Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ

Кафедра автоматизированных систем управления (АСУ)

И. А. Красиков

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ В ЭВМ

Методические указания по выполнению лабораторных работ

2016

Корректор: Осипова Е. А.

Красиков И. А.
Структуры и алгоритмы обработки данных в ЭВМ: методические указания по выполнению лабораторных работ. — Томск: Факультет дистанционного обучения, ТУСУР, 2016. — 24 с.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Согласно учебному плану по дисциплине «Структуры и алгоритмы данных в ЭВМ» (СиАОД) студентам, обучающимся с применением дистанционных образовательных технологий, необходимо выполнить две лабораторные работы.

1. Лабораторная работа № 1 «Бинарные деревья».

2. Лабораторная работа № 2 «Графы».

Целью данных работ является получение практических навыков создания структур данных, изучаемых в рамках данной дисциплины, и применение на практике алгоритмов их обработки.

Лабораторные работы выполняются по вариантам. Выбор варианта осуществляется по следующей формуле:

$$V = (N \times K) \text{ div } 100,$$

где V — искомый номер варианта,

N — общее количество вариантов,

div — целочисленное деление,

при V = 0 выбирается максимальный вариант,

K — код варианта.

Каждая лабораторная работа предполагает написание программы на языке программирования C/C++, предназначенной для решения определенной задачи в соответствии с выбранным вариантом.

Программные среды, поддерживающие языки программирования C/C++, а также ссылки для бесплатного их скачивания, приведены ниже.

1. Microsoft Visual Studio Community 2015

[Ссылка для скачивания:]

https://www.microsoft.com/ru-ru/download/details.aspx?id=48146

2. Dev-C++

[Ссылка для скачивания:]

http://sourceforge.net/projects/orwelldevcpp/

3. Code Blocks v 13.12

[Ссылка для скачивания:]

http://code-blocks.ru.uptodown.com/

Результатом выполнения лабораторной работы будет являться написанная на языке C/C++ программа, реализующая решение индивидуального задания, а также приложенный к ней отчет. Отчет по лабораторной работе должен включать в себя следующие компоненты:

1. Титульный лист.
2. Тема работы.
3. Цель работы.
4. Индивидуальное задание.
5. Алгоритм решения задачи.
6. Результаты работы программы.
7. Выводы.

Приложение А. Листинг программы.

Рекомендуемый порядок выполнения лабораторных работ:

1. Изучить теоретический материал по теме лабораторной работы, представленный в учебном пособии [1].
2. Составить алгоритм решения поставленной задачи.
3. Написать программу на языке C/C++.
4. Отладить и протестировать программу на различных наборах входных данных.
5. Написать отчет по лабораторной работе.

На проверку преподавателю необходимо отправить:

1. Архив с проектом, выполненным студентом, для используемой им среды разработки, который должен включать файлы: исходного кода программы, входных и выходных данных, исполняемый файл программы.

2. Отчет по лабораторной работе.

3. Рецензию на предыдущий вариант работы, если эта работа высылается на проверку повторно после устранения замечаний, указанных проверяющим преподавателем.

Варианты индивидуальных заданий на лабораторные работы приведены ниже, пример оформления титульного листа и структура отчета по лабораторной работе представлены в приложении А.

1 ЛАБОРАТОРНАЯ РАБОТА № 1 «БИНАРНЫЕ ДЕРЕВЬЯ»

Цель лабораторной работы № 1 — получить практические навыки представления в памяти ЭВМ структуры данных «бинарное дерево», реализовать на языке программирования C/C++ алгоритмы работы с деревьями.

1.1 Методические указания

Для выполнения данной лабораторной работы необходимо ознакомиться с главой 5 учебного пособия [1].

Во всех вариантах лабораторной работы № 1 необходимо спроектировать структуру данных «бинарное дерево» и реализовать вывод построенного дерева на экран. Также необходимо реализовать соответствующий конкретному варианту лабораторной работы алгоритм работы с бинарным деревом на языке C/C++. По завершении работы программы необходимо очистить динамическую память, занимаемую бинарным деревом, с помощью функций free() или delete().

