**Лабораторна робота №8В**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**ЗАДАЧІ ОПТИМІЗАЦІЇ НА МЕРЕЖАХ**

**з дисципліни ”Математичні методи дослідження операцій”**

**для студентів спеціальності 122 - ”Комп’ютерні науки”**

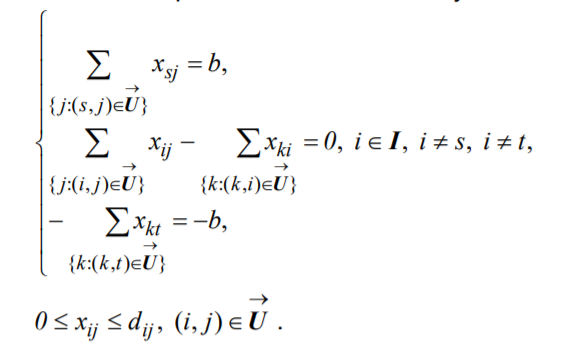
**Теоретичні відомості**

В теорії оптимізації та теорії графів, задача про максимальний потік полягає у знаходженні такого потоку за транспортною мережею, щоб сума потоків з витоку, або, що означає те ж саме, сума потоків до стоку була максимальна.

**Потокова мережа** (англ. flow network) - це орієнтований граф, де кожне ребро має ємність, пропускну спроможність і кожне ребро отримує потік. Загальний потік на ребрі не може перевищувати ємність ребра. В дослідженні операцій орієнтований граф часто називають мережею, вершини — вузлами і ребра — дугами. Потік має задовільняти обмеженю, що загальний вхідний потік вершини дорівнює загальному вихідному, за винятком джерела, що має більший вихідний потік, або стоку, що має більший вхідний потік. Таку мережу можна використати для моделювання руху в дорожній системі, струму в електронних мікросхемах або будь-чого, що рухається через мережу вузлів.

**Постановка задачі про максимальний потік**

Нехай мережа, яка визначається графом **(I, )** і на множині дуг якої задані пропускні спроможності *dij ((i, j) ∈ )*, має єдине джерело *s* з інтенсивністю *b* і єдиний стік *t*. *Тоді* за критерієм існування допустимого потоку стік *t* повинен мати інтенсивність, що дорівнює *-b*, а сам допустимий потік на такій мережі визначатиметься умовами



(1)

(2)

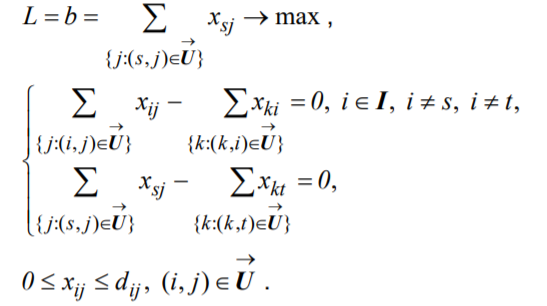
Будемо вважати інтенсивність *b* джерела *s* змінною величиною. Задачу про максимальний потік на мережі з єдиним джерелом і єдиним стоком можна сформулювати так: максимізувати *b:*

*b -> max* (3)

за виконання умов (1), (2). Змістовно це означає відшукати найбільше значення інтенсивності *b*  джерела s , при якому мережа допускає потік.

Позначимо це максимальне значення інтенсивності через *b\**. Потік ***X****\*={xij\*: (i, j) ∈ } ,* що відповідає максимальному значенню інтенсивності *b\** джерела *s* , називають максимальним потоком на мережі із єдиного джерела *s* у єдиний стік *t*, а саме значення − величиною максимального потоку.

Задача про максимальний потік на мережі, подібно задачі про найкоротший шлях на мережі, є окремим випадком задачі про оптимальний потік на мережі. Дійсно, додамо почленно першу і останню рівності у (1) (тобто виключимо *b* ) і приймемо за цільову функцію задачі ліву частину першої рівності. Отримаємо таку задачу, яка формально є задачею про оптимальний потік на мережі,

 (4)

(5)

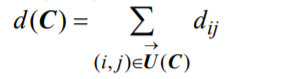
(6)

Говорять, що розріз мереж



породжений підмножиною вершин C ⊂ I , відділяє джерело s від стоку t, якщо s∈C, t ∉C.

Розріз мережі, який відділяє джерело s від стоку і який має найменшу пропускну спроможність



серед всіх таких розрізів називають мінімальним розрізом мережі. Задачу відшукання мінімального розрізу мережі називають задачею про мінімальний розріз мережі.

Алгоритм знаходження максимального потоку в мережі ґрунтується на врахуванні особливостей задачі і полягає в знаходженні на кожному кроці ауґментального (тобто збільшуючого) потоку — тобто до потоку, який збільшує існуючий і на шляху якого існує хоча б одна дуга, пропускна здатність якої буде використана повністю.

Основні приклади методів вирішення задачі про максимальний потік у мережі:

1) **Лінійне програмування.**

Складність залежить від конкретного алгоритму, у симплекс-методі експоненціальна.

Введемо позначення:

* *n*=|*V*| − розмір графа (кількість вершин);
* *m*=|*E*| − потужність графа (кількість ребер);
* ***A*** − матриця інцидентності орграфа мережі розміром *n*×*m*; кожен її елемент *aik*=1, якщо з *i*-ї вершини виходить *k*-а дуга; *aik*=−1, якщо в *i*-у вершину входить *k*-а дуга; і *aik*=0 в інших випадках; у кожному стовпці такої матриці рівно одна одиниця, одна мінус одиниця, а решта нулі;
* *s* − номер вершини-джерела мережі; з цієї вершини повинні тільки виходити дуги, і будь-яка інша вершина повинна бути досяжною з джерела;
* *t* − номер вершини-стока мережі; в цю вершину повинні тільки входити дуги, і з будь-якої іншої вершини повинен бути досяжним стік;
* ***a****s* − *s*-й рядок матриці інцидентності орграфа мережі ***A***; в ньому повинні бути тількм одиниці, оскільки з джерела повинні тільки виходити дуги;
* ***a****t* − *t*-й рядок матриці інцидентності орграфа мережі ***A***; в ньому повинні бути тільки мінус одиниці, оскільки в стік повинні тільки входити дуги;
* ***A****st* − матриця інцидентності орграфа мережі ***A*** з викинутими з неї *s*-м та *t*-м рядками;
* ***e*** − вектор-стовпчик довжини *m*; в кожному його елементі *ek* буде значення потоку в *k*-й дузі;
* ***c*** − вектор-стовпчик довжини *m*; в кожному його елементі *ck*≥0 задається пропускна здатність *k*-ї дуги.

