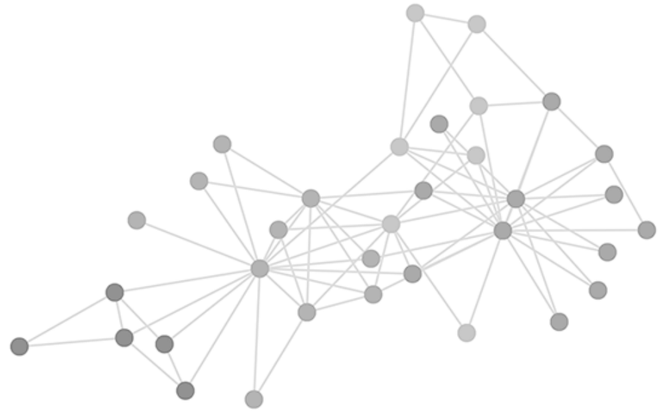
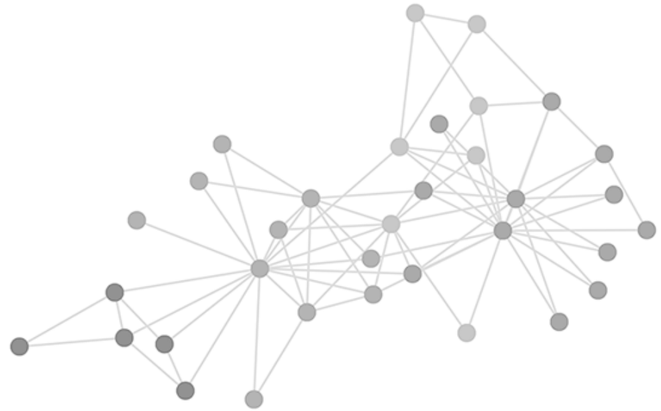
Курсова робота на тему:
Вершинно-позиційне число графів



Керівник курсової роботи:
к. ф.-м. н. Козеренко С. О.
Виконав студент
3-го року навчання спеціальності
122 “Комп‘ютерні науки“
Яременко Петро Всеволодович

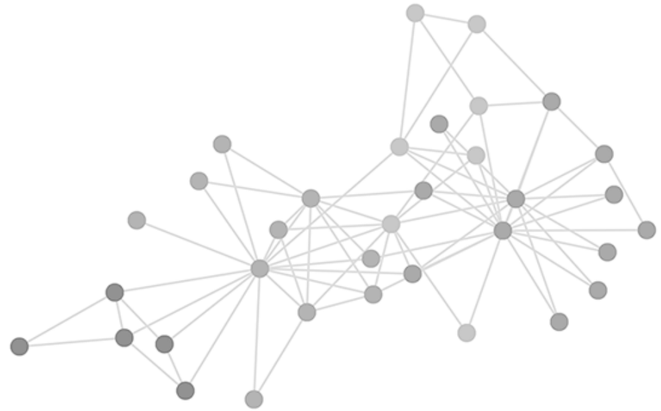
Основні означення:



- Множина вершин S в графі G знаходиться в **загальній позиції**, якщо $\forall u, v \in S$ відстань u, v не перетинається з S\{u, v}.

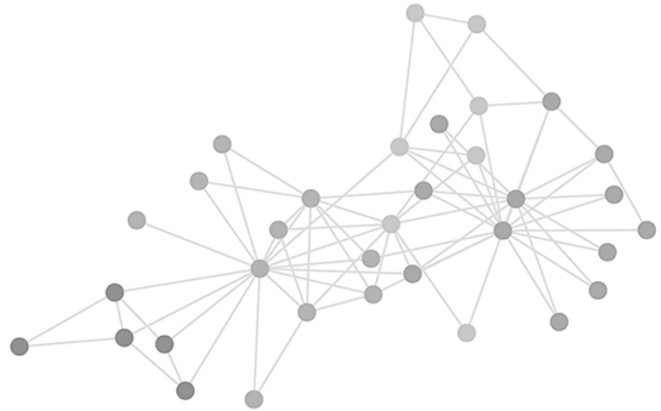
- **Загально-позиційне число gp(G)**- найбільше число вершин в найбільшій загальній позиції G.

