МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)
Структурное подразделение Новосибирского государственного университета –
Высший колледж информатики Университета (ВКИ НГУ)
КАФЕДРА ИНФОРМАТИКИ

РЕАЛИЗАЦИЯ ПРОГРАММНОЙ БИБЛИОТЕКИ ДЛЯ РАБОТЫ С МНОГОУРОВНЕВЫМИ СЕТЯМИ

Квалификация техник-программист

Руководитель Родионов А.С.

«___»_____2022 г.

Студент 4 курса Новиков С.Я.

803в2 «____»_____2022 г.

Нормоконтроль

«____»_____2022 г.

Новосибирск

2022

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ, УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ

Граф – математическая абстракция реальной системы любой природы, объекты которой обладают парными связями. Граф как математический объект есть совокупность двух множеств – множества самих объектов, называемого множеством вершин и множества их парных связей, называемого множеством рёбер. Элемент множества рёбер есть пара элементов множества вершин.

Гиперсеть – математическая модель для описания сложных систем сетевой структуры, с помощью которой решают задачи проектирования и оптимизации различных сетей

ВВЕДЕНИЕ

В современном мире с каждым днем многие процессы и системы становятся все сложнее и более запутанней. Математический аппарат дает много различных способов описать происходящие процессы. Одним из возможных описаний это модель обычной одноуровневой сети, но развитие этих идей создало более сложный подход многоуровневые сети, описывающие связи не только внутри одной группы, но и связь между множеством групп.

При написании данной работы были рассмотрены большинство случаев анализа сетей различной природы (транспортных, дорожных, телекоммуникационных, социальных и пр.), рассматривались графовые, чаще гиперграфовые, модели. Однако, многие сложные системы и процессы включают себя связи на разных уровнях.

Примеры, где отлично подходят многоуровневые сети:

1. транспортные сети;

2. кабельные сети;

3. сети социальных взаимоотношений;

Каждая сеть имеет свою задачу анализа:

1. в транспортных: нахождение маршрута с пересадками;

2. в кабельных: анализ надёжности виртуальных соединений при возможных обрывах проводов;

3. в социальных: нахождение удалённых связей и неявных социальных групп;

Основные исследования в этой области производились в ИВМиМГ СО РАН профессором Попковым В.К.. Им были сформулированы основные определения моделей многоуровневых сетей, одна из которых (абстрактные гиперсети). Эта модель применялась для решения различных задач проектирования и оптимизации сетей связи, транспортных сетей, структурированной кабельной канализации и др. При этом каждый автор

заново описывал и реализовывал гиперсетевые структуры программно, используя как массивы и структуры данных в классических языках программирования типа Паскаль и С, так и объектное представление, и списочные структуры в С++, С#, Java и Delfi. Всё это затрудняет создание единых библиотек и затрудняет преемственность в разработках.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Гиперсеть это математическая модель для описания сложных систем сетевой структуры, с помощью которой, решают задачи проектирования и оптимизации различных сетей, таких как: сети связи, транспортные сети, информационные сети и т.д. Данные задачи актуальны и востребованы. С ростом инфраструктур, подлежащий оптимизации, растет и объем данных, которые требуется обрабатывать. Кроме того, все эти задачи сугубо прикладные, для которых характерно дальнейшая поддержка, модификация и документирование. Данное обстоятельство диктует требования системы, которая будет агрегировать данные, получение после выполнения алгоритмов и по требованию выдавать их либо специалисту, либо другим алгоритмам для дальнейшей работы. На текущий момент целостной системы и каких-либо стандартов для работы с данными, формализованными на языке теории гиперсетей не существует.

Ключевой особенностью гиперсетей является возможность описывания имеющих более 2 структур каждая из которых может быть вложена в другую. Например, рассматривая транспортную сеть города мы имеем дело сетью улиц, в которую отображены (вложены) множество маршрутов общественного транспорта. Традиционно, при решении задач в перечисленных областях использовались графы и гиперсети в качестве математической модели. Отдельные задачи хорошо изучены и решены в рамках теории графов. Но на практике, когда необходимо построить реальную систему, мы не можем абстрагироваться от всех взаимосвязей и внешних факторов, которые оказывают влияние на итоговую структуру гиперсети. В некоторых задачах мы имеем дело с нестационарными структурами – структуры, которые изменяются с течением времени. В таких случаях поставить четкую математическую задачу практически невозможно. Даже сама формулировка окажется громоздкой, не говоря уже о решении. Либо придется пренебрегать каким-либо воздействием. и решать частную

задачу. Для решения некоторого ряда задач можно использовать другие известные модели, такие как с вложенные графы. Но гиперсети являются более широкой моделью (вложенные графы могут быть описаны так же в рамках гиперсетей), которая может, достаточно лаконично описать сложные иерархические, многоуровневые сетевые модели. Например, гиперсети могут хорошо описывать город, где улицы – это первичная сеть, а вторичной сетью может являться общественный транспорт: автобусные маршруты, трамваи, маршрутки.

.