Тоді задача про максимальний потік у мережі може бути сформульована як задача лінійного програмування:

|  |  |
| --- | --- |
| *z* = ***a****s****e*** → max; | (1) |

|  |  |
| --- | --- |
| ***A****st****e*** = ***0***; | (2) |

|  |
| --- |
| ∀*k*: 0 ≤ *ek* ≤ *ck* |

2) **Алгоритм Форда-Фалкерсона**

Складність залежить від алгоритму пошуку шляху, що збільшується. Потребує *O*(max| *f* |) таких пошуків.

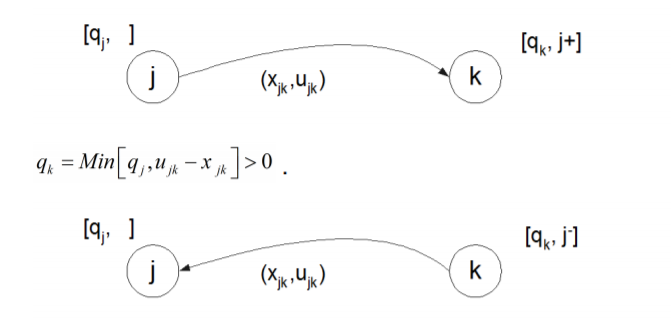
Алгоритм складається з наступних кроків:

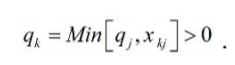
**Крок 1. Побудова початкового потоку.**

В якості початкового потоку за будь-яких умов може бути обраний нульовий — тобто потік з нульовими значеннями в кожній дузі мережі.

**Крок 2. Розташування позначок.**

На кожному кроці методу розташування позначок вузол може належати до однієї з наступних трьох категорій: “не позначений”; “позначений та непереглянутий”; “позначений та переглянутий”. На початку роботи алгоритму всі вузли “не позначені”. Джерело s отримує позначку [∞, -] . Надалі обираємо довільний позначений та непереглянутий вузол j. Два вузли є сусідніми, якщо вони з’єднані дугою. Сусідні з j непозначені вузли позначаємо. Позначка проставляється наступним чином в залежності від напрямку дуги:





Якщо ж *qk=0*, то вузол *k* не може бути позначений, і позначка біля нього не ставиться.

Якщо всі вузли переглянуті, то вузол *j* буде позначеним та переглянутим (позначка у вигляді *[qj, ]*).

Процес продовжуємо до того моменту, поки не досягнемо витоку — у цьому випадку переходимо до кроку 3. Якщо ж неможливо позначити ні одного з вузлів, що залишилися, і витік ще не досягнутий, то **стоп** — побудований максимальний потік.

**Крок 3. Зміна потоку.**

Якщо витік має позначку *[qt, k+]* , то потік дугою *(k, t)* збільшується на значення *qt*, якщо ж *[qt, k-]* — то зменшуємо на qt. Таким чином послідовно просуваємося дугами від витоку до джерела, і в залежності від знаку позначки додаємо або віднімаємо значення *qt* (з позначки при витоку !) поки не досягнемо *s*. Витираємо старі позначки та переходимо до кроку 2.

3) **Алгоритм Едмондса-Карпа**

Складність – О(VE2)

Цей алгоритм є модифікацією алгоритму Форда-Фалкерсона. Ми вибираємо найкоротший шлях, що збільшується за допомогою пошуку у ширину.

4) **Алгоритм Дініца**

Складність – О(V2E), O(V) для одиничних пропускних здатностей, O(VElog(V)) з використанням динамічних дерев Слетора і Тар’яна.

Удосконалення алгоритму Едмондса-Карпа (але хронологічно був знайдений раніше). На кожній ітерації, використовуючи пошук у ширину, визначаємо відстані від джерела до всіх вершин у залишковій мережі. Будуємо граф, який містить лише такі ребра залишкової мережі, на яких ця відстань зростає на 1. Виключаємо з графа усі тупикові вершини з інцидентними їм ребрами, поки всі вершини стануть не тупиковими. (Тупиковою називається вершина, в яку не входить і з якої не виходить жодне ребро, крім джерела і стоку.) На отриманому графі відшукуємо найкоротший шлях, що збільшується (їм буде будь-який шлях з s в t). Виключаємо із залишкової мережі ребро з мінімальною пропускною здатністю, знову виключаємо тупикові вершини, і так поки ще існують шляхи, що збільшуються.

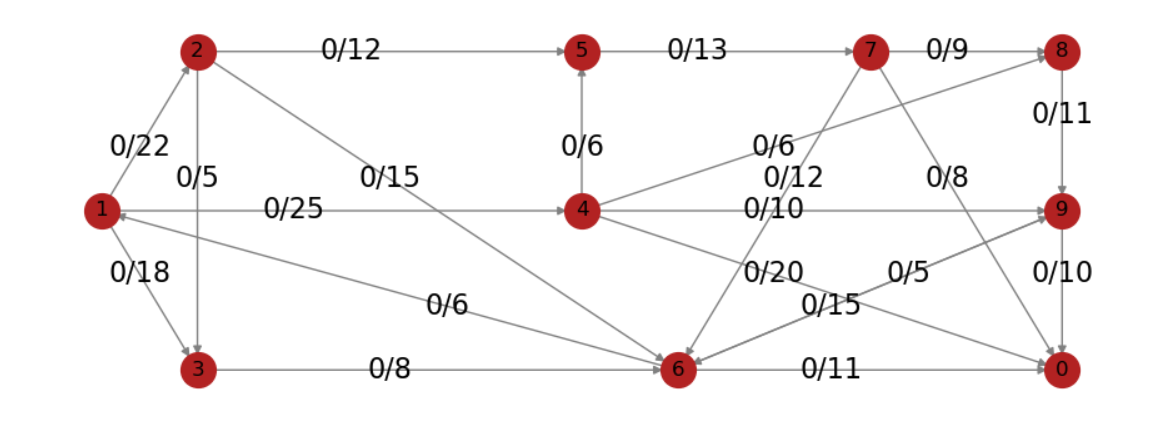
5) **Алгоритм просовування предпотоку**

Складність – О(V2E).

