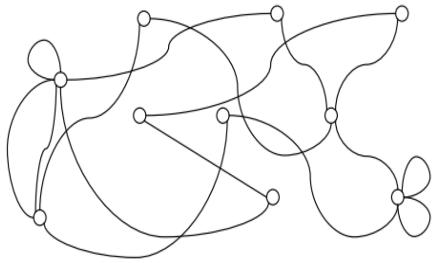
16.



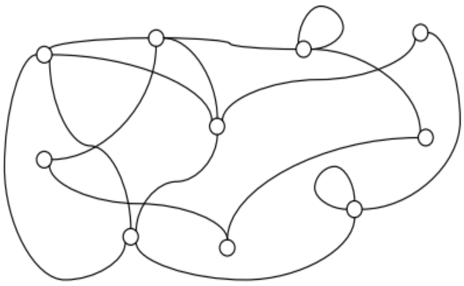
17.



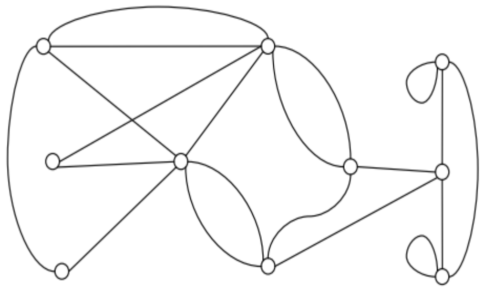
18.



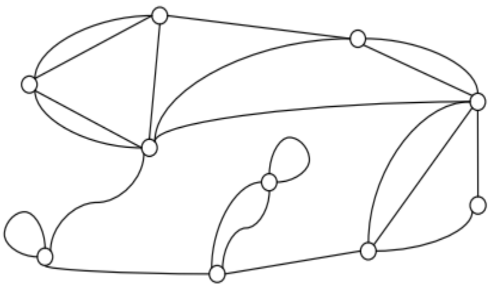
19.



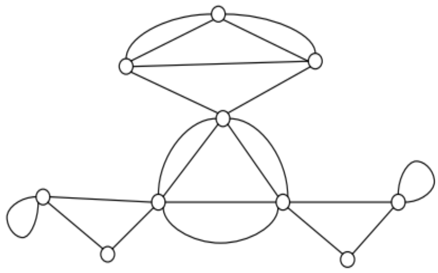
20.



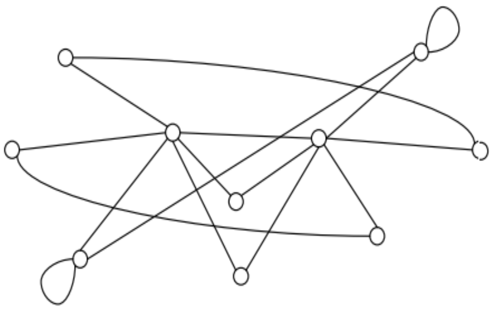
21.



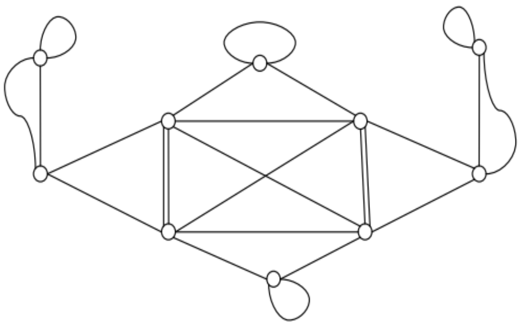
22.



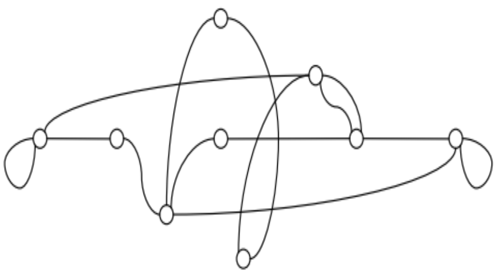
23.



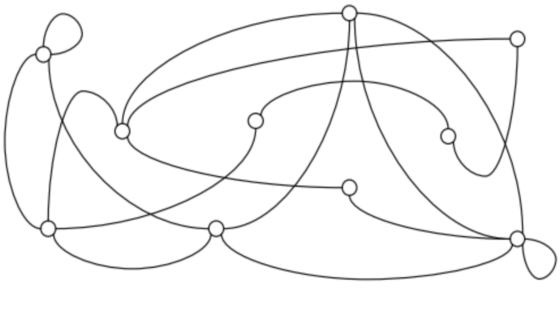
24.



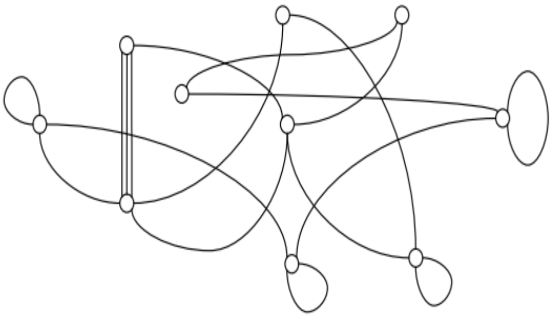
25.



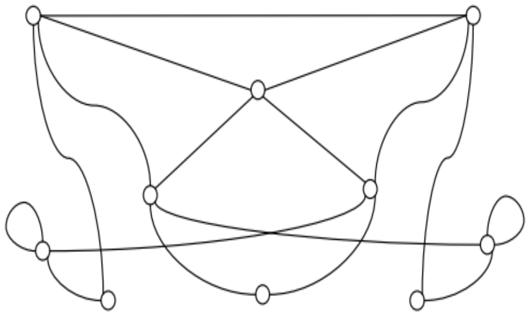
26.



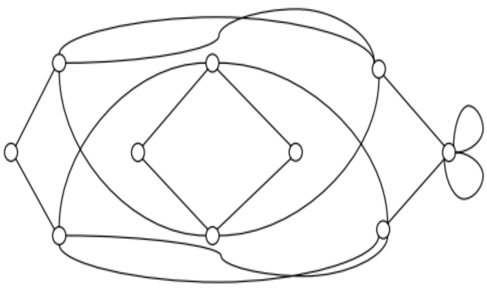
27.



