**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ**

**ДВНЗ «НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»**

**Лабораторна робота №4**

**З дисципліни «Дискретна математика»**

Оптимізація на мережах

Виконав

студент групи 123-17-1

Дік Микита Павлович

м.Дніпро

2018г.

**Мета роботи**: ознайомлення з методами оптимізації мереж

**1. Короткі теоретичні відомості**

**Пошук максимального потоку.**

Нехай S є довільна, частково орієнтована мережа, кожному ребру u якої приписане невід'ємне число c(u) - *пропускна спроможність*. *Потоком* у мережі S називається пара (f, w), де w - деяка орієнтація всіх неорієнтованих ребер мережі, а f(u) - задана на множині всіх ребер функція з невід'ємними значеннями, що не перевершують пропускних спроможностей, і така, що в кожній внутрішній вершині α виконується закон Кірхгофа, відповідно до якого сума значень потоку по ребрах, що входить у вершину, дорівнює сумі потоків по ребрах, що виходить із вершини. Іншими словами, для f(u) виконуються умови:

0 ≤ f(u) ≤ c(u) для усіх вершин мережі;

 R(α) = 0 для усіх внутрішніх вершин, де

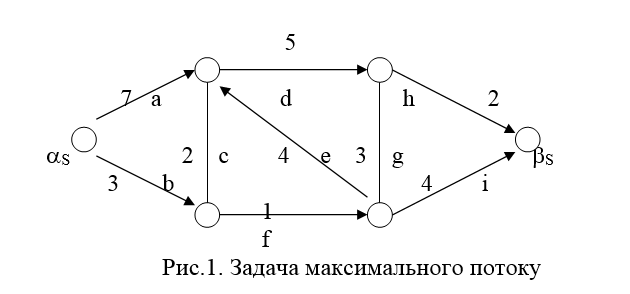
а Γ(α) (відповідно Γ'(α)) - множина всіх ребер, що виходять із α (відповідно вхідних у α) при орієнтації w.

Оскільки сума значень R(α) по усіх вершинах мережі, включаючи полюси, дорівнює нулю (кожне ребро є вихідним для однієї вершини і вхідним для іншої), то R(αs) = - R(βs). Значення R = R(αs) називається *величиною потоку*.

Розглянемо задачу визначення максимального значення Rmax потоку через мережу S при заданих значеннях пропускних спроможностей. Відповідь може бути отримана у термінах перетинів мережі.

*Перетином* мережі називається множина ребер, при видаленні яких мережа стає незв'язною, причому полюса потрапляють у різні компоненти зв'язності. У мережі на рис. 1 прикладами перетинів є {d, e, f}, {b, c, e, g, h}, {d, g, h, i}.

Перетин називається *простим*, якщо при видаленні будь-якого його ребра, він перестає бути перетином. Так, перетини {d, e, f} і {b, c, e, g, h} - прості, а перетин {d, g, h, i} не є таким. Очевидно, що для кожного ребра простого перетину можна зазначити ланцюг, що проходить через це ребро, але не проходить через інші ребра даного перетину.



Якщо у зв'язній мережі віддалиться простий перетин, то мережа розпадеться рівно на дві частини: ліву і праву, що містить αS і βS відповідно. Кожне ребро простого перетину зв'язує вершини з різних частин. Будемо називати ребро перетину *прямим*, якщо воно в мережі не орієнтоване або орієнтоване зліва праворуч, і *оберненим* у противному випадку. Буде орієнтоване ребро прямим або оберненим, залежить від вибору перетину. Так, у прикладі ребро е в перетинах {d, e, f} і {b, c, e, g, h} - обернене, а в перетині {a, c, e, g, i}- пряме.

Кожному простому перетину W припишемо пропускну спроможність c(W), рівну сумі пропускних спроможностей усіх його прямих ребер. У прикладі на рис.2.12 перетин {d, e, f} має пропускну спроможність 5+1=6, а перетин {b, c, e, g, h} - 3+2+3+2=10.

Теорема про максимальну пропускну спроможність мережі сформульована Фордом і Фалкерсоном так: максимальний розмір потоку Rmax через мережу S дорівнює мінімальній пропускній спроможності cmin її простих перетинів. Ця теорема покладена в основу задачі визначення максимальної пропускної спроможності мережі.

Розглянемо *алгоритм Форда - Фалкерсона* для розв'язання цієї задачі.

Крок 0.Нехай джерела позначені, але не переглянуті, а всі інші вузли не позначені.

Крок 1*.* Вибрати довільний позначений, але не переглянутий вузол i.

