МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКІЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра АТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт по курсу

«ОСНОВИ ТЕОРІЇ МЕРЕЖ І СИСТЕМ ЗВ'ЯЗКУ»

для студентів напрямку 6.050903 “Телекомунікації”

денної форми навчання

Затверджені на засіданні кафедри
«Автоматика і телекомунікації»
протокол №3 від 19.03.2009

Затверджені на засіданні
Навчально-видавничої ради ДОННТУ
протокол № 3 від 24.06.09

Донецьк – 2009

Методичні вказівки до виконання лабораторних робіт з курсу “Основи теорії мереж та систем зв’язку” для студентів напрямку 6.050903 “Телекомунікації” денної форми навчання. /Воропаєва В.Я., Бессараб В.І., Левченко Л.В. – Донецьк, ДонНТУ, 2009. - 60 с.

Укладачі:
Воропаєва В.Я., Бессараб В.І., Левченко Л.В.

Затверджені на засіданні кафедри
«Автоматика і телекомунікації»
протокол №3 від 19.03.2009

ОПИС ПРОГРАМИ NET ANALYST 1.2

1. Призначення програми

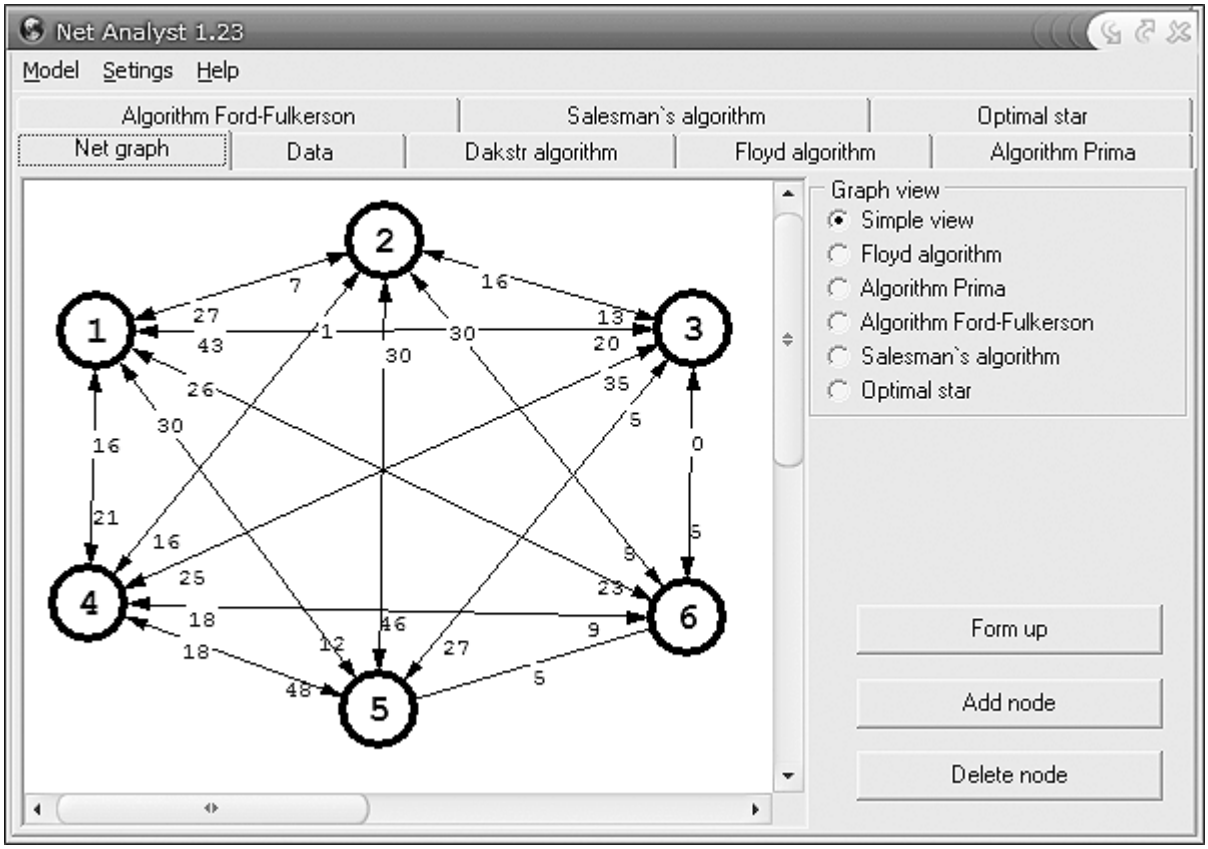
Програма призначена для аналізу та моделювання телекомунікаційної мережі. Вона дозволяє досліджувати мережу за допомогою алгоритмів Дейкстри, Флойда, Прима, Форда-Фалкерсона, Комівояжера та оптимальної зірки.

2. Обмеження та умовні позначення

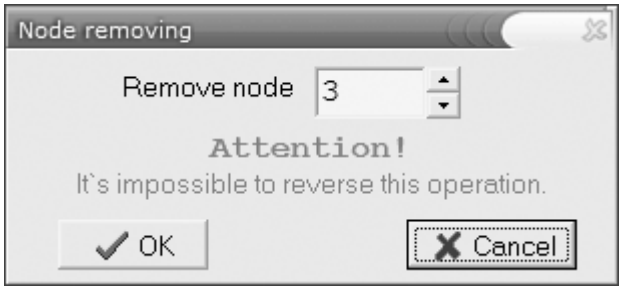
Максимальна кількість вузлів у мережі: 20. Можливо працювати як з симетричними, так і з несиметричними мережами. При зазначенні відстані (пропускної спроможності) значення слід вибирати з діапазону [0; 99]. Значення, що лежать за межами цього діапазону, будуть визначені як нескінченність та позначені дефісом «-». Відповідно, для позначення відсутності зв'язку (або необмеженої пропускної здатності) використовується дефіс «-» або будь-яке число, що виходить за діапазон [0; 99].

3. Інтерфейс програми

Головне вікно програми надає в графічному вигляді інформацію про мережу, що досліджується, а також результат аналізу мережі за допомогою одного з обраних алгоритмів (опції **Graph view**). Кнопка «**Form up»** дозволяє упорядкувати вузли мережі, кнопка «**Add node»** додає до мережі новій вузол, а «**Delete node»** видаляє будь-який вузол (відмінити цю операцію неможливо!).

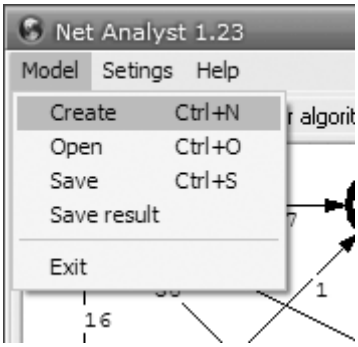


Головне вікно програми Net Analyst

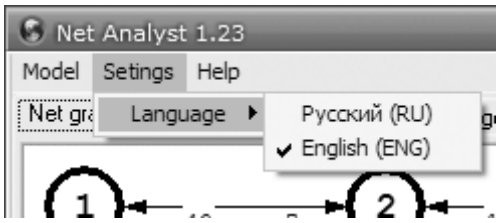


Попередження при видаленні вузла.

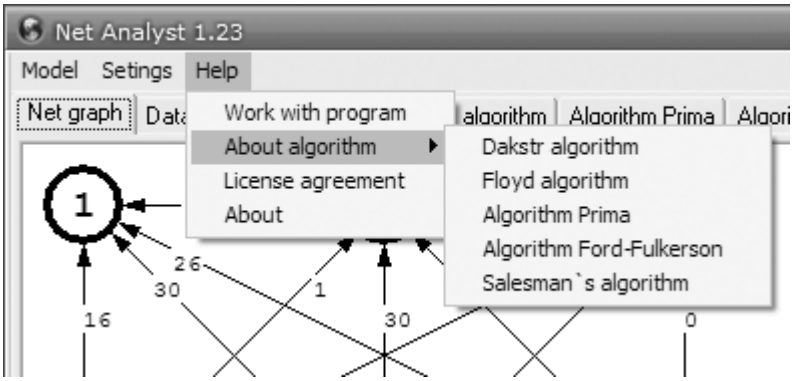
Головне меню програми складається з трьох підменю. Меню «**Model**» дозволяє створити нову модель, відкрити вже існуючу або зберегти модель мережі.



Меню «**Model**»



Меню «**Settings**»

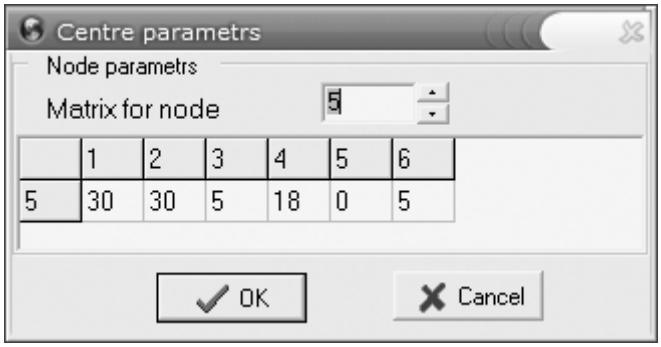


Меню «**Help**»

Меню «**Settings**» дозволяє встановити необхідну мову програми. Меню «**Help**» дозволяє отримати коротку інформацію про можливості програми та докладну інформацію про алгоритми, що використовуються в програмі.

4. Робота з програмою

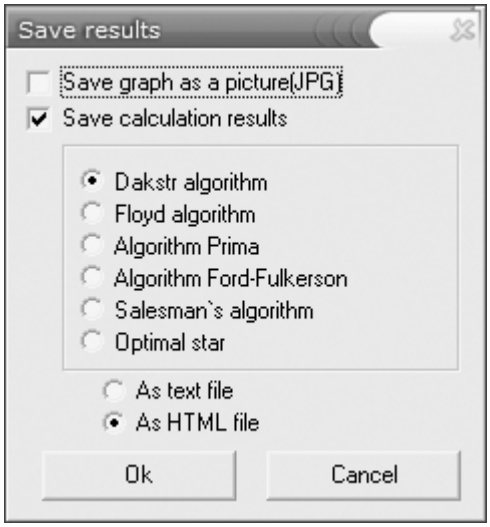
При створенні нової мережі найбільш зручним є матричне представлення мережі, що може бути вибране на другій вкладці «**Data**». При необхідності здійснити деякі зміни в мережі, можна натиснути правою кнопкою на необхідному вузлі. При цьому з’явиться вікно з матрицею для цього вузла, в яку можна ввести необхідні зміни.



Дані для конкретного вузла мережі

Результати обчислень можна проглянути на графі мережі або отримати в табличному вигляді для кожного кроку обчислень. У разі зміни будь-яких початкових даних буде проведено автоматичний перерахунок параметрів мережі.

Для збереження результатів обчислень необхідно вибрати опцію «**Save result»,** яка дозволяє зберегти результати для конкретного алгоритму (в текстовий або HTML файл) та графічне зображення мережі.



Меню «**Save result**»

ЛАБОРАТОРНА РОБОТА №1

ЗАДАЧА ПРО НАЙКОРОТШИЙ ШЛЯХ. АЛГОРИТМ ДЕЙКСТРИ

Мета: *провести аналіз мережі з використанням алгоритму Дейкстри та визначити найкоротші шляхи з першого вузла до всіх інших.*

1. ОСНОВНІ ПОЛОЖЕННЯ

Алгоритм Дейкстри широко використовується для знаходження найкоротших шляхів від заданого вузла телекомунікаційної мережі до інших вузлів, зокрема в протоколах маршрутизації IP-мереж [1].

1.1. Постановка задачі

Розглянемо телекомунікаційну мережу у вигляді графу $G = (N,A)$, де $N = \{1,2,\ldots,n\}$ – множина вузлів, A – множина дуг. Вважаємо, що ця мережа має одне інформаційне джерело – вузол s та один стік – вузол t.

Цілочисельна функція f_{ij}, визначена на множині A, називається потоком в мережі $G = (N,A)$, якщо:

$$f_{ij} \geq 0 \text{ для всіх } (i,j) \in A \tag{1.1}$$

$$\sum_j f_{ij} - \sum_j f_{ji} = 0, \text{ для всіх } i \in N,\ i \neq s, i \neq t, \tag{1.2}$$

Згідно цього визначення, якщо вузол j не виступає ні стоком, на джерелом, то величина потоку, що втікає в нього, повинна дорівнювати величині потоку, що витікає з нього, тобто виконуватися правило збереження потоку (1.2).

Однією з найбільш важливих потокових задач є задача знаходження шляху з джерела до стоку такого, що вартість (час) проходження потоку заданої величини по цьому шляху буде мінімальна. Припустимо, що кожній дузі (i,j) заданої мережі відповідає деяке число C_{ij}, що називається узагальненою вартістю дуги.

Фіктивним, або безкоштовним, дугам приписується вартість $C_{ij} = 0$, а кожній парі вузлів ***(i,j)***, для яких не існує дуги, що з'єднує їх, приписується вартість $C_{ij} = \infty$.

Задача, що вирішується в даній роботі, полягає в знаходженні такого шляху з джерела ***s*** до стоку ***t***, для якого вартість проходження одиниці потоку буде мінімальна.

Математично ця задача може бути записана так:

$$\text{мінімізувати } \sum_i \sum_j C_{ij} f_{ij} \tag{1.3}$$

за умови, що

$$\sum_i f_{si} - \sum_i f_{is} = 1, \tag{1.4}$$

$$\sum_i f_{ij} - \sum_i f_{ji} = 0,\ i \neq s,\ i \neq t, \tag{1.5}$$

$$\sum_i f_{tj} - \sum_i f_{jt} = -1, \tag{1.6}$$

$$f_{ij} \geq 0\ . \tag{1.7}$$

Згідно з обмеженням (1.4), одиниця потоку витікає з джерела. Рівняння (1.5) гарантує збереження даної одиниці потоку при проходженні в мережі. Згідно з рівнянням (1.6), одиниця потоку потрапляє до стоку. В якості найкоротшого шляху може бути взята послідовність суміжних дуг (i, j), для яких $f_{ij} = 1$. Для розв'язання цієї типової задачі лінійного програмування скористуємось спеціальним засобом, відомим під назвою ***алгоритм Дейкстри*** **[3]**.

Алгоритм базується на приписуванні вузлам ***тимчасових***, **або** ***постійних позначок***. Спочатку кожному вузлу, окрім джерела, приписується тимчасова позначка, яка дорівнює довжині найкоротшої дуги, що веде з джерела в даний вузол. Джерелу приписується постійна позначка, значення якої дорівнює нулю. Кожному вузлу, у який не можна потрапити безпосередньо з джерела, приписується тимчасова позначка ∞, а всім іншим вузлам - тимчасові позначки $C_{sj},\ j \neq s$. Якщо визначено, що вузол належить до найкоротшого шляху, його позначка стає постійною. Алгоритм базується на наступному простому факті:

якщо відомий найкоротший шлях з вузла s у вузол j та вузол k належить до цього шляху, то найкоротший шлях з s до k є частиною першого шляху, що закінчується в вузлі k. Алгоритм починає працювати при $j = s$. Після цього величина j послідовно збільшується на одиницю і при $j = t$ алгоритм завершує роботу.

