Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Информационных технологий и управления
Кафедра Интеллектуальных информационных технологий

К защите допустить:
Заведующий кафедрой
_____ Д. В. Шункевич

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту
по дисциплине «Проектирование программного обеспечения интеллектуальных систем»:

Нахождение долей неориентированного графа

Студент: А. В. Титов
Группа: 321703

Руководитель: С. А. Никифоров

Минск 2024

АНАЛИЗ

Для проверки графа на двудольность используется теорема о двудольности графов, которая утверждает, что граф двудольный тогда и только тогда, когда он не содержит нечетных циклов. Это означает, что граф можно разделить на два непересекающихся подмножества вершин (доли), таким образом, что каждое ребро графа соединяет вершину из одной доли с вершиной из другой.

Определение двудольности графа

– Граф считается двудольным, если его вершины можно разделить на два подмножества таким образом, что все рёбра соединяют вершины из разных подмножеств.

Алгоритм проверки на двудольность

Один из способов проверки на двудольность - это использование обхода в глубину (DFS) или в ширину (BFS) для раскраски вершин графа в два цвета, гарантируя, что соседние вершины имеют разные цвета. Если на протяжении обхода не возникло противоречий (вершины одного цвета не соединены ребром), то граф является двудольным.

Шаги проверки на двудольность

– Начать с произвольной вершины и раскрасить её в один цвет.
– Раскрашивать соседние вершины в противоположный цвет.
– Продолжать этот процесс для всех вершин и их соседей.
– Если в процессе обхода обнаружено противоречие (вершины одного цвета соединены ребром), граф не является двудольным.

Один из аналогов решения задачи двудольности графа заключается в использовании матрицы смежности. Путем анализа структуры этой матрицы можно выявить наличие нечетных циклов, что поможет определить, является ли граф двудольным.

Был выбран алгоритм закрашивания элементов, так как он лучше всего подходит для реализации программы.

ПРОЕКТИРОВАНИЕ

Двудольный граф - граф, вершины которого можно разбить на два множества так, что каждое ребро соединяет вершины из разных множеств. Пример двудольного графа приведен на рисунке 0.1

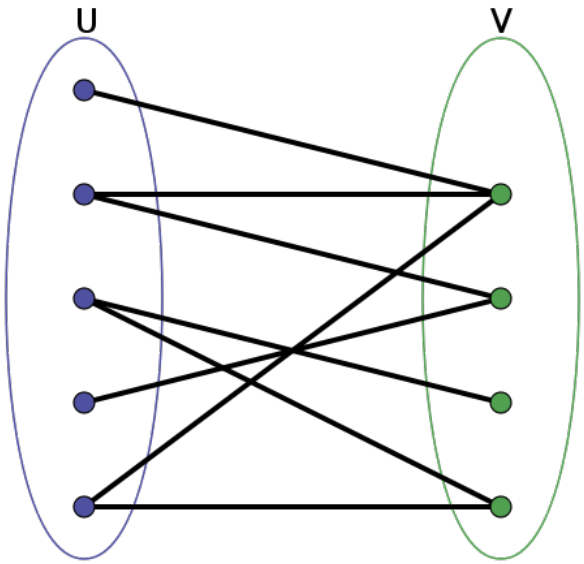

Рисунок *0.1* – Разбиение вершин на два множества

Часто в контексте двудольных графов используется понятие цвета вершины. Разбитие графа на две доли называется покраской его вершин в два различных цвета. Каждое ребро должно соединять вершины различного цвета. Пример раскрашенного графа приведен на рисунке 0.2

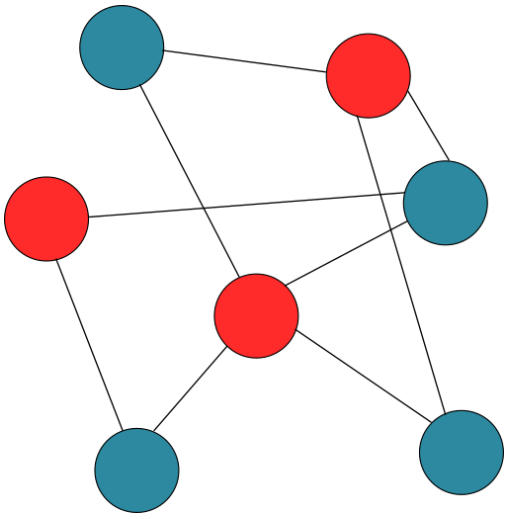

Рисунок *0.2* – Раскраска в два цвета

Существует ещё один признак двудольности графа: граф является двудольным тогда и только тогда, когда в нём отсутствуют циклы нечётной длины.

