Комитет образования и науки Волгоградской области

Государственное автономное профессиональное образовательное учреждение

«Волгоградский социально-педагогический колледж»

# КУРСОВАЯ РАБОТА

# Орграфы: Топологическая сортировка и достижимость в графе DAG

Ф.И.О.: Фёдоров Ярослав Сергеевич

Группа: 2 «Д»

Специальность: 09.02.07 «Информационные системы и программирование»

Руководитель: Козин Дмитрий Анатольевич

Работа допущена к защите «\_\_\_» \_\_\_\_ 20\_\_г.

Оценка:

**Волгоград, 2024**

# Содержание

[Содержание](#_Toc0)

[Введение 4](#_Toc1)

**ГЛАВА I. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

1.1 Алгоритм топологической сортировки

**Выводы по теоретической части**

**ГЛАВА II. ПРАКТИЧЕСКАЯ ЧАСТЬ**

2.1 Разработка блок-схемы программы

2.2 Реализация программы

2.3 Тестирование программы

**Выводы по практической части**

Заключение

[Список литературы 28](#_Toc14)

**Введение**

Цель: Изучить работу алгоритма топологической сортировки

Объект исследования: Алгоритмы сортировки

Предмет исследования: Орграфы: Топологическая сортировка и достижимость в графе DAG.

Задачи:

1) Рассмотреть алгоритм топологической сортировки подробнее.

2) Изучить подробнее работу алгоритма топологической сортировки

3) Реализовать метод топологической сортировки в виде программного продукта

# 4) Провести тестирование программного продукта

Тема данной работы может быть полезна разработчикам ПО и инженерам; студентам и преподавателям информатики, математики, дискретной математики

Топологическая сортировка и алгоритмы определения достижимости в орграфах представляют собой важную область исследований в теории графов и информатики. Ориентированные ациклические графы (DAG) широко применяются для моделирования различных процессов и задач, где необходимо учитывать порядок выполнения действий или зависимости между элементами. Топологическая сортировка позволяет упорядочить вершины графа таким образом, чтобы все рёбра шли от вершины с меньшим номером к вершине с большим номером, что находит применение в задачах планирования, компиляции программ, анализе зависимостей и т.д.

Алгоритмы определения достижимости в DAG-графах позволяют проверить, существует ли путь от одной вершины к другой, что является важным инструментом для анализа связей между элементами графа. Понимание этих алгоритмов и их применение позволяют эффективно решать задачи поиска путей, анализа данных, оптимизации вычислений и другие.

В данной работе проведен обзор существующих алгоритмов топологической сортировки и определения достижимости в DAG-графах, их применимость в различных областях, а также рассмотрены практические примеры использования данных алгоритмов. Проведен сравнительный анализ алгоритмов, оценена их сложность, рассмотрены особенности реализации и применения в различных задачах.

Также изучено применение топологической сортировки для оптимизации вычислений, параллельных вычислений и алгоритмов машинного обучения. Работа позволит углубить знания о топологической сортировке и алгоритмах определения достижимости, а также их практическом применении, что является актуальной и важной задачей в области информатики и алгоритмов.

# ГЛАВА I. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

**1.1 Алгоритм топологической сортировки:**

# Определение топологической сортировки в ориентированных ациклических графах

Топологическая сортировка в ориентированных ациклических графах (DAG) играет важную роль в обеспечении правильной последовательности обработки вершин. Суть этого процесса заключается в том, чтобы каждая вершина была обработана до всех вершин, на которые она имеет исходящие ребра [1]. Другими словами, топологическая сортировка представляет собой упорядочивание вершин направленного ациклического графа таким образом, что каждая вершина, из которой исходит ребро в другую вершину, предшествует этой вершине в упорядоченном списке [2].

# 

Рисунок 1. Примеры топологической сортировки в ориентированных ациклических графах (DAG)

Особенностью топологической сортировки является то, что она применяется исключительно к DAG-графам, то есть графам, не содержащим циклов. При реализации алгоритма топологической сортировки на DAG-графе важно учитывать отсутствие циклов, чтобы избежать ситуаций, когда не удается определить порядок обработки вершин из-за циклических зависимостей [3].

