Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

Методические указания к лабораторным работам,
практическим занятиям и организации самостоятельной работы
для студентов направления «Программная инженерия»
(уровень бакалавриата) заочной формы обучения

2018

Пермякова Наталья Викторовна

Алгоритмы и структуры данных: Методические указания к лабораторным работам, практическим занятиям и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) заочной формы обучения/ Н.В. Пермякова. — Томск, 2018. — 26 с.

Содержание

1 Введение

Учебно-методическое пособие содержит материалы для методической поддержки аудиторных занятий и организации самостоятельной работы студентов.

Целью проведения лабораторных работ, практических занятий и организации самостоятельной работы по дисциплине является изучение классических алгоритмов сортировки и поиска, ознакомление с различными способами хранения и представления данных.

По окончанию изучения дисциплины «Алгоритмы и структуры данных» студент должен:

— **знать** простые алгоритмы сортировки; улучшенные алгоритмы сортировки; алгоритмы поиска подстроки в строке; алгоритмы на BST-деревьях; различные представления очередей с приоритетом;

— **уметь** осуществлять операции сортировки данных; осуществлять поиск данных по заданному ключу; определять вычислительную сложность алгоритмов;

— **владеть навыками** реализации и отладки программ на алгоритмических языках программирования; использования различных структур данных при решении задач.

Лабораторные работы и практические занятия по дисциплине, выполняемые студентами во время обучения, формируют навыки использования конструкций структурного программирования для реализации классических алгоритмов.

Самостоятельная работа студентов по дисциплине содержит несколько видов деятельности — проработка лекционного материала, подготовка и выполнение контрольной работы, подготовка к лабораторным работам и практическим занятиям, самостоятельное изучение тем, подготовка к зачету.

Все виды самостоятельной активности студентов тесно взаимосвязаны. Проработка лекционного материала и тем, вынесенных на самостоятельное изучение, подразумевает изучение законспектированного материала и одновременно является подготовкой к выполнению предусмотренной учебным планом контрольной работы, лабораторных работ.

2 Методические указания к проведению лабораторных работ

2.1 Общие положения

Целью проведения лабораторных работ является формирование и развитие навыков структурного программирования при реализации классических алгоритмов сортировки и поиска.

Основной формой проведения лабораторных работ является разработка алгоритма решения индивидуальной задачи и его программная реализация на языке Си. Процесс программной реализации включает в себя написание программы, отладку программы и тестирование программы.

К основным способам контроля формирования компетенций при выполнении лабораторных работ относятся индивидуальная защита выполненной работы, организация входного контроля уровня подготовки студентов по теоретическому материалу дисциплины, практическое применение которого осуществляется в ходе выполнения лабораторной работы.

Для получения максимальной оценки за лабораторную работу необходимо выполнить и защитить работу во время, отведенное для ее выполнения, согласно расписанию занятий. Допускается досрочное выполнение лабораторной работы по предварительной договоренности с преподавателем.

Выполнение всех лабораторных работ, предусмотренных рабочей программой дисциплины, является условием допуска к итоговому контролю изучения дисциплины — зачету.

2.1 Лабораторная работа «Простые сортировки на месте»

Цель работы: ознакомиться и реализовать простые методы сортировки — сортировки обменом, выбором, вставками, бинарными вставками. Исследовать сложность указанных алгоритмов сортировки.

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к лабораторной работе: для выполнения лабораторной работы необходимо ознакомиться с алгоритмами простых сортировок.

Сортировка обменом. Одним из простых методов сортировки на месте (т.е. при работе не требуется дополнительный объем памяти) является сортировка обменом или «пузырьковая» сортировка.

Суть алгоритма состоит в следующем: сравниваются пары рядом стоящих элементов массива, если первый элемент пары меньше второго, то элементы меняются местами. После первого просмотра массива самый большой элемент встает на свое место, а маленькие по значению элементы на один шаг продвигаются к началу массива. Отсюда метод и получил свое название: «легкие» элементы плавно «всплывают» к началу массива. «Тяжелые» элементы быстро «тонут», встают в конец массива.