1.2. Ітераційна процедура

Для даного вузла j через δ_i будемо позначати оцінку довжини найкоротшого шляху з джерела s у вузол j. Якщо ця оцінка не може бути поліпшена, то відповідне значення ми назвемо постійною позначкою і будемо позначати її символом $[\delta_i]$. У протилежному випадку, назвемо його тимчасовою позначкою. Спочатку процедури постійна позначка приписується лише джерелу. Кожна інша позначка є тимчасовою і її величина дорівнює довжині дуги, що веде з джерела до відповідного вузла:

$$\delta_j = C_{sj}.$$

Для визначення найближчого до джерела вузла оберемо тимчасову позначку з мінімальним значенням та оголосимо її постійною позначкою:

$$[\delta_i] = \min \delta_i.$$

Після цього до тих пір, доки стоку не буде приписана постійна позначка, необхідно виконати наступні дві процедури:

1. Розглянути вузли, що залишилися з тимчасовою позначкою. Порівняти величину кожної *тимчасової позначки* із *сумою* величини останньої з *постійних позначок* і *довжини* дуги, що веде з відповідного постійно позначеного вузла у вузол, що розглядається. Мінімальна з цих двох величин визначається як нова тимчасова позначка вузла.

2. Серед тимчасових позначок обрати ту, значення якої мінімальне, і оголосити її постійною позначкою. Якщо при цьому постійна позначка приписується вузлу t, то алгоритм завершує роботу. У протилежному випадку перейти до кроку 1.

Цю процедуру можна оформити у вигляді таблиці розв'язання, в якій стовпці відповідають вузлам мережі, рядки - крокам ітеративного процесу, а її елементи - постійним та тимчасовим позначкам (див. Приклад 1).

Приклад 1. Знайти найкоротші шляхи з першого вузла до решти вузлів в мережі, граф якої приведено на рис. 1.1. Будемо вважати, що джерелом в мережі є вузол 1, а стоком – вузол 5.

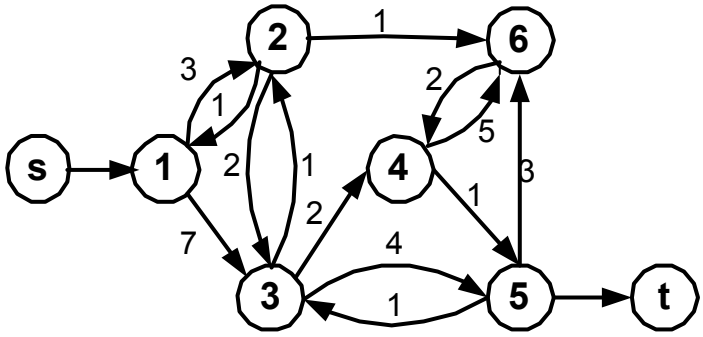


Рис. 1.1 – Граф мережі

Процедуру вирішення задачі зведено в таблицю 1.1.

Таблиця 1.1

Рішення прикладу 1

№ кроку	s	1	2	3	4	5	6	T
0	[0]	∞	∞	∞	∞	∞	∞	∞
1	[0]	0	∞	∞	∞	∞	∞	∞
2	[0]	[0]	∞	∞	∞	∞	∞	∞
3	[0]	[0]	3	7	∞	∞	∞	∞
4	[0]	[0]	[3]	7	∞	∞	∞	∞
5	[0]	[0]	[3]	5	∞	∞	4	∞
6	[0]	[0]	[3]	5	∞	∞	[4]	∞
7	[0]	[0]	[3]	5	6	∞	[4]	∞
8	[0]	[0]	[3]	[5]	6	∞	[4]	∞
9	[0]	[0]	[3]	[5]	6	9	[4]	∞
10	[0]	[0]	[3]	[5]	[6]	9	[4]	∞
11	[0]	[0]	[3]	[5]	[6]	7	[4]	∞
12	[0]	[0]	[3]	[5]	[6]	[7]	[4]	∞
13	[0]	[0]	[3]	[5]	[6]	[7]	[4]	7
14	[0]	[0]	[3]	[5]	[6]	[7]	[4]	[7]

Робота алгоритму починається з того, що джерелу *s* приписується постійна позначка [0], а вузлам $j = 1,2,...,6,\ t$ - тимчасові позначки $\delta_j = C_{sj}$. Таким чином, $\delta_1 = 0$ та $\delta_j = \infty$ для $j = 2,...,t$.

Оскільки значення $\delta_1 = 0$ є мінімальним серед всіх тимчасових позначок, вузлу **1** приписується постійна позначка [0].

Вузли **2** та **3** безпосередньо зв'язані з вузлом **1**, останнім з постійно позначених вузлів. Вони можуть отримати нові тимчасові позначки.

Відзначимо, що $\delta_1 + C_{12} = 0 + 3 < \infty$ та $\delta_1 + C_{13} = 0 + 7 < \infty$. Тому вузлам **2** та **3** приписуються нові тимчасові позначки $\delta_2 = 3$ та $\delta_3 = 7$ відповідно.

З найменших тимчасових позначок обираємо постійну. Після порівняння $\delta_2 < \delta_3$, вузлу **2** приписується постійна позначка $\delta_2 = 3$.

Поновлюємо тимчасові позначки. Вузли **3** і **6** безпосередньо зв'язані з вузлом **2**. Крім того, $\delta_2 + C_{23} = 3 + 2 = 5 < 7$ та $\delta_2 + C_{26} = 3 + 1 = 4 < \infty$. Тому вузлам **3** та **6** приписуються нові тимчасові позначки $\delta_3 = 5$ та $\delta_6 = 4$ відповідно. Оскільки, $\delta_6 < \delta_3$ вузол **6** стає позначеним постійно, тобто $\delta_6 = 4$.

На даному кроку тимчасовими є позначки $\delta_3 = 5$ та $\delta_4 = \delta_5 = \delta_t = \infty$. Останнім вузлом, якому була приписана постійна позначка, є вузол **6**. Він безпосередньо пов'язаний лише з вузлом **4**. Відзначимо, що $\delta_6 + C_{64} = 4 + 2 = 6 < \infty$. Отже, вузлу **4** приписується нова тимчасова позначка $\delta_4 = 6$. Порівнюємо усі тимчасові позначки $\delta_3 = \min\{\delta_3, \delta_4, \delta_5, \delta_t\}$, вузлу **3** приписується постійна позначка $\delta_3 = 5$. Аналогічно проведемо подальші обчислення.

Алгоритм закінчує роботу, коли вузлу t приписується постійна позначка $\delta_t = 7$. Отже, довжина найкоротшого шляху з вузла s в вузол t дорівнює **7**. Цей шлях складається з дуг, для кожної з яких різниця між значеннями постійних позначок її кінцевих вузлів дорівнює довжині цієї дуги. Цей факт можна використати при побудові ***зворотного ходу*** алгоритму Дейкстри, тобто при визначенні, через які саме вузли проходить найкоротший шлях.

Отже, якщо δ_i та δ_j - постійні позначки вузлів i та j відповідно, то умова, при виконанні якої ці вузли належать найкоротшому шляху, може бути записана так:

$$\delta_j = \delta_i + C_{ij} \qquad (1.8)$$

Співвідношення (**1.8**) можна використати рекурсивно, рухаючись від вузла t до вузла s. Визначивши з матриці вартостей вузол, що безпосередньо передує t в найкоротшому шляху, будемо повторювати дану процедуру, доки не досягнемо вузла s.

Матриця вартостей для графа, заданого на рис. 1.1, приведена в таблиці 1.2.

Таблиця 1.2

Матриця вартостей заданого графа.

C	**1**	**2**	**3**	**4**	**5**	**6**
1	0	3	7	∞	∞	∞
2	1	0	2	∞	∞	1
3	∞	1	0	2	4	∞
4	∞	∞	∞	0	1	5
5	∞	∞	1	∞	0	3
6	∞	∞	∞	2	∞	0

Наприклад, визначимо найкоротший шлях між джерелом s та вузлом **5**.

Позначка 5-го вузла - [7]. У стовпці 5-го вузла матриці вартостей знаходимо, що 5й вузол з'єднаний з третім і четвертим. Для цих вузлів виконуємо наступну операцію: від позначки 5-го вузла віднімаємо величину вартості відповідної дуги і, якщо отримана різність співпадає з позначкою для вузла, що розглядається, то цей вузол входить в найкоротший шлях.

В 5-й вузол можна попасти з 3-го або 4-го. Перевіримо, чи належить 3-й вузол найкоротшому шляху: $C_{35} = 4$.

Якби 3-й вузол належав найкоротшому шляху, то постійна позначка для цього вузла мала б бути: 7-4=3.

Але, як видно з таблиці 1.1 постійна позначка 3-го вузла дорівнює 5. Отже 3й вузол не належить найкоротшому шляху.

Аналогічна перевірка для 4-го вузла: $C_{45} = 1$, 7-1=6.

Це і є постійною позначкою для 4-го вузла. Отже він належить найкоротшому шляху.

Повторюємо цю процедуру для 4-го і наступних вузлів, як показано на рис.1.2.

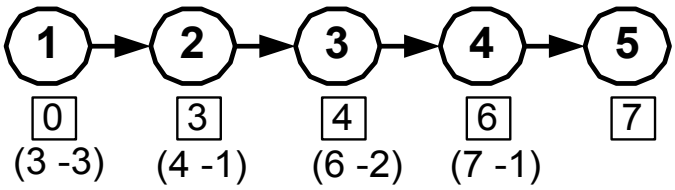


Рис.1.2 Зворотний хід алгоритму Дейкстри

Отже, найкоротший шлях у мережі утворюється послідовністю вузлів:

1-2-6-4-5.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ .

1. Запустіть NET ANALYST.
2. Створіть нову мережу, та введіть дані відповідно до вашого варіанту.
3. Збережіть дані вашого графа на диску.
4. Виберіть пункт «**Dakstr algorithm**», розгляньте кожний крок рішення і остаточну таблицю.
5. Здійсніть модифікацію мережі:
    a) збільшіть відстань до вузла, що отримав першим постійну позначку, та проаналізуйте, які найкоротші шляхи змінились та наскільки;
    b) додайте в мережу ще один вузол, пов’язаний щонайменше з трьома існуючими вузлами мінімальними відстанями, та проаналізуйте, які найкоротоші шляхи тепер проходять через доданий вузол;
    c) вилучіть з початкового графу вузол, що входив до більшості найкоротоших шляхів, та з’ясуйте, який з вузлів тепер найчастіше зустрічається в найкоротших шляхах.

6. Отримайте остаточні таблиці для трьох варіантів модифікації.
7. Самостійно розрахуйте найкоротші шляхи з першого вузла у всі інші в початковому графі і порівняйте результати з результатами, отриманими з допомогою NET ANALYST.

3. ЗМІСТ ЗВІТУ.

1. Мета роботи.
2. Граф мережі згідно з вашим варіантом та матриця вартостей для цього графа.
3. Покрокове рішення та фінальна таблиця, отримані за допомогою NET ANALYST.
4. Модифіковані графи, та фінальні таблиці для них.
5. Самостійний розрахунок вхідного графа.
6. Висновки.

4. КОНТРОЛЬНІ ПИТАННЯ

1. Навести приклад використання алгоритму Дейкстри
2. Дати визначення потоку в мережі
3. Навести правило збереження потоку
4. Записати задачу знаходження найкоротшого шляху. Пояснити зміст цільової функції та обмежень

ЛАБОРАТОРНА РОБОТА №2

ЗАДАЧА ПРО БАГАТОПОЛЮСНИЙ НАЙКОРОТШИЙ ЛАНЦЮГ. АЛГОРИТМ ФЛОЙДА.

Мета: *Засвоїти алгоритм Фолйда знаходження найкоротшого ланцюга для будь-якої пари вузлів телекомунікаційної мережі.*

1. ОСНОВНІ ПОЛОЖЕННЯ

В попередній роботі розглянуто алгоритм, що дозволяє знайти найкоротші шляхи з першого вузла мережі до решти вузлів. Розглянемо задачу знаходження найкоротших ланцюгів між усіма парами вузлів мережі ***G=(N, A)*** (задачу про багатополюсний найкоротший ланцюг). Отже, ***N={1, 2, ..., n}*** – множина вузлів мережі. Дуги з множини ***A*** можуть бути орієнтованими або неорієнтованими. Однак, оскільки напрямок потоку в неорієнтованих дугах неможливо визначити заздалегідь, то кожну таку дугу слід замінити двома орієнтованими дугами з протилежними напрямками та довжиною, що дорівнює довжині неорієнтованої дуги.

Якщо довжині кожної дуги відповідає відстань або вартість одиниці потоку по цій дузі C_{ij}, то найкоротшим ланцюгом між двома довільними вузлами є ланцюг, вартість одиниці потоку для якого є мінімальною.

Позначимо через d^*_{ik} довжину найкоротшого ланцюга з вузла ***i*** до вузла ***k***.

Припустимо, що величина d_{ik} представляє найкращу оцінку довжини найкоротшого ланцюга, що з'єднує вузли ***i*** й ***k***. Тоді довжина $\overline{d_{ik}} = d_{ij} + d_{jk}$ кожного найкоротшого ланцюга, що проходить через проміжний вузол ***j***, або перевищує величину d_{ik} , або поточна довжина найкоротшого ланцюга дорівнює $\overline{d_{ik}}$. Однак якщо довжина $\overline{d_{ik}}$ нового ланцюга (через деякий транзитний вузол) менше d_{ik} , то поточне значення d_{ik} слід замінити на $\overline{d_{ik}}$.

Алгоритм Флойда [3] працює наступним чином. Спочатку за довжину найкоротшого ланцюга між двома довільними вузлами ***i*** та ***k*** приймається величина довжини дуги ***(i, k)***, що з'єднує ці вузли (рис. 2.1). Потім послідовно перевіряються всі можливі транзитні вузли, що розташовані між ***i*** та ***k***. Якщо довжина ланцюга, що проходить через деякий транзитний вузол, менше поточного значення d_{ik}, то оцінці d_{ik} привласнюється нове значення. Ця процедура повторюється для всіх можливих пар вузлів, доки не будуть отримані всі значення d^*_{ik} .

Для будь-яких трьох різних вузлів ***i, j*** та ***k*** сформульовані вище умови можуть бути записані в вигляді нерівності:

$$d_{ij} + d_{jk} \geq d_{ik} \quad i \neq j \neq k, \tag{2.1}$$

оскільки в протилежному випадку найкоротший ланцюг и вузла ***i*** в ***k*** повинен містити вузол ***j*** і тоді величина d_{ik} не дорівнювала б довжині найкоротшого ланцюга. В алгоритмі Флойда початковим значенням d_{ik} є величина C_{ik} , а потім ця оцінка послідовно покращується, доки не буде знайдено найкоротший ланцюг між вузлами ***i*** та ***k***.