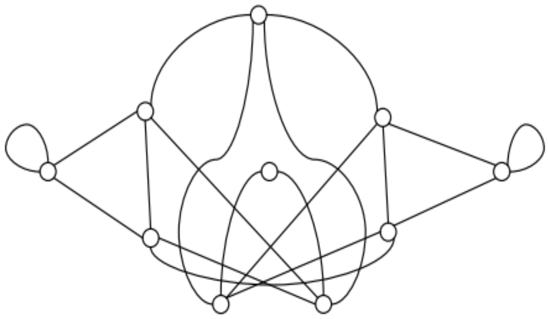
28.



29.



30.



Практичне заняття № 3

Тема: Обхід графа. Пошук найкоротшого шляху між вершинами.

Мета: Навчити студентів здійснювати обхід графа вглиб і вшир, використовувати алгоритм Дейкстри для знаходження найкоротшого шляху між вершинами.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Алгоритм обходу графу вглиб і вшир.
2. Алгоритм Дейкстри.

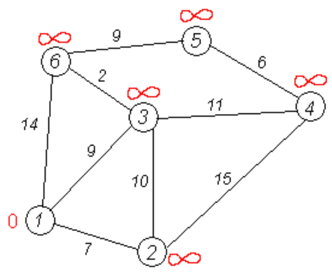
Теоретичні відомості

Алгоритм Дейкстри

Перед початком виконання алгоритму усі вершини графа не позначені. У ході алгоритму кожній вершині присвоюється число, яке дорівнює найкоротшому шляху до даної вершини. Розглянемо виконання алгоритму на прикладі. Хай потрібно знайти відстані від 1-ої вершини до всіх інших. Кружечками позначені вершини, лініями — шляхи між ними («дуги»). Над дугами позначена їх «ціна» — довжина шляху. Надписом над кружечком позначена поточна найкоротша відстань до вершини.

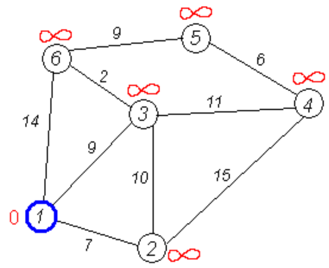
Крок 1

Відстань до всіх вершин графа $V = \infty$. Відстань до a = 0. Жодна вершина графа ще не опрацьована.

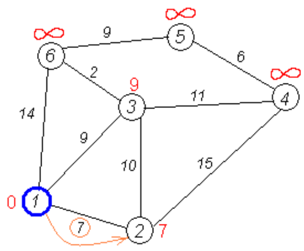


Крок 2

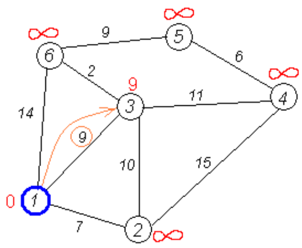
Знаходимо таку вершину (із ще не оброблених), поточна найкоротша відстань до якої мінімальна. В нашому випадку це вершина 1. Обходимо всіх її сусідів і, якщо шлях в сусідню вершину через 1 менший за поточний мінімальний шлях в цю сусідню вершину, то запам'ятовуємо цей новий, коротший шлях як поточний найкоротший шлях до сусіда.

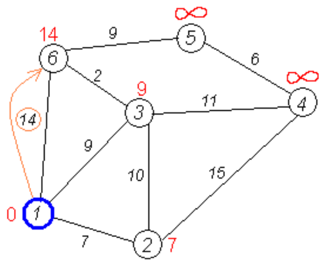


Крок 3. Перший по порядку сусід 1-ї вершини — 2-а вершина. Шлях до неї через 1-у вершину дорівнює найкоротшій відстані до 1-ї вершини + довжина дуги між 1-ю та 2-ю вершиною, тобто 0 + 7 = 7. Це менше поточного найкоротшого шляху до 2-ї вершини, тому найкоротший шлях до 2-ї вершини дорівнює 7.

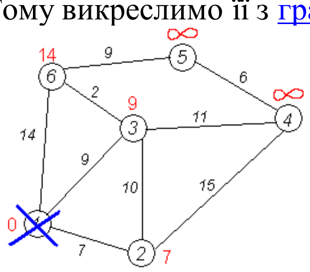


Кроки 4, 5. Аналогічну операцію проробляєм з двома іншими сусідами 1-ї вершини — 3-ю та 6-ю.

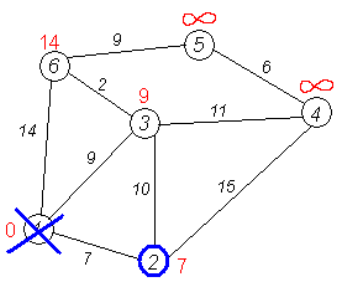


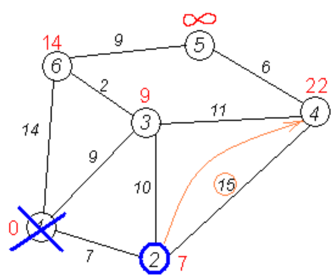


Крок 6. Всі сусіди вершини 1 перевірені. Поточна мінімальна відстань до вершини 1 вважається остаточною і обговоренню не підлягає (те, що це дійсно так, вперше довів [Дейкстра](#)). Тому викреслимо її з [графа](#), щоб відмітити цей факт.



Крок 7. Практично відбувається повернення до кроку 2. Знову знаходимо «найближчу» необроблену (невикреслену) вершину. Це вершина 2 з поточною найкоротшою відстанню до неї = 7.





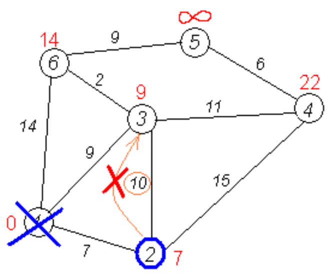
Знову намагаємося зменшити відстань до всіх сусідів 2-ої вершини, намагаючись пройти в них через 2-у. Сусідами 2-ої вершини є 1, 3, 4.

