**Московский авиационный институт**

**(национальный исследовательский университет)**

**Институт №8 «Компьютерные науки и прикладная математика»**

**Кафедра 806 «Вычислительная математика и программирование»**

**КУРСОВОЙ ПРОЕКТ**

**по курсу "Дискретная математика"**

**II семестр**

**на тему «Нахождения максимальной клики в графе»**

**Студент: Клименко В.М.**

**Группа: М8О-103Б-22, № 11**

**Руководитель: Яшина Н. П., доцент 805 кафедры**

**Москва, 2023**

Contents

[Теоретические сведения 3](#_Toc135918526)

[Описание алгоритма 4](#_Toc135918527)

[Программная реализация 6](#_Toc135918528)

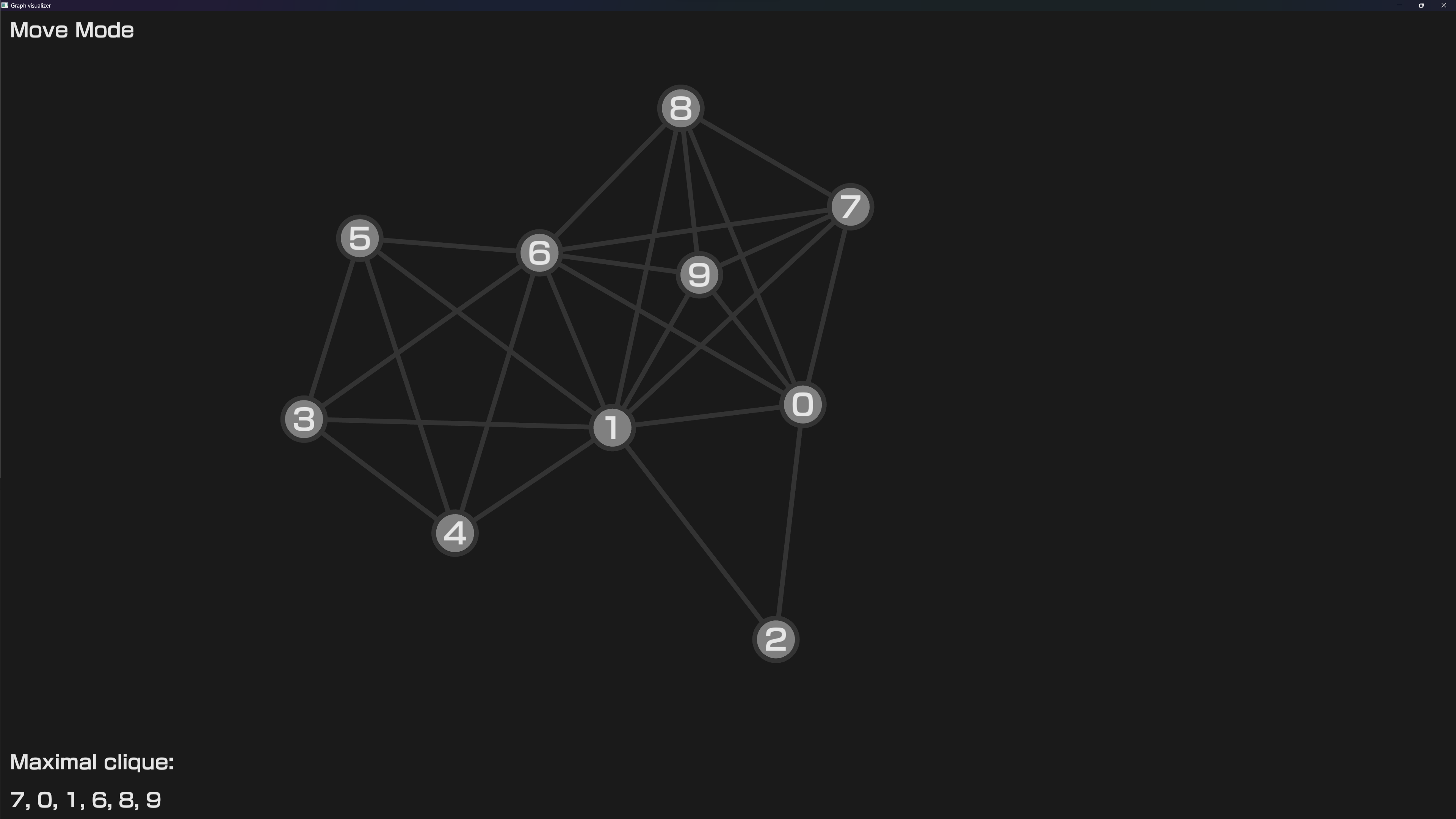
[Практическое применение 10](#_Toc135918529)

# Теоретические сведения

Задан простой неориентированный граф. Нужно найти максимальную клику и вывести ее.