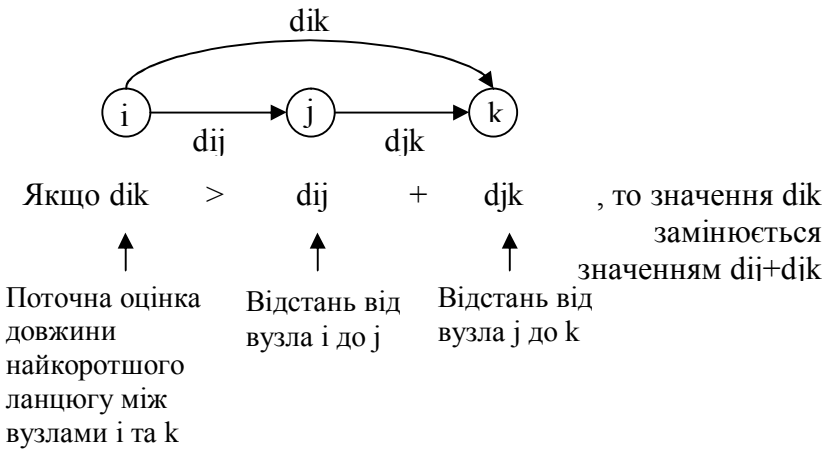


Рис. 2.1 - Приклад процедури порівняння

Алгоритм Флойда дозволяє вирішити задачу побудови багатополюсного найкоротшого ланцюга для мережі з **n** вузлів за **n** ітерацій. Для формального описання ітеративної процедури пошуку рішення, позначимо символом d^j_{ik} оцінку довжини найкоротшого ланцюга з вузла ***i*** до вузла ***k,*** отриману на ***j***-й ітерації. Ця

оцінка обирається як найменше значення з оцінки цього ланцюга на попередній (***j-1)***-й ітерації та довжини ланцюга через транзитний вузол ***j***:

$$d_{ik}{}^{j} = min\ [d_{ik}{}^{j-1}; d_{ij}{}^{j-1} + d_{jk}{}^{j-1}],\ i \neq j \neq k \qquad (2.2)$$

Якщо операцію (2.2) виконати з даною парою вузлів ***i*** й ***k*** і усіма можливими транзитними вузлами ***j (i ≠ j ≠ k),*** збільшуючи номери ***j*** від ***1*** до ***n*,** то значення d^{n}_{ik}, отримане на останній ітерації, буде дорівнювати довжині найкоротшого ланцюга з вузла ***i*** до вузла ***k***.

Будемо використовувати матричний метод, який дозволить запам'ятовувати довжини найкоротших ланцюгів та їх маршрути. На кожній ітерації алгоритму будуються дві матриці. Перша – *матриця довжин* – містить поточні оцінки довжини найкоротших ланцюгів, тобто на ***j-й*** ітерації ця матриця виглядає:

$$D^{j} = [d^{j}_{ik}].$$

Алгоритм починає роботу при $D^{o} = [d^{\circ}_{ik}]$, де $d^{\circ}_{ik} = C_{ik}$. Потім виконується порівняння (2.2) для усіх елементів матриці D° і обчислюється D^{1} і т. д. доки для кожної пари вузлів не буде виконано критерій оптимальності (2.1).

Друга матриця – *матриця маршрутів* – необхідна для находження транзитних вузлів найкоротших ланцюгів. На ***j-й*** ітерації вона визначається як $R^{j} = r^{j}_{ik}$, где $\mathbf{r^{j}_{ik}}$ — перший транзитний вузол найкоротшого ланцюга з ***i*** в ***k,*** що обирається серед вузлів множини ***{1,2,...,j} (i ≠ j ≠ k).*** На початку алгоритма $R^{\circ} = [r^{\circ}_{ik}]$, где $r^{\circ}_{ik} = k$. На ***j-й*** ітерації вузол r^{j}_{ik} обирається з наступного співвідношення:

$$r_{ik}^{j} = \begin{Bmatrix} j, & якщо \quad d^{\,j-1}_{ik} \ \rangle \ d_{ij}^{j-1} + d_{jk}^{j-1} \\ r_{ik}^{j-1} & в\ протилежному\ випадку \end{Bmatrix} \qquad (2.3)$$

Таким чином, після побудови початкових матриць D° и R° треба для кожного ***j =1, 2,...,n,*** виконати 2 кроки:

Крок 1. Визначити ***j-й*** вузол як базовий. (Тобто на цій ітерації саме цей вузол перевіряється як «кандидат» в транзитні вузли між іншими вузлами).

Викреслити ***j-й*** рядок та ***j-й*** стовпчик матриці відстаней D^{j-1}, а також рядки і стовпчики D^{j-1}, що містять в базовому рядку чи стовпчику елементи, рівні ∞.

Крок 2. Кожний невикреслений елемент d_{ik}^{j-1} $(k \neq j)$ матриці відстаней порівнюється з сумою d_{ij}^{j-1} та d_{jk}^{j-1} елементів, розташованих на перетині розглянутого стовпчика і базового рядка та розглянутого рядка і базового стовпчика відповідно.

Якщо виконується нерівність $d_{ij}^{j-1}+d_{jk}^{j-1} \geq d_{ik}^{j-1}$, тобто шлях через транзитний вузол ***j*** гірше, ніж попередня оцінка, то перейти до розгляду наступного невикресленого елемента. В протилежному випадку включення вузла ***j*** в маршрут зменшує довжину найкоротшого ланцюга з вузла ***i*** до вузла ***k,*** отже необхідно поновити значення відповідних елементів в обох матрицях:

елементу d_{ik}^{j-1} *матриці довжин* присвоїти значення $d_{ik}^{j}=d_{ij}^{j-1}+d_{jk}^{j-1}$, а відповідному елементу *матриці маршрутів* — значення ***j.*** Після перегляду усіх елементів знову перейти до кроку **1** при ***j=j+l.*** При ***j=n*** будуть побудовані матриця відстаней і матриця маршрутів, що надають рішення задачі.

Приклад 2.

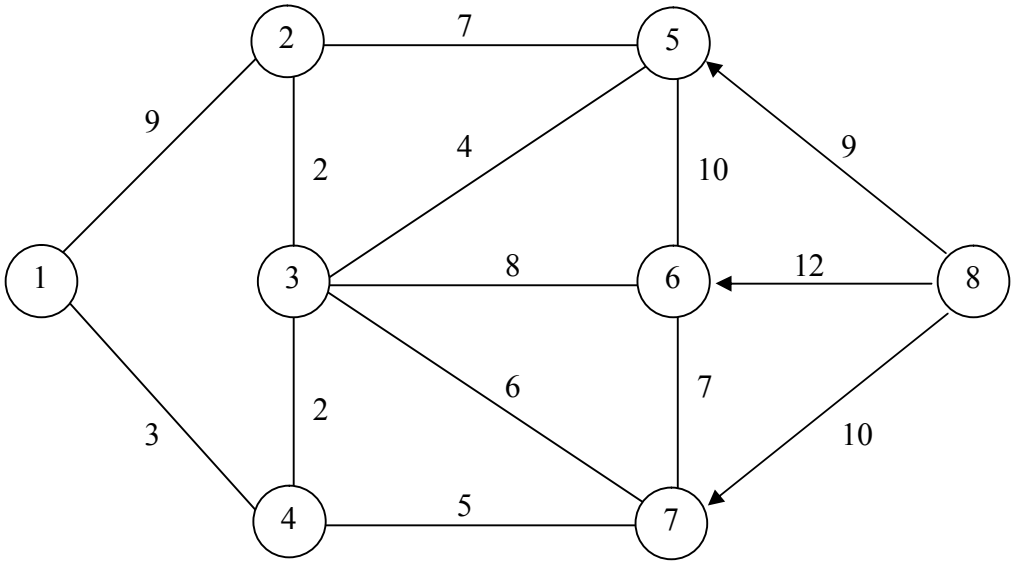


Рисунок 2.2 – Граф мережі

Задано граф мережі (рис. 2.2) , в якій потрібно знайти найкоротші шляхи між усіма парами вузлів.

Для рішення даної задачі потрібно на кожному кроці складати матриці найкоротших відстаней (***D***) і матриці маршрутів (***R***). Матриці відстаней фіксують зміни довжин маршрутів, а матриця маршрутів останній вузол, через який проходить найкоротший шлях.

Початкові значення матриць: $\boldsymbol{D^0}$ – матриця відстаней (вартостей) $\boldsymbol{C_{ij}}$ (якщо немає прямого зв'язку, то ставиться «∞»), і $\boldsymbol{R^0}$ – матриця, де в кожнім стовпці проставлено номер цього стовпця.

Крок № 0: j=0

$\boldsymbol{D^0}$

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	9	∞	3	∞	∞	∞	∞
2	9	0	2	∞	7	∞	∞	∞
3	∞	2	0	2	4	8	6	∞
4	3	∞	2	0	∞	∞	5	∞
5	∞	7	4	∞	0	10	∞	∞
6	∞	∞	8	∞	10	0	7	∞
7	∞	∞	6	5	∞	7	0	∞
8	∞	∞	∞	∞	9	12	10	0

$\boldsymbol{R^0}$

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	2	3	4	5	6	7	8
2	1	2	3	4	5	6	7	8
3	1	2	3	4	5	6	7	8
4	1	2	3	4	5	6	7	8
5	1	2	3	4	5	6	7	8
6	1	2	3	4	5	6	7	8
7	1	2	3	4	5	6	7	8
8	1	2	3	4	5	6	7	8

Крок № 1: j=1.

З матриці $\boldsymbol{D^0}$ викреслюємо ті рядки і стовпці, де на перетинанні з першим стовпцем і першим рядком присутня «∞», тобто стовпці 3, 5, 6, 7, 8 і рядки 3, 5, 6, 7, 8.

	1	2	4
1	0	9	3
2	9	0	∞
4	3	∞	0

Для визначення найкоротшої відстані (для значень з незаштрихованої області) потрібно вибирати мінімальне значення з поточного значення, і суми значень, що знаходяться в ***j***-му рядку та ***i***-му стовпці заштрихованої області.

$$d^1 24 = min\,[d^0 24;\ d^0 21 + d^0 14] = min[\infty; 9+3] = 12,$$

$$d^1 42 = min\,[d^0 42;\ d^0 41 + d^0 12] = min[\infty; 3+9] = 12,$$

Далі всі зміни потрібно внести в матрицю відстаней і матрицю маршрутів. Для матриці маршрутів вноситься номер кроку в ті комірки, де були внесені зміни в матриці відстаней. Дана зміна в матриці маршрутів означає, що, приміром, з вузла 2 у вузол 4 першим проміжним вузлом на шляху проходження маршруту буде вузол 1 (дане твердження не є остаточним, у процесі виконання алгоритму це значення може ще помінятися).

D^1

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	9	∞	3	∞	∞	∞	∞
2	9	0	2	***[12]***	7	∞	∞	∞
3	∞	2	0	2	4	8	6	∞
4	3	***[12]***	2	0	∞	∞	5	∞
5	∞	7	4	∞	0	10	∞	∞
6	∞	∞	8	∞	10	0	7	∞
7	∞	∞	6	5	∞	7	0	∞
8	∞	∞	∞	∞	9	12	10	0

R^1

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	1	2	3	***[1]***	5	6	7	8
3	1	2	3	4	5	6	7	8
4	1	***[1]***	3	4	5	6	7	8
5	1	2	3	4	5	6	7	8
6	1	2	3	4	5	6	7	8
7	1	2	3	4	5	6	7	8
8	1	2	3	4	5	6	7	8

Крок № 2: j=2.

Викреслюємо стовпці 6, 7, 8 і рядки 6,7,8.

	1	**2**	**3**	**4**	**5**
1	0	9	∞	3	∞
2	9	0	2	12	7
3	∞	2	0	2	4
4	3	12	2	0	∞
5	∞	7	4	∞	0

$d^2 13 = \min [d^1 13; d^1 12 + d^1 23] = \min [\infty; 9+2] = 11,$

$d^2 15 = \min [d^1 15; d^1 12 + d^1 25] = \min [\infty; 9+7] = 16,$

$d^2 31 = \min [d^1 31; d^1 32 + d^1 21] = \min [\infty; 2+9] = 11,$

$d^2 45 = \min [d^1 45; d^1 42 + d^1 25] = \min [\infty; 12+7] = 19,$

$d^2 51 = \min [d^1 51; d^1 52 + d^1 21] = \min [\infty; 7+9] = 16,$

$d^2 54 = \min [d^1 54; d^1 52 + d^1 24] = \min [\infty; 7+12] = 19,$

Усі зміни вносимо в матриці.

D^2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	9	***[11]***	3	***[16]***	∞	∞	∞
2	9	0	2	12	7	∞	∞	∞
3	***[11]***	2	0	2	4	8	6	∞
4	3	12	2	0	***[19]***	∞	5	∞
5	***[16]***	7	4	***[19]***	0	10	∞	∞
6	∞	∞	8	∞	10	0	7	∞
7	∞	∞	6	5	∞	7	0	∞
8	∞	∞	∞	∞	9	12	10	0

R^2

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	2	***[2]***	4	***[2]***	6	7	8
2	1	2	3	1	5	6	7	8
3	***[2]***	2	3	4	5	6	7	8
4	1	1	3	4	***[2]***	6	7	8
5	***[2]***	2	3	***[2]***	5	6	7	8
6	1	2	3	4	5	6	7	8
7	1	2	3	4	5	6	7	8
8	1	2	3	4	5	6	7	8

Кроки №№ 3-8: j = 3,…,8...

З 6 ітерації матриця ***D*** не міняється, отже, $\boldsymbol{D^5}$ і $\boldsymbol{R^5}$ – оптимальні.

$\boldsymbol{D^5}$

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	7	5	3	9	13	8	∞
2	7	0	2	4	6	10	8	∞
3	5	2	0	2	4	8	6	∞
4	3	4	2	0	6	10	5	∞
5	9	6	4	6	0	10	10	∞
6	13	10	8	10	10	0	7	∞
7	8	8	6	5	10	7	0	∞
8	18	15	13	15	9	12	10	0

$\boldsymbol{R^5}$

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	4	4	4	4	4	4	8
2	4	2	3	3	3	3	3	8
3	4	2	3	4	5	6	7	8
4	1	3	3	4	3	3	7	8
5	3	3	3	3	5	6	3	8
6	4	3	3	3	5	6	7	8
7	4	3	3	4	3	6	7	8
8	5	5	5	5	5	6	7	8

Після одержання оптимальних матриць відстаней і маршрутів можна визначити найкоротшу відстань з одного вузла в інший і відповідний маршрут. Приміром, потрібно знайти найкоротший шлях між вузлами ***1-5***. По матриці D^5 знаходиться найкоротша відстань: $\boldsymbol{D_{15}^5 = 9}$. По матриці R^5 знаходиться маршрут проходження: в комірці ***1-5*** значення 4 ($\boldsymbol{R_{15}^5 = 4}$), тобто наступним після 1-го вузла буде 4-й, далі потрібно подивитися комірку 4-5 ($\boldsymbol{R_{45}^5 = 3}$), тобто після 4-го вузла йде 3-й.

Далі $\boldsymbol{R_{35}^5 = 5}$. У підсумку виходить маршрут: ***1-4-3-5***.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ .

1. Запустіть NET ANALYST.
2. Створіть нову мережу, та введіть дані відповідно до вашого варіанту.
3. Збережіть дані вашого графа на диску.
4. Виберіть пункт «**Floyd algorithm**», розгляньте кожний крок рішення і фінальні таблиці.
5. За матрицею маршрутів визначіть найкоротші шляхи між усіма парами вузлів мережі.
6. Здійсніть модифікацію графу згідно вказівці викладача.
7. Отримайте фінальну таблицю.
8. Самостійно розрахуйте 3 кроки алгоритму і порівняйте результати з результатами, отриманими з допомогою NET ANALYST.

3. ЗМІСТ ЗВІТУ.

1. Мета роботи.
2. Граф мережі та матриця вартостей згідно варіанту.
3. Покрокове рішення та остаточні таблиці, отримані за допомогою NET ANALYST.
4. Найкоротші шляхи між усіма парами вузлів мережі, визначені за матрицею маршрутів.
5. Зміни, проведені у вашому графі, та фінальна таблиця для отриманого графа.
6. Самостійний розрахунок найкоротшого ланцюга для будь-якої пари вузлів (перши три кроки).
7. Висновки.

4. КОНТРОЛЬНІ ПИТАННЯ

1. В чому полягає відмінність алгоритму Дейкстри та алгоритму Флойда?
2. Навести правило порівняння та зміни поточної оцінки найкоротшого шляху між парою вузлів мережі.
3. За скільки ітерацій може бути вирішена задача побудови багатополюсного найкоротшого ланцюга за алгоритмом Флойда?
4. Які матриці будуються при використанні алгоритму Флойда? Як і коли змінюються їх значення?

ЛАБОРАТОРНА РОБОТА №3

АЛГОРИТМИ СИНТЕЗУ МІНІМАЛЬНИХ ОДНОЗВ'ЯЗНИХ МЕРЕЖ

***Мета**: Засвоїти алгоритм Прима синтезу мінімального остовного дерева та алгоритм синтезу мінімальної зірки.*

1. ОСНОВНІ ПОЛОЖЕННЯ

1.1. Постановка задачі синтезу мінімального остовного дерева

Зв'язністю мережі називається мінімальна кількість різних шляхів, що поєднують 2 будь-які вузли мережі. Зв'язність мережі дорівнює мінімальному рангу вузлів мережі. (***Ранг вузла*** – кількість поєднаних з вузлом дуг)

Назвемо ***деревом*** [3] зв'язану множину неорієнтованих дуг, що не містить циклів. Якщо задана множина з **m** вузлів, з'єднаних неорієнтованими дугами, то для побудови дерева необхідно виділити підмножину, складену з **(m-1)** дуг. Іншими словами, кожний вузол дерева з'єднаний з іншим вузлом тільки одним шляхом. **Отже зв'язність дерева дорівнює 1**.

Розглянемо мережу, що містить **n** вузлів, сукупність яких утворює множину **S**. ***Остовним деревом*** називається зв'язна множина, складена з **(n-1)** дуги та **n** вузлів. З будь-якої підмножини множини **S** може бути утворене дерево, що, однак, не є остовним, якщо включає не всі вузли мережі.

Припустимо, що кожній дузі, що з'єднує вузли **i** та **j** з множини **S**, приписане число $\mathbf{C_{ij}}$, що називається відстанню або вартістю дуги. ***Найкоротшим остовом*** називається такий остов мережі, у якого сума вартостей $\mathbf{C_{ij}}$ всіх його дуг мінімальна.

Існують декілька алгоритмів для розв'язання задачі побудови найкоротшого остовного дерева [2], зупинимось на алгоритмі, відомому під назвою "Алгоритм Прима".

Алгоритм починає роботу з вибору довільного вузла мережі та найкоротшої дуги з множини дуг, що з'єднують цей вузол з іншими вузлами. З'єднаємо два вузла вибраною дугою. Оберемо найближчий до будь-якого з цих вузлів третій вузол. Додамо цей вузол та відповідну дугу до мережі. Продовжуємо даний процес до тих пір, доки всі вузли не будуть з'єднані між собою. Алгоритм, оснований на поглинанні найкоротших дуг, може бути описаний наступним чином [3].

1.2. Алгоритм Прима

1. Розділимо усі вузли вхідної мережі **S** на дві множини: **Σ**- множина з'єднаних вузлів; **Ω** - множина не з'єднаних вузлів. Спочатку всі вузли належать множині **Ω**;
2. Вибираємо довільний вузол з множини **Ω** і з'єднуємо його з найближчим сусіднім вузлом; ці два вузла переносимо з множини **Ω** до множини **Σ**;
3. Серед всіх дуг, що з'єднують вузли з множини **Σ** з вузлами з множини **Ω**, вибираємо найкоротшу дугу. Кінцевий вузол цієї дуги, що лежить в множині **Ω** переносимо в множину **Σ**;
4. Виконуємо крок 3 до тих пір, доки всі вузли не будуть належати множині **Σ**.
5. Сумарна довжина усіх вибраних дуг дає ***L*** – довжину мінімального остовного дерева.

Приклад 3.1.

Побудувати найкоротше остовне дерево для мережі, граф якої приведено на рис. 3.1.

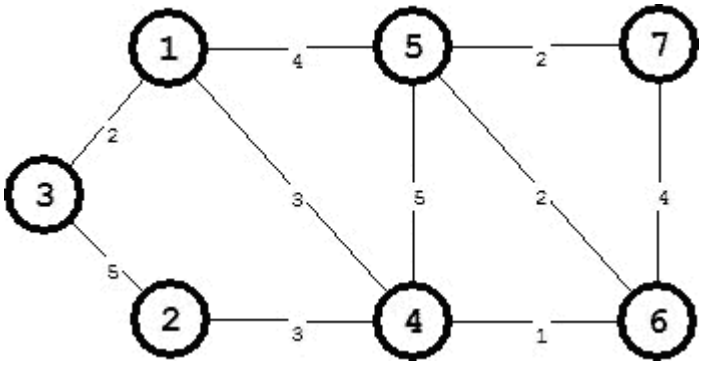


Рисунок 3.1. – Граф мережі

Запишемо по наведеному графу матрицю вартостей (табл.3.1).

Табл. 3.1

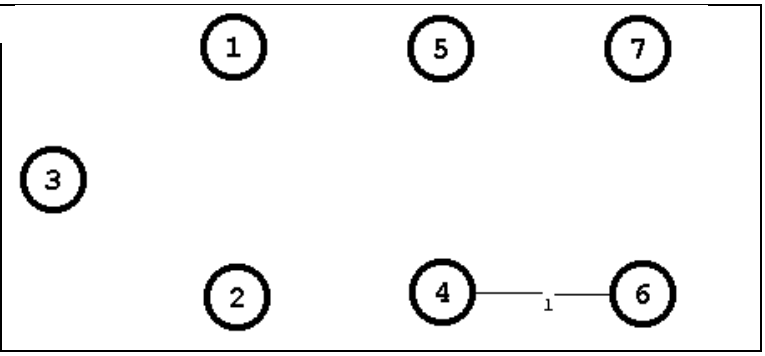
Матриця вартостей

C	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	∞	2	3	4	∞	∞
2	∞	0	5	3	∞	∞	∞
3	2	5	0	∞	∞	∞	∞
4	3	3	∞	0	5	1	∞
5	4	∞	∞	5	0	2	2
6	∞	∞	∞	1	2	0	4
7	∞	∞	∞	∞	2	4	0

Вибираємо найменше значення $\mathbf{C_{ij}}$ (в прикладі [1]), відмічаємо рядки **i** та **j (**в прикладі знаком *), а стовпці **i** та **j** викреслюємо і в подільшому не розглядаємо (в прикладі беремо в дужки (4), (6)). Серед значень $\mathbf{C_{ij}}$, що входять в позначені рядки, вибираємо найменше, викреслюємо стовпець, в якому це найменше значення знаходиться, відмічаємо рядок з номером останнього викресленого стовпця і т.д. Алгоритм закінчує роботу, коли позначені всі рядки та викреслено всі стовпці.

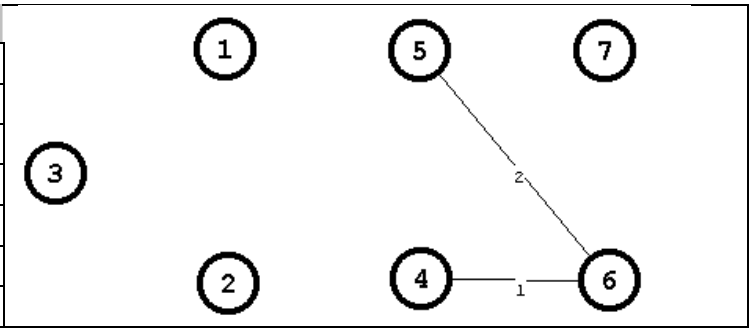
1. Σ={4,6}

C	1	2	3	(4)	5	(6)	7	
1	0	∞	2	3	4	∞	∞	
2	∞	0	5	3	∞	∞	∞	
3	2	5	0	∞	∞	∞	∞	
4	3	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	2	
6	∞	∞	∞	1	2	0	4	*
7	∞	∞	∞	∞	2	4	0	



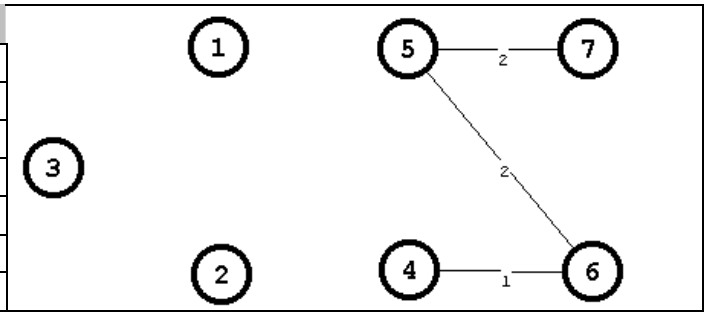
2. Σ={4,5,6}

C	1	2	3	(4)	(5)	(6)	7	
1	0	∞	2	3	4	∞	∞	
2	∞	0	5	3	∞	∞	∞	
3	2	5	0	∞	∞	∞	∞	
4	3	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	2	*
6	∞	∞	∞	1	[2]	0	4	*
7	∞	∞	∞	∞	2	4	0	



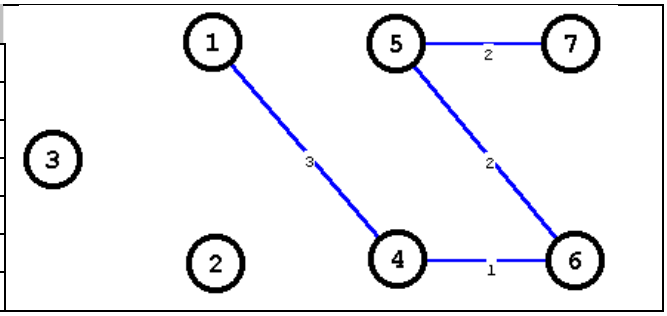
3. Σ={4,5,6,7}

C	1	2	3	(4)	(5)	(6)	(7)	
1	0	∞	2	3	4	∞	∞	
2	∞	0	5	3	∞	∞	∞	
3	2	5	0	∞	∞	∞	∞	
4	3	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	[2]	*
6	∞	∞	∞	1	[2]	0	4	*
7	∞	∞	∞	∞	2	4	0	*



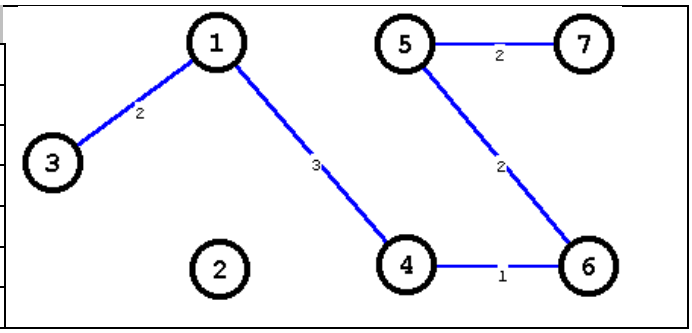
4. Σ={1,4,5,6,7}

C	(1)	2	3	(4)	(5)	(6)	(7)	
1	0	∞	2	3	4	∞	∞	*
2	∞	0	5	3	∞	∞	∞	
3	2	5	0	∞	∞	∞	∞	
4	[3]	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	[2]	*
6	∞	∞	∞	1	[2]	0	4	*
7	∞	∞	∞	∞	2	4	0	*



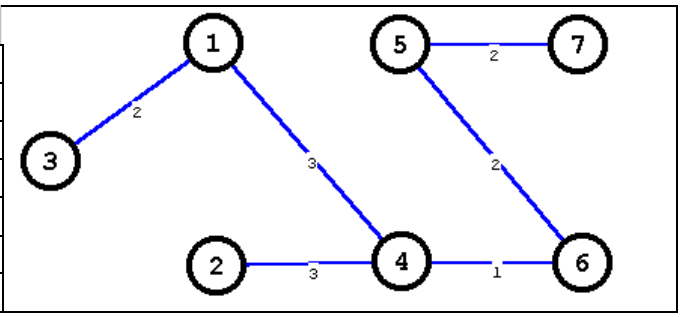
5. Σ={1,3,4,5,6,7}

C	(1)	2	(3)	(4)	(5)	(6)	(7)	
1	0	∞	[2]	3	4	∞	∞	*
2	∞	0	5	3	∞	∞	∞	
3	[2]	5	0	∞	∞	∞	∞	*
4	[3]	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	[2]	*
6	∞	∞	∞	1	[2]	0	4	*
7	∞	∞	∞	∞	2	4	0	*



6. Σ={1,2,3,4,5,6,7}

C	(1)	(2)	(3)	(4)	(5)	(6)	(7)	
1	0	∞	[2]	3	4	∞	∞	*
2	∞	0	5	3	∞	∞	∞	*
3	[2]	5	0	∞	∞	∞	∞	*
4	[3]	3	∞	0	5	[1]	∞	*
5	4	∞	∞	5	0	2	[2]	*
6	∞	∞	∞	1	[2]	0	4	*
7	∞	∞	∞	∞	2	4	0	*



Довжина найкоротшого остовного дерева:

L=1+2+2+3+2+3=13.

1.3. Постановка задачі та алгоритм синтезу мінімальної зірки

Оптимальна (мінімальна) зірка є окремим випадком остовного дерева, коли один вузол – центр зірки – безпосередньо зв'язан з кожним іншим вузлом. Для визначення центра мінімальної зірки, треба для кожного вузла обчислити сумарну довжину всіх дуг до інших вузлів, та вибрати мінімальне значення.

Приклад 3.2.

Для мережі (рис. 3.2) побудувати мінімальну зірку.