Замість потоку оперує з передпотоком. Різниця в тому, що для будь-якої вершини u, крім джерела і стоку, сума потоків, що входять до неї для потоку повинна бути строго нульовою (умова збереження потоку), а для передпотока — невід'ємною. Ця сума називається *надмірним потоком*у вершину, а вершина з позитивним надмірним потоком називається *переповненою*. Крім того, для кожної вершини алгоритм зберігає додаткову характеристику, *висоту*, яка є цілим невід'ємним числом. Алгоритм використовує дві операції: *просування*, коли потік по ребру, що йде з більш високої в нижчу вершину, збільшується на максимально можливу величину, і *підйом*, коли переповнена вершина, просування з якої неможливо через недостатню висоту, підіймається. Просування можливо, коли ребро належить залишковій мережі, коли воно веде з більш високої вершини в більш низьку, і вихідна вершина переповнена. Підйом можливий, коли вершина, що піднімається, переповнена, але жодна з вершин, в котру з неї ведуть ребра залишкової мережі, не нижче за неї. Він вчиняється до висоти на 1 більшою, ніж мінімальна з висот цих вершин. Спочатку висота джерела V, по всім ребрам, що виходять з джерела, тече максимально можливий потік, а решта висоти і потоки нульові. Операція просування і підйому виконуються до тих пір, поки це можливо.

**Вибір методу розв’язування задачі про максимальний потік потрібно робити, опираючись на умови задачі. Вище наведена інформація про складність алгоритмів для певних часткових випадків.**

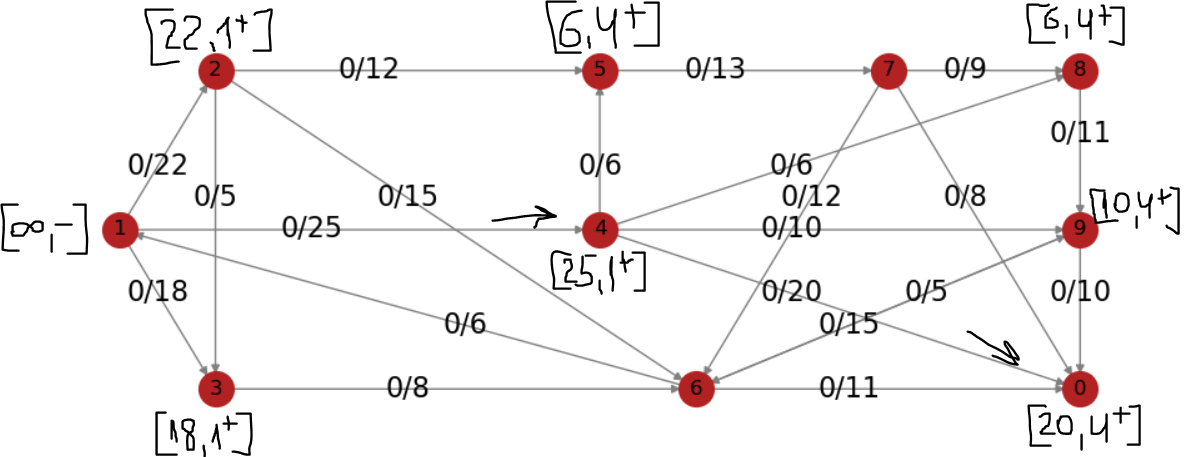
**Завдання:** Знайти максимальний потік у мережі за допомогою алгоритму розташування позначок. Джерело – вузол 1, витік – вузол -0.



**Аналітичне розв’язання:**

*#Графи побудовано за допомогою бібліотеки Networkx(Python)*

**Крок 1.**

****

Позначаємо та переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2, 3, 4.

Позначка на вузлі 2: [22, 1+]. 22 = min(∞, 22-0).

Позначка на вузлі 3: [18, 1+]. 18 = min(∞, 18-0).

Позначка на вузлі 4: [25, 1+]. 25 = min(∞, 25-0).

Обираємо довільний з них, наприклад, вузол 4, позначаємо його сусідів – вузли 5, 8, 9, 0.

Позначка на вузлі 5: [6, 4+]. 6 = min(25, 6-0).

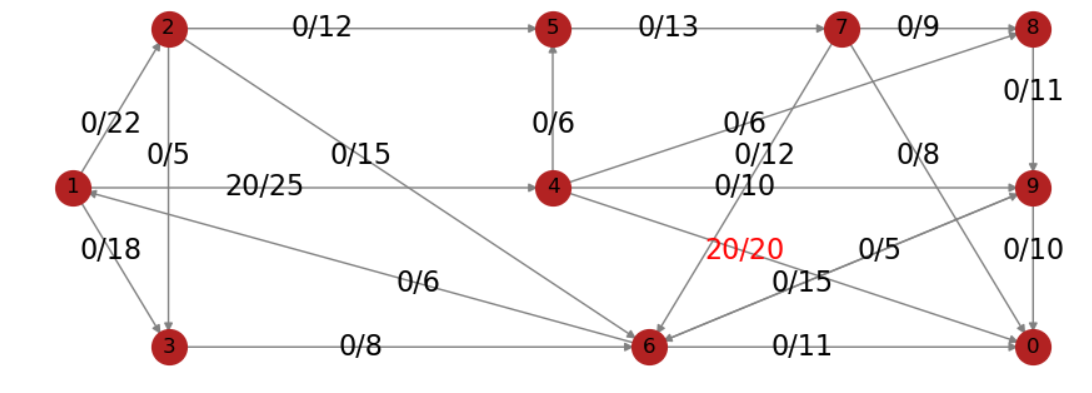
Позначка на вузлі 8: [6, 4+]. 6 = min(25, 6-0).

Позначка на вузлі 9: [10, 4+]. 10 = min(25, 10-0).