1.2 Варианты заданий на лабораторную работу № 1

Вариант № 1

Задать последовательность чисел. Построить бинарное дерево, содержащее эти числа. Произвести обход дерева сверху вниз и вывести результат обхода на экран. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 2

Дана последовательность чисел. Построить бинарное дерево, содержащее эти числа. Произвести обход дерева слева направо и вывести ре-

зультаты на экран. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 3

Дана последовательность чисел. Построить бинарное дерево, содержащее эти числа. Произвести обход дерева справа налево и вывести результаты на экран. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 4

Даны две последовательности чисел. Построить бинарное дерево, содержащее числа первой последовательности. Для каждого числа второй последовательности узнать, входит ли оно в дерево. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 5

Дана последовательность чисел. Написать программу, которая формирует бинарное дерево, выводит построенное дерево на экран и подсчитывает число вершин на *n*-ом уровне сформированного дерева. Корень считать вершиной 0-ого уровня. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 6

Дана последовательность чисел. Построить бинарное дерево поиска, содержащее эти числа. Произвести обход дерева слева направо. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 7

Дана последовательность чисел, написать программу, которая формирует бинарное дерево поиска, выводит построенное дерево на экран и подсчитывает число вершин на *n*-ом уровне сформированного дерева. Корень считать вершиной 0-ого уровня. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 8

Дана последовательность чисел. Построить бинарное дерево поиска, содержащее эти числа. Для числа, введённого с клавиатуры, сказать, присутствует ли оно в дереве. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 9

Дана последовательность чисел. Построить бинарное дерево, содержащее эти числа. Удалить из дерева число, введённое с клавиатуры, и вывести оставшиеся числа в дереве. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 10

Дана последовательность чисел. Построить бинарное дерево поиска, содержащее эти числа. Для числа, введённого с клавиатуры, произвести удаление из дерева, вывести оставшиеся числа обходом слева направо. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 11

Дана последовательность чисел, написать программу, которая формирует АВЛ-дерево, выводит построенное дерево на экран и подсчитывает

число вершин на *n*-ом уровне сформированного дерева. Корень считать вершиной 0-ого уровня. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 12

Дана последовательность чисел. Написать программу формирования и вывода бинарного дерева на экран. Найти в построенном бинарном дереве все вершины, для которых выполняется условие: количество потомков в левом поддереве отличается от количества потомков в правом поддереве на 1. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 13

Дана последовательность чисел. Написать программу формирования и вывода бинарного дерева на экран. Для построенного дерева вывести на экран только элементы, являющиеся листьями. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 14

Задать последовательность чисел. Написать программу, выполняющую построение и вывод бинарного дерева на экран. Для построенного дерева найти все вершины, имеющие поддеревья одинаковой высоты, и для каждой из этих вершин вывести список потомков. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 15

Задать последовательность чисел. Написать программу, выполняющую построение и вывод бинарного дерева поиска на экран. Для построенного дерева вывести на экран все элементы, находящиеся на *k*-ом

уровне, где число *k* задается с клавиатуры. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 16

Дана последовательность чисел. Написать программу, выполняющую построение и вывод АВЛ-дерева на экран. Для построенного дерева вывести на экран только нечетные элементы, находящиеся на *k*-ом уровне, где число *k* задается с клавиатуры. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 17

Задать последовательность чисел. Написать программу, выполняющую построение и вывод АВЛ-дерева на экран. Для построенного дерева найти значения максимального, минимального, среднего арифметического значения элементов. Реализовать функцию добавления нового элемента в АВЛ-дерево. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 18

Дана последовательность чисел. Написать программу, выполняющую построение и вывод АВЛ-дерева на экран. Для построенного дерева реализовать функцию поиска элемента по указанному с клавиатуры значению. Реализовать функцию удаления найденного элемента из АВЛ-дерева. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 19

Задать последовательность чисел. Написать программу, выполняющую построение и вывод бинарного дерева на экран. Для построенного де-

рева реализовать функцию удаления поддерева, корнем которого является узел с номером, введенным с клавиатуры. После выполнения программы очистить память, занятую древовидной структурой.

Вариант № 20

Дана последовательность чисел. Написать программу, выполняющую построение и вывод АВЛ-дерева на экран. Для построенного дерева реализовать функцию нахождения пути от корня до вершины с номером *k,* где *k* вводится с клавиатуры. После выполнения программы очистить память, занятую древовидной структурой.