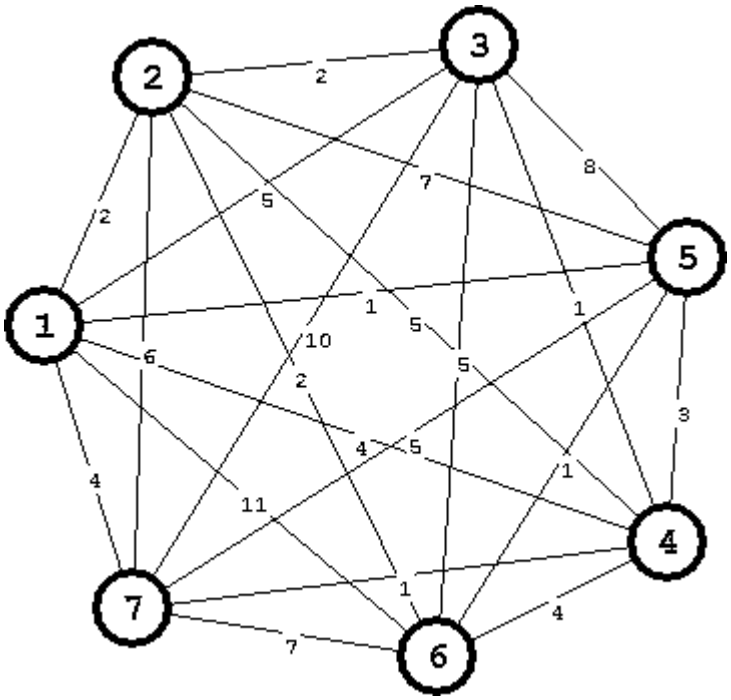


Рисунок 3.2 – Граф мережі

Додамо в матрицю вартостей рядок з сумою довжин дуг – отримаємо таблицю рішення (табл.3.2)

Табл. 3.2

Таблиця рішення

C	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	0	2	5	4	1	11	4
2	2	0	2	5	7	2	6
3	5	2	0	1	8	5	10
4	4	5	1	0	3	4	1
5	1	7	8	3	0	1	5
6	11	2	5	4	1	0	7
7	4	6	10	1	5	7	0
Сума	27	24	31	**18**	25	30	33

Таким чином оптимальною буде зірка з центром в вузлі 4.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ.

1. Запустіть NET ANALYST.
2. Створіть нову мережу, та введіть дані відповідно до вашого варіанту.
3. Збережіть дані вашого графа на диску.
4. Виберіть пункт «**Prima algorithm**», виконайте покрокове рішення і отримайте фінальну таблицю.
5. Самостійно розрахуйте найкоротше остовне дерево і порівняйте результати з результатами, отриманими за допомогою NET ANALYST.
6. Доповніть задану мережу довільними значеннями до повнозв’язної.
7. Побудуйте для неї мінімальне дерево та мінімальну зірку за допомогою NET ANALYST.
8. Самостійно розрахуйте мінімальне дерево та мінімальну зірку.
9. Порівняйте отримані результати – довжину дерева та довжину зірки.

3. ЗМІСТ ЗВІТУ.

1. Мета роботи.
2. Граф мережі та матриця вартостей згідно варіанту.
3. Покрокове розв’язання алгоритму Прима та остаточна таблиця, отримані за допомогою NET ANALYST.
4. Самостійний розрахунок мінімального дерева для вхідного графа.
5. Матриця вартостей для модифікованої (повнозв’язної мережі).
6. Мінімальне дерево та мінімальна зірка, , отримані за допомогою NET ANALYST.
7. Результати самостійниз розрахунків.
8. Висновки.

4. КОНТРОЛЬНІ ПИТАННЯ

1. Дати визначення термінам: зв’язність мережі, ранг вузла. Як пов’язані ці поняття?
2. Дати визначення дерева, остовного дерева, найкоротшого остова, зірки. Чому дорівнює зв’язність кожної названої мережі?
3. Які недоліки та переваги мають однозв’язні мережі?
4. За скільки ітерацій може бути вирішена задача побудови найкоротшого остова?
5. Де використовується алгоритм Прима?
6. На якомі рівні мережевої ієрархії використовується топологія зірка?
7. Яка мережева топологія є найбільш економічною?

ЛАБОРАТОРНА РОБОТА № 4

ЗАДАЧА ПРО МАКСИМАЛЬНИЙ ПОТІК. АЛГОРИТМ ФОРДА-ФАЛКЕРСОНА

Мета: *визначити максимальний потік у мережі з обмеженою пропускною здатністю гілок. Перевірити теорему про мінімальний розріз.*

1. ОСНОВНІ ПОЛОЖЕННЯ

1.1 Постановка задачі

Розглянемо граф (див. л.р. № 1) ***G={N, A},*** де ***N*** - множина вузлів, ***A*** - множина дуг. Граф ***G*** має одне джерело ***s*** і один стік ***t***. Дуги ***(i,j)*** мають обмежену пропускну здатність U_{ij}**.**

$$f_{ij} \le U_{ij} \tag{4.1}$$

де f_{ij} - можливий потік через дугу ***(i,j)***

U_{ij} - задане значення пропускної здатності для дуги ***(i,j)***.

Величина потоку з джерела не обмежена; у кожному проміжному вузлі виконується умова збереження потоку.

$$\sum_i f_{ij} - \sum_i f_{ji} = 0, \quad i \ne s, \quad i \ne t, \tag{4.2}$$

Задача полягає в визначенні таких дугових потоків, щоб загальний потік з джерела ***s*** в стік ***t*** був максимальним:

$$\sum_{i \ne j} f_{ij} \to \max \tag{4.3}$$

Задача вирішується з допомогою ітераційної процедури розстановки позначок вузлів. Кожна позначка вказує величину потоку та його джерело й може бути як позитивною, так і негативною.

g_i , g_j - кількість одиниць потоку

Позитивна позначка збільшує потік по дузі $\mathbf{b_{ij}}$, але так, щоб сумарний потік не перевищив U_{ij} ; негативна позначка зменшує потік по дузі, але так, щоб він не став негативним.

Вузли помічаються послідовно від 1 до **n**.

Якщо вузол a_i позначений, то a_j можна позначити з нього по прямій дузі на величину не більшу, ніж $g_i = \min\lfloor g_i; U_{ij} - f_{ij} \rfloor$, а по зворотній дузі на величину $g_j = \min\lfloor g_i; f_{ij} \rfloor$.

На кожної ітерації джерело помічається позначкою $[\infty;-]$ та послідовно розставляються позначки інших вузлів по одному з можливих шляхів з джерела до стоку. Коли доходить черга до стоку, йому приписується мітка $[+U_n;k]$. Тобто потік в мережі можна збільшити на U_n; після чого всім дугам розглянутого шляху приписується $f_{ij} = U_n$. Після цього всі мітки стираються і переходять до наступної ітерації.

Алгоритм закінчує роботу, якщо жоден вузол мережі вже не може бути позначений з джерела.

1.2 Теорема про максимальний потік та мінімальний розріз.

Нехай заданий граф ***G={N, A}***. Розіб'ємо множину ***N*** на дві підмножини, які не пересікаються $\{N\} = \{N_C\} \cup \{\overline{N_C}\}$, так щоб джерело та стік графа не знаходились в одній підмножині. Всі дуги, що з'єднують ці дві підмножини, утворюють множину $\{A_C\}$.

Множина дуг, яку необхідно видалити з графа, щоб порушити його зв'язність, називається ***перетином***. Тобто $\{A_C\}$ є перетин.

Кількість дуг в перетині називається ***рангом перетину***, а сумарна пропускна спроможність всіх дуг перетину - ***розрізом***.

Очевидним є той факт, що потік з джерела в стік буде обмежений зверху розрізом. Більш того, має місце наступна теорема: максимальний потік з джерела в стік дорівнює мінімальному розрізу, що розділяє джерело та стік.

Дана теорема корисна для аналізу "вузьких місць" у мережі. Тобто, якщо збільшити пропускну здатність будь-якої дуги з мінімального перетину, то можна збільшити загальний потік в мережі.

Приклад 4. Для мережі з обмеженими пропускними здатностями, що задана графом (рис.4.1), визначити максимальний потік з джерела до стоку.

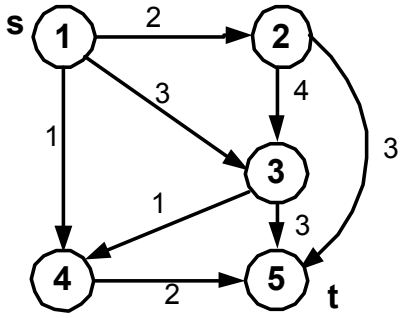


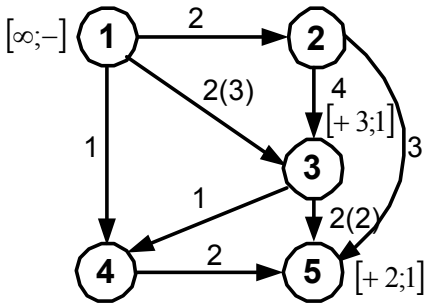
Рисунок 4.1 – Граф мережі з обмеженими пропускними здатностями

1 ітерація. Шлях: 1-3-5

Позначки: $a_1[\infty;-], a_3[+3;1], a_5[+2;3]$

Потоки в дугах: $f_{12}=2; f_{35}=2$

Загальний потік збільшився на ΔV =2



2 ітерація. Шлях: 1-2-3-4-5

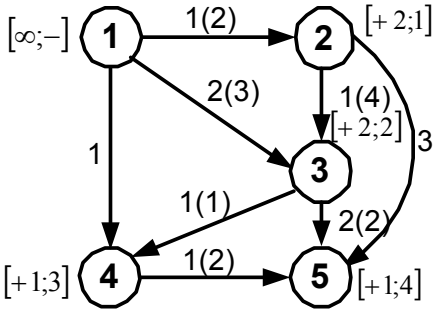
Позначки:

$a_1[\infty;-], a_2[+2;1], a_3[+2;2], a_4[+1;3], a_5[+1;4]$

Потоки в дугах:

$f_{12}=1; f_{23}=1; f_{34}=1; f_{45}=1$

Загальний потік збільшився на ΔV=1

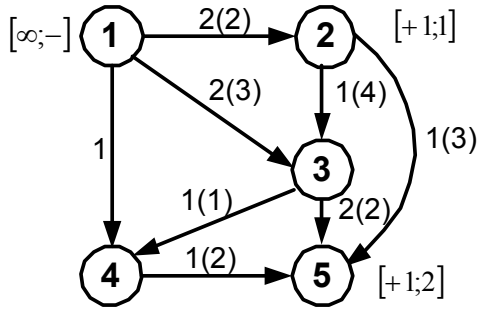


3 ітерація. Шлях: 1-2-5

Позначки: $a_1[\infty;-], a_2[+1;1], a_5[+1;2]$

Потоки в дугах: $f_{12}=2; f_{25}=1$

Загальний потік збільшився на $\Delta V=1$



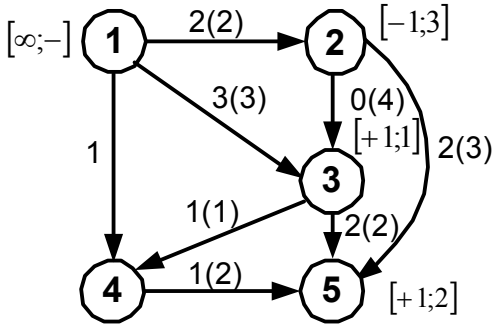
4 ітерація. Шлях: 1-3-2-5

Позначки:

$a_1[\infty;-], a_3[+1;1], a_2[-1;3], a_5[+1;2]$

Потоки в дугах: $f_{13}=3; f_{23}=0; f_{25}=2$

Загальний потік збільшився на $\Delta V=1$

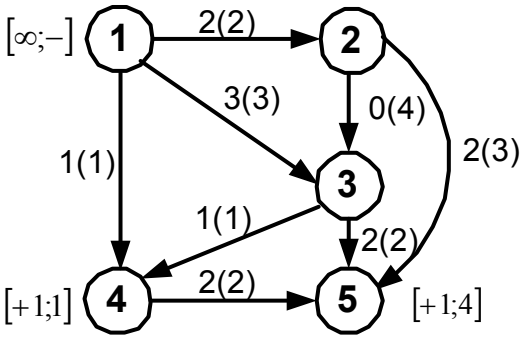


5 ітерація. Шлях: 1-4-5

Позначки: $a_1[\infty;-], a_4[+1;1], a_5[+1;4]$

Потоки в гілках: $f_{14}=1; f_{45}=2$

Загальний потік збільшився на $\Delta V=1$



6 ітерація. Ще будь-який шлях вибрати неможливо.

Максимальний потік в мережі дорівнює сумі збільшень потоків ΔV на кожній ітерації, тобто V=6. В даному прикладі можна визначити максимальний потік, використовуючи теорему про максимальний потік та мінімальний розріз, яким є розріз b_{12}, b_{13}, b_{14}.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ.

1. Запустіть NET ANALYST.
2. Створіть нову мережу, та введіть данні згідно варіанту.
3. Збережіть дані вашого графа на диску.
4. Виберіть пункт «**Ford-Fulkerson algorithm**», виконайте покрокове рішення і отримайте фінальну таблицю.
5. Знайдіть мінімальний розріз та порівняйте його значення з величиною максимального потоку.
6. Зробіть модифікацію графу, збільшуючи пропускну здатність дуг, що належать до мінімального перетину, доки мінімальний перетин не зміниться.
7. Визначити, при яких значеннях пропускної здатності дуг мінімальний перетин змінюється.
8. Самостійно розрахуйте максимальний потік з першого вузла в останній і порівняйте результати з результатами, отриманими за допомогою NET ANALYST.

3. ЗМІСТ ЗВІТУ.

1. Мета роботи.
2. Граф мережі та матриця пропускних здатностей згідно варіанту.
3. Покрокове розв’язання та фінальна таблиця, отримані за допомогою NET ANALYST.
4. Мінімальний розріз вхідного графа.
5. Зміни, проведені у вашому графі, та фінальна таблиця для отриманого графа.
6. Аналіз, наскільки можна збільшити пропускну здатність вхідного графа до зміни мінімального перетину
7. Самостійний розрахунок вхідного графа.
8. Висновки.

4. КОНТРОЛЬНІ ПИТАННЯ

1. Навести умову збереження потоку в транзитних вузлах.
2. Записати задачу знаходження максимального потоку.
3. Сформулювати теорему про максимальний потік та мінімальний розріз
4. Сфера використання теореми про максимальний потік та мінімальний розріз.
5. Навести правило визначення дугових трафіків в мережі.

ЛАБОРАТОРНА РОБОТА № 5

СИНТЕЗ БАГАТОЗВ'ЯЗАНОЇ МЕРЕЖІ. ЗАДАЧА КОМІВОЯЖЕРА.

Мета: *вивчити алгоритм гілок та меж (задачу комівояжера) для синтезу мінімального кільця. Провести синтез багатозв'язаної мережі з обмеженням на число переприйомів. Розрахувати таблицю трафіків*

1. ОСНОВНІ ПОЛОЖЕННЯ.

1.1 Задача комівояжера.

Кільцем називається мережева топологія, при якій кожний вузол мережі поєднан з двома іншими вузлами, тобто для будь-якої пари вузлів існує 2 різних маршрута. Таким чином, зв'язність кільця дорівнює 2. Для заданної матриці відстаней між вузлами може бути побудовано кілька кілець, серед яких одне або декілька мають мінімальну довжину, тобто є ***оптимальними***.