2 ПОСТАНОВКА ЗАДАЧИ

На сегодняшний момент не существует универсального решения, для унификации взаимодействий с гиперсетями. Не существует единого стандарта их описания при реализации алгоритмов решения прикладных задач.

Целью работы является создание единой системы объектов, позволяющей достаточно экономно по памяти и при этом удобно в использовании описывать гиперсети и использовать эти описания в решении различных задач, которые:

- позволят сохранять данные в виде гиперсети;
- обеспечат удобный механизм получения и обновления данных;
- обеспечат возможность частичного получения данных в различных представлениях;

Для реализации поставленной цели, в рамках работы будут решены следующие задачи:

- проектирование архитектуры программного комплекса;
- разработка и реализация алгоритмов для генерации часто используемых представлений. Таких как: список инцидентности, матрица смежности;
- разработка и реализация для общения пользователя с системой;

Важным аспектом работы является разработка удобочитаемых текстовых форматов для загрузки-выгрузки данных гиперсетей. С помощью этой функции будет вводиться или выводится вся информация о графе в нескольких представлениях, например, список смежности и матрица инцидентности.

3 АНАЛОГИ

В интернете можно найти системы для отрисовки и расчета параметров сети, но они не предоставляют нужного функционала для гиперсетей, поэтому было принято решение создать свою библиотеку взаимодействия с гиперсетью. Примеры таких систем:

1. graphonline.ru - сайт для визуализации графа и применения на нем встроенных алгоритмов.

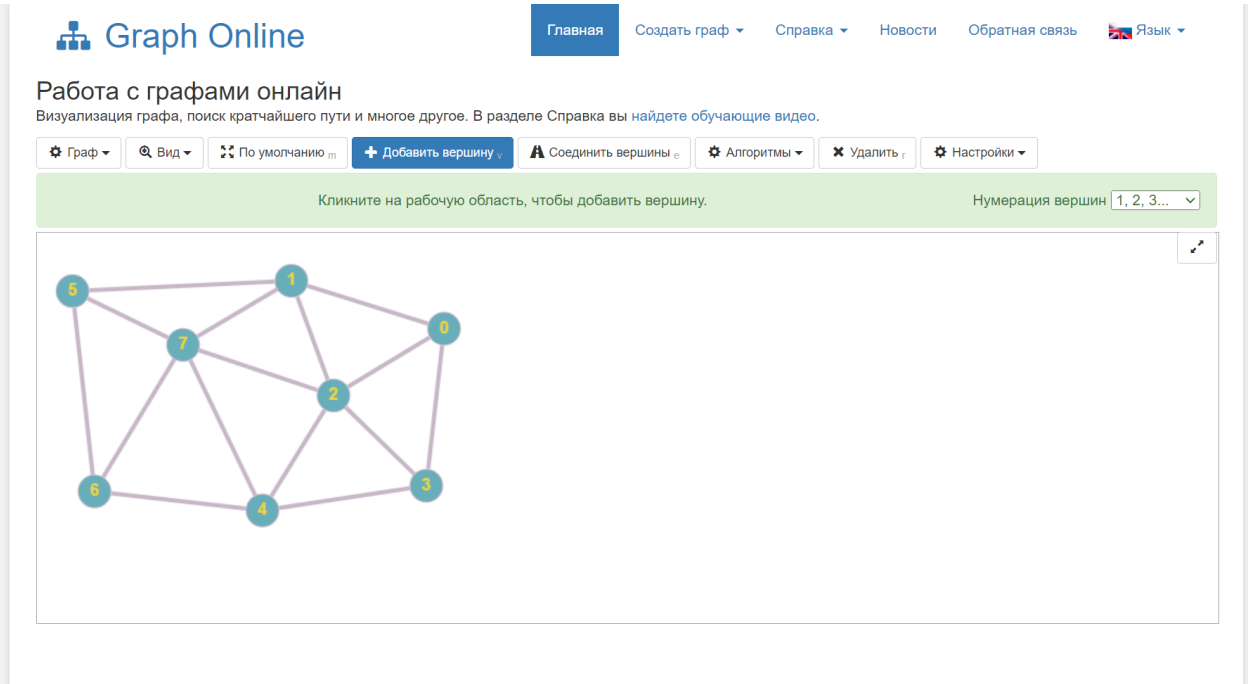

Рисунок 1 – Скриншот страницы сайта graphonline.ru

Этот сайт имеет графический интерфейс и набор некоторых алгоритмов, но отсутствует поддержка гиперсетей, что делает его не пригодным для работы с гиперсетями.

2. programforyou.ru/graph-redactor – сайт для работы с графами.