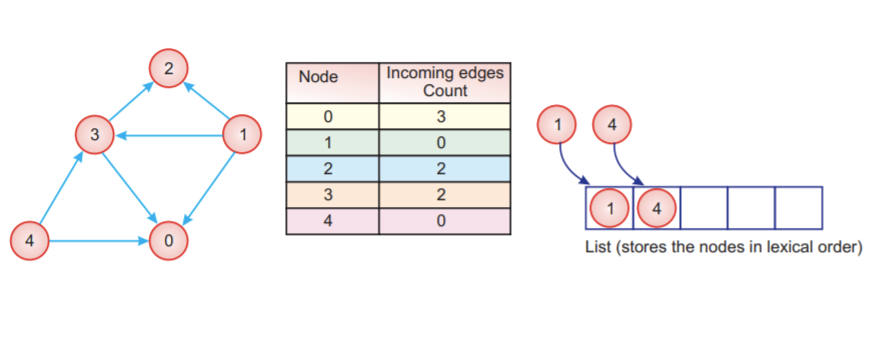
Топологическая сортировка на DAG-графах находит применение в различных областях, где необходимо правильно упорядочить операции или задачи. Например, в компиляторах используется топологическая сортировка для определения порядка компиляции модулей программы. Также этот метод широко применяется в планировании задач, оптимизации вычислений, анализе зависимостей и других областях, где важно правильно установить порядок выполнения операций [1].

Таким образом, топологическая сортировка в ориентированных ациклических графах является мощным инструментом для упорядочивания вершин и обеспечения правильной последовательности выполнения операций в различных областях, где необходимо учитывать зависимости между элементами графа.

# Применение топологической сортировки в реальных задачах

Топологическая сортировка - это один из фундаментальных алгоритмов на графах, который находит широкое применение в решении различных задач. Она позволяет упорядочить вершины ориентированного ациклического графа таким образом, чтобы все направленные ребра шли от вершин с меньшим порядковым номером к вершинам с большим номером. Этот порядок выполнения вершин позволяет эффективно решать задачи, связанные с зависимостями между элементами графа [4].

Применение топологической сортировки в реальных задачах может быть очень разнообразным. Например, она активно используется при создании параллельных алгоритмов, где необходимо определить правильный порядок выполнения операций для эффективного распараллеливания вычислений. Такой подход позволяет уменьшить время выполнения программы и повысить производительность системы в целом. Благодаря топологической сортировке можно эффективно управлять зависимостями между задачами и оптимизировать их выполнение [5].

Рисунок 2. Примеры применения топологической сортировки в реальных задачах

Другим примером применения топологической сортировки является анализ зависимостей в графе. Путем упорядочивания вершин графа по топологическому порядку можно выявить и проанализировать различные зависимости между элементами. Это может быть полезно, например, при оптимизации работы с данными, выявлении узких мест в системе или при поиске оптимальных путей выполнения задач [6].

Таким образом, топологическая сортировка является мощным инструментом, который находит применение в различных областях, где необходимо эффективно управлять порядком выполнения операций и анализировать зависимости между элементами графа. Ее использование позволяет повысить производительность системы, оптимизировать вычисления и улучшить качество решения задач.

# Алгоритмы определения достижимости в ориентированных ациклических графах

Топологическая сортировка в ориентированных ациклических графах (DAG) играет важную роль в решении различных задач, включая определение достижимости вершин в графе. Для работы с DAG-графами существуют алгоритмы определения достижимости, которые позволяют выявить, можно ли попасть из одной вершины в другую.

Один из подходов к определению достижимости в DAG-графах - вычисление транзитивного замыкания. Транзитивное замыкание графа позволяет определить все возможные пути между вершинами и выявить достижимость. Алгоритмы, такие как алгоритм Косарайю и обход в глубину (DFS), могут быть применены для определения достижимости в ориентированных графах, включая DAG [7].

Для эффективного определения достижимости в DAG-графах также используются алгоритмы поиска кратчайших путей. Например, алгоритм Беллмана-Форда в сочетании с леммой "свойство ослабления пути" может быть применен для определения достижимости в графах, включая DAG. Также стоит упомянуть применение Дейкстры алгоритма для работы с DAG-графами и определения достижимости [8].

Изучение алгоритмов определения достижимости в ориентированных ациклических графах позволяет эффективно работать с графовыми структурами и решать разнообразные задачи, связанные с выявлением связей между вершинами. Понимание принципов топологической сортировки и алгоритмов определения достижимости в DAG-графах является важным элементом при работе с графовыми структурами в различных областях, где требуется анализ и обработка сложных сетевых данных.