Задача комівояжера застосовується для ***синтезу оптимального кільця*** і може бути сформульована наступним чином. Комівояжер повинен виїхати з заданого міста, побувати у кожному з інших **n-1** міст рівно один раз та повернутися у вихідне місто. Задача полягає в визначенні послідовності об'їзду міст, при якій комівояжеру треба проїхати найменшу сумарну відстань. При цьому припускається, що відстань для кожної пари міст відома. Замість довжини шляху можна розглядати будь-які інші критерії ефективності, такі, як вартість, час пересування і т.д.

Неважко помітити, що всього існує **(n-1)!** можливих маршрутів, серед яких один або декілька оптимальні. В більшості випадків можна вважати, що відстань між містами ***i*** та ***j*** є симетрична, тобто відстань від міста ***i*** до міста ***j*** рівна відстані від міста ***j*** до міста ***i***. Однак для алгоритму, що подано нижче, дане припущення не обов'язкове.

Приведений алгоритм називається алгоритмом гілок та меж. Вперше цей алгоритм був розроблений Літтлом. Алгоритм працює наступним чином. Спочатку визначається деяке допустиме рішення, після чого множина всіх маршрутів, що залишалися розбивається на все більш дрібні підмножини. На кожному кроку розбивання легко обчислюється нижня границя довжини поточного найкращого маршруту. З допомогою знайдених меж здійснюється подальше розбивання підмножини допустимих маршрутів і в кінцевому підсумку визначається оптимальний маршрут. Якщо знаходиться маршрут, довжина якого не перевищує найменшої нижньої границі всіх інших маршрутів, то дане проміжне рішення стає «найкращим» допустимим рішенням. Основними процедурами алгоритму є обчислення нижніх меж довжин маршрутів, виділення субоптимальних рішень та розгалуження з метою отримання нових (більш коротких) маршрутів.

Підмножини множини всіх маршрутів можна розглядати як вузли дерева, а процес розбивання - як розгалуження дерева. Тому даний засіб називається засобом пошуку по дереву рішень, або алгоритмом гілок і меж.

1.2. Постановка задачі синтезу багатозв’язної мережі.

Розглянуті вище мережеві топології (дерево, зірка, кільце) є базовими, їх оптимальний синтез здійснюється на першому етапі проектування телекомунікаційних мереж. На другому етапі в базову мережу додаються лінії зв’язку для виконання певних обмежень (наприклад, для збільшення надійності або зменшення кількості транзитних вузлів, тощо). Отримані мережі є ***багатозв’язними*** (кількість маршрутів між будь-якою парою вузлів перевищує 2). Додаються, як правило, дуги мінімальної вартості, що не увійшли в базову мережу.

Крім визначення топології, при проектуванні телекомунікаційної мережі слід висунути вимоги до її обладнання, для чого треба визначити необхідну пропускну здатність каналів зв’язку. Це здійснюється через розрахунок ***матриці***

міжвузлових трафіків мережі. Основою для такого розрахунку є ***матриця інформаційних тяжінь*** – потреб у зв'язку для будь-якої пари вузлів.

Отже задача синтезу багатозв'язної мережі може бути сформульована наступним чином:

для множини вузлів, що задана матрицею відстаней та матрицею інформаційних тяжінь, побудувати телекомунікаційну мережу, що забезпечує необхідний трафік та відповідає висунутим умовам та обмеженням.

Розглянемо послідовність дій для синтезу багатозв'язної мережі з обмеженням на число транзитних вузлів між будь-якою парою вузлів мережі (на кількість "переприйомів").

1.3. Алгоритм синтезу багатозв'язної мережі з обмеженням на число транзитних вузлів

Рішення поставленої задачі може здійснюватися за наступним алгоритмом:

1. Побудувати оптимальну мережу базової топології (кільце, дерево, зірку).
2. Перевірити виконання обмеження для кожної пари вузлів. Для чого:
    a) за алгоритмом Флойда визначити найкоротші шляхи між будь-якої пари вузлів;
    b) підрахувати кількість переприйомів в кожному k-му маршруті n_k;
    c) якщо $n_k > n_{доп}$ (умова не отримується) перейти до кроку 3, інакше – до кроку 4.
3. Внести зміни в базову мережу, для чого:
    a) додати в мережу мінімальну з невикористанних дуг;
    b) виконати алгоритм Флойда та підрахувати кількість маршрутів, для чких умова не дотримана)
    c) якщо додана дуга зменшує кількість таких маршрутів, залишити її в рішенні, інакше – відкинути та перейти до іншої;
    d) крок повторюється доки усі маршрути не будуть задовольняти умову $n_k \leq n_{доп}$

4. Розрахувати матрицю трафіків f_{ij}
    a) обнулити значення елементів матриці: $f_{ij} = 0$;
    b) розглянути усі маршрути, отримані на кроці 3, як послідовність дуг: $(i \to j) = (i \to k),(k \to l),\ldots,(m \to j)$;
    c) для кожної дуги $(k \to l)$, що входить в розглянутий маршрут $(i \to j)$ збільшити значення трафіка на величину інформаційного тяжіння між кінцевими вузлами маршрута: $f_{kl} = f_{kl} + h_{ij}$
5. Обрати пропускну здатність U_{ij} каналів зв'язку мережі як найменше з нормованих (типових) значень, що відповідає умові:
    a) для дуплексних каналів: $U_{ij} \geq f_{ij} + f_{ji}$
    b) для напівдуплексних каналів: $U_{ij} \geq max\ (f_{ij}, f_{ji})$
    c) для симплексних каналів: $U_{ij} \geq f_{ij}$

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ.

1. Запустіть NET ANALYST.
2. Створіть нову повнозв'язну мережу та введіть дані відповідно до варіанту, довільно додаючи значення в матрицю відстаней.
3. Збережіть дані вашого графа на диску.
4. Виберіть пункт «**Salesman's algorithm**» для синтезу оптимального кільця, розгляньте кожний крок рішення і фінальну таблицю.
5. На основі отриманого кільця побудуйте багатозв'язну мережу, вибираючи мінімальні ребра, що не ввійшли в кільце, так щоб число переприйомів в кожному шляху не перевищувало 2.
6. Задайте матрицю інформаційних тяжінь.
7. Розрахуйте матрицю трафіків для мережі, отриманої в п.5.
8. Виберіть пропускні здатності каналів зв'язку для розрахованих трафіків.

3. ЗМІСТ ЗВІТУ.

1. Мета роботи.
2. Матриця вартостей та матриця інформаційних тяжінь.
3. Фінальна таблиця та оптимальне кільце, отримані за допомогою NET ANALYST.
4. Синтез багатозв’язної мережі.
5. Розрахунок матриці трафіків.
6. Матриця пропускних здатностей каналів зв’язку.
7. Висновки

4. КОНТРОЛЬНІ ПИТАННЯ

1. Дати визначення оптимальтного кільця. Чому дорівнює зв’язність кільця?
2. Навести етапи синтезу багатозв’язної мережі.
3. Які топології використовуються в якості базових при синтезі телекомунікаційних мереж?
4. Як розраховуються трафіки між вузлами в мережі?
5. Пояснити вибір необхідних пропускних здатностей каналів зв’язку залежно від типу каналу.

ЛАБОРАТОРНА РОБОТА № 6
РОЗРАХУНОК МІЖВУЗЛОВИХ НАВАНТАЖЕНЬ (МАТРИЦІ ІНФОРМАЦІЙНИХ ТЯЖІНЬ)

Мета:

*1). Значення вхідного (α_{inc}) і вихідного (α_{out}) трафіку для **k**-го вузлу, що задані у таблиці варіантів, розподілити між рештою вузлів транспортної мережі.*

2). Засвоїти методику побудування матриці інформаційних тяжінь.

1. ОСНОВНІ ПОЛОЖЕННЯ

1.1. Види вузлових навантажень

Схему вузла із вказаними напрямками вхідного, вихідного і внутрішнього трафіку можна показати схемою, що представлена на рисунку 1.1.

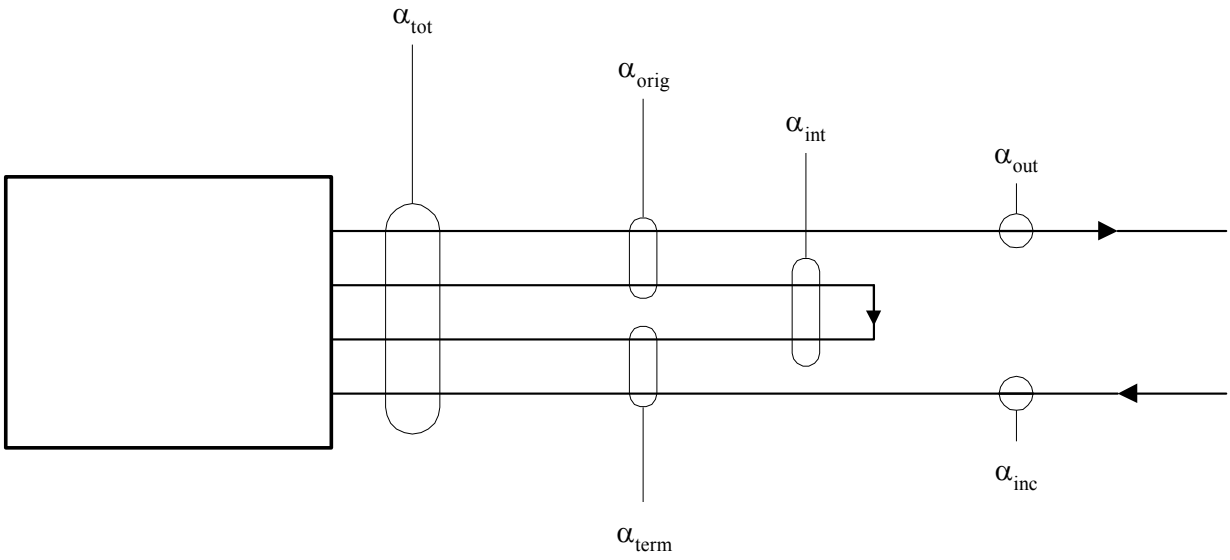


Рисунок 1.1 – Розподілення навантаження у вузлі за напрямками

α_{out} – вихідний трафік, (зовнішній), Ерл;

α_{inc} – вхідний трафік, (зовнішній), Ерл;

α_{int} – внутрішній трафік, Ерл;

α_{tot} – загальний трафік, Ерл;

α_{orig} – трафік, що генерується (зовнішній і внутрішній), Ерл;

α_{term} – трафік, що приймається (зовнішній і внутрішній), Ерл.

$$\left.\begin{aligned} TCR &= \alpha_{tot} / N_{sub} \\ PO &= \frac{\alpha_{orig}}{\alpha_{tot}} \\ PI &= \frac{\alpha_{\text{int}}}{\alpha_{tot}} \end{aligned}\right\} \qquad (1.1)$$

Використовуючи (1.1), співвідношення між видами навантаження і числом абонентів вузла мають наступний вигляд (1.2)

$$\begin{aligned} \alpha_{tot} &= TCR * N_{sub} \\ \alpha_{orig} &= TCR \cdot PO * N_{sub} \\ \alpha_{term} &= TCR \cdot (1 - PO) * N_{sub} \\ \alpha_{out} &= TCR \cdot (PO - PI/2) * N_{sub} \\ \alpha_{inc} &= TCR \cdot (1 - PO - PI/2) * N_{sub} \\ \alpha_{int} &= TCR \cdot PI * N_{sub} \end{aligned} \qquad (1.2)$$

Для кожного вузла мережі розрахунковим навантаженням є вхідний α_{inc} (перші 10 значень у таблиці варіантів) і вихідний зовнішній трафіки α_{out} (останні 10 значень у таблиці варіантів).

1.2. Перевірка балансу вихідних і вхідних вузлових навантажень

Оскільки вихідний і вхідний трафіки прогнозуються незалежно один від одного, то необхідно перевірити загальний баланс навантажень на мережу, тобто, необхідно виконання умови

$$\sum_k \alpha_{out} = \sum_k \alpha_{inc} \qquad (1.3)$$

Якщо дисбаланс значний (більш 5%), то у рівнянні (1.3) задані значення вхідних вузлових навантажень α_{inc} необхідно помножити на поправочний коефіцієнт R

$$R = \sum_k \alpha_{out} \Big/ \sum_k \alpha_{inc} . \qquad (1.4)$$

Отримані значення вхідних і вихідних вузлових навантажень зводяться у таблицю 1.1.

Таблиця 1.1 – Навантаження вузлів транспортної мережі

Вузол	α_{inc}, Ерл	α_{out}, Ерл
1		
…		
N		

2. МЕТОДИКА ПОБУДУВАННЯ МАТРИЦІ ІНФОРМАЦІЙНИХ ТЯЖІНЬ

Якщо має місце рівне взаємне тяжіння між абонентами всієї мережі, то навантаження, що виходить від ***i–го*** вузла до ***j–го***, пропорційне частці вхідного трафіку до ***j-го*** вузла відносно сумарного навантаження всієї мережі, тобто

$$\alpha_{ij} = \alpha_{out_i} \frac{\alpha_{inc_j}}{\sum\limits_{\substack{k=1 \\ k \neq j}}^{N} \alpha_{inc_k}} . \qquad (2.1)$$

Розраховані за (2.1) значення міжвузлових навантажень зводяться до таблиці 2.1. При цьому значення, що відповідають елементам головної діагоналі, не заповнюються (внутрішнє навантаження вузла не враховувалось при визначенні вхідних і вихідних навантажень), а до суми знаменника для кожного ***j-го*** стовпця не включається відповідне значення α_{inc_j}, яке є у чисельнику (1.5). При заповненні таблиці 2.1 слід перевіряти відповідність суми міжвузлових тяжінь за рядками і стовпцями вихідним і вхідним навантаженням. У випадку розходжень процедуру заповнення таблиці 2.1 слід виконувати ітеративно – послідовно корегуючи α_{ij} так, щоб виконувалось сходження для стовпців і рядків.

Таблиця 2.1 – Матриця інформаційних тяжінь

Вузол	1	…	k	…	N	α_{out}
1	-					α_{out1}
…		-				
k			-			α_{outk}
…				-		
N					-	α_{outN}
α_{inc}	α_{inc1}	…	α_{inck}	…	α_{incN}	

ЛАБОРАТОРНА РОБОТА № 7
РОЗРАХУНОК ЦИФРОВИХ МУЛЬТИПЛЕКСОРІВ

Мета:

1). Розрахувати пропускну здатність цифрових каналів для забезпечення транспортування цифрового потоку заданої величини.

2). Розрахувати і привести схему мультиплексорів.

1. ЗАВДАННЯ ДО РОБОТИ

1. Вузли B, C, D, E з'вязані через вузол A (рисунок 1.1). Матриця інформаційних тяжінь між ними задана (таблиця 1.1) у цифрових потоках E1 (ІКМ-30, 2Мбіт/с). Вузол A – транзитний, інформаційні потоки не починаються і не закінчуються у ньому.
2. Розрахувати пропускну здатність цифрових каналів BA, CA, DA, EA для забезпечення транспортування цифрового потоку заданної величини. При цьому:
    - слід використовувати наступні рівні ієрархії:
        - E1 = ІКМ-30 = 2 Мбіт/с
        - E2 = ІКМ-120 = 8 Мбіт/с
        - E3 = ІКМ-480 = 34 Мбіт/с
        - E4 = ІКМ-1920 = 140 Мбіт/с
    - пропускна здатність задіяних каналів має бути повністю використана;
    - більш високі рівні ІКМ переважні за низькі;
    - жодна система не повинна закінчуватися у вузлі A;
    - решта вузлів обладнані необхідними мультиплексорами.
3. Розрахувати і привести схему мультиплексорів, що необхідно встановити у вузлі A для забезпечення розрахованих у п.2 цифрових каналів.