Крок 8. Перший (по порядку) сусід вершини 2 — 1-а вершина. Але вона вже оброблена (або викреслена — див. крок 6). Тому з 1-ою вершиною нічого не робимо.

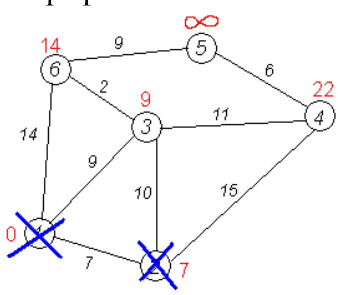
Крок 8 (з іншими вхідними данними)

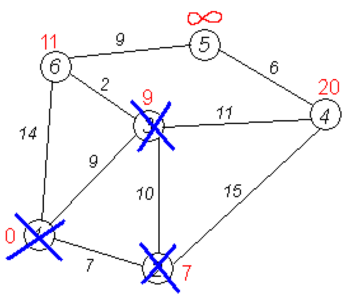
Інший сусід вершини 2 — вершина 4. Якщо йти в неї через 2-у, то шлях буде = найкоротша відстань до 2-ої + відстань між 2-ою і 4-ою вершинами = 7 + 15 = 22. Оскільки 22 < ∞, встановлюємо відстань до вершини № 4 рівним 22.

Крок 9. Ще один сусід вершини 2 — вершина 3. Якщо йти в неї через 2-у, то шлях буде = 7 + 10 = 17. Але 17 більше за відстань, що вже запам'ятали раніше до вершини № 3 і дорівнює 9, тому поточну відстань до 3-ої вершини не міняємо.



Крок 10. Всі сусіди вершини 2 переглянуті, заморожуємо відстань до неї і викреслюємо її з графа



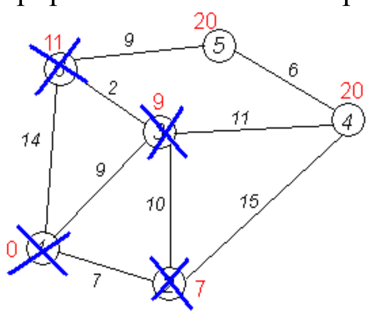


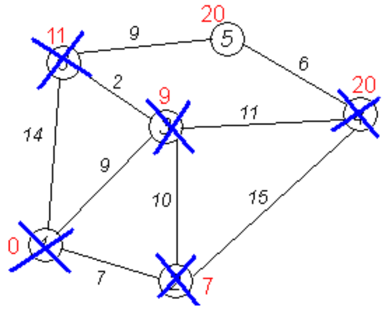
Кроки 11 — 15

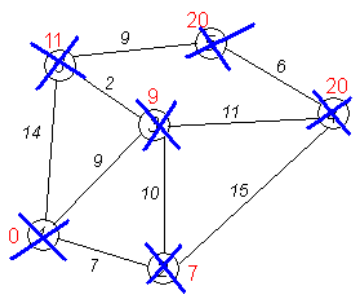
По вже «відпрацьованій» схемі повторюємо кроки 2 — 6. Тепер «найближчою» виявляється вершина № 3. Після її «обробки» отримаємо такі результати:

Наступні кроки

Проробляємо те саме з вершинами, що залишилися.







Завершення виконання алгоритму.

Алгоритм закінчує роботу, коли викреслені всі вершини. Результат його роботи видно на останньому малюнку: найкоротший шлях від 1-ої вершини до 2-ої становить 7, до 3-ої — 9, до 4-ої — 20, до 5-ої — 20, до 6-ої — 11 умовних одиниць.

Індивідуальні завдання

Визначити мінімальний шлях із вершини s орієнтованого графа G до вершини f використовуючи алгоритм. Граф G заданий списком дуг з відповідними їм вагами:

1. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)} ;
2. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)} ;
3. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)} ;
4. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)} ;
5. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)} ;
6. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)} ;
7. {(s, a; 2); (s, b; 1); (a, c; 4); (b, a; 3); (b, c; 2); (b, d; 8); (c, d; 5); (c, f; 9); (d, f; 3)} ;
8. {(s, a; 1); (s, b; 4); (a, b; 3); (a, c; 6); (a, d; 2); (b, d; 1); (c, d; 7); (c, f; 3); (d, f; 9)} ;
9. {(s, a; 5); (s, b; 2); (a, c; 4); (b, a; 1); (b, c; 7); (b, d; 9); (c, d; 3); (c, f; 8); (d, f; 4)} ;
10. {(s, a; 2); (s, b; 7); (a, b; 6); (a, c; 3); (a, d; 4); (b, d; 2); (c, d; 5); (c, f; 3); (d, f; 5)} ;
11. {(s, a; 4); (s, b; 2); (a, c; 7); (b, a; 1); (b, c; 5); (b, d; 2); (c, d; 3); (c, f; 7); (d, f; 2)} ;
12. {(s, a; 5); (s, b; 3); (a, b; 2); (a, c; 1); (a, d; 2); (b, d; 7); (c, d; 6); (c, f; 2); (d, f; 8)} ;
13. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)} ;
14. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)} ;
15. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)} ;
16. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)} ;
17. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)} ;
18. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)} ;
19. {(s, a; 2); (s, b; 1); (a, c; 4); (b, a; 3); (b, c; 2); (b, d; 8); (c, d; 5); (c, f; 9); (d, f; 3)} ;
20. {(s, a; 1); (s, b; 4); (a, b; 3); (a, c; 6); (a, d; 2); (b, d; 1); (c, d; 7); (c, f; 3); (d, f; 9)} ;
21. {(s, a; 5); (s, b; 2); (a, c; 4); (b, a; 1); (b, c; 7); (b, d; 9); (c, d; 3); (c, f; 8); (d, f; 4)} ;
22. {(s, a; 2); (s, b; 7); (a, b; 6); (a, c; 3); (a, d; 4); (b, d; 2); (c, d; 5); (c, f; 3); (d, f; 5)} ;
23. {(s, a; 4); (s, b; 2); (a, c; 7); (b, a; 1); (b, c; 5); (b, d; 2); (c, d; 3); (c, f; 7); (d, f; 2)} ;
24. {(s, a; 5); (s, b; 3); (a, b; 2); (a, c; 1); (a, d; 2); (b, d; 7); (c, d; 6); (c, f; 2); (d, f; 8)} .
25. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)} ;
26. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)} ;
27. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)} ;
28. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)} ;
29. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)} ;
30. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)} .