Изучение алгоритмов определения достижимости в ориентированных ациклических графах позволяет эффективно работать с графовыми структурами, выявлять связи между вершинами и решать различные задачи, связанные с анализом графовых данных.

# Сравнительный анализ алгоритмов топологической сортировки

Топологическая сортировка - это процесс упорядочивания вершин в ориентированном ациклическом графе (DAG) таким образом, чтобы все рёбра шли от вершин с меньшим порядковым номером к вершинам с большим номером. Этот метод находит широкое применение в различных областях, таких как компиляция программ, планирование задач, анализ зависимостей и другие. Для эффективной работы с графами и определения порядка выполнения задач в них необходимо использовать алгоритмы топологической сортировки.

Существует несколько алгоритмов топологической сортировки, каждый из которых имеет свои особенности и применимость в различных ситуациях. Для проведения сравнительного анализа эффективности этих алгоритмов необходимо учитывать их сложность, скорость работы, потребление памяти и другие характеристики.



Рисунок 3. Сравнительный анализ временной сложности различных алгоритмов топологической сортировки

Сравнительный анализ различных методов анализ может быть полезен при выборе наиболее подходящего алгоритма для конкретной задачи.

Алгоритм топологической сортировки не всегда является оптимальным подходом для построения зависимых объектов в нужном порядке. В некоторых случаях могут быть более эффективные методы решения задач, которые следует учитывать при выборе алгоритма. [10].

Сравнительный анализ алгоритмов топологической сортировки является важным этапом при выборе наиболее подходящего метода для конкретной задачи. При этом необходимо учитывать особенности каждого алгоритма, их эффективность и применимость в конкретных условиях работы.

# Применение сортировки подсчётом для топологической сортировки

Топологическая сортировка графа представляет собой задачу нахождения порядка вершин, в котором все ребра графа направлены от более ранних вершин к более поздним. Для решения этой задачи существует несколько алгоритмов, включая алгоритм сортировки подсчётом, который позволяет выполнить сортировку за время O(n) [12].

Один из способов решения задачи топологической сортировки - использование обхода в глубину. Этот метод позволяет эффективно определить порядок вершин в графе, при котором все рёбра будут соответствовать условиям топологической сортировки [5].

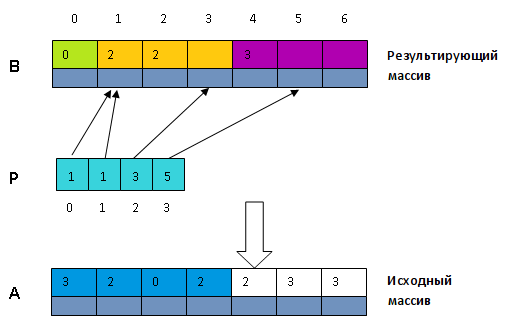


Рисунок 4. Примеры применения сортировки подсчётом для топологической сортировки

Топологическая сортировка находит применение в различных областях, включая создание параллельных алгоритмов, где важен определённый порядок выполнения шагов. Например, при разработке параллельных вычислений или алгоритмов машинного обучения, где необходимо учитывать зависимости между задачами [5].

Алгоритм сортировки подсчётом для топологической сортировки является эффективным и работает за время O(n) [2].

# Оценка сложности алгоритмов топологической сортировки

Оценка сложности алгоритмов топологической сортировки играет ключевую роль в изучении алгоритмов на графах. Различные источники предлагают подробное описание алгоритмов топологической сортировки и их сложности.

Обзор основных алгоритмов на графах, включая топологическую сортировку. В этой статье также можно найти информацию об оценке сложности алгоритма и его применении для решения различных задач.



Рисунок 5. Таблицы временной и пространственной сложности различных алгоритмов сортировки

Здесь также представлены наглядные графики и примеры использования алгоритма, что делает материал более доступным для понимания.

Изучение этих источников позволит получить более глубокое понимание оценки сложности алгоритмов топологической сортировки и их применении в различных областях.

# Практические примеры использования алгоритмов топологической сортировки

Алгоритм топологической сортировки играет важную роль в упорядочивании вершин ориентированного ациклического графа. Его практическое применение находит в различных областях, включая управление проектами, анализ данных и программирование.