После того, как все пары элементов массива просмотрены, массив еще не будет отсортирован, поэтому необходимо просмотреть пары элементов еще раз, и тогда еще один самый большой элемент встанет на свое место.

Если массив состоит из n элементов, то пар в таком массиве ровно *n-1.* На каждом шаге алгоритма самый большой элемент массива становится на свое место. Поэтому количество просматриваемых пар уменьшается с каждым шагом на единицу.

Таким образом, в алгоритме явно просматриваются два вложенных цикла. Внутренний цикл отвечает за просмотр пары элементов, количество шагов этого цикла зависит от внешнего цикла. Чем больше шагов выполнил внешний цикл, тем меньше пар просматривает внутренний цикл. Т.к. при выполнении одного шага внешнего цикла самый большой элемент становится на место, то количество шагов цикла равно количеству элементов массива -1. Однако, массив может быть уже отсортированным, до окончания работы внешнего цикла. Можно досрочно остановить выполнение цикла, если на каком-то *i*- том шаге во внутреннем цикле не произошло ни одного обмена.

Покажем применение сортировки на массиве *1 5 2 4 3.*

1: 1 2 4 3 ***5***

2: 1 2 3 ***4 5***

3: ***1 2 3 4 5***

После третьего шага алгоритм может закончить свою работу, так как не было проведено ни одного обмена.

Сортировка выбором. Сортировка выбором использует другой принцип упорядочивания элементов. На начальном шаге алгоритма выбирается минимальный элемент и ставится на место нулевого элемента. На последующих, *i*- тых шагах выбирается минимальный элемент среди элементов от *i*- того до (*n-1)*-го.

Рассмотрим на примере массива *1 5 2 4 3.*

1: ***1*** *5 2 4 3* Первый минимальный элемент уже стоит на своем месте.

2: ***1 2*** *5 4 3* Второй минимальный элемент — 2, поменяем его местами с 5.

3: ***1 2 3*** *4 5* Третий минимальный элемент — 3, поменяем его местами с 5.

4: ***1 2 3 4*** *5* Четвертый минимальный элемент стоит на своем месте.

После этого сортировка заканчивается, т.к. последний элемент в любом случае будет стоять на своем месте.

Сортировка вставками. Сортировка вставками упорядочивает массив, выполняя следующие действия — на начальном шаге алгоритма считаем, что последовательность из одного, первого элемента упорядоченная последовательность. Найдем место в этой последовательности для второго элемента. После этого упорядочены уже первые два элемента. Далее в текущей упорядоченной последовательности ищутся места для третьего, четвертого и т.д. элементов.

При программировании поиск места для вставляемого элемента лучше всего осуществить с конца упорядоченной последовательности.

Рассмотрим на примере массива *1 5 2 4 3.*

1: ***1 5*** *2 4 3* В упорядоченную последовательность добавляется 5.
2: ***1 2 5*** *4 3* В упорядоченную последовательность добавляется 2.
3: ***1 2 4 5*** *3* В упорядоченную последовательность добавляется 4.
4: ***1 2 3 4 5*** В упорядоченную последовательность добавляется 3.

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Составить алгоритм простой сортировки.
3. Реализовать алгоритм на языке Си, добавив в программу подсчет количества сравнений и перестановок, проведенных алгоритмом.
4. Найти среднее количество сравнений и перестановок, выполняемых программой для сортировки массивов из 100, 200, 300, 500, …, 10000 элементов, результаты сохраните в текстовом файле.
5. Построить графики зависимости сложности алгоритма (количество сравнений + количество перестановок) от количества обрабатываемых данных.
6. Сделать выводы по работе.

2.2 Лабораторная работа «Двоичные деревья - 1»

Цель работы: закрепление навыков программирования динамических структур на примере программирования деревьев двоичного поиска (BST — binary search tree).

Форма проведения: выполнение индивидуального задания.

Рекомендации по подготовке к занятию: перед проведением занятия необходимо изучить теоретический материал, изложенный в [1]. Раздел пособия 4.4 (стр. 191 — 205).

