|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования |
| **«МИРЭА – Российский технологический университет»** |
| **РТУ МИРЭА** |
|  |

|  |  |
| --- | --- |
| **Отчет по выполнению практического задания № 7, часть 2** | |
| **Тема:** | |
| **«Графы: создание, алгоритмы обхода, важные задачи теории**  **графов»** | |
| Дисциплина: «Структуры и алгоритмы обработки данных» | |
|  | Выполнил студент: Лисовский И.В |
|  | Группа: ИКБО-21-23 |

Москва – 2024

СОДЕРЖАНИЕ

[1 ЦЕЛЬ 3](#_Toc183695988)

[2 ЗАДАНИЕ 4](#_Toc183695989)

[2.1 Формулировка задачи 4](#_Toc183695990)

[2.2 Математическая модель решения задачи 1 4](#_Toc183695991)

[2.3 Код программы с комментариями 7](#_Toc183695992)

[2.4 Тестирование программы 9](#_Toc183695993)

[3 ВЫВОД 10](#_Toc183695994)

[4 ЛИТЕРАТУРА 11](#_Toc183695995)

# **1 ЦЕЛЬ**

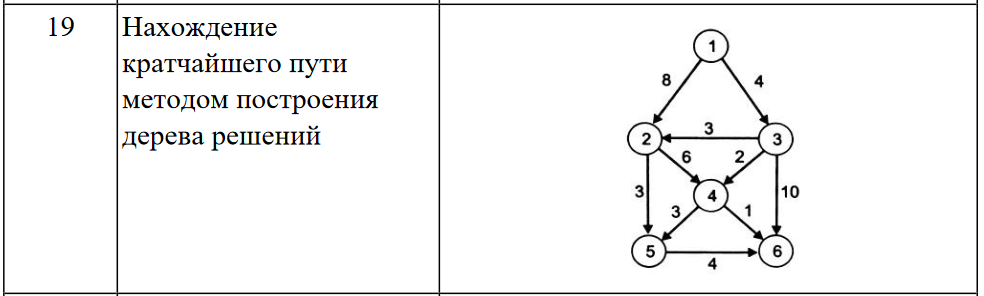
Исследовать метод нахождения кратчайшего пути в графе с помощью построения дерева решений, реализовать алгоритм на языке C++ и проанализировать его эффективность и применение в различных задачах теории графов, таких как маршрутизация и оптимизация.

# **2 ЗАДАНИЕ**

## **2.1 Формулировка задачи**

Составить программу создания графа и реализовать процедуру для работы с графом, определенную индивидуальным вариантом задания. Самостоятельно выбрать и реализовать способ представления графа в памяти. В программе предусмотреть ввод с клавиатуры произвольного графа. В вариантах построения остовного дерева также разработать доступный способ(форму) вывода результирующего дерева на экран монитора. Провести тестовый прогон программы на предложенном в индивидуальном варианте задания графе. Результаты тестирования в виде скриншотов экранов включить в отчет по выполненной работе. Сделать выводы о проделанной работе, основанные на полученных результатах. Оформить отчет с подробным описанием рассматриваемого графа, принципов программной реализации алгоритмов работы с графом, описанием текста исходного кода и проведенного тестирования программы.

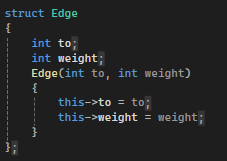
**Вариант №19:**

****

## **2.2 Математическая модель решения задачи 1**

1. **Определение структуры данных:**

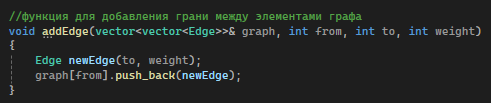
Определяется структура Edge, которая представляет ребро графа. Каждый экземпляр включает: to – вершина, к которой ведет ребро; weight – вес ребра.