Клика – это множество вершин, в котором каждая вершина соединена с каждой. Максимальная клика – клика, с наибольшим количеством вершин, среди всех найденных клик.

Для поиска максимальной клики использован жадный алгоритм, который работает за . Это не так много в реалиях моей программы. Так как у программы есть интерфейс, на котором красиво рисуется граф, физически невозможно его построить настолько большим, что поиск максимальной клики в нем будет высчитываться непозволительно долго.



# Описание алгоритма

Алгоритм поиска максимальной клики:

1. Задаем, что максимальная клика – пустое множество. Начнем пункт 2 со случайной вершины.
2. Если вершина уже есть в максимальной клики, пропускаем эту вершину, иначе создаем новое множество , единственным элементом которого является вершина под номером .
3. Проходим все вершины кроме вершины , проверяем, если является кликой, то добавляем вершину в , иначе идем дальше. По окончанию прохода, проверяем, больше ли мощность мощности . Если больше, то приравниваем к . Если еще не прошли все вершины – выбираем любую другую вершину и возвращаемся к пункту 2, иначе максимальная клика найдена.

Блок-схема алгоритма:

Начало

Выбираем случайную вершину ()

нет

да

нет в максимальной клике?

Создаем множество с

Выбираем случайную вершину ()

нет

да

Множество с является кликой и не в максимальной клике?

Добавляем в множество

нет

да

Мощность множества больше мощности максимальной клики

Максимальная клика приравнивается к множеству

Конец

# Программная реализация

Программа из себя представляет приложение с графикой, в котором можно строить неориентрованные графы, двигать отдельные вершины, выводить номера вершин, которые образуют максимальную клику в введенном графе.

Программа написана на языке программирования Rust. Такой выбор был сделан в виду того, что у меня был опыт написания графических приложений на Расте. Помимо этого, Раст – очень быстрый язык программирования, что конечно не может не радовать.

Для вершины была реализована структура данных Vertex:

pub struct Vertex {

pub id: usize,

pub connected: Vec<Vertex>,

pub coords: Vec2,

pub velocity: Vec2,

pub acceleration: Vec2,

}

В этой структуре данных лежит айди, который используется для обращение к вершинам в разных функциях, а также для графического обозначения вершин. Список вершин, присоединенных к этой вершине, реализованный на векторе. Остальные поля этой структуры данных используются исключительно для графического представления.

Для неориентированного графа была сделана отдельная структура данных Graph:

pub struct Graph {

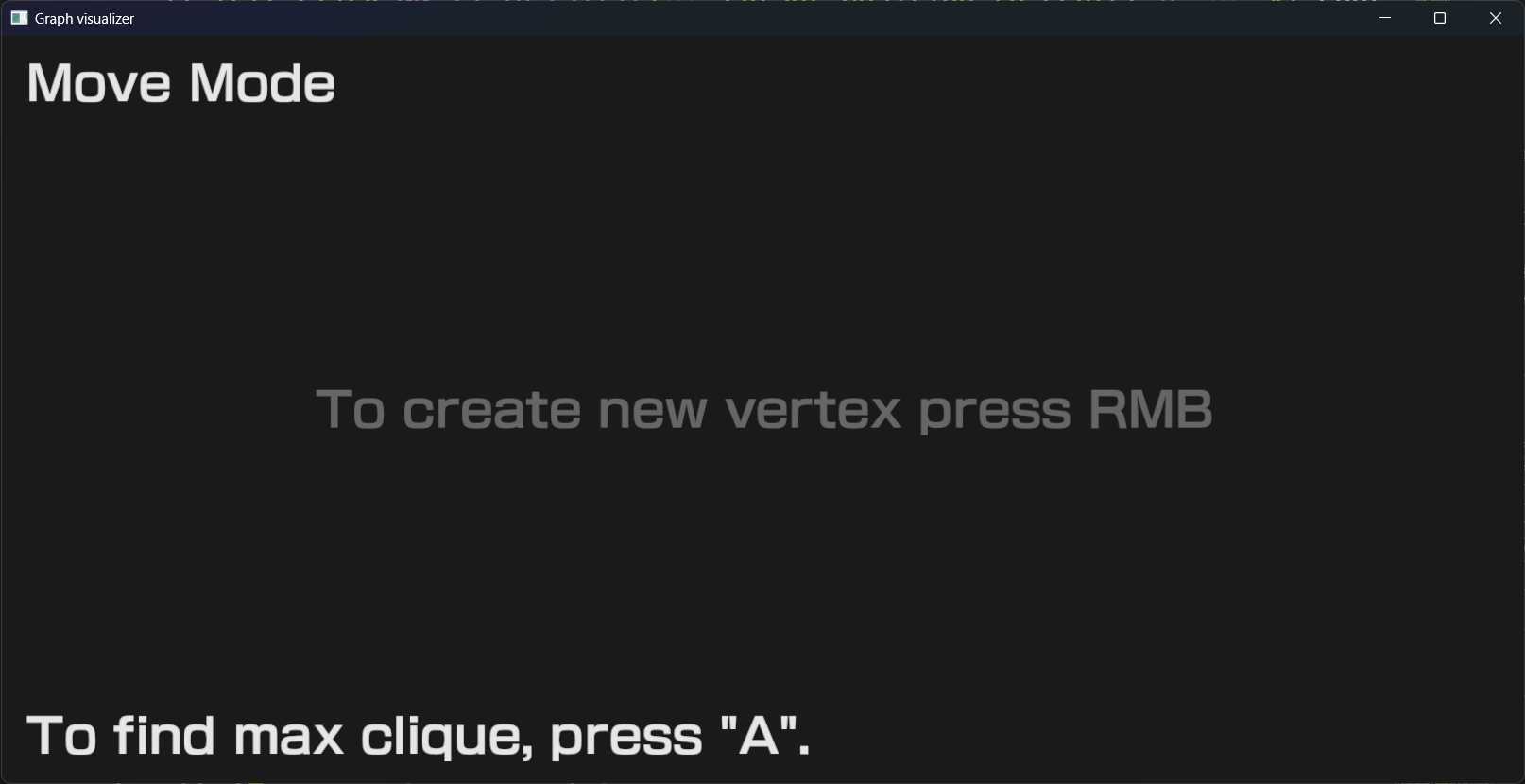
pub vertices: Vec<Vertex>,

pub arcs: HashMap<usize, Vec<usize>>,

}

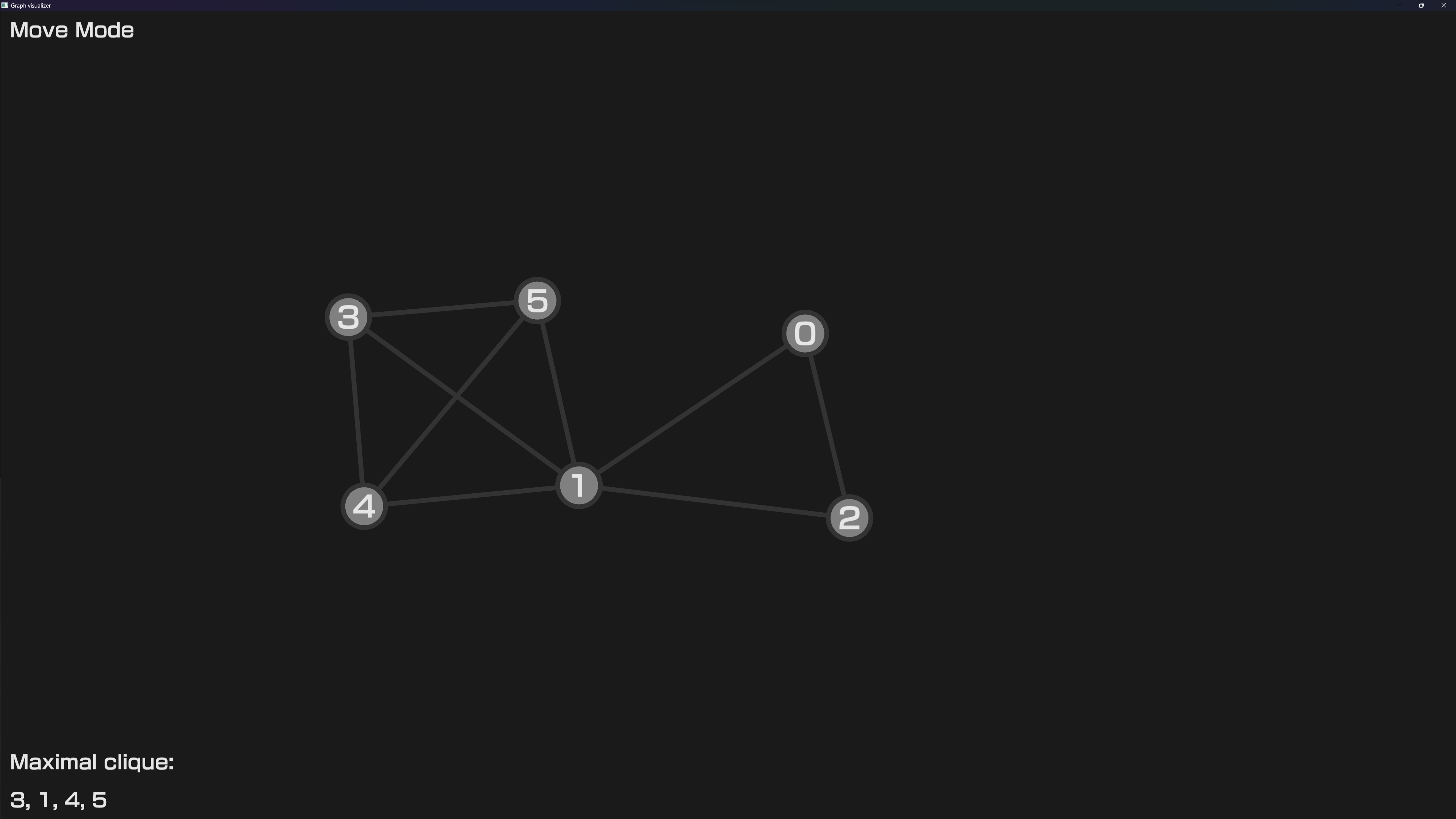
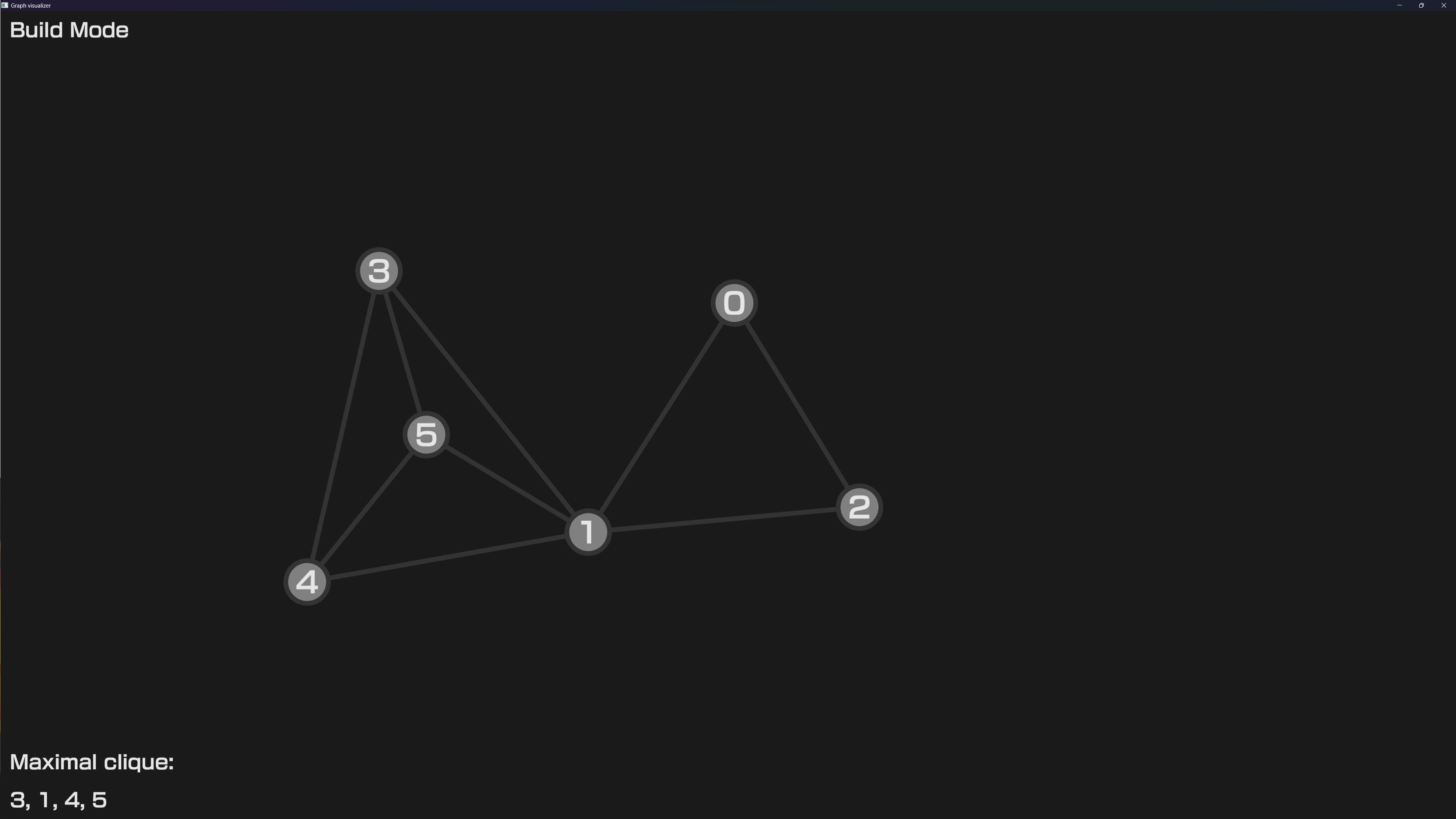
То есть в этой структуре находятся два поля – вектор со всеми вершинами и хэшмап, значение которого по данному ключу – вектор с айди всех вершин, с которыми соединена вершина с айди равным данному ключу.