Крок 2*.* Переглянути всі дуги e (i, j) із пропускною спроможністю α е > 0, що з'єднують вузол i з ще не позначеними вузлами j. Приписати позначки вузлам j і відзначити дуги e j  = e = (i, j). Тепер вузол i позначений та переглянутий, вузли j позначені, але не переглянуті. Якщо при цьому стік виявився позначеним, то необхідний ланцюг знайдений. У противному випадку після перегляду по всіх дугах (i, j) перейти до кроку

Крок 3.Нехай вузол i позначений і переглянутий. Перейти до кроку 1 і повторювати кроки алгоритму доти, поки не залишиться позначених і не переглянутих вузлів. На цьому пошук максимального потоку закінчується.

**Пошук найкоротшого шляху.**

Якщо для мережі кожне ребро характеризується деяким числом, що є відстанню між вузлами мережі, то виникає задача визначення найкоротшої відстані між заданими вузлами, тобто *початку і стоку.*

Розглянемо *алгоритм Дейкстри* для визначення найкоротшого шляху (ланцюга) з початку в стік.

Крок 0. Вибрати як перспективну множину вузлів множину S c = S 0  і покласти d i = 0 для i ∈S 0 та d i = ∞ для i ∉ S 0 .

Крок 1. Вибрати вузол i \* ∈ S c, якому відповідає найменше значення di ( i ∈ S0 ) . Знайдений в такий спосіб розмір d i відповідає найкоротшому шляху з деякого джерела у вузол i\* (довжиною дуги є c e), а дуга e i ( визначена для усіх вузлів i ∈ S c , крім джерел ) є остання дуга шляху . Якщо i \* - стік , то процедура пошуку найкоротшого шляху закінчується .

Крок 2. Переглянути дуги e = ( i \*, j ) і замінити оцінку d j на min {d j , d i + c e}. Якщо d j була дорівнена ∞ , увести вузол j у S c. Якщо d j зменшилася, увести позначення e j = e = (i\*, j).

Крок 3. Видалити i\* із S c і перейти до кроку 1 , якщо множина S c не порожня. На цьому пошук найкоротшого шляху закінчується.

**2. Варіанти індивідуальних завдань** :

Кожен студент повинен довільно задати мережу з такими характеристиками:

* кількість вершин не менше восьми;
* кількість ребер не менше дванадцяти.

Для обраної мережі довільно задати значення пропускної спроможності її ребер та відстані між вершинами з числового проміжку {1..9} умовних одиниць.

Знайти максимальний потік та найкоротший шлях.

**3. Склад звіту з роботи**

1. Короткі теоретичні відомості.

2. Індивідуально побудовану мережу з варіантом вхідних даних.

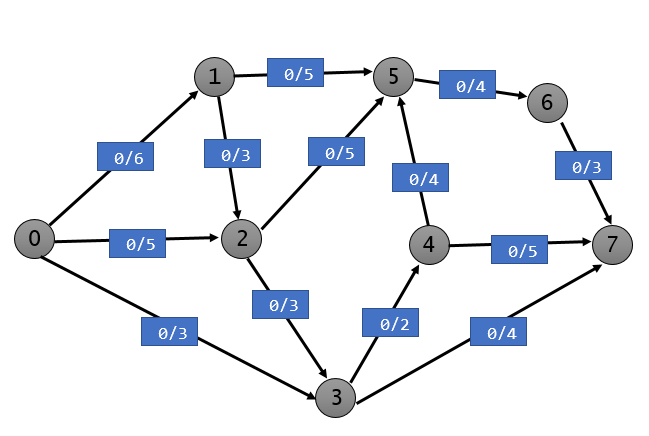
3. Повні розрахунки на мережі з пошуку max потока та найкоротшого шляху.

4. Висновки з лабораторної роботи.

**Хід роботи**

**Максимальний потік в мережі**

Вхідні дані ( крок 0 )



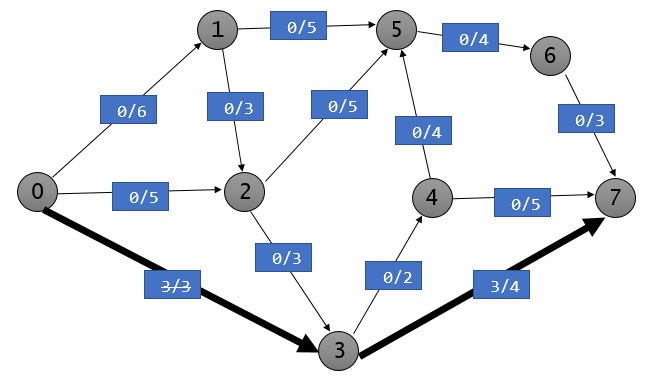
**Крок 1.** Обираємо маршрут ‘ 0 -> 3 -> 7 ’

p1 = min { (0,3), (3,7) } = min {3,4} = 3;

Pmax = p1 = 3;

Тоді : ребро (0,3) = 3/3; Ребро (3,7) = 3/4 .