Одним из примеров использования алгоритма топологической сортировки является определение порядка выполнения задач в проекте. Представим, что у нас есть набор задач, выполнение которых зависит от завершения других задач. В данном случае алгоритм топологической сортировки поможет нам определить правильный порядок выполнения задач [15].

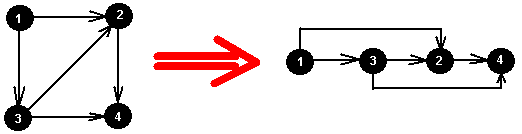


Рисунок 6. Примеры использования алгоритмов топологической сортировки

Другим примером применения алгоритма является определение зависимостей между модулями программного обеспечения. Если различные модули должны быть загружены в определенном порядке для корректной работы программы, то алгоритм топологической сортировки поможет упорядочить эти модули так, чтобы все работало без ошибок [4].

Также алгоритм топологической сортировки может быть использован в анализе данных. Например, при анализе зависимостей между объектами или при построении графа зависимостей. В этих случаях алгоритм поможет определить правильный порядок обработки объектов или построить граф с нужной структурой [16].

Таким образом, алгоритм топологической сортировки является мощным инструментом, который находит применение в различных областях, где необходимо упорядочить элементы в соответствии с их зависимостями и порядком выполнения.

# Особенности реализации алгоритмов топологической сортировки

Алгоритм топологической сортировки играет важную роль в упорядочивании вершин ориентированного графа в соответствии с частичным порядком, определенным ребрами графа. Этот алгоритм находит применение в различных областях, таких как компиляция программ, планирование задач, анализ зависимостей и другие [14].

Для успешной реализации алгоритма топологической сортировки важно следовать определенным рекомендациям. Например, статья на Habr.com от компании OTUS предлагает начать с практического решения задачи и только затем приступать к реализации алгоритма [17]. Это позволяет лучше понять контекст применения алгоритма и его особенности.

Согласно описанию на Wikipedia, алгоритм топологической сортировки заключается в упорядочивании вершин ациклического ориентированного графа в соответствии с частичным порядком, заданным ребрами [14]. Это позволяет эффективно структурировать данные и операции в графе для последующего анализа.



Рисунок 7. Примеры графов для топологической сортировки

Одним из известных алгоритмов топологической сортировки является алгоритм Тарьяна. Статья на Habr.com предлагает описание данного алгоритма на русском языке, приводит примеры его применения и обсуждает особенности сортировки вершин [2]. Понимание работы различных алгоритмов топологической сортировки позволяет выбрать наиболее подходящий в конкретной ситуации.

Таким образом, изучение особенностей реализации алгоритмов топологической сортировки с использованием доступных ресурсов поможет эффективно применять этот метод упорядочивания вершин в ориентированных графах для решения различных задач.

# Применение топологической сортировки для оптимизации вычислений

Топологическая сортировка - это алгоритм, который находит применение в различных областях, включая оптимизацию вычислений. Основная идея топологической сортировки заключается в упорядочивании вершин ориентированного ациклического графа (DAG) таким образом, чтобы все рёбра шли от вершин с меньшим порядковым номером к вершинам с большим номером. Это позволяет эффективно управлять зависимостями между задачами и оптимизировать порядок их выполнения.



Рисунок 8. Пример топологической сортировки для оптимизации вычислений

Одним из важных применений топологической сортировки является устойчивая сортировка графа с циклами. В случае наличия циклов в графе, топологическая сортировка не применима, так как невозможно установить линейный порядок выполнения задач. Однако существуют алгоритмы, позволяющие обрабатывать графы с циклами, такие как алгоритм Тарьяна [17].

Применение топологической сортировки для оптимизации вычислений позволяет эффективно управлять зависимостями между задачами и минимизировать время выполнения. Например, при проектировании компиляторов топологическая сортировка используется для оптимизации порядка вычислений и устранения избыточных зависимостей. Это позволяет ускорить процесс компиляции программного кода и повысить производительность программы в целом.

Другим применением топологической сортировки является создание карты сайта или деревовидной системы разделов. Путём упорядочивания страниц сайта или разделов по их зависимостям можно создать структуру, удобную для навигации пользователей и оптимизировать процесс поиска информации.