Скриншоты программы:



В программе можно двигать вершины, причем при перемещении вершины, соединенные с выбранной тоже двигаются: если они слишком близко к передвигаемой, то вершины "отплывают" от выбранной, если слишком далеко, то пододвигаются ближе. Значения критических расстояний вынесены в константы MINIMAL\_DISTANCE и MAXIMUM\_DISTANCE. Помимо этих двух констант, есть еще несколько, отвечающих за движение вершин на экране.

Пример передвижения вершин:



Для структур данных было реализовано несколько функций, необходимых для построения самого графа, но самое интересное это конечно функция поиска максимальной клики:

pub fn max\_clique(&self) -> Vec<usize> {

let mut max\_clique: Vec<usize> = vec![];

for i in 0..self.len() {

if max\_clique.contains(&i) { continue; }

let mut current\_clique: Vec<usize> = vec![];

current\_clique.push(i);

for j in 0..self.len() {

if i == j { continue; }

if self.is\_clique(&current\_clique, j) {

current\_clique.push(j);

}

}

if current\_clique.len() > max\_clique.len() {

max\_clique = current\_clique.clone();

}

}

max\_clique

}

Помимо функции поиска максимальной клики, реализована вспомогательная функция проверки на то является ли список вершин с вершиной под номером кликой.

fn is\_clique(&self, vec: &Vec<usize>, j: usize) -> bool {

let mut answ: bool = true;