Результат кроку 1



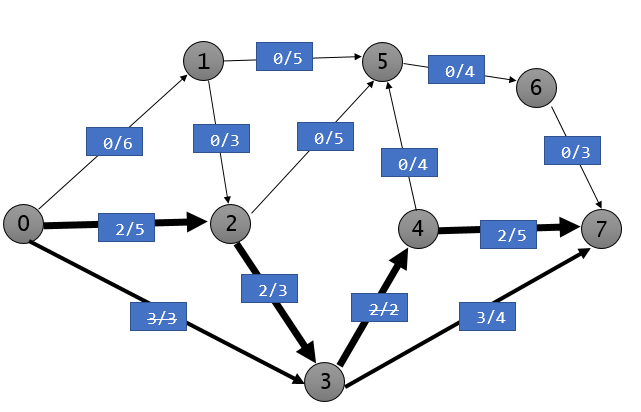
Крок 2. Обираємо маршрут ‘ 0 -> 2 -> 3 -> 4 -> 7 ’

p2 = min { (0,2), (2,3), (3,4), (4,7) } = min {5, 3, 2, 5} = 2;

Pmax = p1 + p2 = 3 + 2 = 5;

Тоді : (0,2) = 2/5; (2,3) = 2/3; (3,4) = 2/2; (4,7) = 2/5

Результат кроку 2

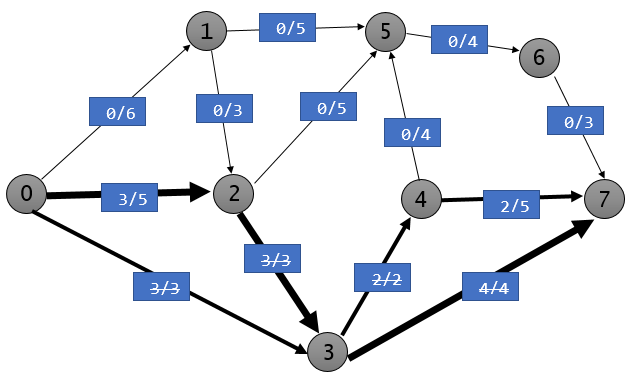


**Крок 3**. Обираємо маршрут ‘ 0 -> 2 -> 3 -> 7 ’

P3 = min {3, 1, 1} = 1;

Pmax = p1 + p2 + p3 = 3 + 2 + 1 = 6;

Результат кроку 3

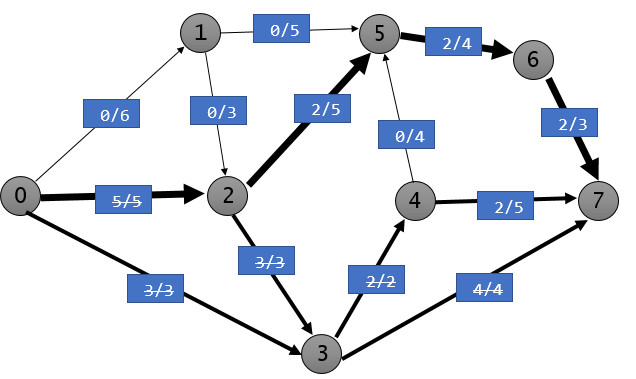


**Крок 4.** Обираємо маршрут ‘ 0 -> 2 -> 5 -> 6 -> 7 ’

P4 = min {2, 5, 4, 3} = 2;

Pmax = p1 + p2 + p3 + p4 = 3 + 2 + 1 + 2 = 8;

Результат кроку 4

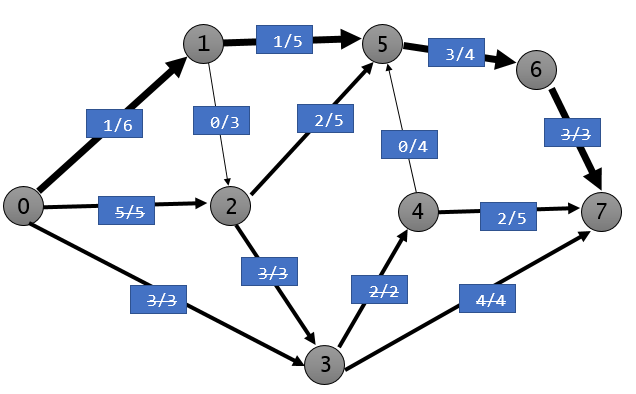


**Крок 5.** Обираємо маршрут ‘ 0 -> 1 -> 5 -> 6 -> 7 ’

P5 = min {6, 5, 2, 1} = 1;