Вершинно-позиційне число



- Для будь-якого графу та фіксованої вершини x ∈ V (G), множина Sx ⊆ V (G) є ***x-позиційною множиною***, якщо ∀y ∈ Sx кожна вершина з Sx\{y} лежить на найкоротшому шляху x,y в G. ***X-позиційне число*** G є потужністю x-позиційної множини і позначається ***px(G)*** або px. ***X-позиційна множина потужностей px(G)*** називається ***px-множиною***. Найбільше px(G) серед x ∈ G називається ***верхнім вершинно-позиційним числом vp(G)***, або просто вершинно позиційним числом G. Одночасно, найменше px(G) серед x ∈ G називається ***нижнім вершинно-позиційним числом vp-(G).***

Алгоритмічна реалізація:



Для пошуку вершинно-позиційного числа нам потрібно знайти x-позиційні числа та множини для усіх вершин графу.

Пошук вершинно-позиційного числа складається з чотирьох етапів:

1. Фільтрація ребер за критерієм однакової відстані до вершини x
2. Додавання ребер між двома вершинами, що мають найкоротший шлях до x
3. Взяття максимальної незалежної множини від множини вершин результуючого в пункті 2 графу
4. Застосування кожного попереднього пункту до кожної вершини x в графі G

Алгоритмічна реалізація:

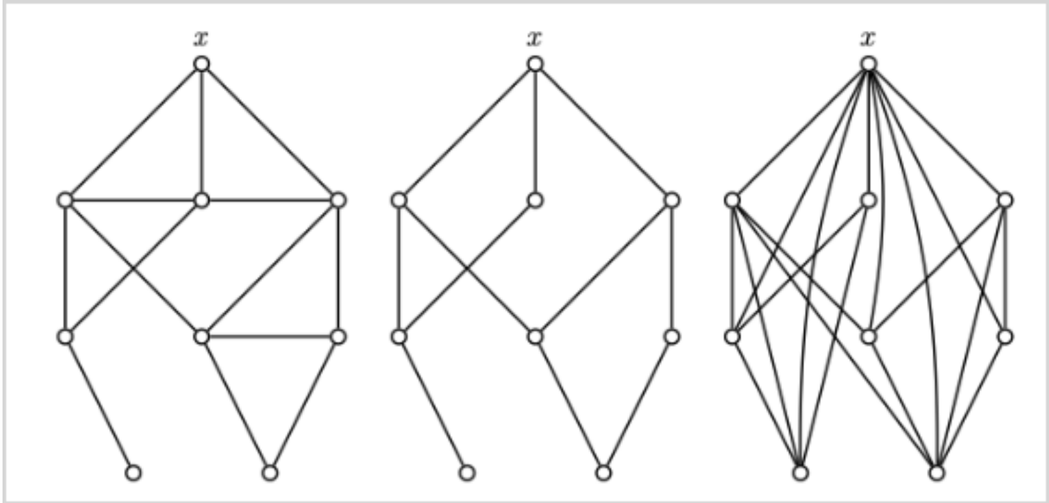


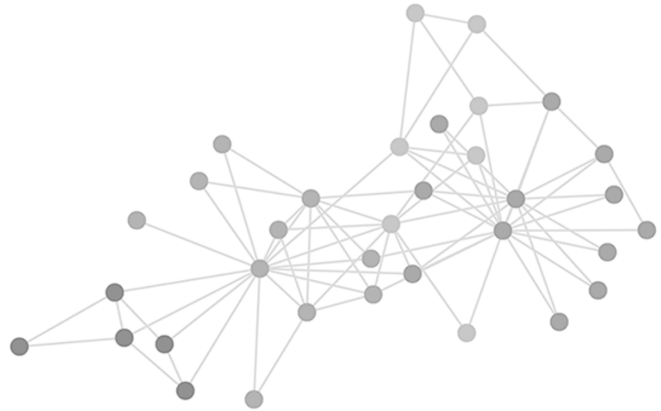
Рис. 2: Граф G, $\tilde{G}_x$, $\tilde{G}_x^*$

Введемо певні означення:

- Дано граф G і вершину x ∈ V (G). Відфільтрований граф G˜є графом, з тими ж вершинами, що і G, але трансформований за допомогою видаляння всіх ребер, що поєднують вершини на однаковій відстані від x.
- Дано граф G і вершину x ∈ V (G), граф G∗x має ті ж вершини, що і V(G), отримані з відфільтрованого графу G˜за допомогою додавання ребер між двома вершинами, що мають найкоротший шлях до x.

Алгоритмічна реалізація:

Максимальна незалежна множина



Найбільша незалежна множина графу G – така незалежна множина, при додаванні нової вершини до якої, множина не буде незалежною.