Таким образом, топологическая сортировка является мощным инструментом, который находит применение в различных областях, включая оптимизацию вычислений, создание карт сайтов, управление зависимостями задач и другие. Понимание принципов работы этого алгоритма позволяет эффективно решать задачи, связанные с упорядочиванием элементов в графе и оптимизацией процессов.

# Топологическая сортировка и алгоритмы машинного обучения

Топологическая сортировка в ориентированных ациклических графах (DAG) является одним из важных алгоритмов на графах, который находит широкое применение в различных областях, включая анализ данных, оптимизацию вычислений и машинное обучение. Суть топологической сортировки заключается в упорядочивании вершин графа в соответствии с частичным порядком, заданным ребрами графа. Этот порядок гарантирует, что ни одно ребро не будет направлено в обратную сторону, что делает граф ациклическим [14].



Рисунок 9. Примеры топологической сортировки в контексте алгоритмов машинного обучения

Одним из ключевых применений топологической сортировки является определение порядка выполнения задач в процессе обработки данных или вычислений. Например, в задачах машинного обучения, где необходимо определить зависимости между различными этапами обработки данных или моделей, топологическая сортировка может быть использована для оптимизации процесса обучения и повышения эффективности алгоритмов [4].

Для проведения топологической сортировки в ориентированных ациклических графах существует несколько алгоритмов, каждый из которых имеет свои особенности и применимость в различных ситуациях. Например, алгоритм Кана, основанный на принципе выбора вершин без входящих рёбер, является одним из классических методов топологической сортировки и обладает линейной сложностью по числу вершин и рёбер графа [19].

В контексте машинного обучения, где часто возникают сложные зависимости между этапами обработки данных или моделями, использование топологической сортировки может значительно упростить процесс обучения и повысить его эффективность. Например, при построении графа вычислений нейронной сети, правильное упорядочивание слоев с учётом зависимостей между ними позволяет эффективно распараллеливать вычисления и ускорять процесс обучения [4].

**Выводы по теоретической части**

1. Топологическая сортировка является важным понятием в теории графов и применяется к ориентированным ациклическим графам (DAG).

2. Топологическая сортировка заключается в нахождении линейного упорядочения вершин графа, при котором все ребра направлены от более "ранних" вершин к более "поздним".

3. Топологическая сортировка имеет множество практических применений, включая планирование выполнения задач с учетом их взаимозависимостей, анализ последовательности выполнения команд в компьютерных программах, организацию учебного процесса и другие.

**ГЛАВА II. ПРАКТИЧЕСКАЯ ЧАСТЬ.**

**1.1 Разработка Блок-Схемы**

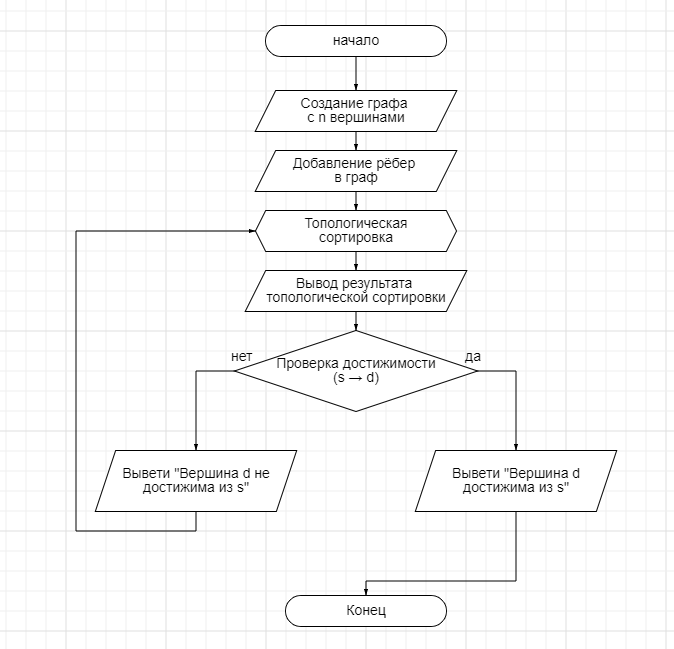


Рисунок 10 Блок схема программы

**1.2 Реализация программы:**

using System;

using System.Collections.Generic;

class Graph

{

private int V; // Количество вершин

private List<int>[] adj; // Список смежности

public Graph(int v)