Табл.1.1

Між	**Загальне число потоків E1**		
	випадок a)	**випадок b)**	**випадок c)**
B і C	6	7	70
B і D	3	5	25
B і E	7	7	40
C і D	16	17	37
C і E	4	9	13
D і E	4	4	28

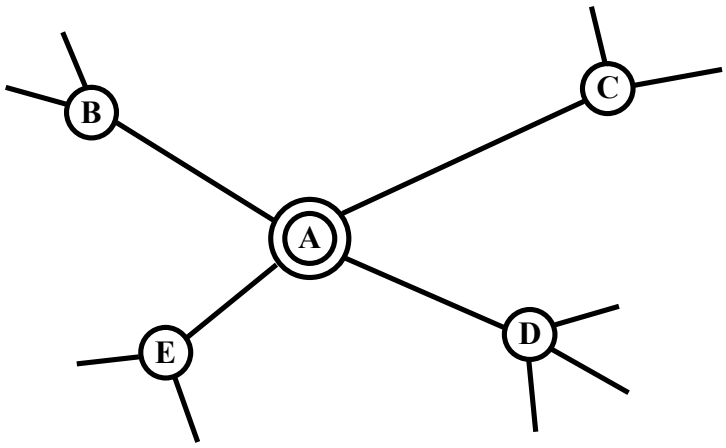


Рисунок 1.1 – Схема мережі

2. РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Процедуру розв’язання розглянемо на прикладі випадку а).

1. Розраховується кількість цифрових потоків між вузлами і заноситься у таблицю 2.1, яка має три типи комірок:
   а) комірка для кожної пари вузлів, що знаходиться на перетині відповідних рядка і стовпця (наприклад, BC) і показує:

- загальне число N потоків E1 між цими вузлами згідно завданню (N=6 для вузлів B і C);
- число каналів різного рівня ієрархії N1, N2, N3, N4, що необхідні для передачі міжвузлового потоку, враховуючи, що переважними є більш високі рівні ієрархії

6 E1 = 1 E2 + 2 E1
180 = 120 + 60

N	N1	6	
	N2		
	N3		1
	N4		2

b) комірка Σ_1 для сумарної кількості цифрових потоків і каналів різного рівня кожного вузла у випадку, коли вузол A не обладнаний мультиплексором (проміжне значення). У цій комірці приводиться сума усіх відповідних комірок першого типу.

c) комірка Σ_2 для сумарної кількості цифрових потоків і каналів різного рівня кожного вузла у випадку, якщо вузол A обладнаний мультиплексором. Тут для загального числа цифрових потоків (воно буде таке сааме, як і у Σ_1) визначається, якого рівня ієрархії канали необхідно використовувати, починаючи з більш високих.

2. Визначається, які цифрові канали можуть проходити через вузол A без мультиплексування/демультиплексування, тобто транзитом і заповнюється таблиця 2.4. Для цього порівнюються значення, отримані в комірках Σ_1 и Σ_2 для кожного вузла.

Для вузла B необхідно забезпечувати мультиплексування при передачі до решти вузлів (оскільки без мультиплексування для нього необхідні 2 канали E2 і 8 каналів E1, а з мультиплексором у вузлі A можна використати 1 канал E3.

Таблиця 2.1

B		C		D		E		Σ_1		Σ_2		
		6		3		7		16		16	1	B
			1				1		2			
			2		3		3		8			
6				16	1	4		26	1	26	1	C
	1						1		2		2	
	2								2		2	
3		16	1			4		23	1	23	1	D
							1		1		1	
	3								3		3	
7		4		4				15		15		E
	1		1		1				3		3	
	3								3		3	
								80		80		Σ
									2		3	
									8		6	
									16		8	

Для вузлів C, D, E отримані значення у відповідних комірках співпадають. Перевіряємо, які канали не зв’язані з вузлом B.

Для вузла C це:

- канал E3 зв’язує вузли C і D, тобто є транзитним. Записуємо «C/D» до рядку «Транзитні системи» у комірку для ЦК 3-го рівня (ІКМ-480) у таблицю 2.4. Дані таблиці 2.1 переносимо до проміжної таблиці 2.2, залишаючи тільки верхню діагональ. Після чого у таблиці 2.2 обводимо усі значення, що пов’язані з цим каналом: це 1 у комірках для N3 (C, D), (C, Σ_1), (C, Σ_2), (D, Σ_1), (D, Σ_2) (усього 5 разів). Поруч записується кількість каналів відповідного рівня для даного вузла (або пари вузлів), що залишаються для об’єднання мультиплексором. Ця

кількість знаходиться як різниця між розрахованим числом каналів і числом транзитних каналів. Для випадку CD: 1-1=0, тому поруч нічого не дописується.

- 2 канали E2. Один зв’язує вузли C і B, тобто не може бути транзитним (як показано вище, усі канали, які пов’язані з вузлом B, не можуть бути транзитними). Другий зв’язує вузли C і E, тобто є транзитним. Аналогічно відмічаємо відповідні значення у таблиці 2.2 і заносимо «C/E» до рядку «Транзитні системи» до комірки для Цк 2-го рівню (ІКМ-120) до таблиці 2.4.
- 2 канали E3. Обидва зв’язують вузли C і B, тобто не можуть бути транзитними.

Для вузла D (після того, як частина каналів вже обведена і врахована із вузла C) проаналізуємо канали:

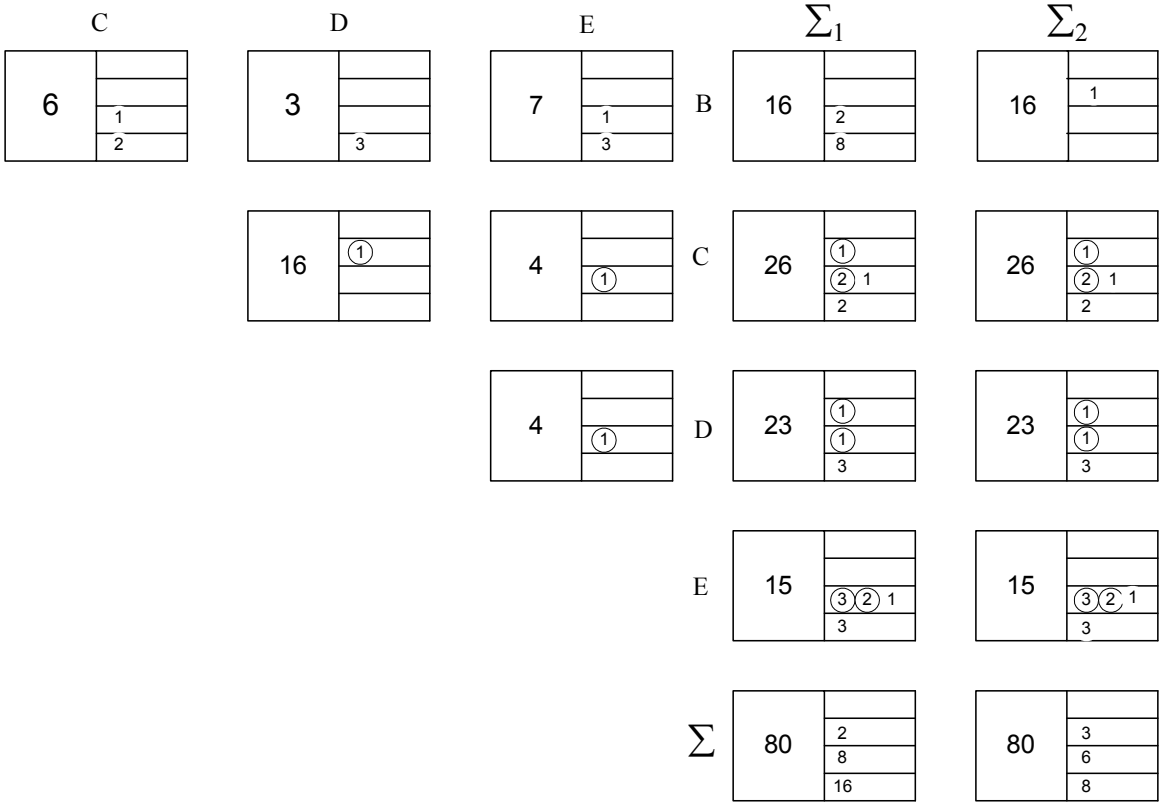
- 3 канали E3. Усі пов’язують вузли D і B, тобто не можуть бути транзитними.

Для вузла E (після того, як частина каналів вже обведена і врахована із вузлів C і D) проаналізуємо канали:

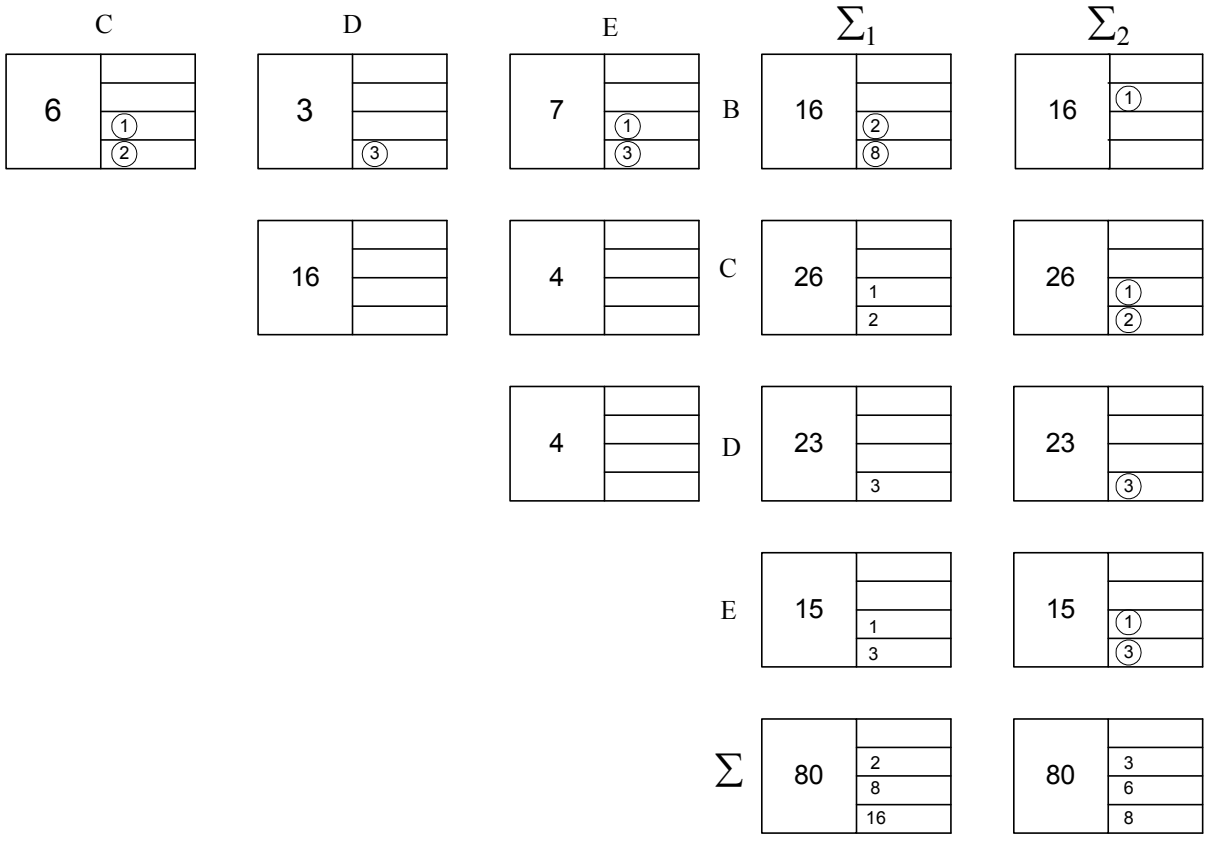
- 2 канали E2 (один вже врахований як транзит «C/E»). Один зв’язує вузли B і E, тобто не транзитний. Другий зв’язує вузли D і E, відмічається як транзит «D/E» для ІКМ-120 і обводимо у таблиці 2.2 відповідні значення.

Таблиця 2.2 тепер буде виглядати так:

Таблиця 2.2



Таблиця 2.3



3. Решта (що не обведені) позначення кількості каналів перенесемо до таблиці 2.3. Всі вони будуть мультиплексуватися у вузлі А і у випадку а) зв’язані з вузлом В.

Розрахунок мультиплексорів вузла А зводимо до таблиці 2.4. У ній по стовпцях приведені шаблони для заповнення вхідних і вихідних портів мультиплексорів М2 Е2/Е1 (120/30), М3 Е3/Е2 і М4 Е4/Е3 (1920/480). При цьому кожна комірка має 4 поля для вхідних портів і 1 поле для вихідного порта. І вхідними, і вихідними портами можуть бути вузли (В, С, D, Е) або проміжні мультиплексори (М201, М202, М301,…), що необхідні для послідовної реалізації ієрархії цифрових каналів.

Із аналізу таблиці 2.3 видно, що необхідно об’єднати 8 каналів Е1 (2 для вузлів С і В, 3 для вузлів D і В, 3 для вузлів Е і В). Для цього використовується 2 мультиплексори Е2/Е1, позначені М201 і М202, на входи яких заводяться вузли С, D, Е (при цьому обводяться усі відповідні числа у таблиці 2.3). На виході – 2 канали Е2 (таблиця 2.4). Але до вузла В необхідно завести канал Е3.

Об’єднуючи 2 канали Е2 із виходів М201 і М202 з каналами Е2 з вузлів Е і С у мультиплексорі М301 Е3/Е2 (480/120), отримаємо канал Е3, який і заводимо до вузла В.

Коли рішення, що приймаються, не очевидні, використовується стовпець «Імовірно» таблиці 2.4, до якої заносяться імовірні варіанти рішення.

На рисунку 2.1 приведені схематичні зображення розрахованих у таблиці 2.4 мультиплексорів.