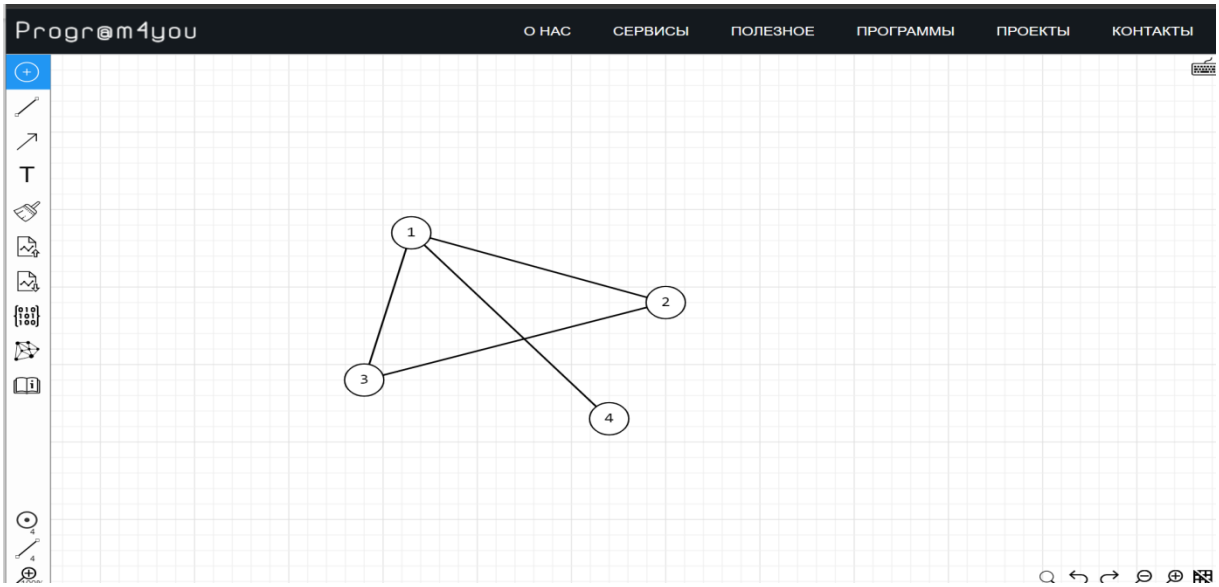

Рисунок 2 – Скриншот страницы сайта programforyou.ru/graph-redactor

Сайт владеет более обширным функционалом, по сравнению с предыдущим сайтом, такие, как чтение или сохранение графа в файл, более обширный выбор представления графа. Но, как у предыдущего сайта, отсутствует поддержка гиперсетей .

4 ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ СРЕДСТВУ

К разрабатываемым модуля были выведены следующие функциональные требования:

1. создание первичной сети;
2. добавление графа вторичной сети в первичную сеть;
3. добавление ветвей в граф вторичной сети;
4. связывание вершин ребрами;
5. механизм удобного взаимодействия с гиперсетью;
6. сохранения гиперсети в постоянной памяти;
7. генерация часто используемых представлений графа;
8. загрузка-выгрузка из текстового формата данных гиперсетей;

Также будут разработаны специальные алгоритмы, с помощью которых будет производиться тестирование всех компонентов библиотеки. Список реализуемых алгоритмов:

- Укладка вторичных сетей в первичную.
- Получение списка поврежденных вторичных сетей при разрушении заданных ребер первичной сети.

5 НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ К ПРОГРАММНОМУ СРЕДСТВУ

5.1 Требования к программному обеспечению

Кроссплатформенность – возможность работы на следующих операционных системах:

- Windows 7 и новее.
- Linux с версией ядра 4.4 и новее.

5.2 Требования к аппаратному обеспечению

Низкие системные требования, включающие в себя работу на большинстве устройств, для обеспечения качественной работы в составе большинства проектов.

5.3 Требования к надежности

В процессе выполнения расчетов не должны возникать критические ошибки. Программа должна по результатам работы выдавать точный результат.

6 ХАРАКТЕРИСТИКА ВЫБРАННЫХ ПРОГРАММЫХ СРЕД И СРЕДСТВ

Для реализации модулей было используется язык C#, и среда разработки Visual Studio. Также в качестве системы управления версий был выбран Git.

C# – объектно-ориентированный язык программирования, относящийся к C-подобным. Преимуществом данного языка можно считать быстродействие и безопасность. ближайшими конкурентами могут послужить Java, у которой есть проблемы со скоростью, и C++, который имеет прямой доступ к памяти, увеличивая количество ошибок при разработке приложения.

Microsoft Visual Studio – интегрированная среда разработки программного обеспечения. Предоставляет удобные средства для создания и редактирования исходного кода, и бесплатна для пользования студентам. Но из-за наличия большого количества инструментов и функционала, требует много оперативной памяти. Поддерживает множество плагинов для улучшения и упрощения процесса написания кода. Она является наиболее технологичной и практичной средой разработки на языке C#.

Git – популярная система управления версиями, хорошо отличающаяся от конкурентов своей быстротой, безопасностью и удобством. С помощью нее можно вносить изменения в проект, не боясь последствий, из-за возможности откатить их в прошлое, до текущих изменений состояния. Эта возможность позволяет вносить изменения, не боясь остаться с не рабочим приложением.