Практичне заняття № 4

Тема: Обхід дерева. Префіксна та постфіксна форма запису виразу.

Мета: Навчити студентів записувати вирази у вигляді дерева, здійснювати обхід дерева.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Який граф називається деревом?
2. Які види обходу дерева ви знаєте?
3. Що таке постфіксна і префіксна форма запису виразу?

Теоретичні відомості

Дерево - це нелінійна структура даних, що використовується для представлення ієрархічних зв'язків, що мають відношення «один до багатьох».

Самий верхній вузол дерева називається коренем. Верхній вузол для нижнього вузла називається предком, а нижній вузол для верхнього - нащадком. Вершини (вузли), що не мають нащадків, називаються листками дерева. Вершини, що мають нащадків, називаються внутрішніми. Дві вершини дерева з'єднуються гілкою. Кількість гілок від кореня до вершини є довжиною шляху до цієї вершини.

Довжина шляху від кореня до будь-якої вершини називається глибиною цієї вершини. Максимальна глибина вершин дерева називається висотою дерева.

Кількість безпосередніх нащадків у вершини (вузла) дерева називається степенем вершини (вузла). Максимальний степінь всіх вершин є степенем дерева.

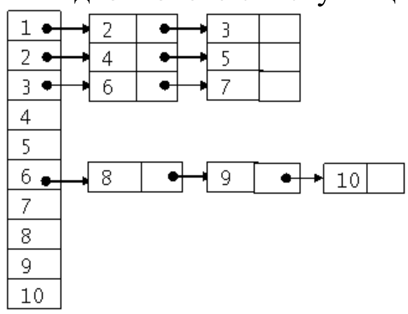
Кожному вузлу дерева можна співставити ім'я вузла і значення вузла, тобто дані, які зберігаються в цьому вузлі. Причому, якщо значенням є різнорідні дані (записи або об'єднання), то значенням вузла можна вважати значення одного з полів цих даних, яке називається ключем.

У пам'яті дерева можна подати у вигляді зв'язків з предками (батьками); зв'язного списку нащадків (дітей) або структури даних.

Подання дерева у вигляді зв'язків з предками:

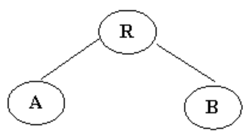
№ вершина	1	2	3	4	5	6	7	8	9	10
Батько	0	1	1	2	2	3	3	6	6	6

Приклад подання дерева у вигляді зв'язного списку нащадків :



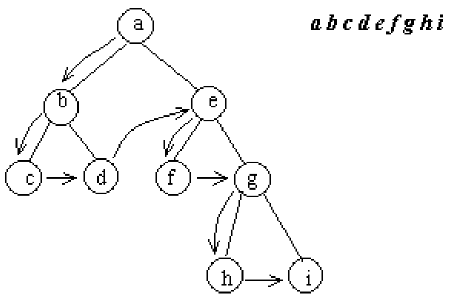
Якщо у кожної вершини дерева є не більше двох нащадків (ліві і праві піддерева), то таке дерево називається двійковим або бінарним.
Двійкові дерева широко використовуються у програмуванні.
Основні операції з деревами: обхід дерева, пошук по дереву, включення в дерево, виключення з дерева.
Обхід (відвідування) вершин дерева можна здійснити наступним чином:

- Зліва направо: A, R, B (інфіксний обхід, симетричний обхід);
- Зверху вниз: R, A, B (префіксній обхід, обхід в прямому порядку);
- Знизу вверх: A, B, R (постфіксний обхід, обхід в зворотному порядку).



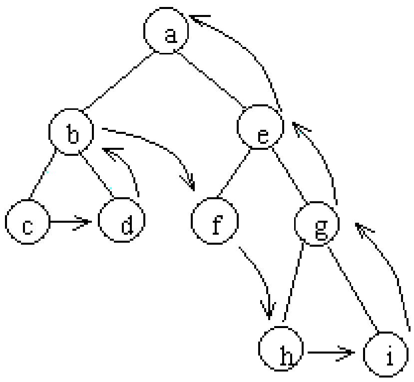
Прямий порядок проходження бінарного дерева можна визначити таким чином:

- потрапити в корінь;
- пройти в прямому порядку ліве піддерево;
- пройти в прямому порядку праве піддерево.



Проходження бінарного дерева в зворотному порядку можна визначити в аналогічній формі:

- пройти в зворотному порядку ліве піддерево;
- пройти в зворотному порядку праве піддерево;
- потрапити в корінь.



c d b f h i g e a

Індивідуальні завдання

Обчислити значення виразу в польському та зворотному польському записі:

1. * + – 9 4 7 / ↑ 2 5 8 ; 4292*^/528-+
2. + * – 8 6 9 / ↑ 6 1 3 ; 3748/+*523^*-
3. – + * 5 2 8 / ↑ 3 2 9 ; 3724*+*5424^/-
4. * + – 6 2 4 / ↑ 2 4 8 ; 5221*^/528-+
5. + * – 7 5 4 / ↑ 6 2 4 ; 542/+*523^*-
6. – * + 2 3 5 / ↑ 4 2 8 ; 2431*^/528-+
7. * + – 5 9 7 / ↑ 8 1 4 ; 2742/+*523^*-
8. + * – 4 9 2 / ↑ 9 1 3 ; 3222*^/528-+
9. – + * 3 5 2 / ↑ 2 3 8 ; 4763/+*523^*-
10. + – * 5 1 4 / ↑ 8 2 4 ; 5231*^/528-+
11. * + – 8 3 1 / ↑ 4 3 8 ; 7284/+*523^*-
12. – + * 7 6 9 / ↑ 2 5 4 ; 3214*^/528-+
13. + * – 7 2 8 / ↑ 6 1 2 ; 5284/+*523^*-
14. + – * 9 7 6 / ↑ 8 1 2 ; 6521*^/528-+
15. – + * 8 2 7 / ↑ 6 2 9 ; 8542/+*523^*-

3. Записати вираз у вигляді дерева та здійснити його обхід:
4.