2 ЛАБОРАТОРНАЯ РАБОТА № 2 «ГРАФЫ»

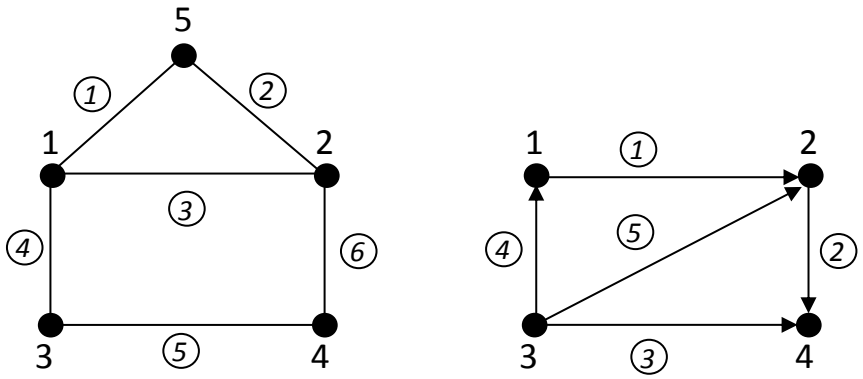
Цель лабораторной работы № 2 — получить практические навыки представления графов в памяти ЭВМ, реализовать на языке программирования C/C++ алгоритмы работы с графами.

2.1 Методические указания

Для выполнения данной лабораторной работы необходимо ознакомиться с главой 6 учебного пособия [1].

Во всех вариантах лабораторной работы № 2 необходимо спроектировать структуру данных, которая будет использоваться для представления графа в памяти ЭВМ. Граф должен задаваться в текстовом файле так, как это сказано в задании на лабораторную работу.

Рассмотрим способы задания графа во входных файлах на примере ориентированного и неориентированного графов (рис. 2.1).



а *б*

Рис. 2.1 — Примеры графов:

а — неориентированный граф; *б* — ориентированный граф

1. Матрица инциденций

В первой строке файла даны два числа: n — количество вершин графа, m — количество рёбер графа. Далее содержится матрица инциденций размером ($n \times m$) (табл. 2.1).

Таблица 2.1 — Примеры файлов, задающих графы (рис. 2.1) с помощью матрицы инциденций

Неориентированный граф	Ориентированный граф
5 6 1 0 1 1 0 0 0 1 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0	4 5 1 0 0 –1 0 –1 1 0 0 –1 0 0 1 1 1 0 –1 –1 0 0

2. Матрица смежности

Первая строка файла содержит число n, обозначающее количество вершин графа.

Последующие n строк содержат матрицу смежности графа G.

Для неориентированного графа элемент g_{ij} может принимать значения:

1 — в графе имеется ребро между вершинами i и j; 0 — в графе отсутствует ребро между данными вершинами.

Для ориентированного графа число g_{ij} может принимать значения:

1 — в графе имеется дуга, ведущая из вершины i в вершину j; 0 — таковая дуга в графе отсутствует (табл. 2.2).

Таблица 2.2 — Примеры файлов, задающих графы (рис. 2.1) с помощью матрицы смежности

Неориентированный граф	Ориентированный граф
5 0 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 1 0 0 0	4 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0

3. Список рёбер

Первая строка файла содержит число *n*, обозначающее количество рёбер графа. Последующие *n* строк содержат пары чисел, обозначающих рёбра для ориентированного и дуги для неориентированного графов (табл. 2.3).

Таблица 2.3 — Примеры файлов, задающих графы (рис. 2.1) с помощью списка ребер

Неориентированный граф	Ориентированный граф
6 1 2 1 3 1 5 2 4 2 5 3 4	5 1 2 2 4 3 1 3 2 3 4

4. Списки смежности

Первая строка файла содержит число *n*, обозначающее количество рёбер графа.

Последующие *n* строк содержат списки смежности, где *i*-ая строка списка задается следующим образом:

$Ni \;\; gi1 \;\; gi2 \;\; ... \;\; giNi$, где Ni — количество смежных с *i*-ой вершиной вершин графа; $gi1 \;\; gi2 \;\; ... \;\; giNi$ — последовательность (список) вершин, смежных с *i*-ой вершиной графа *G* (табл. 2.4).