Листинг 1.1 — Структура граней графа

1. **Добавления ребер:**

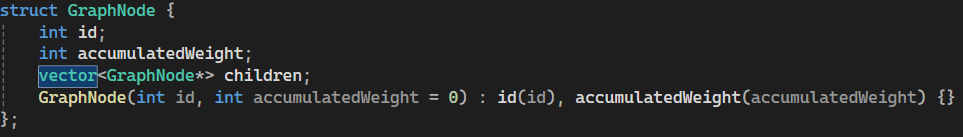
Создается новое ребро и добавляется в список смежности соответствующей вершины.



Листинг 1.2 — Добавления ребра графа

1. **Определение структуры узла дерева решений:**

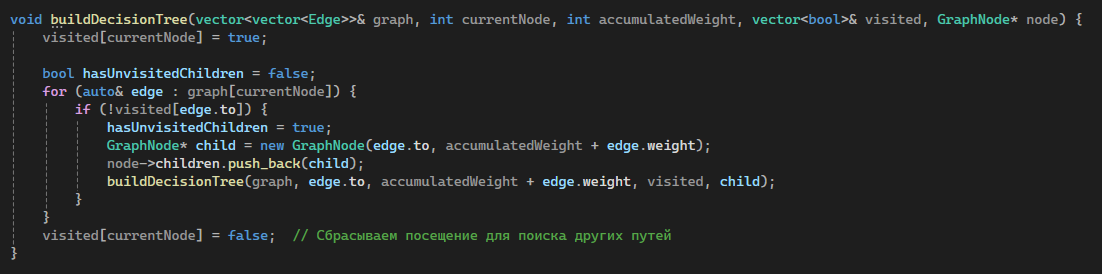
Структура GraphNode описывает узел дерева решений, который используется для накопления пути от начальной вершины до текущей. Структура: id — идентификатор вершины; accumulatedWeight — суммарный вес пути до данного узла; children — вектор дочерних узлов.



Листинг 1.3 — Структура узла дерева решений

1. **Построение дерева решений:**

Функция выполняет рекурсивный обход графа, создавая дерево решений. Алгоритм помечает вершину как посещённую и перебирает все её соседние вершины. Если сосед еще не посещен, создается дочерний узел, и происходит рекурсивный вызов для дальнейшего обхода. Параметры: graph — граф в виде списка смежности; currentNode — текущая вершина для обхода; accumulatedWeight — текущий накопленный вес; visited — вектор для отслеживания посещенных вершин (избежание циклов); node — текущий узел дерева решений.