Таблиця 2.4

Мультиплексори						«Імовірно»
1920	480	480	120	120	30	
M4		M301		M201		30
		B	C	M301	C	/
			E		C	/
			M201		D	/
			M202		D	/
M4		M3		M202		
				M301	D	
					E	
					E	
					E	
M4		M3		M2		120
						/
						/
						/
						/
M4		M3		M2		
M4		M3		M2		480
						/
						/
						/
						/
M4		M3		M2		

Транзитні системи			
1920	480	120	30
/	C / D	C / E	/
/	/	D / E	/
/	/	/	/
/	/	/	/
/	/	/	/

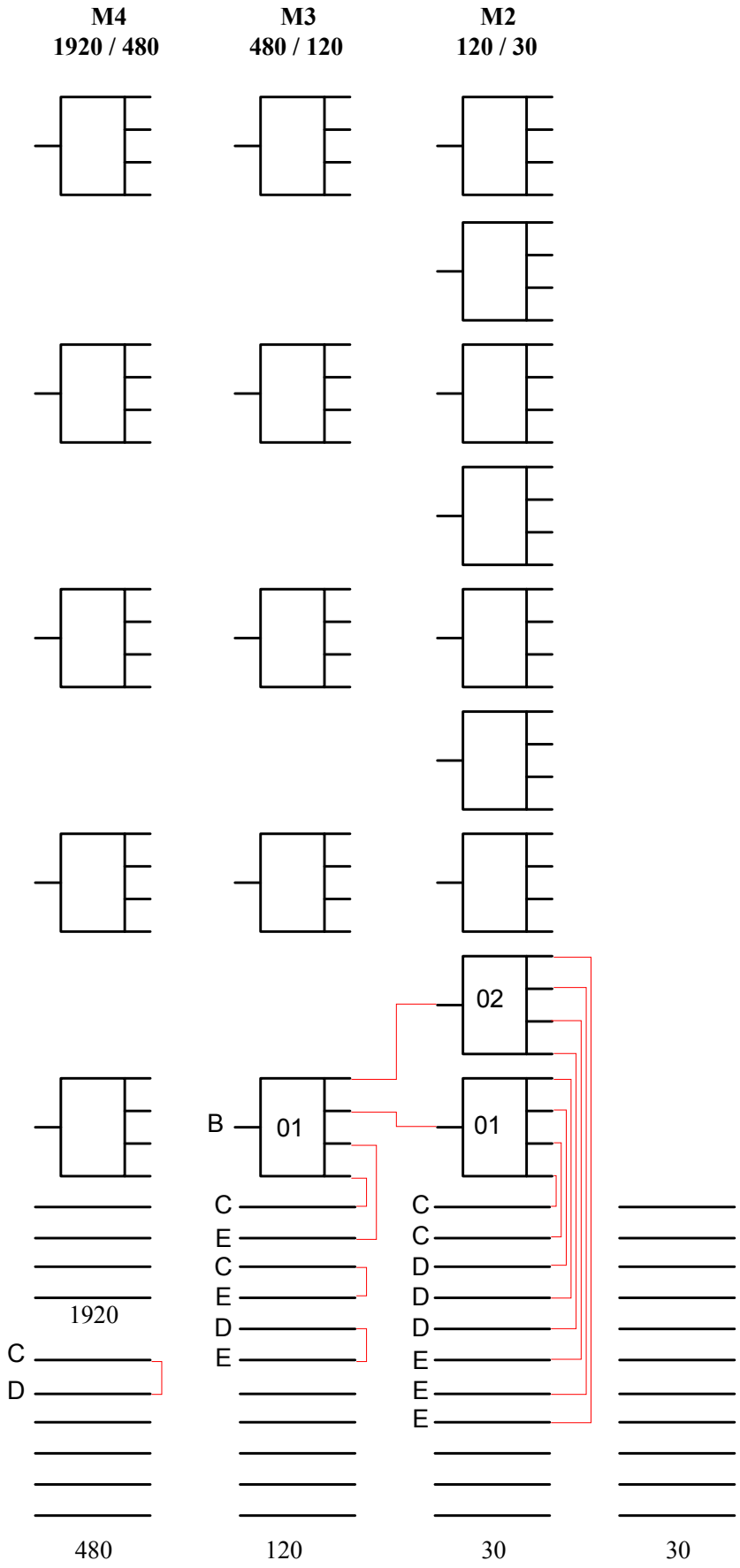

M4
1920 / 480
M3
480 / 120
M2
120 / 30
02
01
B
01
C
E
C
E
D
E
C
C
D
D
D
E
E
E
1920
C
D
480
120
30
30

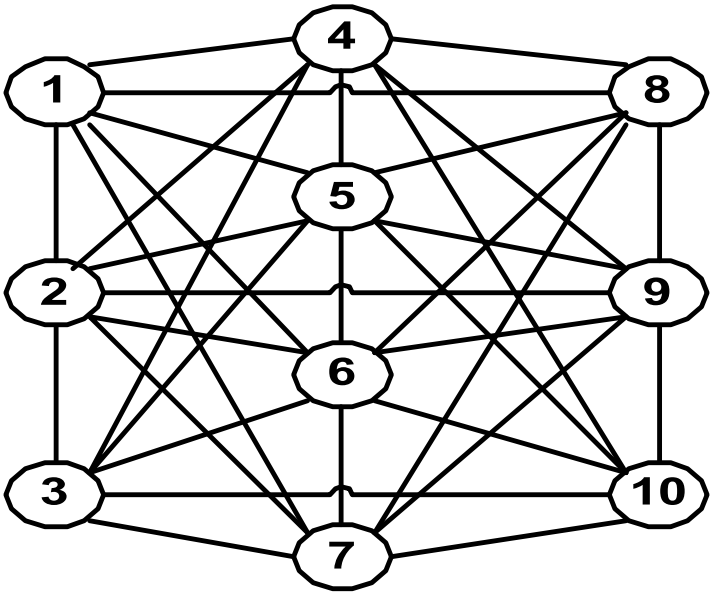
Рисунок 2.1

Перелік літератури

1. В.Г. Олифер, Н.А. Олифер «Компьютерные сети. Принципы, технологии, протоколы», С-Пб., Питер, 2000, 672 с.
2. Ю.А. Кулаков, Г.М. Луцкий « Компьютерные сети», К., Юниор, 1998, 384 с.
3. Д.Филлипс., А. Гарсиа-Диас, «Методы анализа сетей», М., «Мир», 1984, 496 с.

ВАРІАНТИ ЗАВДАНЬ

З наведеного графа у відповідності з таблицею варіантів отримайте Ваш граф, видалив з початкового графа гілки, що в таблиці варіантів позначені символом *.



Таблиця варіантів до лабораторних робіт 1-5

№ варі-анта	B12	B13	B14	B15	B16	B17	B18	B19	B1-10	B23	B24	B25	B26	B27	B28	B29	B2-10	B34	B35	B36	B37	B38	B39	B3-10
1	*	4	2	*	*	8	*	3	*	*	*	3	*	7	*	*	5	1	3	*	*	2	2	10
2	*	3	*	3	*	8	*	1	*	3	*	3	*	7	3	*	6	1	3	*	5	*	6	*
3	1	5	2	*	*	*	*	2	4	3	*	3	4	*	*	*	2	1	*	4	*	2	8	10
4	*	2	2	3	*	*	*	1	4	3	2	*	4	*	3	*	3	*	3	*	5	2	5	*
5	*	2	*	*	7	8	*	4	*	3	*	3	*	7	*	9	*	1	*	4	*	*	*	10
6	*	1	*	3	*	8	*	4	*	3	*	3	*	7	3	*	4	1	3	*	5	*	*	*
7	*	3	2	*	*	8	*	1	*	*	*	3	*	7	*	*	5	1	3	*	*	2	5	10
8	*	2	*	3	*	*	7	4	*	3	2	3	*	7	3	*	9	*	3	4	*	*	*	*
9	1	3	*	*	*	8	*	2	*	3	2	3	*	*	*	9	*	*	3	*	*	*	7	*
10	*	1	2	*	7	*	*	5	2	*	2	3	*	7	*	*	3	*	3	*	5	2	*	*
11	*	2	*	*	7	8	*	3	3	3	*	3	*	7	*	9	5	1	*	4	*	*	8	10
12	*	1	2	3	*	*	*	4	4	3	2	*	4	*	3	*	8	*	3	*	5	2	*	*
13	*	4	2	*	*	8	*	6	5	*	*	3	*	*	*	*	*	1	3	*	*	2	2	10
14	*	4	*	3	*	*	7	2	3	3	2	3	*	7	3	*	2	1	*	4	*	*	3	*
15	*	1	2	*	*	8	*	*	1	*	*	3	*	*	*	*	3	1	3	*	*	2	6	10
16	1	4	*	*	*	*	7	2	2	3	2	*	4	*	*	9	4	*	3	*	5	*	*	10
17	*	2	*	*	7	8	*	*	1	3	*	3	*	7	*	9	5	1	*	4	*	*	*	10
18	*	5	2	*	*	*	7	2	4	*	2	3	*	7	*	*	*	1	*	4	5	2	5	*
19	1	3	2	*	*	*	*	*	4	3	*	3	4	*	*	*	3	1	*	4	5	2	*	*
20	1	4	*	3	*	*	*	2	1	3	2	*	4	*	3	9	5	*	*	4	5	*	4	*
21	*	6	2	*	*	8	*	*	4	3	*	3	*	7	*	*	6	*	3	4	*	2	*	*
22	*	1	*	3	*	*	7	2	2	3	*	3	*	7	3	*	*	1	3	4	*	*	2	*
23	1	2	*	*	*	*	7	*	5	3	2	*	4	*	*	9	4	*	3	*	5	*	2	10
24	*	1	*	3	*	*	7	*	3	3	2	*	*	7	3	*	2	1	3	4	*	*	6	*
25	*	2	2	*	*	*	7	2	4	*	2	3	*	7	*	*	6	1	*	4	5	2	8	*
26	*	3	2	*	*	8	*	2	6	3	*	3	4	*	*	*	*	*	3	*	*	2	*	*
27	*	4	*	3	*	*	7	2	*	3	2	3	*	7	3	*	*	*	3	4	*	*	*	*
28	*	5	2	3	*	*	*	*	*	3	2	*	4	*	3	*	3	*	3	*	5	2	3	*

№ варі-анта	B45	B46	B47	B48	B49	B4-10	B56	B57	B58	B59	B5-10	B67	B68	B69	B6-10	B78	B79	B7-10	B89	B8-10	B9-10
1	2	4	*	*	5	*	3	3	3	*	9	4	*	5	*	*	4	*	3	*	2
2	*	3	*	1	*	10	*	1	3	*	*	4	*	*	7	3	4	6	3	5	*
3	*	5	*	1	*	*	*	2	3	5	*	4	*	5	7	3	*	*	3	*	2
4	*	2	5	1	*	10	3	1	*	5	*	4	*	5	7	*	4	*	3	*	2
5	*	2	*	1	*	*	*	4	*	5	9	4	*	5	*	3	4	*	*	*	2
6	*	1	*	1	*	10	*	4	3	*	*	4	*	*	7	3	4	6	*	4	2
7	2	3	*	1	*	*	3	1	*	5	9	4	*	5	*	*	4	*	3	4	2
8	*	2	5	1	*	10	*	4	3	5	*	*	4	5	7	*	4	6	*	*	*
9	*	3	*	1	5	10	*	2	3	*	9	4	*	5	*	3	*	6	*	*	2
10	2	1	*	*	*	10	3	5	3	*	*	4	4	*	7	*	4	*	3	*	2
11	*	2	5	1	*	*	*	3	*	5	9	4	4	*	*	3	4	*	*	3	2
12	*	1	*	1	*	10	3	4	*	5	*	4	*	5	7	3	*	*	3	5	2
13	2	4	*	*	5	*	3	6	3	*	9	4	*	5	*	*	4	6	3	8	2
14	*	4	5	1	*	10	*	*	3	5	*	*	4	5	7	*	4	6	*	3	*
15	2	1	4	1	*	*	3	*	3	*	9	4	*	5	*	*	4	6	3	1	2
16	2	4	*	*	5	*	*	*	3	*	9	4	*	*	7	*	4	*	*	2	2
17	*	2	*	1	*	*	*	5	*	5	*	4	*	5	*	3	4	6	*	1	2
18	*	5	8	*	*	10	*	*	3	*	9	4	*	5	7	*	*	6	3	4	2
19	*	3	*	*	5	*	*	*	3	5	*	4	*	5	7	3	*	*	3	4	2
20	*	4	6	1	*	*	3	*	*	5	9	4	*	5	*	3	*	6	*	1	2
21	2	6	*	1	*	10	3	5	3	*	*	*	4	*	7	*	4	6	3	4	*
22	*	1	7	1	*	10	*	*	3	5	*	*	4	5	7	*	4	6	*	2	*
23	2	2	*	*	5	*	*	*	3	5	*	4	*	*	7	*	4	*	*	5	2
24	*	1	3	1	*	10	*	5	3	5	*	*	4	5	7	*	4	6	*	3	*
25	*	2	4	*	*	10	*	*	3	*	9	4	*	5	7	*	4	*	3	4	2
26	2	3	5	1	*	10	3	*	3	*	*	4	4	*	7	*	4	6	3	6	*
27	*	4	*	1	*	10	*	5	3	5	*	4	*	5	7	*	4	6	*	4	*
28	*	5	2	1	*	10	3	*	*	5	*	4	4	*	7	*	4	*	3	2	2

Таблиця варіантів до лабораторної роботи №6

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	30	35	85	45	50	35	40	20	50	60	25	15	20	45	50	20	35	45	20	85	40	45	32	90	40
2	45	60	40	70	85	20	45	15	65	25	35	10	15	25	85	15	60	25	15	40	45	33	35	32	65
3	25	20	55	60	70	40	55	10	40	30	45	25	25	35	70	10	20	35	10	55	55	37	61	47	55
4	20	50	65	30	45	45	70	35	85	45	55	20	35	55	45	35	50	55	35	65	70	50	78	58	25
5	35	45	20	25	90	55	30	50	120	90	65	20	10	15	90	50	45	15	50	20	30	75	25	28	68
6	50	65	15	55	40	60	25	40	35	35	75	35	40	60	40	40	65	60	40	15	25	40	35	22	37
7	55	30	30	45	35	15	20	70	40	40	85	30	45	90	35	70	30	90	70	30	20	48	15	25	40
8	25	25	45	15	60	10	45	25	60	25	80	45	35	25	60	25	25	25	25	45	45	27	42	40	20
9	35	45	50	35	25	25	40	35	50	15	70	40	25	30	25	35	45	30	35	50	40	12	42	60	45
10	40	20	70	65	55	35	35	40	30	20	60	25	15	12	55	40	20	12	40	70	35	30	28	80	75
1	27	40	90	40	42	28	32	17	47	45	22	12	17	40	42	17	40	40	17	90	32	60	40	85	45
2	40	65	32	65	95	17	35	12	68	33	37	12	12	30	95	12	65	30	12	32	35	25	45	40	70
3	35	30	47	55	75	50	61	8	35	37	54	28	32	30	75	8	30	30	8	47	61	30	55	55	60
4	18	43	58	25	38	55	78	40	90	50	51	30	37	60	38	40	43	60	40	58	78	45	70	65	30
5	33	38	28	68	100	65	25	60	100	75	60	25	8	20	100	60	38	20	60	28	25	90	30	20	25
6	45	17	22	37	32	32	35	45	40	40	65	40	45	65	32	45	17	65	45	22	35	35	25	15	55
7	50	30	25	40	25	68	15	75	50	48	78	35	40	80	25	75	30	80	75	25	15	40	20	30	45
8	35	38	40	20	75	10	42	20	68	27	72	37	40	20	75	20	38	20	20	40	42	25	45	45	15
9	37	55	60	45	15	10	42	32	57	12	70	32	30	35	15	32	55	35	32	60	42	15	40	50	35
10	45	17	80	75	65	30	28	33	25	30	75	20	10	20	65	33	17	20	33	80	28	20	35	70	65