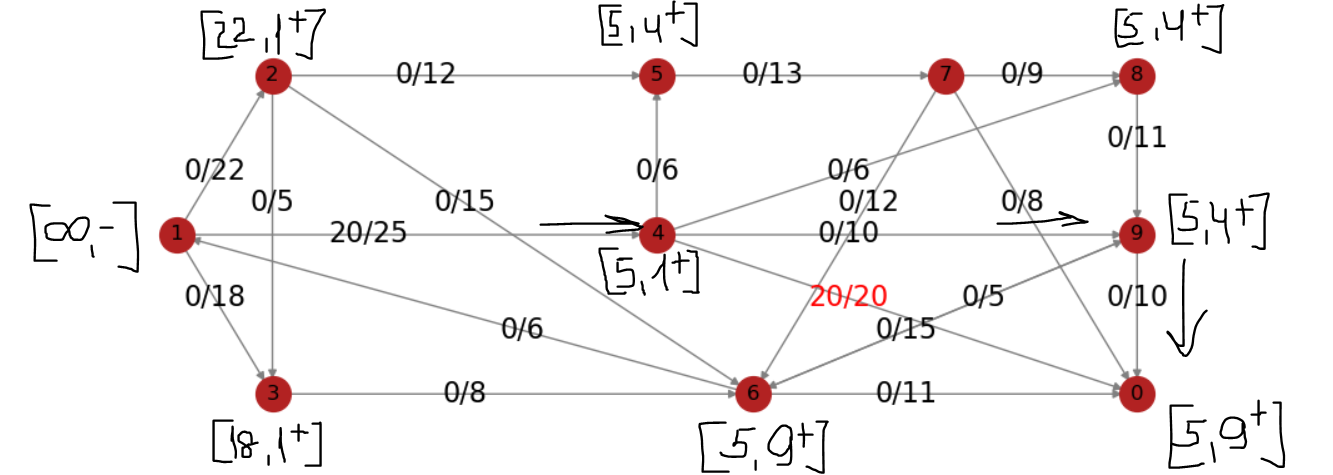
Позначка на вузлі 0: [20, 4+]. 20 = min(25, 20-0).

Переглядаємо вузол 4.

Обираємо довільний з його сусідів, наприклад, вузол 0. Це витік. Переглядаємо його. Збільшуємо потік на шляху 1-4-0 на 20.



**Крок 2.**

****

Шукаємо наступний збільшуючий шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2, 3, 4.

Позначка на вузлі 2: [22, 1+]. 22 = min(∞, 22-0).

Позначка на вузлі 3: [18, 1+]. 18 = min(∞, 18-0).

Позначка на вузлі 4: [5, 1+]. 25 = min(∞, 25-20).

Обираємо довільний з них, наприклад, вузол 4, позначаємо його сусідів – вузли 5, 8, 9 (вузол 0 позначати не можна, оскільки q0 = min(5, 20 - 20) = 0).

Позначка на вузлі 5: [5, 4+]. 6 = min(5, 6-0).

Позначка на вузлі 8: [5, 4+]. 6 = min(5, 6-0).

Позначка на вузлі 9: [5, 4+]. 10 = min(5, 10-0).

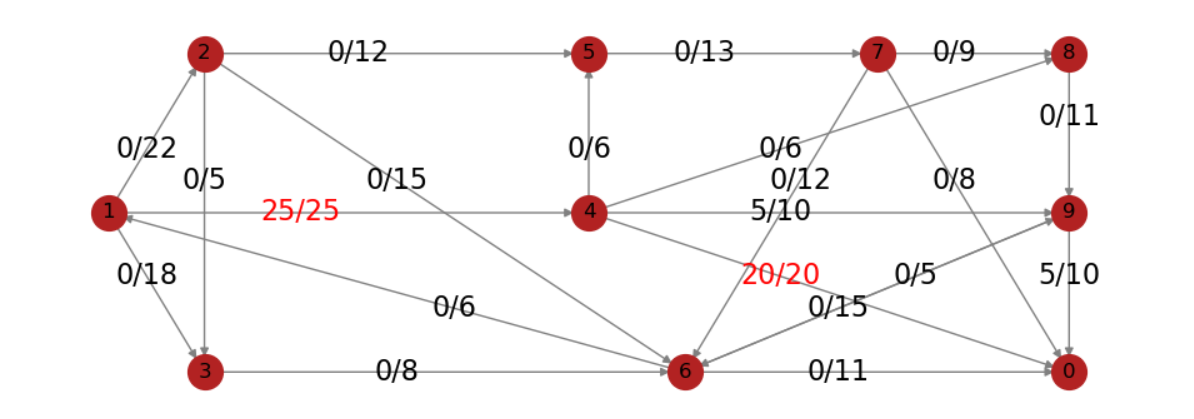
Переглядаємо вузол 4, обираємо довільний із його позначених сусідів, наприклад, вузол 9. Позначаємо його сусідів – вузли 0 та 6.

Позначка на вузлі 6: [5, 9+]. 5 = min(5, 5-0).

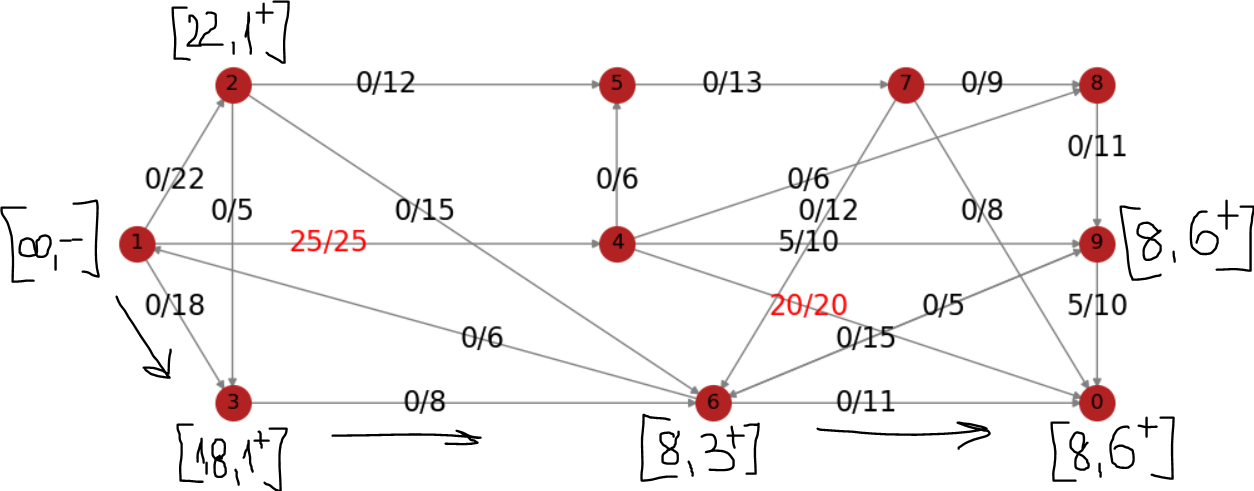
Позначка на вузлі 0: [5, 9+]. 5 = min(5, 10-0).