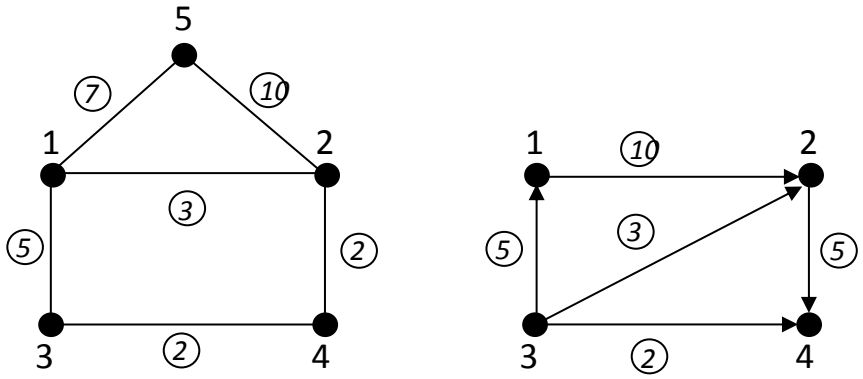
Таблица 2.4 — Примеры файлов, задающих графы (рис. 2.1) с помощью списков смежности

Неориентированный граф	Ориентированный граф
5 3 2 3 5 3 1 4 5 2 1 4 2 2 3 2 1 2	4 1 2 1 4 3 1 2 4 0

5. Матрица весов

Представление матрицы весов аналогично представлению матрицы смежности, с тем отличием, что в матрице весов элемент g_{ij} принимает значение веса (стоимости) ребра или дуги, если данное ребро или дуга существует между вершинами *i* и *j*, и 0 в остальных случаях.

Далее приведены примеры взвешенных графов (рис. 2.2).



а *б*

Рис. 2.2 — Примеры взвешенных графов:
а — неориентированный граф; *б* — ориентированный граф

Ниже приведены примеры файлов, содержащих матрицы весов для графов, представленных на рисунке 2.2 (табл. 2.5).

Таблица 2.5 — Примеры файлов, задающих графы (рис. 2.2) с помощью матрицы весов

Неориентированный граф	Ориентированный граф
5 0 3 5 0 7 3 0 0 2 10 5 0 0 2 0 0 2 2 0 0 7 10 0 0 0	4 0 10 0 0 0 0 0 5 5 3 0 2 0 0 0 0

Результатом работы программы будет являться решение задачи, поставленной в лабораторной работе. Алгоритм работы с графом должен быть реализован на языке C/C++. По завершении работы программы необходимо очистить динамическую память, занимаемую графом, с помощью функций free() или delete().

2.2 Варианты заданий на лабораторную работу № 2

Вариант № 1

Используя алгоритм Дейкстры, найти длины кратчайших путей во взвешенном неориентированном графе от заданной вершины до всех остальных. Граф задать в текстовом файле матрицей весов.

Вариант № 2

Для заданной с клавиатуры начальной вершины найти и вывести Эйлеров путь в неориентированном графе. Граф задать в текстовом файле списками смежности.

Вариант № 3

Используя метод поиска в ширину, в неориентированном графе, найти и вывести все вершины, достижимые из заданной. Номер начальной вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей смежности. Достижимые вершины выводить в порядке их посещения.

Вариант № 4

Используя метод поиска в глубину, найти и вывести путь в ориентированном графе между двумя вершинами. Номера начальной и конечной вершин ввести с клавиатуры. Граф задать в текстовом файле матрицей весов.

Вариант № 5

Используя алгоритм Дейкстры, найти длины кратчайших путей во взвешенном ориентированном графе от заданной вершины до всех остальных. Граф задать в текстовом файле матрицей весов.

Вариант № 6

Используя метод поиска в глубину, найти и вывести путь в неориентированном графе между двумя заданными вершинами. Номера начальной и конечной вершин ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций.

Вариант № 7

Используя метод поиска в глубину, в неориентированном графе, найти и вывести все вершины, достижимые из заданной. Номер начальной вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций. Достижимые вершины выводить в порядке их посещения.

Вариант № 8

Найти длины кратчайших путей в неориентированном графе, все рёбра которого имеют единичный вес, от заданной вершины до всех остальных. Начальную вершину ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций.