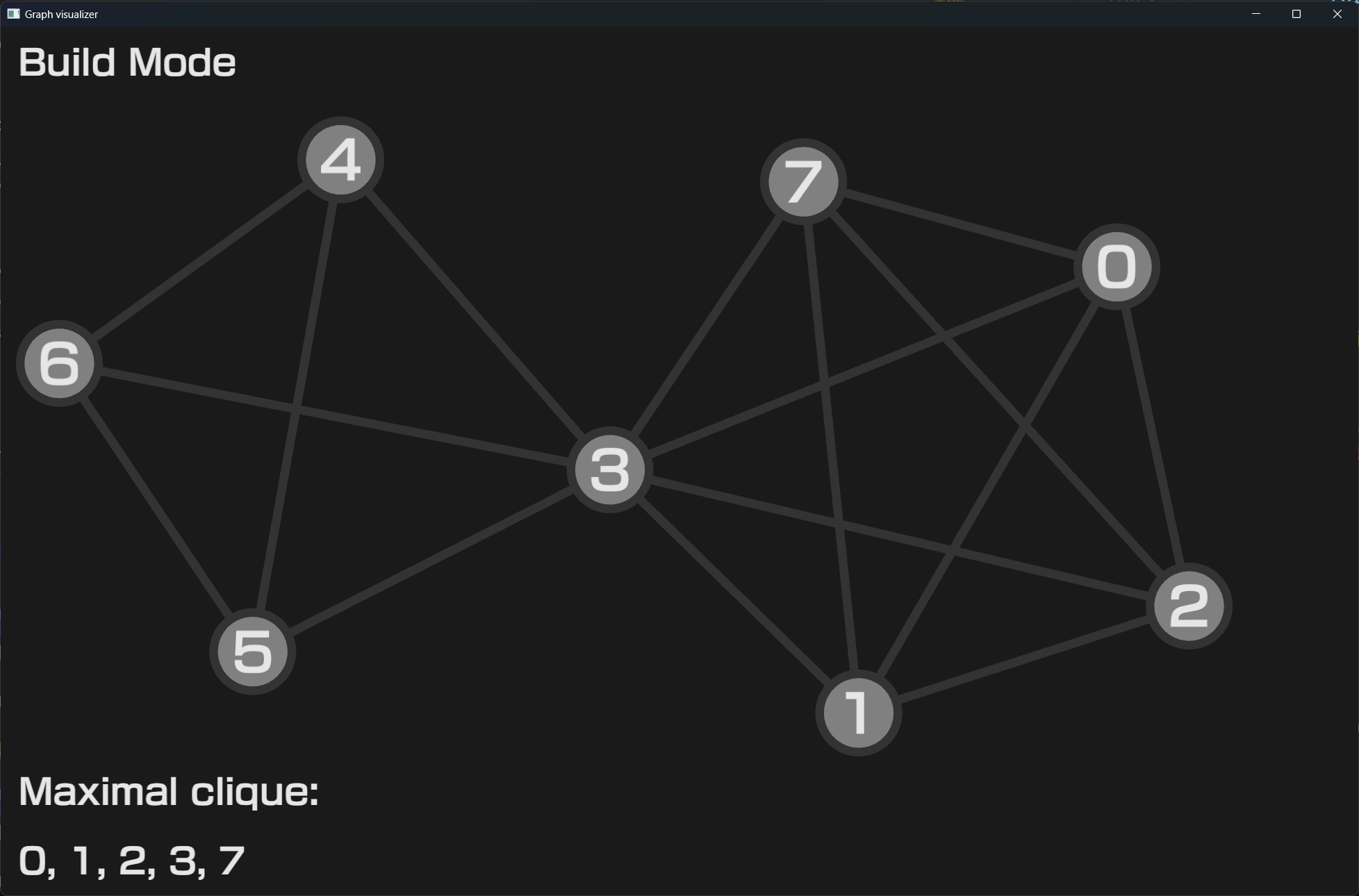
for k in vec {

answ &= self.arcs[&j].contains(k) || self.arcs[k].contains(&j);

}

answ

}



# Практическое применение

В реальном мире эта проблема может встать, например, во время организации вечеринки:

У Маши день рождения, она хочет позвать много-много друзей к себе в гости попить чаю и поесть тортик. Однако, из-за того что Маша – очень общительный и разносторонний человек, она дружит с совсем разными компаниями (в компаниях все люди дружат друг с другом), а как известно для хорошего праздника нужно подобрать хороших гостей. Задача – найти наибольшую компанию.

Для решения такой задачи можно представить дружбу людей в виде графа: люди будут вершинами, а ребра показывают, какие два человека дружат. В нашем случае компании людей – клики в графе, получается задача свелась к поиску максимальной клики.

# Полезные ссылки

* [Гитхаб с кодом программы](https://github.com/vitosotdihaet/graph-visualizer)
* [Википедия про клики](https://ru.wikipedia.org/wiki/Клика_(теория_графов))