Алгоритм: Начинаем покраску с произвольной вершины, которую красим в произвольный цвет. При прохождении по каждому ребру красим следующую вершину в противоположный цвет. Если при переборе соседних вершин мы нашли вершину, уже покрашенную в тот же цвет, что и текущая, то в графе существует нечётный цикл, а значит, он не является двудольным. Пример покрашенных графов приведен на рисунке 0.3

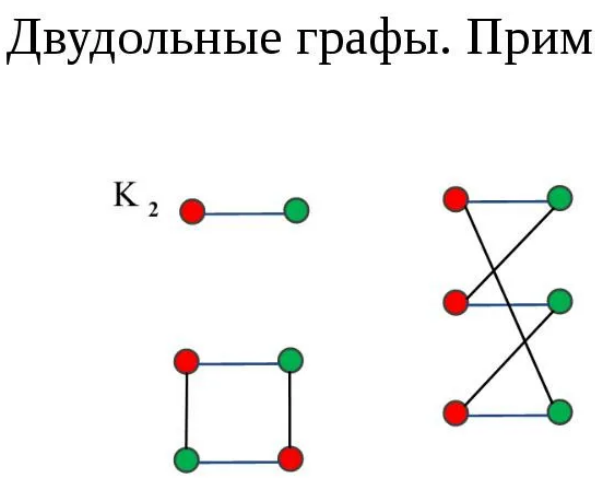

Рисунок *0.3* – Раскрашенные графы

РАЗРАБОТКА

Логика кода состоит всего из двух почти одинаковых функций: покраска вершины в красный и покраска вершины в синий цвет. Запускаем обход графа от любой начальной вершины. Добавляем взятую вершину в посещенные(visited) и множество красных вершин(red-colour). При помощи итератора ищем всех ее соседей. Для полученных соседей из первого итератора, при помощи второго итератора, мы проверяем. Посещены ли эти соседи ранее. Если да, то Node-is-visited становится true, а если нет, запускаем от соседа функцию Blue-node, тем самым делая рекурсивный обход. Если при помощи итератора мы установим, что соседняя вершина такого же цвета, что и в наша. Программа сразу же выводит, что граф не является двудольным. Реализация кода показана на рисунке 0.4.

```
void MyAgent::Red_node(ScAddr conceptAddr, bool & bipartite_graph)
{
    m_memoryCtx.CreateEdge(ScType::EdgeAccessConstPosPerm, MyKeynodes::visited, conceptAddr); //добавляем в посещенные
    m_memoryCtx.CreateEdge(ScType::EdgeAccessConstPosPerm, MyKeynodes::red_colour, conceptAddr); //добавляем в мн-во красный цвет


    ScIterator3Ptr it3 = m_memoryCtx.Iterator3(
        conceptAddr,
        ScType::EdgeUCommonConst,
        ScType::NodeConst
    );

    while (it3->Next()) {  //просматриваем соседей
        ScAddr Node = it3->Get(2);
        ScIterator3Ptr it3_check_class = m_memoryCtx.Iterator3( //итератор для проверки, является ли вершина посещенной
            ScType::NodeConstClass,
            ScType::EdgeAccessConstPosPerm,
            Node
        );

        bool Node_is_visited = false;

        while (it3_check_class->Next()){
            ScAddr NodeVisited = it3_check_class->Get(0);
            if (NodeVisited == MyKeynodes::visited)
                Node_is_visited = true;

            else if (NodeVisited == MyKeynodes::red_colour){
                bipartite_graph = false;
                return;
            }
        }
        if (!Node_is_visited) Blue_node(Node, bipartite_graph);

    }
}
```

Рисунок *0.4* – Функция покраски вершины в красный

Тесты:

Результаты первого теста показаны на рисунках 0.5 и 0.6

```
[22:59:04][Info]: Signal handler
[23:00:39][Info]: MyAgent: starting
[23:00:39][Info]: Graph is bipartite. My Agent: end
```