Вариант № 9

Найти и вывести кратчайший путь в ориентированном графе, все дуги которого имеют единичный вес, между двумя заданными вершинами. Начальную и конечную вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций.

Вариант № 10

Используя метод поиска в ширину, найти и вывести путь в неориентированном графе между двумя заданными вершинами. Номера начальной и конечной вершин ввести с клавиатуры. Граф задать в текстовом файле матрицей смежности.

Вариант № 11

Используя алгоритм Дейкстры, найти кратчайший путь между двумя заданными вершинами во взвешенном неориентированном графе. Начальную и конечную вершины пути ввести с клавиатуры. Граф задать в текстовом файле матрицей весов.

Вариант № 12

Построить стягивающее дерево неориентированного графа методом поиска в глубину и вывести список рёбер дерева. Граф задать в текстовом файле списком рёбер.

Вариант № 13

Найти длины кратчайших путей в ориентированном графе, все дуги которого имеют единичный вес, от заданной вершины до всех остальных. Начальную вершину ввести с клавиатуры. Граф задать в текстовом файле матрицей смежности.

Вариант № 14

Построить стягивающее дерево неориентированного графа методом поиска в ширину и вывести список рёбер дерева. Граф задать в текстовом файле матрицей инциденций.

Вариант № 15

Используя метод поиска в ширину, найти и вывести путь в ориентированном графе между двумя вершинами. Номера начальной и конечной вершин ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций.

Вариант № 16

Построить стягивающее дерево неориентированного графа методом поиска в ширину и вывести список рёбер дерева. Граф задать в текстовом файле списками смежности.

Вариант № 17

Построить стягивающее дерево неориентированного графа методом поиска в глубину и вывести список рёбер дерева. Граф задать в текстовом файле матрицей смежности.

Вариант № 18

Найти и вывести кратчайший путь в неориентированном графе, все рёбра которого имеют единичный вес, между двумя заданными вершинами. Начальную и конечную вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей смежности.

Вариант № 19

Используя метод поиска в глубину, в ориентированном графе найти и вывести все вершины, достижимые из заданной. Номер начальной вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей смежности. Достижимые вершины выводить в порядке их посещения.

Вариант № 20

Используя метод поиска в ширину, в ориентированном графе найти и вывести все вершины, достижимые из заданной. Номер начальной вершины ввести с клавиатуры. Граф задать в текстовом файле матрицей инциденций. Достижимые вершины выводить в порядке их посещения.

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Красиков И. А. Структуры и алгоритмы обработки данных в ЭВМ / И. А. Красиков — Томск : ФДО, ТУСУР, 2016. — 290 с.

2. Горитов А. Н. Структуры и алгоритмы обработки данных в ЭВМ: методические указания по выполнению лабораторных работ дисциплины «Структуры и алгоритмы обработки данных в ЭВМ» для студентов, обучающихся по специальности 230105 — «Программное обеспечение вычислительной техники и автоматизированных систем» / А. Н. Горитов. — Томск : ТУСУР, 2012. — 14 с.

Дополнительная литература

1. Макконелл Дж. Основы современных алгоритмов / Дж. Макконелл. — 2-е изд., доп. — Москва : Техносфера, 2004. — 368 с. (13 экз.)

2. Окулов С. М. Программирование в алгоритмах / С. М. Окулов. — 2-е изд., доп. — М. : БИНОМ. Лаборатория знаний, 2006. — 384 с. (30 экз.)

3. Вирт Н. Алгоритмы и структуры данных / Н. Вирт. — М. : Мир, 1989. — 360 с. (50 экз.)

4. Алгоритмы: построение и анализ : пер. с англ. / Томас Х Кормен [и др.]. — 3-е изд. — М. : Вильямс, 2013. — 1328 с. : ил. — (Парал. тит. англ.). — ISBN 978-5-8459-1794-2 (рус.).

ПРИЛОЖЕНИЕ А

Структура отчета

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

НАЗВАНИЕ РАБОТЫ

Отчет по лабораторной работе № X по дисциплине «Структуры и алгоритмы обработки данных в ЭВМ»

Выполнил: _____

_____

«____» _____ 20__ г.

Проверил: _____

_____

«____» _____ 20__ г.

Томск — 20__

СОДЕРЖАНИЕ