Листинг 1.4 — Функция для построения дерева решений

## **2.3 Код программы с комментариями**

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  struct Edge  {  int to;  int weight;  Edge(int to, int weight)  {  this->to = to;  this->weight = weight;  }  };  struct GraphNode {  int id;  int accumulatedWeight;  vector<GraphNode\*> children;  GraphNode(int id, int accumulatedWeight = 0) : id(id), accumulatedWeight(accumulatedWeight) {}  };  //функция для добавления грани между элементами графа  void addEdge(vector<vector<Edge>>& graph, int from, int to, int weight)  {  Edge newEdge(to, weight);  graph[from].push\_back(newEdge);  }  void printGraph(vector<vector<Edge>>& graph)  {  for (int i = 0; i < graph.size(); i++)  {  for (int j = 0; j < graph[i].size(); j++)  {  cout << "Вершины " << i << " -> " << graph[i][j].to << " | Вес: " << graph[i][j].weight << "\n";  }  }  }  void buildDecisionTree(vector<vector<Edge>>& graph, int currentNode, int accumulatedWeight, vector<bool>& visited, GraphNode\* node) {  visited[currentNode] = true;  bool hasUnvisitedChildren = false;  for (auto& edge : graph[currentNode]) {  if (!visited[edge.to]) {  hasUnvisitedChildren = true;  GraphNode\* child = new GraphNode(edge.to, accumulatedWeight + edge.weight);  node->children.push\_back(child);  buildDecisionTree(graph, edge.to, accumulatedWeight + edge.weight, visited, child);  }  }  visited[currentNode] = false; // Сбрасываем посещение для поиска других путей  }  void printDecisionTree(GraphNode\* node, vector<int>& path) {  path.push\_back(node->id);  if (node->children.empty()) {  // Выводим полный путь и суммарный вес  cout << "Путь: ";  for (int id : path) {  cout << id << " ";  }  cout << "| Суммарный вес: " << node->accumulatedWeight << "\n";  }  else {  for (auto& child : node->children) {  printDecisionTree(child, path);  }  }  path.pop\_back();  }  // Функция для удаления дерева и освобождения памяти  void deleteTree(GraphNode\* node) {  if (node) {  for (auto& child : node->children) {  deleteTree(child);  }  delete node;  }  }  int main()  {  setlocale(0, "");  vector<vector<Edge>> graph(6);  addEdge(graph, 0, 1, 8);  addEdge(graph, 0, 2, 4);  addEdge(graph, 2, 1, 3);  addEdge(graph, 2, 3, 2);  addEdge(graph, 2, 5, 10);  addEdge(graph, 1, 3, 6);  addEdge(graph, 1, 4, 3);  addEdge(graph, 3, 4, 3);  addEdge(graph, 3, 5, 1);  addEdge(graph, 4, 5, 4);  printGraph(graph);  vector<bool> visited(graph.size(), false);  GraphNode\* decisionTree = new GraphNode(0);  buildDecisionTree(graph, 0, 0, visited, decisionTree);  // Печать дерева решений  cout << "\nДерево решений:\n";  vector<int> path;  printDecisionTree(decisionTree, path);  // Очистка выделенной памяти  deleteTree(decisionTree);  return 0;  } |

Листинг 2 — Код программы

## **2.4 Тестирование программы**

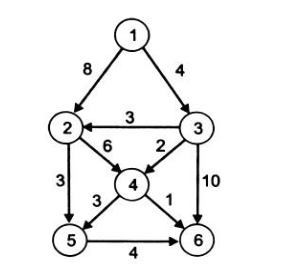


Рисунок 1 — Предложенный граф для тестирования



Рисунок 2 — Тестирование программы

# **3 ВЫВОД**

В выводе по практической работе со структурой данных граф, выполненной на C++, можно отметить следующее: В ходе работы была реализована структура данных графа и применён алгоритм поиска кратчайшего пути методом дерева решений. Полученные результаты продемонстрировали эффективность выбранного метода при решении задачи. Анализ алгоритма показал его возможности в поиске оптимальных решений, а также выявил потенциальные улучшения, такие как использование различных подходов к реализации алгоритма (например, алгоритм Дейкстры или A\*). В целом, работа позволила глубже понять особенности работы с графами и их практическое применение в программировании.

# **4 ЛИТЕРАТУРА**

1. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона, 2010.

2. Кнут Д. Искусство программирования. Тома 1-4, 1976-2013.

3. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для про-граммистов и любопытствующих, 2017.

4. Кормен Т.Х. и др. Алгоритмы. Построение и анализ, 2013.

5. Лафоре Р. Структуры данных и алгоритмы в Java. 2-е изд., 2013.

6. Макконнелл Дж. Основы современных алгоритмов. Активный обуча-ющий метод. 3-е доп. изд., 2018.

7. Скиена С. Алгоритмы. Руководство по разработке, 2011.

8. Хайнеман Д. и др. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2017.

9. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. Ин-форматика и вычислительная биология, 2003.

По языку С++:

10. Страуструп Б. Программирование. Принципы и практика с использова-нием C++. 2-е изд., 2016.

11. Павловская Т.А. C/C++. Программирование на языке высокого уровня, 2003.

12. Прата С. Язык программирования С++. Лекции и упражнения. - 6-е изд., 2012.

13. Седжвик Р. Фундаментальные алгоритмы на C++, 2001-2002

14. Хортон А. Visual C++ 2010. Полный курс, 2011.

15. Шилдт Г. Полный справочник по C++. 4-е изд., 2006.