Рисунок *0.5* – Вывод результата в консоль

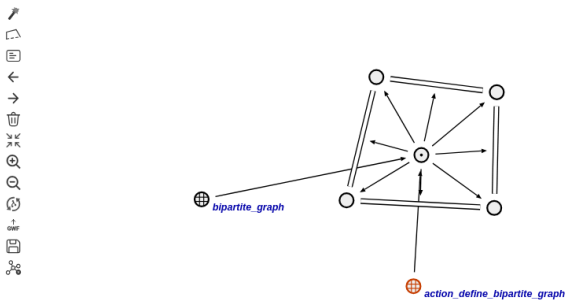

Рисунок *0.6* – Данный граф является двудольным

Результаты второго теста показаны на рисунках 0.7 и 0.8

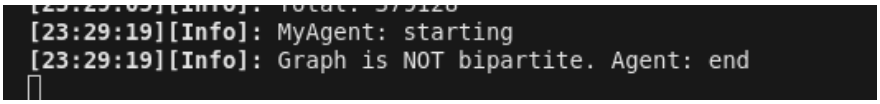

Рисунок *0.7* – Вывод результата в консоль

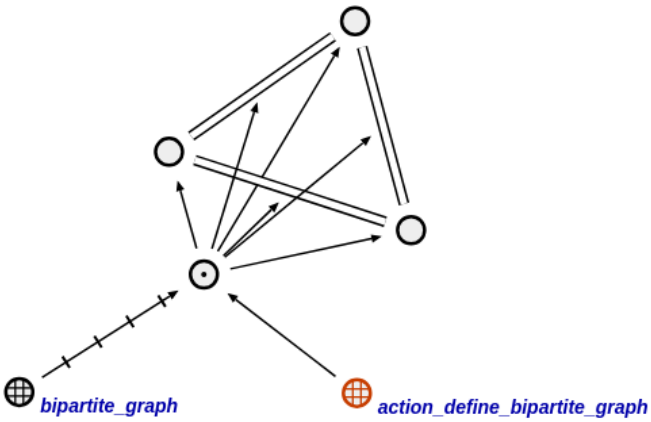

Рисунок *0.8* – Данный граф не является двудольным

Результаты второго теста показаны на рисунках 0.9 и 0.10

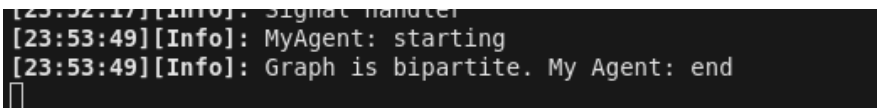

Рисунок *0.9* – Вывод результата в консоль

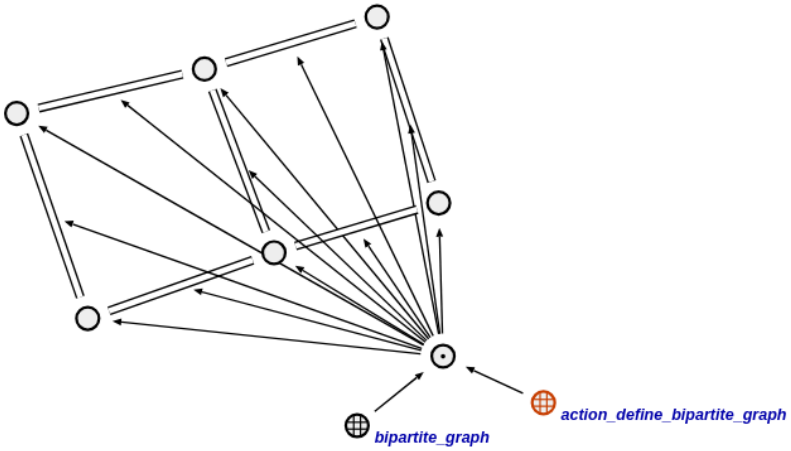

Рисунок *0.10* – Данный граф является двудольным

ЗАКЛЮЧЕНИЕ

В ходе выполнения расчетной работы была изучена тема "дольность графа". Разработан алгоритм определения двудольности графа, а так же написан код на основе этого алгоритма и проведено тестирование кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Weber, Richard. Time Series / Richard Weber. — 1999. http://www.statslab.cam.ac.uk/~rrw1/timeseries/t.pdf.

[2] Колосов, А. А. Обнаружение радиосигналов / А. А. Колосов; Ed. by А. А. Колосов. — Радио и связь, 1989. — Vol. 289.

[3] Палицын, В. А. — Технико-экономическое обоснование дипломных проектов: Метод. пособие для студ. всех спец. БГУИР. В 4-х ч. Ч. 4: Проекты программного обеспечения. — БГУИР, 2006.

[4] SpeakEasy. http://en.wikipedia.org/wiki/SpeakEasy.