7 АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

7.1 Математическая постановка

Простая гиперсеть H = (X,V,R;P,F,W) состоит из следующих объектов (см. рис 1): X = (x_1, . . . ,x_n) – набор вершин; V = (v_1, . . . ,v_m) – набор ветвей (ребер графа первичной сети); R = (r_1, . . . ,r_g) – набор ребер (ребер графа вторичной сети); P:V → X×X – отображение, которое определяет граф PN = (X,V), который называется первичной сетью; W:R → X×X – отображение, которое определяет граф SN = (X,R), который называется вторичной сетью; F : R → 2^V – отображение ребер в маршруты PN.

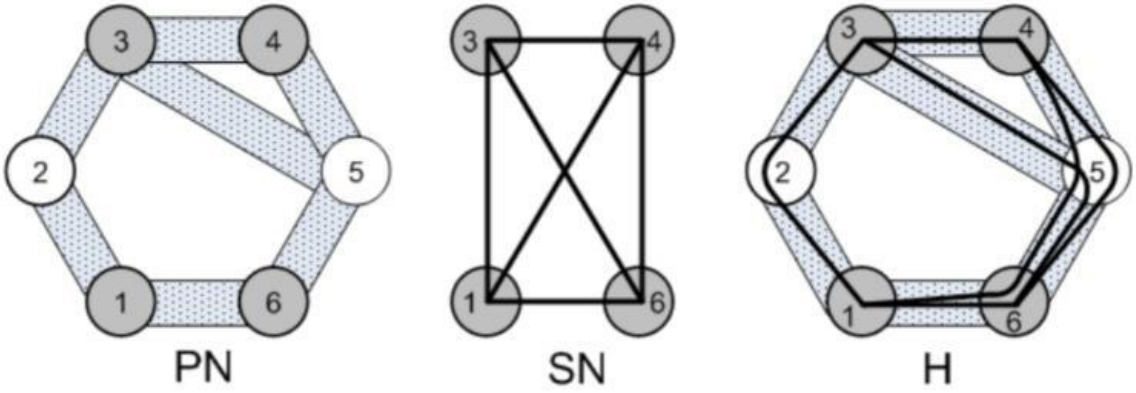

Рисунок 3 – Пример гиперсети

7.2 Алгоритмы

Для демонстрации работы библиотеки, была разработана программа, с помощью которой будет производиться проверка ее работоспособности и также может служить примером ее использования, вот список алгоритмов, которые будут реализованы в этой программе:

- Укладка вторичных сетей в первичную.
- Получение списка поврежденных вторичных сетей при разрушении заданных ребер первичной сети.

7.3 Алгоритм укладки вторичной сети в первичную

Целью работы этого алгоритма является объединение множества вторичных сетей в одну первичную. Результатом такого объединения является удобная работа с объединенными вторичными сетями как с одним графом. Это, например, позволяет определить существует ли путь от одной вершины до другой, при этом обе вершины могут находиться в разных вторичных сетях. На рисунке 4 можно увидеть пример работы алгоритма, на котором видно, как вторичные сети гиперсети объединяются в одну первичную сеть.

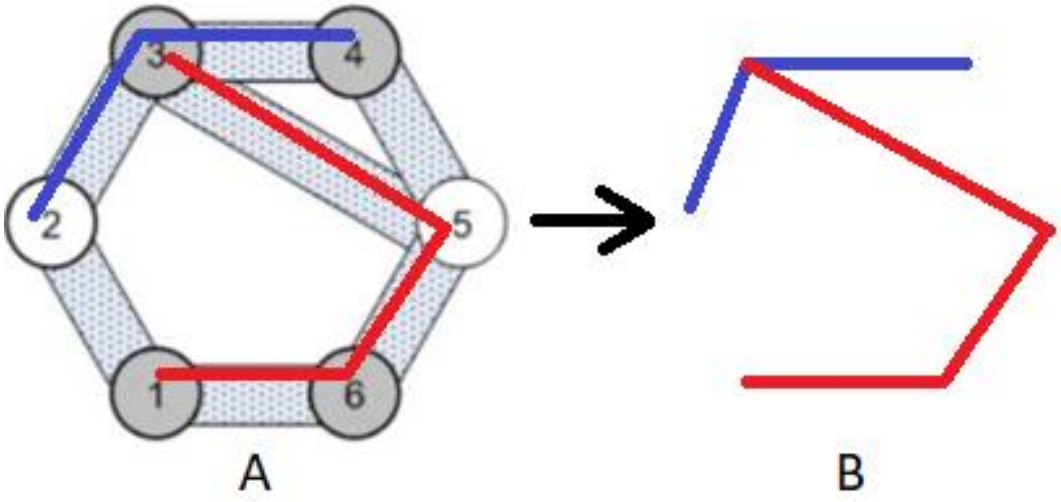

Рисунок 4 – Пример работы алгоритма укладки

7.4 Алгоритм получения списка поврежденных вторичных сетей при разрушении заданных ребер первичной сети.

После удаления ребер из первичной сети, во вторичной сети могут разорваться некоторые ветви, и для сохранения ее целостности будет использоваться этот алгоритм, который может помочь определить сети, которые будут разорваны и определения нового пути для их ветвей.