1. $\frac{6\sin a}{c^3}+(a-b)tgc$

2. $\sqrt{a^2-b}-\frac{b+c}{5\cos b}$

3. $\frac{a}{b-tga}+5\left(c-ab^5\right)$

4. $5^{a+b}-3-\frac{a-c}{8+tgb}$

5. $a\left(b^2+c\right)-\frac{3^a}{\sqrt{b+c}}$

6. $a^4tgb-\frac{3b}{(c-a)^3}$

7. $\frac{6a-b^2}{c}+(a-b\cdot c)^6$

8. $\left(\frac{a}{c}-bca\right)^3-\sqrt{c-ba}$

9. $\sqrt{ab-\frac{c}{c-a}}+\frac{a-b+c}{abc}$

10. $x_1\oplus x_2\to\left(x_1\wedge\bar{x}_2\right)\vee\left(\bar{x}_1\wedge x_2\right)$

11. $\left(x_2\wedge\left(x_3\vee x_2\right)\right)\oplus\left(x_2\to x_1\right)$

12. $(\left(x_2\uparrow x_1\right)\wedge\left(x_3\vee x_2\right))\oplus\left(x_2\to x_1\vee x_3\right)$

13. $a\left(b\ +2c\right)-\frac{2^a}{\sqrt{b-c^2}}$

14. $2(\sin a+tgb)-\frac{3b}{(c-a)^3}$

15. $6\sqrt{a\ -b^3}-\frac{b+c}{5-\cos b}$

16. $\frac{6\sin a}{c^3-4}+(a-3b)tgc$

Практичне заняття № 5

Тема: Каркаси. Мінімальний каркас.

Мета:Ознайомити студентів з поняттям каркасу, навчити використовувати алгоритм Краскала для побудови мінімального каркаса.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Що таке каркас? Як його можна побудувати?

2. Який каркас називається мінімальним?

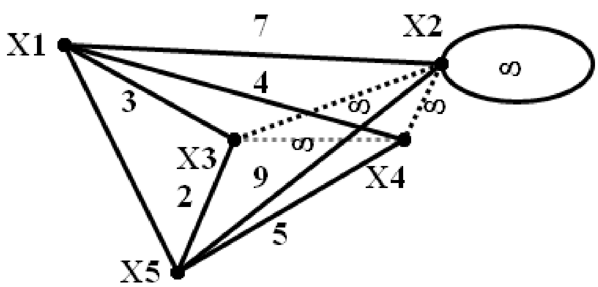
3. Суть алгоритму Краскала.

Теоретичні відомості

Метод Краскала використовується для побудови мінімального каркаса (каркаса найменшої ваги). Ребра графа записуються в порядку зростання їхньої довжини. Алгоритм

1. Створити список ребер R, що містить довжину ребра, номери вхідної (i), та вихідної вершини ребра (j), ознаку включення ребра в дерево.
2. Переглядається список R і вибирається з нього ребро, ще не включене в каркас, яке не утворює цикл з уже включеними в каркас ребрами.

Приклад. Побудувати мінімальний каркас графа.



Ребра записані в порядку зростання ваги

N	1	2	3	4	5	6	7
ребро	(x_3, x_5)	(x_1, x_3)	(x_1, x_4)	(x_1, x_5)	(x_4, x_5)	(x_1, x_2)	(x_2, x_5)
довжина	2	3	4	4	5	7	9

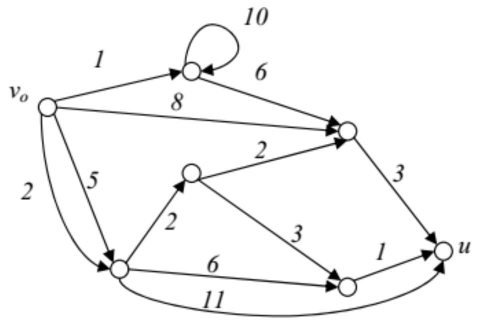
Розвязування задачі:

N кроку	Список вершин	ребро		Вага ребер
1	$\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\}$	(x_3, x_5)	так	2
2	$\{x_1\},\{x_2\},\{x_3, x_5\},\{x_4\}$	(x_1, x_3)	так	5
3	$\{x_1, x_3, x_5\},\{x_2\},\{x_4\}$	(x_1, x_4)	так	9
4	$\{x_1, x_3, x_4, x_5\},\{x_2\}$	(x_1, x_5)	ні	9
5	$\{x_1, x_3, x_4, x_5\},\{x_2\}$	(x_4, x_5)	ні	9
6	$\{x_1, x_3, x_4, x_5\},\{x_2\}$	(x_1, x_2)	так	16
	$\{x_1, x_2, x_3, x_4, x_5\}$			16

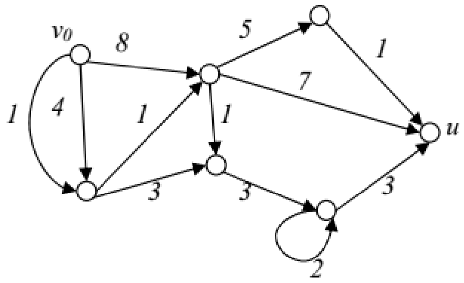
Індивідуальні завдання

Побудувати мінімальний каркас графа:

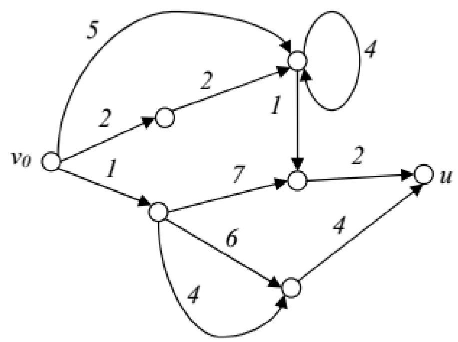
1.