Порядок выполнения работы

1. Получить индивидуальный вариант.
2. Ответить на вопросы входного контроля
3. Составить алгоритм решения задачи индивидуального варианта.
4. Реализовать алгоритм на языке Си.
5. Выполнить тестирование полученного программного продукта
6. Защитить работу.

Вопросы входного контроля

1. Сформулируйте принцип построения BST-дерева.
2. Запишите структуру языка Си для описания узла двоичного дерева, если информационным полем будет число типа float.
3. Покажите на примере работу алгоритма добавления нового элемента в дерево
4. Перечислите возможные способы обхода дерева.

2.3 Лабораторная работа «Двоичные деревья - 2»

Цель работы: закрепление навыков программирования динамических структур на примере программирования деревьев двоичного поиска (BST — binary search tree).

Форма проведения: решение практических задач.

Рекомендации по подготовке к занятию: перед проведением занятия необходимо изучить теоретический материал, изложенный в [1]. Раздел пособия 4.4 (стр. 206 — 227).

Порядок проведения занятия

1. Получить индивидуальный вариант.
2. Ответить на вопросы входного контроля
3. Составить алгоритм решения задачи индивидуального варианта.
4. Реализовать алгоритм на языке Си.
5. Выполнить тестирование полученного программного продукта
6. Защитить работу.

Вопросы входного контроля

1. Продемонстрируйте на примере вставку нового элемента в корень дерева.

2. Запишите алгоритм ротации узла дерева.

3. Какие деревья называются сбалансированными деревьями?

4. Поясните понятие разделения дерева относительно наименьшего элемента. Можно ли выполнить разбиение дерева из 15 элементов, относительно 19-го наименьшего элемента.

3 Методические указания к проведению практических занятий

3.1 Общие положения

Практические занятия по дисциплине формируют навыки практического применения теоретических знаний, полученных во время изучения дисциплины. Специфика изучаемой дисциплины обуславливает форму проведения занятий — решение практических задач.

Для каждого практического занятия определяется тема, вопросы для проведения фронтального опроса, порядок проведения занятия. Первая часть занятия, как правило, проводится в виде семинара, в ходе которого проводится обсуждение темы занятия и опрос студентов. Во второй части занятия студенты решают практические задания по индивидуальным вариантам.

3.2 Практическое занятие «Поиск»

Цель занятия: ознакомление с принципами организации поиска подстроки в строке.

Форма проведения: решение практических задач.

Рекомендации по подготовке к занятию: перед проведением занятия необходимо изучить теоретический материал, изложенный в [1]. Раздел пособия 1.9 (стр. 55 — 66).

Порядок проведения занятия

1. Обсуждение основных принципов алгоритмов поиска строк.
2. Выборочный опрос студентов.
3. Решение задач (реализация, отладка программы).
4. Тестирование реализованных задач.
5. Анализ полученных результатов.

Примеры задач

Вариант 1

Напишите программу, реализующую алгоритм поиска Боуера-Мура.

Вариант 2

Напишите программу, реализующую алгоритм поиска Кнута, Морриса и Пратта.

Вариант 3

Напишите программу, реализующую алгоритм прямого поиска подстроки в строке.

Вопросы для проведения опроса на занятии

1. Расскажите алгоритм прямого поиска подстроки в строке.

2. Сколько сравнений выполнит алгоритм прямого поиска при поиске подстроки «пример» в тексте «рассматриваемый в учебном пособии пример может помочь в формировании представления о работе алгоритма»?

3. Запишите таблицу сдвигов, которая будет построена при реализации алгоритма Боуера-Мура, выполняющего поиск подстроки в тексте, который составлен из символов русского алфавита, знаков препинания и пробелов.

4. Расскажите принцип работы алгоритма Кнута, Морриса и Пратта.

5. Постройте таблицу сдвигов алгоритма Кнута, Морриса и Пратта, которая будет организована для поиска подстроки «аквариумистика».

4 Методические указания для организации самостоятельной работы

4.1 Общие положения

Самостоятельная работа является важной составляющей в изучении дисциплины и состоит из следующих видов деятельности: проработка лекционного материала для подготовки к тестированию и контрольным работам, подготовка к лабораторным работам и практическим заданиям, выполнение контрольных работ, самостоятельное изучение тем курса.