Переглядаємо вузол 9, обираємо довільний з його позначених сусідів, наприклад, вузол 0. Це витік. Переглядаємо його.

Збільшуємо потік на шляху 1-4-9-0 на 5.



**Крок 3**.



Шукаємо наступний ауґментальний шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2 та 3 (вузол 4 позначити не можна, оскільки q4 = min(∞, 25 - 25) = 0).

Позначка на вузлі 2: [22, 1+]. 22 = min(∞, 22-0).

Позначка на вузлі 3: [18, 1+]. 18 = min(∞, 18-0).

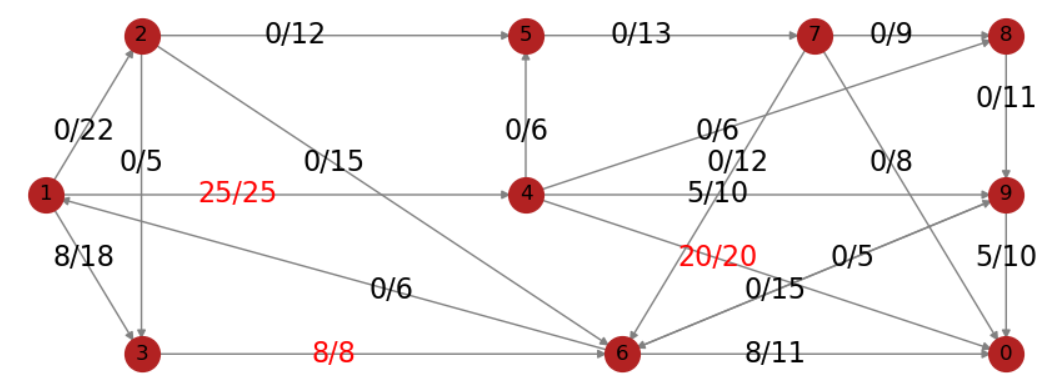
Обираємо довільний з них, наприклад, вузол 3, позначаємо його сусіда – вузол 6. Позначка на вузлі 6: [8, 3+]. 8 = min(18, 8-0).

Переглядаємо вузол 3. Позначаємо непозначених сусідів вузла 6 – вузли 9 та 0.

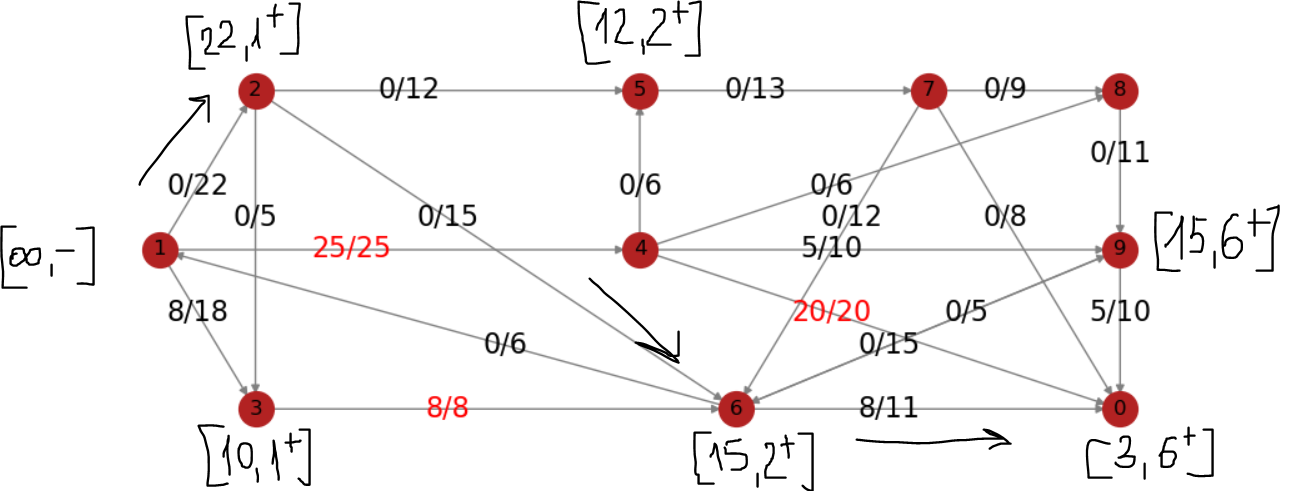
Позначка на вузлі 9: [8, 6+]. 8 = min(8, 15-0).

Позначка на вузлі 0: [8, 6+]. 8 = min(8, 11-0).

Переглядаємо вузол 6. Обираємо довільний з позначених і не переглянутих сусідів вузла 6, наприклад, вузол 0. Це витік. Переглядаємо його і збільшуємо потік на шляху 1-3-6-0 на 8.



**Крок 4.**



Шукаємо наступний збільшуючий шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2 та 3 (вузол 4 позначити не можна, оскільки q4 = min(∞, 25 - 25) = 0).

Позначка на вузлі 2: [22, 1+]. 22 = min(∞, 22-0).

Позначка на вузлі 3: [10, 1+]. 10 = min(∞, 18-8).

Обираємо довільний з них, наприклад, вузол 2. Позначаємо його сусідів – вузли 5 і 6.

Позначка на вузлі 5: [12, 2+]. 12 = min(22, 12-0).

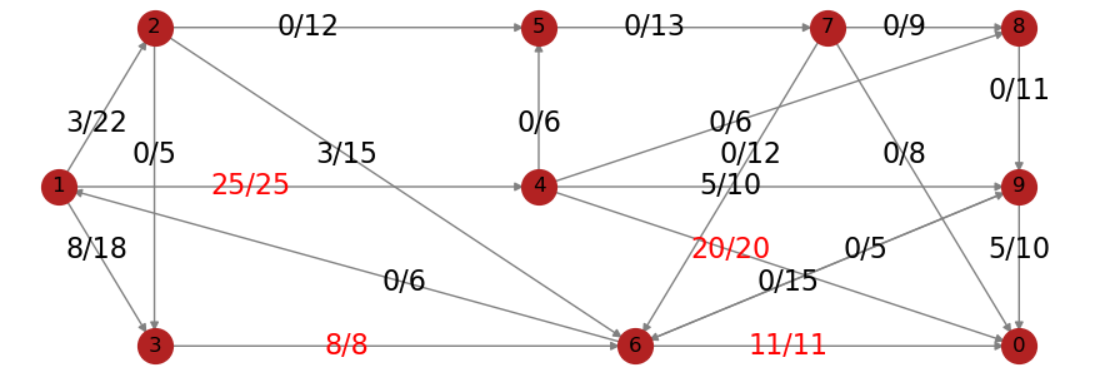
Позначка на вузлі 6: [15, 2+]. 15 = min(22, 15-0).