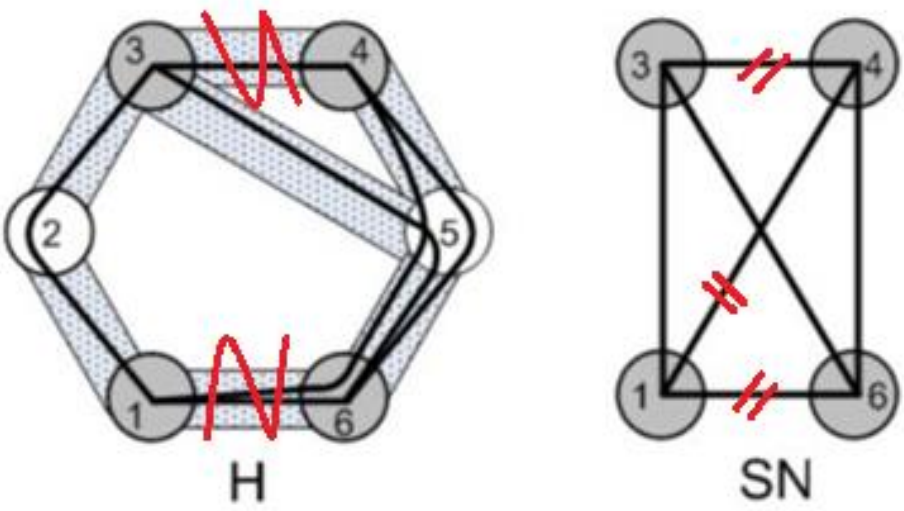

Рисунок 5 – Пример разрушенной вторичной сети

На рисунке 5 продемонстрирован пример, где после удаления ребер из первичной сети, была разрушена вторичная сеть. Алгоритм должен это выявить и сообщить пользователю.

8 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Так как целью работы является создание библиотеки, то ниже будут описаны входные и выходные данные для имеющихся методов.

8.1 GraphWriter.WriteGraphToFile

Записывает гиперсеть в файл. На вход принимает путь до файла, куда необходимо сохранить гиперсеть, класс гиперсети, тип представления графа в текстовом файле, матрица инцидентности, либо список смежности. Результатом выполнения метода является текстовый файл с сохраненной гиперсетью внутри.

8.2 GraphReader.ReadGraphFromFile

Считывает гиперсеть из файла. На вход принимает путь до файла, с которого необходимо считать гиперсеть. Результатом выполнения метода является считанная из файла гиперсеть.

8.3 GraphExtension.AsMatrix

Возвращает первичную сеть гиперсети в виде матрицы инцидентности. На вход принимает первичную сеть. Результатом выполнения метода является первичная сеть в виде матрицы инцидентности.

8.4 GraphExtension.DijkstrasAlgorithm

Находит кратчайшие пути от одной вершины графа до других. На вход принимает первичную сеть и номер верный, от которой необходимо найти кратчайший путь. Результатом выполнения является словарь расстояний до других вершин.

8.5 GraphExtension.AsList

Возвращает первичную сеть гиперсети в виде списка смежности. На вход принимает первичную сеть. Результатом выполнения метода является первичная сеть в виде матрицы инцидентности.

8.6 GraphExtension.ConnectivityComponent

Находи компонент связности графа первичной сети. На вход принимает первичную сеть. Результатом является словарь компонентов связности.

8.7 HyperGraphManipulation

Этот класс внутри хранит гиперсеть

В этом классе определены все методы добавления вершин, ребер, ветвей, вторичных сетей из гиперсети, а также их удаление. Такие методы принимают на вход одну из структур и добавляют их в гиперсеть. После добавления возвращает обновленную структуру, с уникальным идентификатором. Так как гиперсеть редко обновляется, но занимает много места, то в качестве основной структуры хранения данных был выбран отсортированный массив, с помощью которого удается достаточно экономно по памяти уместить всю информацию в гиперсети и проводить быстрый поиск нужных данных.

9 ОТЛАДКА И ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Библиотека умеет обрабатывать ошибки, которые могу возникнуть при считывании гиперсети из файла. Все эти ошибки помимо основного сообщения, хранят еще дополнительную информацию о ее местоположении. Например, если библиотека считает букву, хотя ожидает число, то оно выведет соответствующую ошибку.

Список всех возможных ошибок:

- Ошибка ввода отличного символа от числа.
- Некорректный тип представления входных данных.
- Отличное количество ребер от ранее заданного.
- Не удается создать несуществующее ребро между вершинами.

```
1   6a      // Введено некорректное число
2   List1   // Некорректный тип представления данных
3   6
4   1 2 1
5   2 3 5
6   3 7 8   // Попытка создать ребро между несуществующими вершинами
7   4 5 23
8   5 6 54  // Неверное количество ребер (5 вместо 6)
9   1
10  ShortedMatrix
11  6
12  1, 3, 1 : 1 2 3
13  3, 4, 1 : 3 4
14  4, 6, 1 : 4 5 6
15  6, 1, 1 : 6 1
16  3, 6, 1 : 3 4 5 6
17  1, 4, 1 : 1 6 5 4
```

Рисунок 6 – Пример некорректного файла.

```
One or more errors occurred. (Unexpected number. Line: 1)
One or more errors occurred. (Wrong input type, expected 'list' or 'matrix'. Line: 2)
One or more errors occurred. (Node id must be between 0 or 6. Line: 6)
One or more errors occurred. (List length too small. Line: 9)
```

Рисунок 7 – Пример вывода ошибок библиотекой.