Самостоятельная работа над теоретическим материалом направлена систематизацию и закрепление знаний, полученных на лекционных занятиях и на получение новых знаний по дисциплине, путем самостоятельного изучения тем.

Самостоятельная работа по подготовке к лабораторным работам и практическим занятиям направлена на изучение методического и теоретического материала по теме лабораторной работы или практического занятия.

Выполнение контрольных работ — полностью самостоятельная работа, направленная на получение навыков самостоятельного составления алгоритмов, реализацию программ, их дальнейшей отладки и тестирования.

4.2 Проработка лекционного материала, подготовка к лабораторным работам и практическим занятиям

Проработка лекционного курса является одной из важных активных форм самостоятельной работы. Этот вид самостоятельной работы может быть организован следующим образом:

- прочитайте конспект лекции, согласуя Ваши записи с информацией на слайдах лекции;
- попробуйте выполнить самостоятельно примеры программ, разобранных на лекции;
- если в лекции рассматривался какой-либо алгоритм, попытайтесь выполнить этот алгоритм на тестовых данных без использования компьютерной программы; такой способ проработки материалов лекции покажет, правильно ли Вы поняли идею алгоритма;

- изучите дополнительные учебные материалы, рекомендованные преподавателем;
- попытайтесь ответить на контрольные вопросы, которыми, как правило, заканчиваются разделы учебных пособий или учебников;
- если после выполненной работы Вы считаете, что материал освоен не полностью, сформулируйте вопросы и задайте их преподавателю.

Методические указания к ведению конспектов лекций. Лекции по дисциплине проводятся с использованием слайдов. Но это не означает, что лекцию можно просто слушать. Ведение конспектов значительно повышает качество последующей проработки лекционного материала. В силу специфики дисциплины на слайдах лекций очень много алгоритмов, кодов программ, примеров демонстрации работы изучаемых алгоритмов. Но этот материал может быть бесполезен, если Вы не делаете записи в течение лекции, потому что большинстве случаев, комментарии по представленным на слайдах примерам, лектор выполняет в устной форме.

Можно рекомендовать распечатывать слайды перед лекцией и вести конспект непосредственно на бумажном варианте слайд-презентации.

Одной из форм текущего мониторинга уровня знаний по дисциплине являются контрольные работы.

Самостоятельная работа по подготовке к лабораторным работам и практическим занятиям по дисциплине состоит в изучении методических материалов по темам соответствующих видов аудиторных занятий.

Рекомендуется перед выполнением лабораторной работы изучить лекционный и методический материал по теме занятия, ознакомиться с алгоритмами, реализацию которых необходимо выполнить во время проведения занятия. Обратите особое внимание на порядок выполнения работы. Поскольку конечным результатом всех лабораторных работ является компьютерная программа, самостоятельно разработайте структурную схему будущей программы, выполните заготовку проекта, подготовьте самостоятельно тестовые данные.

Перед проведением практического занятия подготовьте ответы на вопросы, вынесенные для проведения входного контроля уровня знаний по теме занятия. Если при подготовке к занятию остались нерешенные вопросы, обратитесь за консультацией к преподавателю.

4.3 Подготовка и выполнение контрольных работ

4.3.1 Общие положения

Выполнение контрольных работ для студентов заочной формы обучения — это полностью самостоятельный вид деятельности студента. Темами контрольных работ являются темы дисциплины, не охваченные циклом лабораторных работ и практических занятий.

Выполнение контрольной работы состоит в написании программ по индивидуальному варианту. Обучающиеся самостоятельно разрабатывают алгоритм решения задания, реализуют разработанный алгоритм на языке программирования Си, отлаживают и тестируют написанную программу. Цель контрольной работы — провести сравнительный анализ времени выполнения различных алгоритмов сортировок.

Для достижения цели необходимо выполнить следующие задачи:

- реализовать на языке программирования Си алгоритмы указанных в варианте контрольной работы сортировок, организовав программу таким образом, чтобы количество сравнений и перестановок, выполняемых алгоритмом для массива размерности n, сохранялось в текстовом файле;
- протестировать работу программ на массивах с размерностью 100, 1000, 2000, … 50000 элементов;
- используя информацию о количестве сравнений и перестановок, выполняемых алгоритмом, построить графики зависимости времени выполнения алгоритма от размерности массива; для построения графиков возможно использование специального программного обеспечения;
- сделайте выводы о работе реализованных алгоритмов, на основе выполненных графиков.