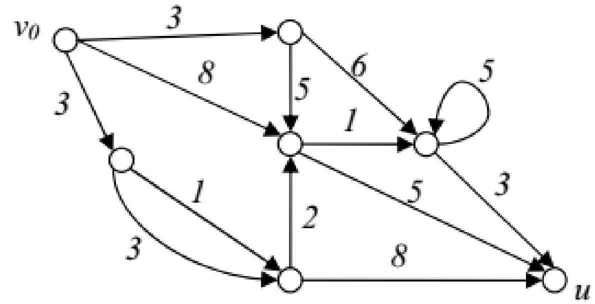
2.



3.

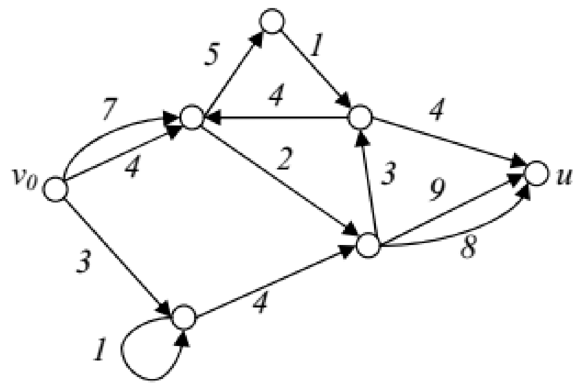


4.



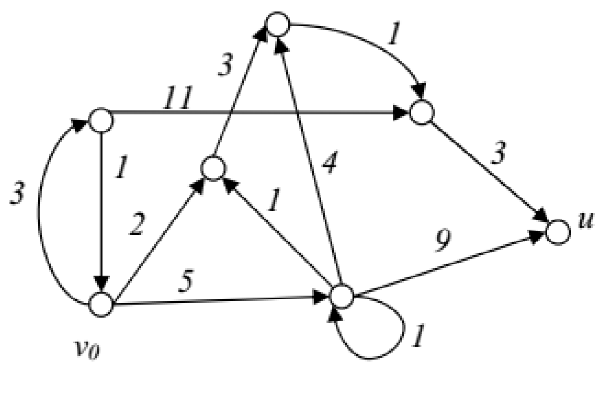
v_0
3
8
3
6
5
5
1
1
5
3
3
2
8
u

5.



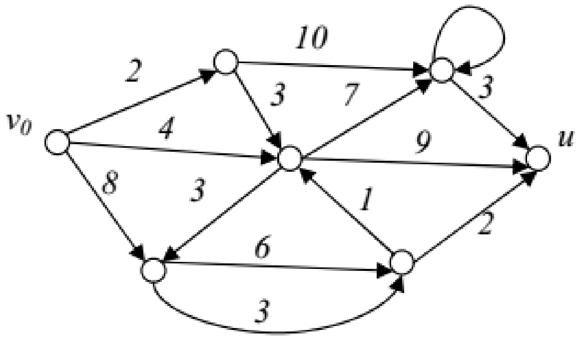
5
1
7
4
4
v_0
4
2
3
9
u
3
8
4
1

6.



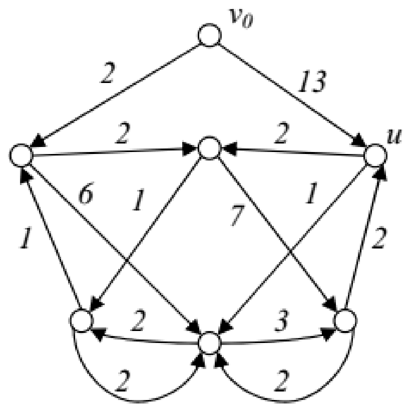
1
3
11
3
1
4
3
u
2
1
9
5
1
v_0

7.

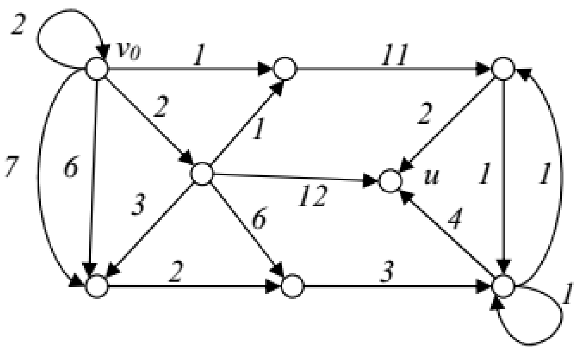


10
2
3
7
3
v_0
4
9
u
8
3
1
2
6
3

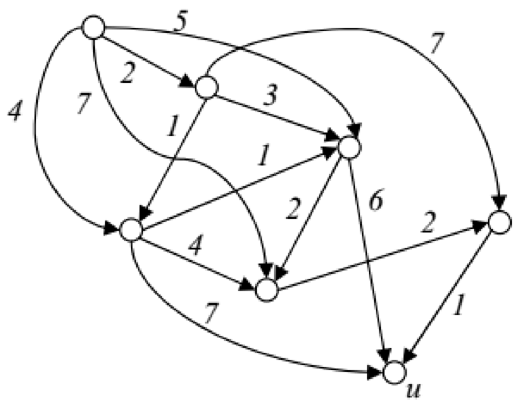
19.


v_0
2
13
2
2
u
6
1
1
7
1
2
2
3
2
2

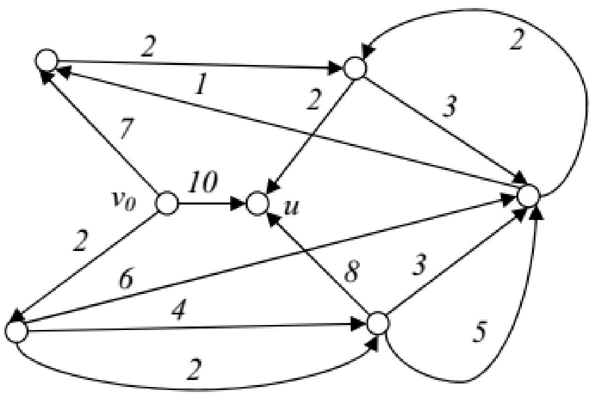
20.