Максимальна незалежна множина графу G – така незалежна множина, більше якої не існує незалежних множин.

Data: граф G
Result: максимальна незалежна множина
$mis \leftarrow \{\}$;
$nodes \leftarrow \{V(G)\}$;
while *nodes != empty* **do**
    $v \leftarrow nodes[0]$;
    видаляємо перший елемент зі списку *nodes* ;
    додаємо v до mis;
    $nodes \leftarrow nodes \backslash N_G(v)$;
end
return *mis*

Алгоритмічна реалізація:

Фільтрація ребер

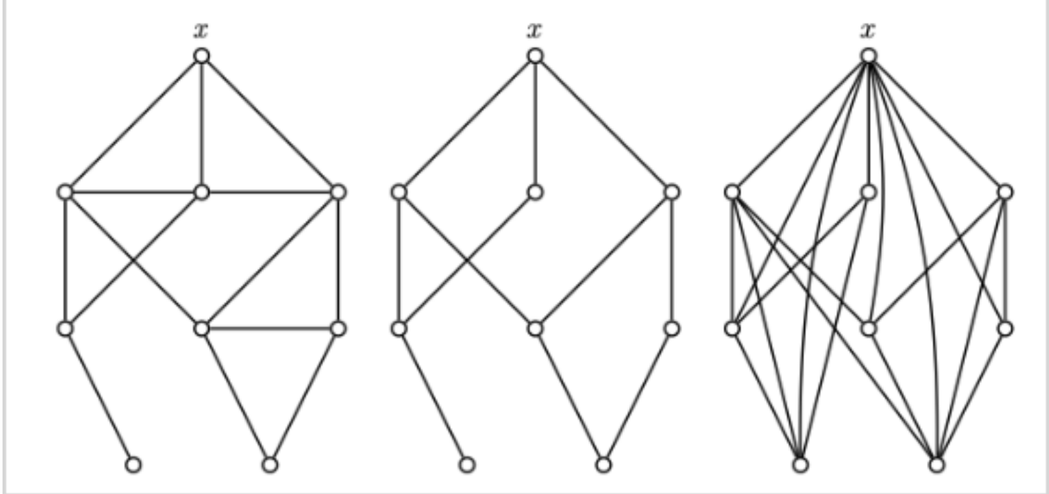


Рис. 2: Граф G, $\tilde{G}_x$, $\tilde{G}_x^*$

В алгоритмі ми фітруємо ребра (u, v) нашого графа G за критерієм рівності відстаней від вершини u до вершини x та вершини v до вершини x, тобто якщо вони рівні, то видаляємо ребро (u, v).

Data: граф G, вершина $v \in V(G)$
Result: граф $\tilde{G}_x$
for $uv \in E(G)$ **do**
  if $d(u, v) = d(v, x)$ **then**
    видаляємо ребро uv з графу G;
  end
end
return $\tilde{G}_x$

Алгоритмічна реалізація:

Додавання ребер

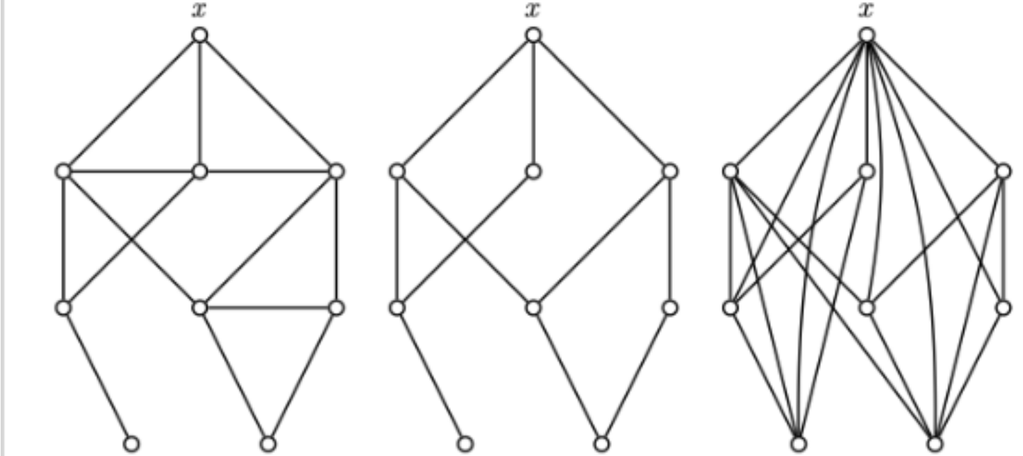


Рис. 2: Граф G, $\tilde{G}_x$, $\tilde{G}_x^*$

Алгоритм будується на пошуці в глибину. Перебираючи всі вершини u графу, для кожної ми визначаємо словник співставлень visited, де ключами є вершини, а значеннями True або False в залежності від того, що відвідували ми цю вершину, чергу q, яка складається з u. Значення вершини u в словнику visited буде True. Далі поки черга не пуста, ми беремо з неї вершину v та перебираємо циклом вершини n, які є сусідами вершини v. Якщо вершину n ще не відвідували, то додаємо її до черги, та перевіряємо, чи є вона в кожному з найкоротших шляхів від v до x, якщо так, то додаємо ребро u,n до графу відфільтрованого графу G˜. Вершина n відвідана, присвоюємо їй значення T rue в словнику visited. x
Результатом отримаємо граф G∗x.

Data: граф $\tilde{G}_x$, вершина $vertex \in V(G)$
Result: граф G_x^*
for $u \in V(\tilde{G}_x)$ **do**
  $visited \leftarrow \forall v \in V(\tilde{G}_x)$ присвоюємо значення $False$ для ключа v;
  $q \leftarrow$ черга;
  q.enqueue(u);
  $visited[u] \leftarrow True$;
  while $q.empty() == False$ **do**
    $v \leftarrow q$.dequeue();
    for $n \in N_G(v)$ **do**
      if $visited[n] == False$ **then**
        $q.enqueue(n)$;
        $paths \leftarrow$ всі шляхи з вершини v до вершини $vertex$ в графі $\tilde{G}_x$;
        for $path \in paths$ **do**
          if $n \in V(path)$ **then**
            додаємо ребро u, n до графу G;
          end
        end
      end
    end
  end
end
return G_x^*

Висновки:

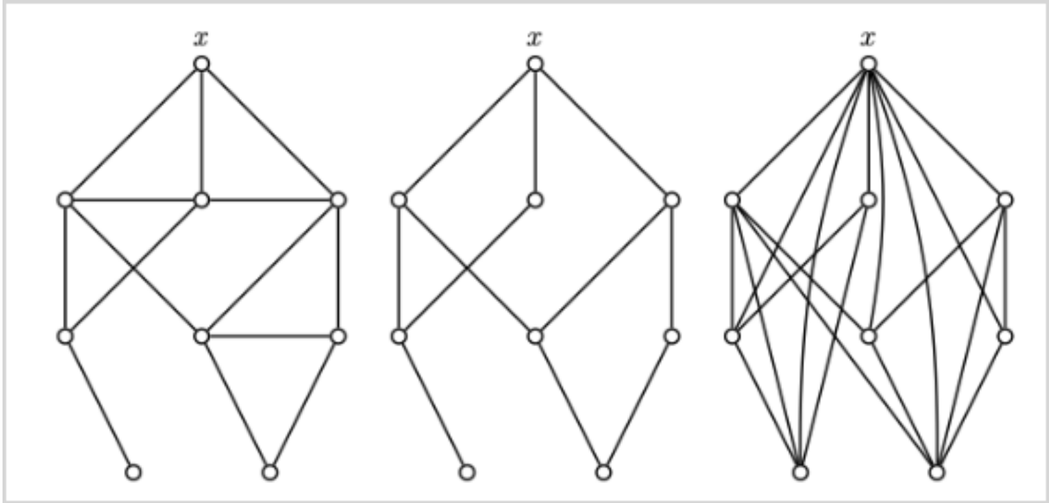


Рис. 2: Граф G, $\tilde{G}_x$, $\tilde{G}_x^*$

В роботі досліджено та реалізовано вершинно-позиційні множини та числа графів. Основна мета роботи досягнута. Вона полягала в дослідженні вершинно- позиційної множини та вершинно позиційного числа та реалізації пошуку вищеза- значених характеристик. В роботі наведений алгоритм та власне реалізація мовою Python. Алгоритм А шукає максимальну незалежну множину за Означенням 7.1 з Розділу 7, Алгоритм Б фільтрує ребра за Означенням 6.2 з Розділу 6, а Алгоритм В в свою чергу додає нові ребра за Означенням 6.3. Також у цьому розділі наведено реалі- зацію вищезазначених алгоритмів мовою Python. До роботи прикріплений Jupyter Notebook з візуалізацією.