10 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Так как результат разрабатываемого программного средства – библиотека, то для ее использования, пользователь должен загрузить ее, и добавить ссылку на проект, после этого необходимо будет подключить пространство имен библиотеки. Пример добавления библиотеки в проект указан на рисунке 8.

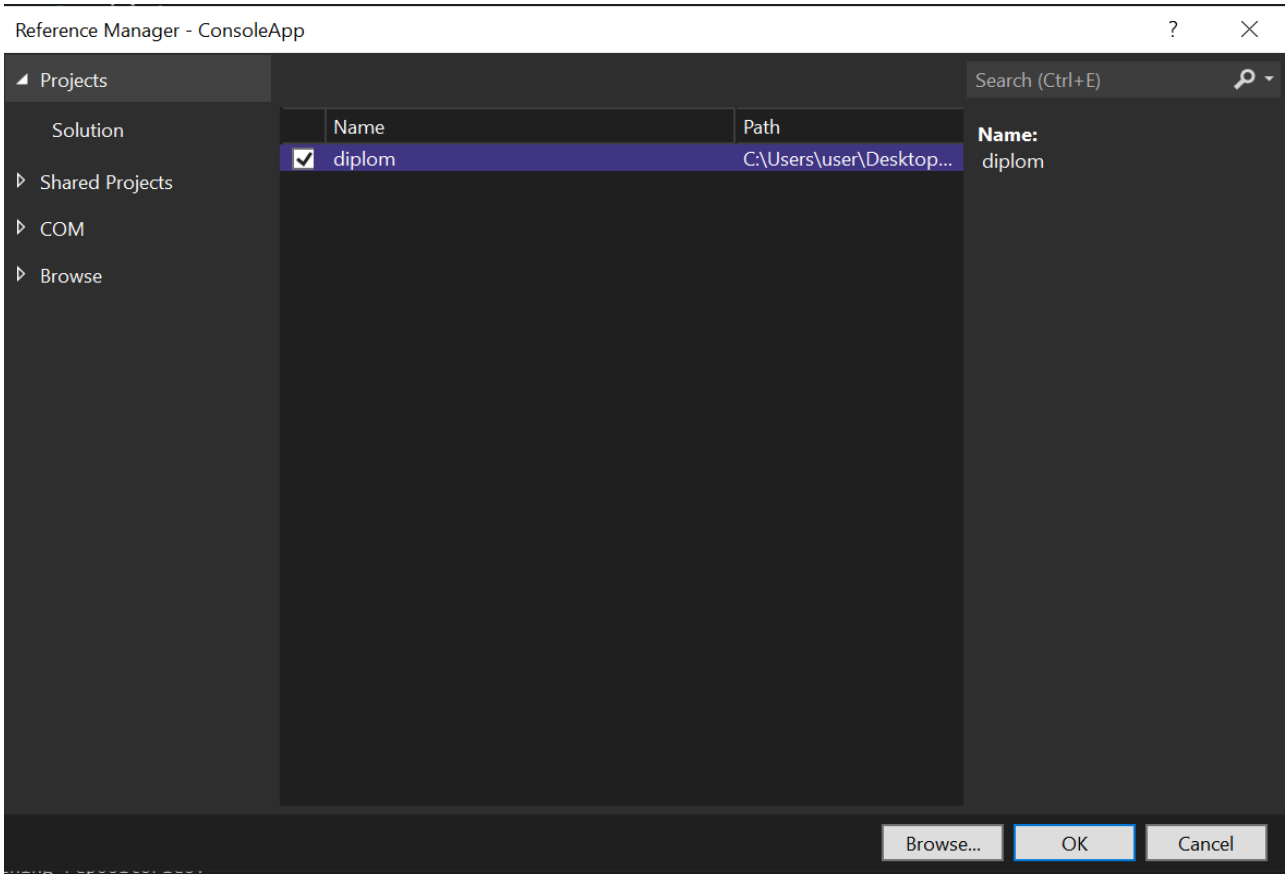

Рисунок 8 – Пример добавления библиотеки в проект

После добавление библиотеки, появится возможность использовать ее методами. На рисунке 9 представлен пример использования гиперсети, чтения гиперсети из файла, применения алгоритма, удаления вторичной сети из первичной, и сохранения гиперсети без вторичной сети в файлы в формате матрицы инцидентности. На рисунках 9 и 10 представлена гиперсеть до и после выполнения программы.