2
v_0
1
11
2
2
1
7
6
u
1
1
12
3
6
4
2
3
1

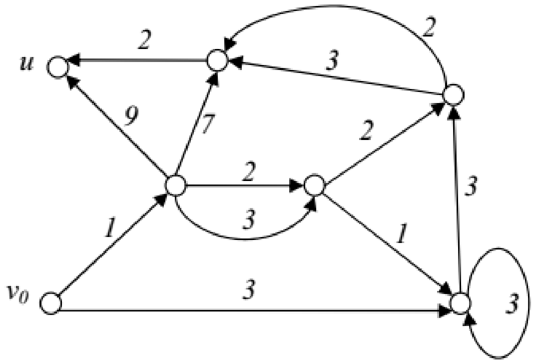
21.


5
7
2
4
7
3
1
1
6
2
2
4
1
7
u

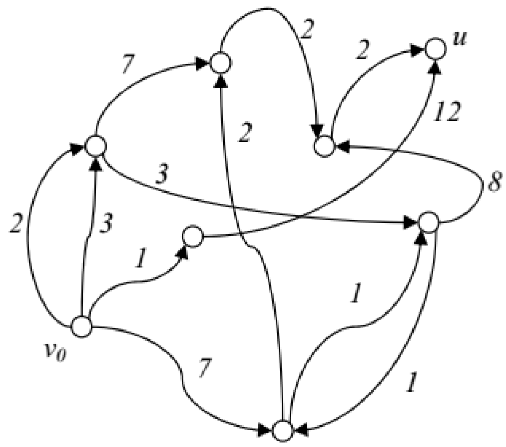
22.


2
2
1
2
3
7
10
v_0
u
2
8
3
6
4
5
2

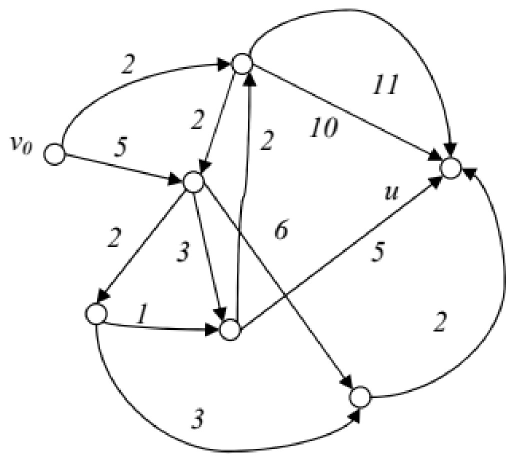
27.


u
2
2
3
9
7
2
2
3
3
1
1
v_0
3
3

28.


2
u
7
2
12
2
3
8
2
3
1
1
v_0
7
1

29.


2
11
2
10
v_0
5
2
u
2
6
3
5
1
2
3

Практичнезаняття № 6

Тема: Транспортні мережі. Задача про максимальний потік. Алгоритм Форда-Фалкерсона розв'язування задачі про максимальний потік.

Мета: Ознайомити студентів з поняттям транспортної мережі, оволодіти сутністю методу розв'язання задачі про максимальний потік у мережі.

Література

1. Бардачов Ю.М., Соколова Н.А. Дискретна математика. – К. : Вища школа, 2013.
2. Новиков Н.А. Дискретная математика для программистов. – СПб. : Питер, 2012.
3. Ядренко М.Й. Дискретна математика. - К. : КНУ, 2008.

Контрольні питання

1. Сформулювати задачу про максимальний потік.
2. Записати математичну модель задачі про максимальний потік.

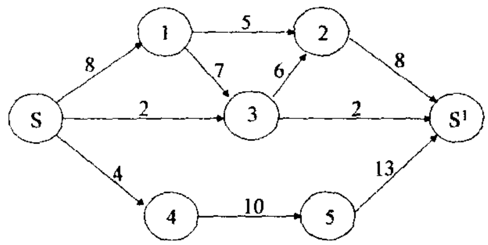
Теоретичні відомості

Задача про максимальний потік в мережі формулюється так: при заданій топології мережі і відомій пропускній спроможності дуг знайти найбільшу величину потоку і його розподіл по дугам мережі.

Алгоритм визначення максимального потоку

1. Побудувати початковий припустимий план-потік. Рішення можна начати, наприклад, з нульового потоку.
2. Перевіряємо, чи нема ненасичених шляхів, що ведуть з S в S'. Якщо вершина S' не попала в множину вершин S, досяжних по ненасиченим ребрам з S, то побудований поток максимальний. Задача вирішена. Якщо вершина S' попала в множину S, то потік можна збільшити.
3. Відокремлюємо шлях, складений з ненасичених ребер, ведучий з S в S'. Збільшуємо потік цього шляху на величину Δ_1. Переходимо до нового потоку, збільшенного на Δ_1, і т.д.

   Ідею алгоритму можно реалізувати табличним або графічним способом. Розглянемо графічний спосіб розв'язання задачі. Нехай задано мережу



1. Побудуємо початковий припустимий потік X_o, для цього розглянемо, наприклад, шляхи

"1" S-1-3-2-S' min{10,7,6,8}=6

"2" S-4-5-S' min{4,10,13}=4

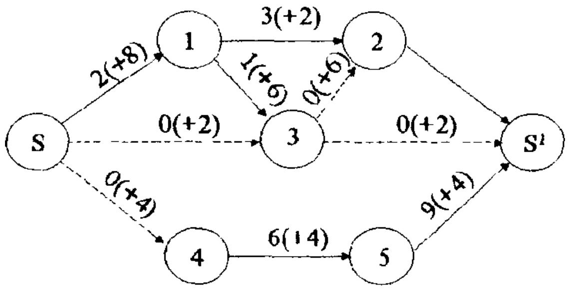
По шляху "1" пропустили min{10,7,6,8}=6

По шляху "2" пропустили min{4,10,13}=4

Отримуємо потік X_o величиною v_0=10 одиниць.