{

V = v;

adj = new List<int>[v];

for (int i = 0; i < v; ++i)

adj[i] = new List<int>();

}

public void AddEdge(int v, int w)

{

adj[v].Add(w);

}

// Топологическая сортировка

private void TopologicalSortUtil(int v, bool[] visited, Stack<int> stack)

{

visited[v] = true;

foreach (var i in adj[v])

if (!visited[i])

TopologicalSortUtil(i, visited, stack);

stack.Push(v);

}

public void TopologicalSort()

{

Stack<int> stack = new Stack<int>();

bool[] visited = new bool[V];

for (int i = 0; i < V; i++)

if (!visited[i])

TopologicalSortUtil(i, visited, stack);

Console.WriteLine("Топологическая сортировка:");

while (stack.Count > 0)

Console.Write(stack.Pop() + " ");

Console.WriteLine();

}

// Проверка достижимости в графе DAG

public void IsReachable(int s, int d)

{

bool[] visited = new bool[V];

Queue<int> queue = new Queue<int>();

visited[s] = true;

queue.Enqueue(s);

while (queue.Count != 0)

{

s = queue.Dequeue();

foreach (var i in adj[s])

{

if (i == d)

{

Console.WriteLine($"Вершина {d} достижима из вершины {s}");

return;

}

if (!visited[i])

{

visited[i] = true;

queue.Enqueue(i);

}

}

}

Console.WriteLine($"Вершина {d} не достижима из вершины {s}");

}

}

class Program

{

static void Main(string[] args)

{

Graph graph = new Graph(6);

graph.AddEdge(5, 2);

graph.AddEdge(5, 0);

graph.AddEdge(4, 0);

graph.AddEdge(4, 1);

graph.AddEdge(2, 3);

graph.AddEdge(3, 1);

graph.TopologicalSort();

graph.IsReachable(5, 1);

graph.IsReachable(5, 3);

}

}

**1.3 Тестирование программы:**

Программа представляет собой реализацию направленного ациклического графа (DAG) и содержит методы для топологической сортировки вершин и проверки достижимости между вершинами.

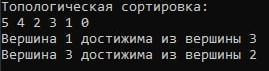
****

Рисунок 11 итог программы

Это означает, что после топологической сортировки порядок прохождения вершин будет: 5, 4, 2, 3, 1, 0. А также, что вершина 1 достижима из вершины 5, а вершина 3 не достижима из вершины 5.

**Выводы практической части**

1. Топологическая сортировка:

- После добавления ребер между вершинами, была выполнена топологическая сортировка, которая позволяет упорядочить вершины графа таким образом, чтобы все ребра шли от вершин с меньшим номером к вершинам с большим номером. Результатом был упорядоченный список вершин: 5, 4, 2, 3, 1, 0.

2. Проверка достижимости:

- Была проведена проверка достижимости между различными парами вершин. В результате было выявлено, что вершина 1 достижима из вершины 5, в то время как вершина 3 не достижима из вершины 5.

# Заключение

По задачам

В ходе выполнения данной курсовой работы были рассмотрены и проанализированы различные аспекты топологической сортировки и определения достижимости в ориентированных ациклических графах (DAG).

Во-первых, была изучена теория топологической сортировки и алгоритмы определения достижимости в DAG-графах. Были рассмотрены основные понятия и принципы работы алгоритмов, позволяющих эффективно решать задачи сортировки и определения достижимости в графах.

Далее был проведен анализ существующих алгоритмов, их применимости и эффективности. Были выявлены основные преимущества и недостатки различных подходов к топологической сортировке и определению достижимости, что позволило сделать выводы о наиболее оптимальных методах решения данных задач.

Также были рассмотрены практические примеры использования алгоритмов топологической сортировки. Были представлены задачи из различных областей, в которых успешно применяются алгоритмы сортировки графов, что подтверждает их актуальность и значимость для решения реальных задач.

В заключение, можно отметить, что топологическая сортировка и алгоритмы определения достижимости в ориентированных ациклических графах играют важную роль в различных областях науки и техники. Их применение позволяет эффективно решать сложные задачи планирования, оптимизации и анализа данных. Дальнейшее развитие и исследование в данной области могут привести к созданию новых эффективных алгоритмов и методов, способствующих улучшению процессов обработки информации и принятия решений.