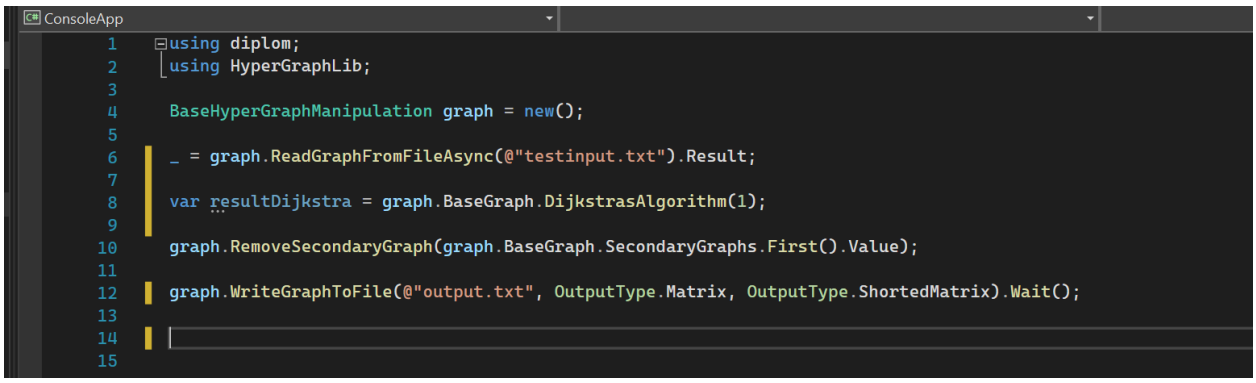

Рисунок 9 – Пример использования библиотеки

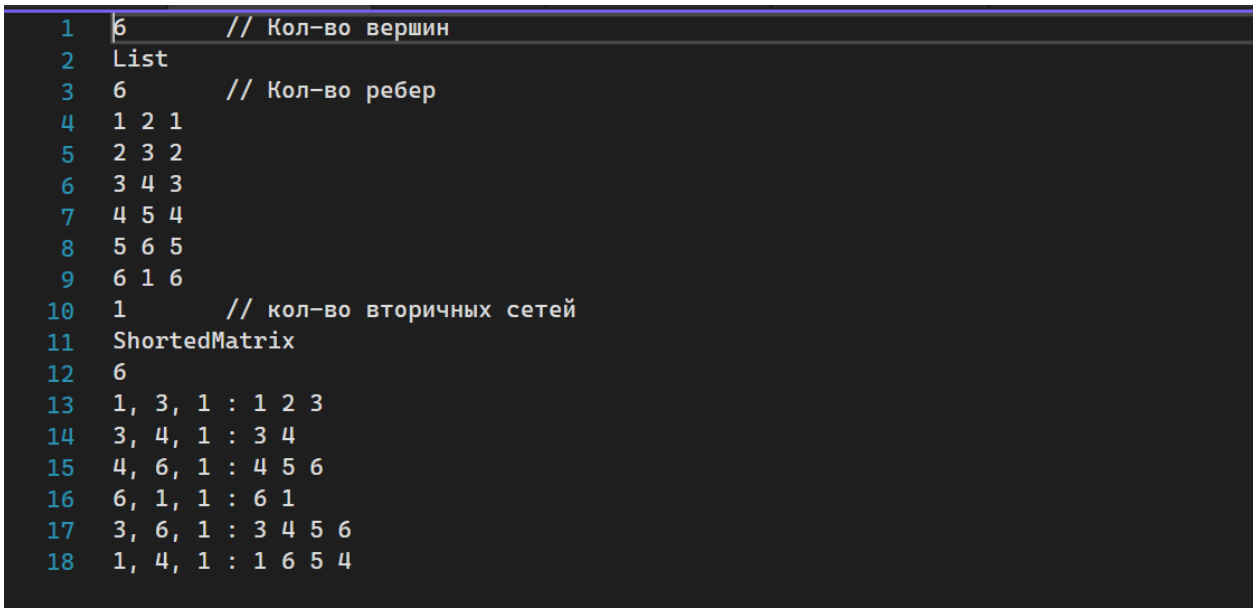

Рисунок 10 – Входной файл с гиперсетью

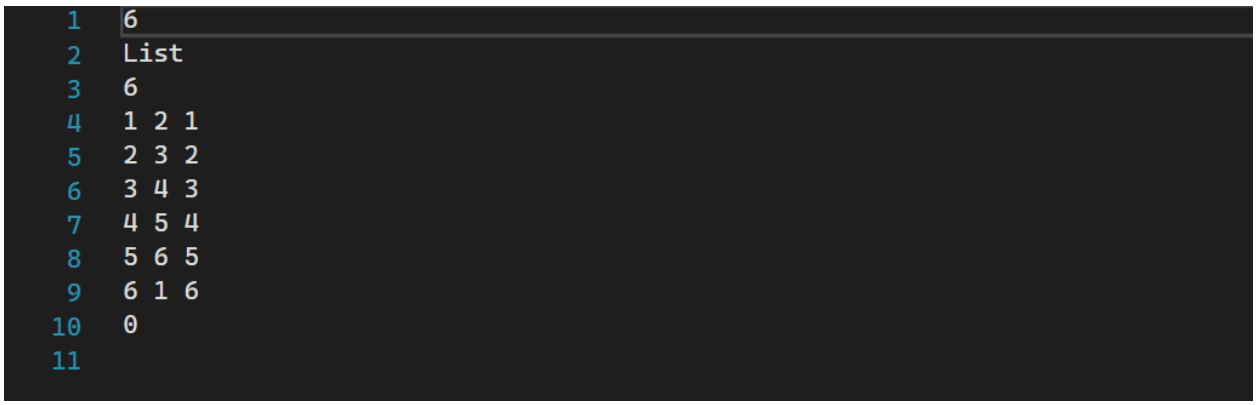

Рисунок 11 – Результат выполнения программы, гиперсеть без вторичной сети

ЗАКЛЮЧЕНИЕ

В результате была разработана библиотека, которая позволяет достаточно экономно по памяти описывать гиперсети и при этом удобна в использовании, которая может:

- сохранять данные в гиперсеть;
- обеспечить удобные механизмы обновления данных;
- обеспечить возможность получения данных в различных представлениях, таких как: матрица инцидентности и список смежности;

Программа обладает достаточным функционалом для удобного использования. Также библиотека была успешно протестирована на нескольких алгоритмах для подтверждения ее работоспособности и эффективности.

Библиотека была выложена в открытый доступ на Github, чтобы предоставить доступ для использования или изменения библиотеки.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Попков В. К. Математичиские модели связности. — Новосибирск : ИВМиМГ СО РАН, 2006. — 490 с.
2. Оре О. Графы и их применение. — Рипол Классик, 2002.
3. Golumbic M. C., Kaplan H., Shamir R. Graph sandwich problems // Journal of Algorithms. — 1995. — Т. 19, № 3. — С. 449—473.
4. Poulovassilis A., Levene M. A nested-graph model for the representation and manipulation of complex objects // ACM Transactions on Information Systems (TOIS). — 1994. — Т. 12, № 1. — С. 35—68.
5. Levene M., Loizou G. A graph-based data model and its ramifications // IEEE Transactions on Knowledge and Data Engineering. — 1995. — Т. 7, № 5. — С. 809—823.
6. Попков В. К. Гиперсети и структурные модели сложных систем // Математические и имитационные модели сложных систем. — Новосибирск, 1981. — С. 26—48.
7. Попков В. К. Применение теории S-гиперсетей для моделирования систем сетевой структуры // Проблемы информатики. — Новосибирск, 2010. — № 4. — С. 17—40.
8. Галямов В. А. О задаче построения структурированных кабельных систем // Новосиб. Ун-та. — Новосибирск, 2005. — С. 36.