Переглядаємо вузол 2, обираємо довільний з його сусідів, наприклад, вузол 6. Позначаємо його непозначених сусідів – вузли 9 і 0.

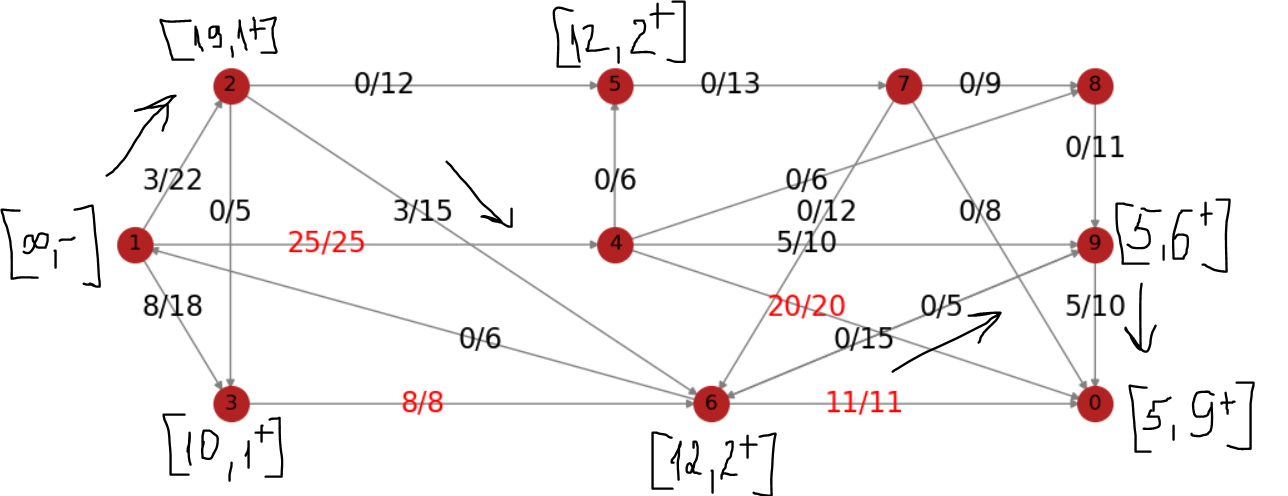
Позначка на вузлі 9: [15, 6+]. 15 = min(15, 15-0).

Позначка на вузлі 0: [3, 6+]. 3 = min(15, 11-8).

Переглядаємо вузол 6, обираємо довільного його позначеного сусіда, наприклад, вузол 0. Це витік. Переглядаємо його і збільшуємо потік на шляху 1-2-6-0 на 3.



**Крок 5.**



Шукаємо наступний збільшуючий шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2 та 3 (вузол 4 позначити не можна, оскільки q4 = min(∞, 25 - 25) = 0).

Позначка на вузлі 2: [19, 1+]. 19 = min(∞, 22-3).

Позначка на вузлі 3: [10, 1+]. 10 = min(∞, 18-8).

Обираємо довільний з них, наприклад, вузол 2. Позначаємо його сусідів – вузли 5 і 6.

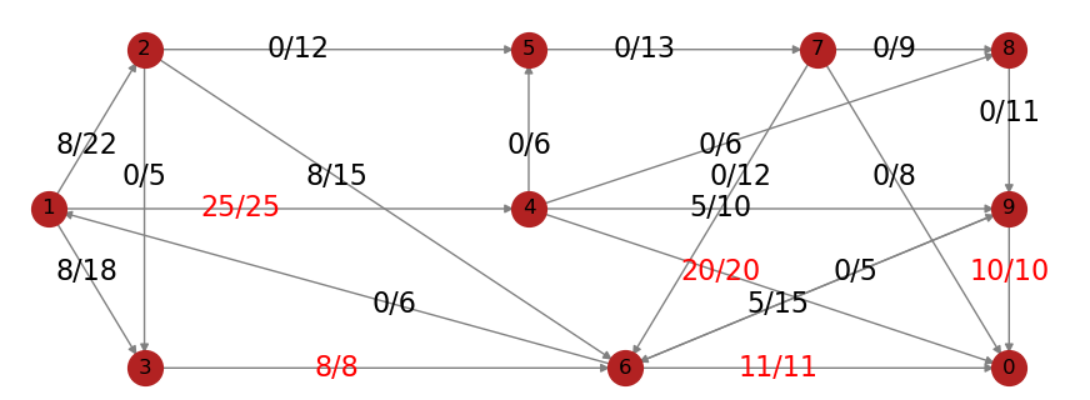
Позначка на вузлі 5: [12, 2+]. 12 = min(19, 12-0).

Позначка на вузлі 6: [12, 2+]. 15 = min(19, 15-3).