Мережа, відповідна потоку X_o, зображена на рисунку. При цьому числа над ребрами означають: без дужок – залишена пропускна можливість, в дужках – величина пропускаючого груза, тобто величина потоку через ребро.

Пунктиром зображено ненасичені ребра, тобто ребра, потік через які дорівнює її пропускній можливості.



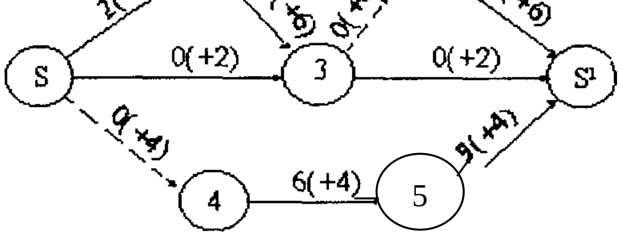
Потік в мережі потужності V_0=10

2. Перевіряємо, чи нема ненасичених шляхів. Такі шляхи є тому, що початковий поток можно збільшити. Для цього вибираємо шляхи "3", "4".

"3" S=>1-2=>S' min{4,5,2}=2

"4" S=>3=>S' min{2,2}=2

По шляху "3" пропускаємо потік величиною 2 одиниці, а по шляху "4" - також 2 одиниці. Отримуємо потік в мережі величиною v_1=14 одиниць, йому відповідає мережа на рисунку



Потік в мережі потужності V_1=14 - максимальний потік

Так як більше ненасичених шляхів з вершин S в S' немає, то отриманий потік є максимальним і задача розв'язана.

Необхідно також перевірити виконання вимог до однорідного потоку в мережі (згідно математичній моделі).

Індивідуальні завдання

Визначити мінімальний потік із вершини s орієнтованого графа G до вершини f. Граф G заданий списком дуг з відповідними їм вагами:

1. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)}
2. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)}
3. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)}
4. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)}
5. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)}
6. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)}
7. {(s, a; 2); (s, b; 1); (a, c; 4); (b, a; 3); (b, c; 2); (b, d; 8); (c, d; 5); (c, f; 9); (d, f; 3)}
8. {(s, a; 1); (s, b; 4); (a, b; 3); (a, c; 6); (a, d; 2); (b, d; 1); (c, d; 7); (c, f; 3); (d, f; 9)}
9. {(s, a; 5); (s, b; 2); (a, c; 4); (b, a; 1); (b, c; 7); (b, d; 9); (c, d; 3); (c, f; 8); (d, f; 4)}
10. {(s, a; 2); (s, b; 7); (a, b; 6); (a, c; 3); (a, d; 4); (b, d; 2); (c, d; 5); (c, f; 3); (d, f; 5)} ;
11. {(s, a; 4); (s, b; 2); (a, c; 7); (b, a; 1); (b, c; 5); (b, d; 2); (c, d; 3); (c, f; 7); (d, f; 2)} ;
12. {(s, a; 5); (s, b; 3); (a, b; 2); (a, c; 1); (a, d; 2); (b, d; 7); (c, d; 6); (c, f; 2); (d, f; 8)} ;
13. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)} ;
14. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)} ;
15. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)} ;
16. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)} ;

15\. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)} ;

16\. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)} ;

17. {(s, a; 2); (s, b; 1); (a, c; 4); (b, a; 3); (b, c; 2); (b, d; 8); (c, d; 5); (c, f; 9); (d, f; 3)} ;

18. {(s, a; 1); (s, b; 4); (a, b; 3); (a, c; 6); (a, d; 2); (b, d; 1); (c, d; 7); (c, f; 3); (d, f; 9)} ;

19. {(s, a; 5); (s, b; 2); (a, c; 4); (b, a; 1); (b, c; 7); (b, d; 9); (c, d; 3); (c, f; 8); (d, f; 4)} ;

20. {(s, a; 2); (s, b; 7); (a, b; 6); (a, c; 3); (a, d; 4); (b, d; 2); (c, d; 5); (c, f; 3); (d, f; 5)} ;

21. {(s, a; 4); (s, b; 2); (a, c; 7); (b, a; 1); (b, c; 5); (b, d; 2); (c, d; 3); (c, f; 7); (d, f; 2)} ;

22. {(s, a; 5); (s, b; 3); (a, b; 2); (a, c; 1); (a, d; 2); (b, d; 7); (c, d; 6); (c, f; 2); (d, f; 8)} .

23. {(s, a; 3); (s, b; 2); (a, c; 5); (b, a; 2); (b, c; 7); (b, d; 6); (c, d; 2); (c, f; 5); (d, f; 1)} ;

24. {(s, a; 2); (s, b; 5); (a, b; 1); (a, c; 3); (a, d; 8); (b, d; 2); (c, d; 4); (c, f; 1); (d, f; 2)} ;

25. {(s, a; 4); (s, b; 1); (a, c; 8); (b, a; 2); (b, c; 3); (b, d; 5); (c, d; 2); (c, f; 9); (d, f; 4)} ;

26. {(s, a; 3); (s, b; 1); (a, b; 5); (a, c; 2); (a, d; 9); (b, d; 4); (c, d; 3); (c, f; 8); (d, f; 3)} ;

27. {(s, a; 1); (s, b; 5); (a, c; 3); (b, a; 7); (b, c; 2); (b, d; 9); (c, d; 4); (c, f; 6); (d, f; 2)} ;

28. {(s, a; 3); (s, b; 4); (a, b; 2); (a, c; 5); (a, d; 1); (b, d; 8); (c, d; 6); (c, f; 2); (d, f; 7)} ;

29. {(s, a; 2); (s, b; 1); (a, c; 4); (b, a; 3); (b, c; 2); (b, d; 8); (c, d; 5); (c, f; 9); (d, f; 3)} ;

30. {(s, a; 1); (s, b; 4); (a, b; 3); (a, c; 6); (a, d; 2); (b, d; 1); (c, d; 7); (c, f; 3); (d, f; 9)} .

Зміст