Pmax = p1 + p2 + p3 + p4 + p5= 3 + 2 + 1 + 2 + 1 = 9;

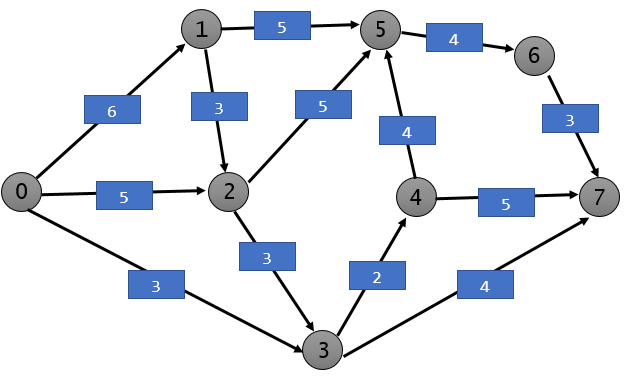
Результат кроку 5



**Результат** : максимальний потік для обраного графа з 8 вершин та 13 дуг дорівнює **9 одиницям**.

**Пошук мінімального шляху**

Вхідні дані ( крок 0 )

****

**Крок 1. Оберемо вершину 0. Приймемо d(0) = 0 .**

**А всі інші вершини -> d(1)=d(2)=d(3)=d(4)=d(5)=d(6)=d(7)=** **∞**

**Крок 2. Поточна змінна y = 0 :**

**d(1) = min { d(1) ; d(0) + d(0,1) } = min{ ∞ ; 0 + 6 }= 6;**

**d(2) = min { d(2) ; d(0) + d(0,2) } = min{ ∞ ; 0 + 5 }= 5;**

**d(3) = min { d(3) ; d(0) + d(0,3) } = min{ ∞ ; 0 + 3 }= 3;**

**d(y) = min { d(1) ; d(2) ; d(3) } = min { 6 ; 5 ; 3 } = 3.**

**Крок 3. Поточна змінна y = 3 :**

**d(1) = min { d(1) ; d(3) + d(3,1) } = min{ 6 ; 3 + ∞ }= 6;**

**d(2) = min { d(2) ; d(3) + d(3,2) } = min{ 5 ; 3 + ∞ }= 5;**

**d(4) = min { d(4), d(3) + d(3,4) } = 3 + 2 = 5;**

**d(5) = min { d(5), d(3) + d(3,5) } = min{ ∞ ; 3 + ∞ }= ∞;**

**d(6) = min { d(6), d(3) + d(3,6) } = min{ ∞ ; 3 + ∞ }= ∞;**

**d(7) = min { d(7) ; d(3) + d(3,7) } = 3 + 4 = 7;**

**d(y) = min { d(1); d(2) ; d(4) ; d(5); d(6); d(7) } = 5;**

**y = 2 або 4.**

**Крок 4. Поточна змінна y = 4 :**

**d(1) = min { d(1) ; d(4) + d(4,1) } = min{ 6 ; 5 + ∞ }= 6;**

**d(2) = min { d(2) ; d(4) + d(4,2) } = min{ 5 ; 5 + ∞ }= 5;**

**d(3) = min { d(3) ; d(4) + d(4,3) } = min{ 3 ; 5 + ∞ }= 3;**

**d(5) = min { d(5) ; d(4) + d(4,5) } = min { ∞ ; 5 + 4 } = 9;**

**d(6) = min { d(6), d(4) + d(4,6) } = min{ ∞ ; 5 + ∞ }= ∞;**

**d(7) = min { d(7) ; d(4) + d(4,7) } = min { 7 ; 5 + 5 } = min { 7 ; 10 } = 7;**

**Крок 5. Поточна змінна y = 2 :**

**d(1) = min { d(1) ; d(2) + d(2,1) } = min{ 6 ; 5 + ∞ }= 6;**

**d(3) = min { d(3) ; d(2) + d(2,3) } = min{ 3 ; 5 + 3 }= 3;**

**d(4) = min { d(4) ; d(2) + d(2,4) } = min{ 5 ; 5 + ∞ }= 5;**

**d(5) = min { d(5) ; d(2) + d(2,5) } = min { 9 ; 5 + 5 } = 9;**

**d(6) = min { d(6), d(2) + d(2,6) } = min{ ∞ ; 5 + ∞ }= ∞;**

**d(7) = min { d(7) ; d(2) + d(2,7) } = 7;**

**Крок 6. Поточна змінна y = 1 :**

**d(2) = min { d(2) ; d(1) + d(1,2) } = min{ 5 ; 6 + 3 }= 5;**

**d(3) = min { d(3) ; d(1) + d(1,3) } = min{ 3 ; 