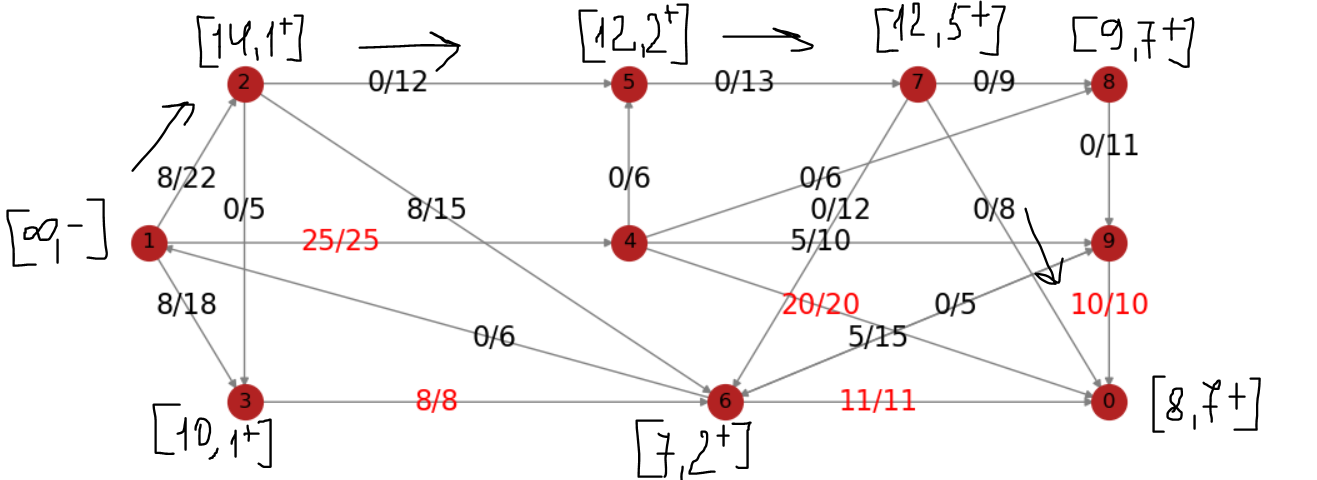
Переглядаємо вузол 2. Обираємо довільний з сусідів, наприклад, вузол 6, позначаємо його сусідній вузол 9 (вузол 0 не можна позначати, оскільки q0 = min(12, 11-11) = 0).

Позначка на вузлі 9: [5, 6+]. 15 = min(12, 5-0).

Переглядаємо вузол 6, позначаємо сусіда вузла 9 – вузол 0. Позначка: [5, 9+]. 15 = min(5, 10-5). Це витік, переглядаємо його. Збільшуємо потік на шляху 1-2-6-9-0 на 5.



**Крок 6.**



Шукаємо наступний збільшуючий шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2 та 3 (вузол 4 позначити не можна, оскільки q4 = min(∞, 25 - 25) = 0).

Позначка на вузлі 2: [14, 1+]. 19 = min(∞, 22-8).

Позначка на вузлі 3: [10, 1+]. 10 = min(∞, 18-8).

Обираємо довільний з них, наприклад, вузол 2. Позначаємо його сусідів – вузли 5 і 6.

Позначка на вузлі 5: [12, 2+]. 12 = min(14, 12-0).

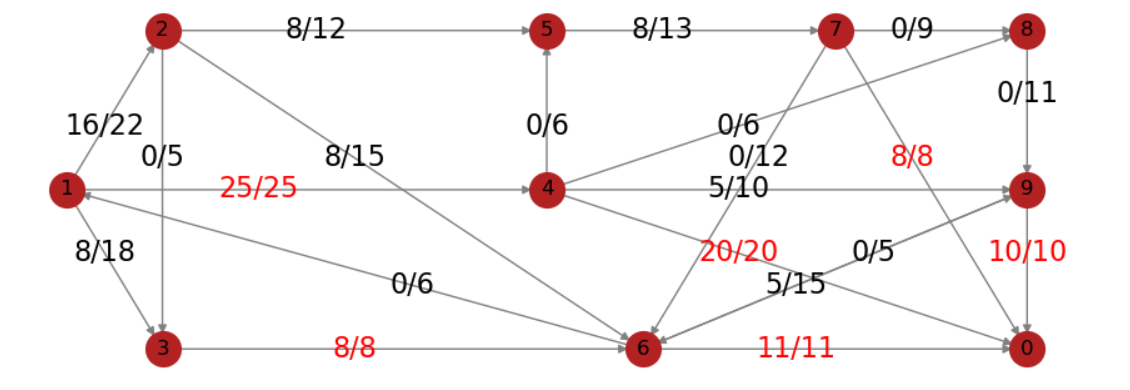
Позначка на вузлі 6: [7, 2+]. 15 = min(14, 15-8).

Переглядаємо вузол 2. Обираємо довільного сусіда, наприклад, вузол 5. Позначаємо його сусіда – вузол 7. Позначка: [12, 5+]. 12=min(12,13-0). Переглядаємо вузол 5. Позначаємо сусідів вузла 7 – вузли 8 та 0.

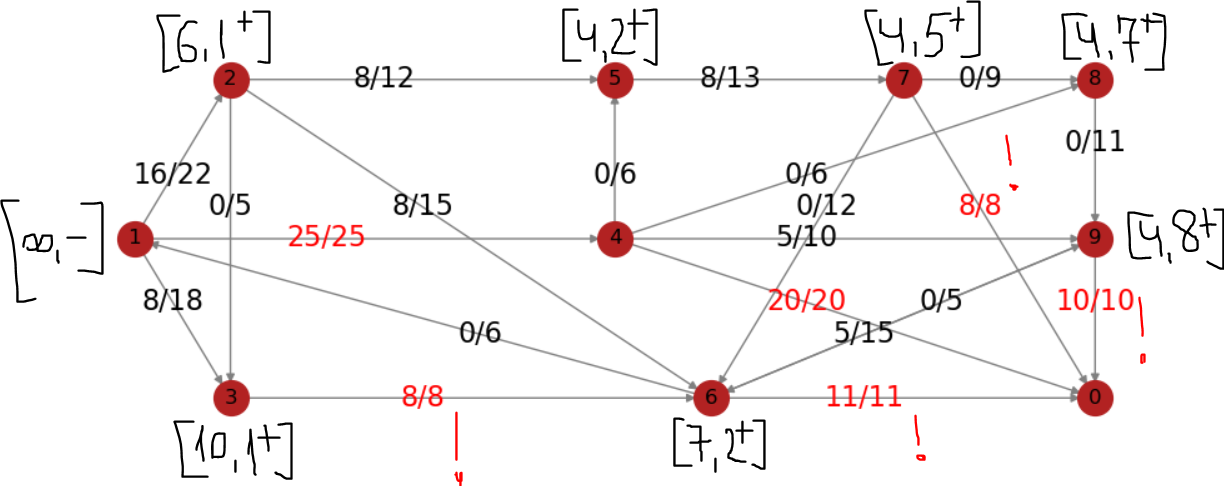
Позначка на вузлі 8: [9, 7+]. 9 = min(12, 9-0).

Позначка на вузлі 0: [8, 7+]. 8 = min(12, 8-0).

Переглядаємо вузол 7. Обираємо довільного сусіда, наприклад, вузол 0. Це витік. Переглядаємо його. Збільшуємо потік на шляху 1-2-5-7-0 на 8.



**Крок 7.**



Шукаємо наступний збільшуючий шлях. Позначаємо і переглядаємо джерело – вузол 1. Позначка: [∞, -].