9. Dijkstra F., Andree B., Koymans K., van der Hama J., Grosso P., de Laat C. A multi-layer network model based on ITU-T G.805 // Computer Networks, 2008. N 52. P 1927 - 1937.
10. Rodionov A., Rodionova O. Random Hypernets in Reliability Analysis of Multilayer Networks // Springer International Publishing Switzerland 2015, Computational Problems in Science and Engineering, Lecture Notes in Electrical Engineering 343, P. 307 - 313.

ПРИЛОЖЕНИ А

```
public class MyCollection<TData> : Dictionary<int, TData>, IConnectionCollection<TData>
   { }

   public class BaseHyperGraphManipulation : IBaseHyperGraphManipulation
   {
      private PrimaryGraph _graph = new()
      {
         Branches = new MyCollection<Branch>(),
         Nodes = new MyCollection<Node>(),
         Edges = new MyCollection<Edge>(),
         SecondaryGraphs = new MyCollection<SecondaryGraph>(),
      };
      public PrimaryGraph BaseGraph
      {
         get => _graph;
         set => _graph = value;
      }

      public int LastId { private set; get; }
      private int GetId()
      {
         return ++LastId;
      }

      public SecondaryGraph AddSecondaryGraph()
      {
         return AddSecondaryGraph(CreateSecondaryGraph());
      }

      public SecondaryGraph AddSecondaryGraph(SecondaryGraph secondaryGraph)
      {
         secondaryGraph.Id = GetId();
         secondaryGraph.PrimaryGraph = _graph;

         _graph.SecondaryGraphs.Add(secondaryGraph.Id, secondaryGraph);

         return secondaryGraph;
      }
      public Branch AddBranch()
      {
         return AddBranch(CreateBranch());
      }
      public Branch AddBranch(Branch branch)
      {
         branch.Id = GetId();
         branch.PrimaryGraph = _graph;

         _graph.Branches.Add(branch.Id, branch);
```

```
    foreach (var i in branch.Edges)
    {
        i.Value.Branches.Add(branch.Id, branch);
    }

    branch.First.Branches.Add(branch.Id, branch);
    branch.Second.Branches.Add(branch.Id, branch);

    branch.SecondaryGraph.Branches.Add(branch.Id, branch);

    return branch;
}

public Node AddNode()
{
    return AddNode(CreateNode());
}

public Node AddNode(Node node)
{
    node.Id = GetId();
    node.PrimaryGraph = _graph;

    _graph.Nodes.Add(node.Id, node);
```