6 + ∞ }= 3;**

**d(4) = min { d(4) ; d(1) + d(1,4) } = min{ 5 ; 6 + ∞ }= 5;**

**d(5) = min { d(5) ; d(1) + d(1,5) } = min { 9 ; 6 + 5 } = 9;**

**d(6) = min { d(6), d(1) + d(1,6) } = min{ ∞ ; 6 + ∞ }= ∞;**

**d(7) = min { d(7) ; d(1) + d(1,7) } = 7;**

**Крок 7. Поточна змінна y = 5 :**

**d(1) = min { d(1) ; d(5) + d(5,1) } = 6;**

**d(2) = min { d(2) ; d(5) + d(5,2) } = 5;**

**d(3) = min { d(3) ; d(5) + d(5,3) } = 3;**

**d(4) = min { d(4) ; d(5) + d(5,4) } = 5;**

**d(6) = min { d(6), d(5) + d(5,6) } = min{ ∞ ; 9 + 4 }= 13;**

**d(7) = min { d(7) ; d(2) + d(2,7) } = 7;**

**Крок 8. Поточна змінна y = 6 :**

**d(1) = min { d(1) ; d(6) + d(6,1) } = 6;**

**d(2) = min { d(2) ; d(6) + d(6,2) } = 5;**

**d(3) = min { d(3) ; d(6) + d(6,3) } = 3;**

**d(4) = min { d(4) ; d(6) + d(6,4) } = 5;**

**d(5) = min { d(5) ; d(6) + d(6,5) } = min { 9 ; 13 + ∞ } = 9;**

**d(7) = min { d(7) ; d(6) + d(6,7) } = min { 7 ; 13 + 3 } = 7;**

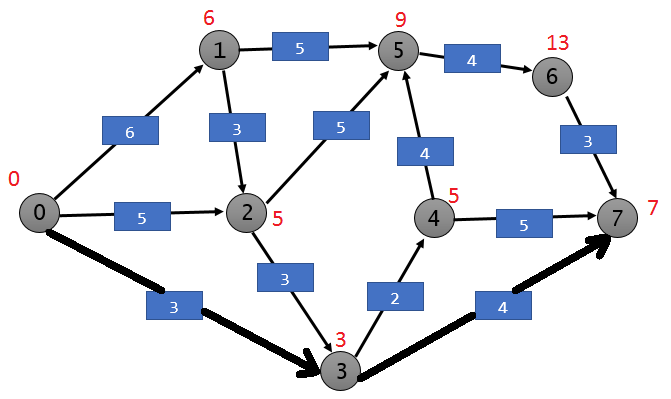
**Отже, отримали такі результати :**

**d(0) = 0; d(1) = 6; d(2) = 5; d(3) = 3;**

**d(4) = 5; d(5) = 9; d(6) = 13; d(7) = 7;**

(Цифри над вершинами – мінімальний

шлях від джерела до вершини )

****

*Мінімальний шлях від джерела до стоку.*

**Висновки :** проблема знаходження максимального потоку та мінімального шляху має величезну практичне значення. Розглянуті вище методи вирішення цих проблем можна застосовувати на транспортних, комунікаційних, електричних та будь-яких інших мережах реального світу.

Використання методу Форда-Фалкерсона (пошуку максимального потоку) дає змогу з`ясувати, які ребра (наприклад, труби) мають завелику пропускну здатність, а які – недостатню, а також визначити, яку максимальну кількість ресурсів можна передати із джерела в стік при максимальному навантаженні системи.

Алгоритм Дейкстри дозволяє вирішити проблему пошуку мінімального шляху, наприклад, при прокладанні найкоротшого шляху з пункту А в пункт Б. Це дозволяє раціонально використати ресурси, наприклад, при транспортуванні товару.

Аналізуючи результати роботи алгоритмів, можна зробити висновки і внести корективи до вже існуючої мережі. Або, іншими словами, оптимізувати її.