Позначаємо його сусідів – вузли 2 та 3 (вузол 4 позначити не можна, оскільки q4 = min(∞, 25 - 25) = 0).

Позначка на вузлі 2: [6, 1+]. 19 = min(∞, 22-16).

Позначка на вузлі 3: [10, 1+]. 10 = min(∞, 18-8).

Обираємо довільний з них, наприклад, вузол 2. Позначаємо його сусідів – вузли 5 і 6.

Позначка на вузлі 5: [4, 2+]. 12 = min(6, 12-8).

Позначка на вузлі 6: [7, 2+]. 15 = min(6, 15-8).

Переглядаємо вузол 2. Обираємо довільного сусіда, наприклад, вузол 5. Позначаємо його сусіда – вузол 7. Позначка: [4, 5+]. 4=min(4,13-8). Переглядаємо вузол 5. Позначаємо сусіда вузла 7 – вузол 8 (вузол 0 не можна позначати, оскільки q0 = min(4, 8-8) = 0). Позначка: [4, 7+]. 4=min(4,9 - 0). Переглядаємо вузол 7. Позначаємо сусіда вузла 8 – вузол 9. Позначка: [4, 8+]. 4=min(4,11-0). Єдиний непозначений сусід вузла 9 – вузол 0, проте його позначити не можна, оскільки q0 = min(4, 10-10) = 0. Цим шляхом витік досягнути неможливо. Спробуємо обрати іншого сусіда вузла 7 – вузол 6. Непозначені сусіди вузла 6 – вузол 9 та вузол 0. Вузол 0 позначати не можна, оскільки q0 = min(7, 11-11) = 0, а єдиний сусід вузла 9 – сусід 0, який не можна позначити. Вузол 5 не має сусідів, окрім вузла 7, а вузол 2, окрім вузла 5, має сусіднім 6. Оскільки нам відомо, що з вузла 6 витоку ми не досягнемо, то обираємо іншого позначеного сусіда вузла 1 – вузол 3. Сусіднім до вузла 3 є вузол 6, але його не можна позначати, оскільки q6 = min(10, 8-8) = 0. Отже, не можна позначати жоден з вузлів, які залишилися, і витік не досягнуто. Це означає, що максимальний потік побудовано.

**Крок 8.**

Максимальний потік = 20 + 5+ 8 + 3 + 5 + 8 = 49.

**Код програми:**

import matplotlib.pyplot as plt

import networkx as nx

def fordfulkerson(graph, source, sink):

maxflow, path = 0, True

while path:

path, increased\_flow = dfs(graph, source, sink)

maxflow += increased\_flow

for v, u in zip(path, path[1:]):

if graph.has\_edge(v, u):

graph[v][u]['flow'] += increased\_flow

else:

graph[u][v]['flow'] -= increased\_flow

results(graph, path, maxflow, increased\_flow)

draw\_graph()

def dfs(graph, source, sink):

undirected = graph.to\_undirected()

explored = {source}

stack = [(source, 0, dict(undirected[source]))]

while stack:

v, zminna, neighbours = stack[-1]

if v == sink:

break

while neighbours:

u, e = neighbours.popitem()

if u not in explored:

break

else:

stack.pop()

continue

in\_direction = graph.has\_edge(v, u)

capacity = e['capacity']

flow = e['flow']

neighbours = dict(undirected[u])

if in\_direction and flow < capacity:

stack.append((u, capacity - flow, neighbours))

explored.add(u)

elif not in\_direction and flow:

stack.append((u, flow, neighbours))

explored.add(u)

reserve = min((f for zminna, f, zminna in stack[1:]), default=0)

path = [v for v, zminna,zminna in stack]

return path, reserve

G = nx.DiGraph()

G.add\_nodes\_from('123456789')

G.add\_edges\_from([

('1', '2', {'capacity': 22, 'flow': 0}),

('1', '3', {'capacity': 18, 'flow': 0}),

('1', '4', {'capacity': 25, 'flow': 0}),

('2', '3', {'capacity': 5, 'flow': 0}),

('2', '5', {'capacity': 12, 'flow': 0}),

('3', '6', {'capacity': 8, 'flow': 0}),

('4', '5', {'capacity': 6, 'flow': 0}),

('4', '8', {'capacity': 6, 'flow': 0}),

('4', '9', {'capacity': 10, 'flow': 0}),

('5', '7', {'capacity': 13, 'flow': 0}),

('6', '1', {'capacity': 6, 'flow': 0}),

('6', '9', {'capacity': 15, 'flow': 0}),

('7', '8', {'capacity': 9, 'flow': 0}),

('7', '6', {'capacity': 12, 'flow': 0}),

('8', '9', {'capacity': 11, 'flow': 0}),

('9', '6', {'capacity': 5, 'flow': 0}),

('4', '0', {'capacity': 20, 'flow': 0}),

('6', '0', {'capacity': 11, 'flow': 0}),

('7', '0', {'capacity': 8, 'flow': 0}),

('9', '0', {'capacity': 10, 'flow': 0}),

('2', '6', {'capacity': 15, 'flow': 0}),

])

layout = {

'1': [0, 0], '2': [1, 3], '3': [1, -3], '4': [5, 0],

'5': [5, 3], '6': [6, -3], '7': [8, 3], '8': [10, 3],

'9': [10, 0], '0': [10, -3]

}

def draw\_graph():

plt.figure(figsize=(12, 4))

plt.axis('off')

nx.draw\_networkx\_nodes(G, layout, node\_color='firebrick', node\_size=400)

nx.draw\_networkx\_edges(G, layout, edge\_color='gray')

nx.draw\_networkx\_labels(G, layout, font\_color='black')

for u, v, e in G.edges(data=True):

label = '{}/{}'.format(e['flow'], e['capacity'])

color = 'black' if e['flow'] < e['capacity'] else 'red'

x = layout[u][0] \* .6 + layout[v][0] \* .4

y = layout[u][1] \* .6 + layout[v][1] \* .4

t = plt.text(x, y, label, size=16, color=color,

horizontalalignment='center', verticalalignment='center')

plt.show()

def results(graph, path, current\_flow, increased\_flow):

print('flow increased by: ', increased\_flow, 'at path: ', path,'; current maximal flow: ', current\_flow)

fordfulkerson(G, '1', '0')

**Робота програми:**

****