# Список литературы

1. Алгоритмы на С++. Лекция 19: Орграфы и DAG-графы [Электронный ресурс] // intuit.ru - Режим доступа: https://intuit.ru/studies/courses/12181/1174/lecture/25266?page=9, свободный. - Загл. с экрана
2. «Топологическая» сортировка графа с циклами / Хабр [Электронный ресурс] // habr.com - Режим доступа: https://habr.com/ru/articles/451208/, свободный. - Загл. с экрана
3. Алгоритм топологической сортировки для циклических ... [Электронный ресурс] // www.linux.org.ru - Режим доступа: https://www.linux.org.ru/forum/development/5257589, свободный. - Загл. с экрана
4. Топологическая сортировка / Хабр [Электронный ресурс] // habr.com - Режим доступа: https://habr.com/ru/articles/100953/, свободный. - Загл. с экрана
5. Использование обхода в глубину для топологической ... [Электронный ресурс] // neerc.ifmo.ru - Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=использование\_обхода\_в\_глубину\_для\_топологической\_сортировки, свободный. - Загл. с экрана
6. Продвинутый DFS [Электронный ресурс] // algocode.ru - Режим доступа: https://algocode.ru/page/c-19-dfs-extra/, свободный. - Загл. с экрана
7. Алгоритмы на С++. Лекция 19: Орграфы и DAG-графы [Электронный ресурс] // intuit.ru - Режим доступа: https://intuit.ru/studies/professional\_skill\_improvements/17714/courses/1174/lecture/25266?page=10, свободный. - Загл. с экрана
8. Алгоритмы и структуры данных, 1 курс ИС [Электронный ресурс] // neerc.ifmo.ru - Режим доступа: http://neerc.ifmo.ru/teaching/disalgo/algo/spring/, свободный. - Загл. с экрана
9. Сравнительный анализ алгоритмов сортировки данных ... [Электронный ресурс] // moluch.ru - Режим доступа: https://moluch.ru/archive/55/7474/, свободный. - Загл. с экрана
10. И снова о топологической сортировке… [Электронный ресурс] // habr.com - Режим доступа: https://habr.com/ru/companies/developersoft/articles/116812/, свободный. - Загл. с экрана
11. Сравнение алгоритмов сортировок. Визуализация и ... [Электронный ресурс] // www.youtube.com - Режим доступа: https://www.youtube.com/watch?v=i9pcxi7mt-o, свободный. - Загл. с экрана
12. Топологическая сортировка - Алгоритмика - Algorithmica [Электронный ресурс] // ru.algorithmica.org - Режим доступа: https://ru.algorithmica.org/cs/graph-traversals/topological-sorting/, свободный. - Загл. с экрана
13. Алгоритмы и алгоритмические языки - Курс АиАЯ [Электронный ресурс] // algcourse.cs.msu.su - Режим доступа: http://algcourse.cs.msu.su/wp-content/uploads/2010/09/lection19-2018.pdf, свободный. - Загл. с экрана
14. Топологическая сортировка [Электронный ресурс] // ru.wikipedia.org - Режим доступа: https://ru.wikipedia.org/wiki/топологическая\_сортировка, свободный. - Загл. с экрана
15. Практическая работа №6 - Топологическая сортировка [Электронный ресурс] // ks.psuti.ru - Режим доступа: https://ks.psuti.ru/downloads/students/distance\_learning/2пкс-28/теория алгоритмов/та\_пр 6.pdf, свободный. - Загл. с экрана
16. Задание 4. Топологическая сортировка вершин графа [Электронный ресурс] // hci.fenster.name - Режим доступа: http://hci.fenster.name/304y/lab4/, свободный. - Загл. с экрана
17. Топологическая сортировка / Хабр [Электронный ресурс] // habr.com - Режим доступа: https://habr.com/ru/companies/otus/articles/499138/, свободный. - Загл. с экрана
18. Топологическая сортировка. Алго [Электронный ресурс] // users.math-cs.spbu.ru - Режим доступа: https://users.math-cs.spbu.ru/~okhotin/teaching/tcs2\_2017/okhotin\_tcs2\_2017\_l5.pdf, свободный. - Загл. с экрана
19. Шаг 88. Топологическая сортировка [Электронный ресурс] // it.kgsu.ru - Режим доступа: https://it.kgsu.ru/c\_din/din\_0088.html, свободный. - Загл. с экрана