Контрольные работы выполняются самостоятельно, во время семестра, а защищаются перед преподавателем во время экзаменационных сессий, как это предусмотрено учебным планом дисциплины. Допускается общение с преподавателем во время семестра посредством электронной почты — выполненные контрольные работы могут быть высланы на предварительную проверку.

Процедура защиты представляет собой индивидуальное собеседование с преподавателем, примерный сценарий которого представлен ниже:

- комментирование студентом кода и логики написанных программ;
- ответы на вопросы преподавателя по коду и логике программы;
- комментирование студентом тестовых данных;
- демонстрация работы программы;

- анализ работы реализованных сортировок, основанный на интерпретации графического представления зависимости времени работы алгоритмов от размерности массивов.

4.3.2 Контрольная работа «Сортировки»

Темы заданий контрольной работы

Оптимизация простых сортировок

Улучшенные методы сортировки

Сортировка слиянием

Поразрядные сортировки

Методические указания к разработке алгоритмов

Шейкерная сортировка. Сортировка перемешиванием, или Шейкерная сортировка — разновидность сортировки обменом. Анализируя метод пузырьковой сортировки, можно отметить два обстоятельства.

Во-первых, если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, её можно исключить из рассмотрения.

Во-вторых, при движении от конца массива к началу минимальный элемент “всплывает” на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо.

Эти две идеи приводят к следующим модификациям в методе пузырьковой сортировки. Границы рабочей части массива (т.е. части массива, где происходит движение) устанавливаются в месте последнего обмена на каждой итерации. Массив просматривается поочередно справа налево и слева направо.

Сортировка бинарными вставками. Сортировка бинарными вставками некоторым образом улучшает алгоритм сортировки вставками, увеличивая скорость нахождения места для вставляемого элемента. Метод использует информацию о том, что часть выборки уже отсортирована. Тогда, можно организовать поиск места для вставки нового элемента следующим образом: пусть переменная F обозначает индекс начала упорядоченной последовательности, а L — индекс конца. Найдем индекс элемента, находящегося в центре отсортированной последовательности — $M = (F + L)/2$ Сравним вставляемый элемент $X[j]$ с элементом $X[M]$. Если $X[j] > X[M]$, то изменим F на M (будем искать в правой части по-

следовательности). Если *X[j]< =X[M]*, то изменим *L* на *M* (будем искать в левой части последовательности). Описанные действия будем проводить до тех пор, пока *L>F*. После выхода из цикла место для вставки элемента найдено.

Рассмотрим работу алгоритма на примере уже упорядоченной последовательности:

1 2 3 5 6 7 8 9 10 ***4.***

F=0, L = 8, M = 4 X[4]=6 > 4, следовательно:

F = 0 L = 4 M = 2 X[2]=3 < 4, следовательно:

F = 2 L = 4 M = 3 X[3]=5 >4, следовательно:

F = 2, L = 3 M = 2 (по правилам деления целых чисел в Си 5/2 = 2) X[2]= 3 < 4, следовательно F = 3 L = 3.

Так как *F* и *L* равны (условие выполнения цикла), то вставляемый элемент должен занимать в упорядоченной последовательности место с номером 3.

Сортировка вставками со сторожевым элементом. Еще одна оптимизация метода простых вставок. Очевидно, что на каждом шаге внутреннего цикла выполняется два сравнения — вставляемый элемент сравнивается с просматриваемым элементом и отслеживается возможный выход за границы массива. Второго сравнения можно избежать, проведя перед сортировкой некоторые мероприятия.

1 вариант. Перед сортировкой можно найти минимальный элемент и поменять его местами с первым элементом массива. Таким образом, ни один из оставшихся элементов массива не будет больше первого элемента и второе сравнение во внутреннем цикле можно опустить.

2 вариант. При инициализации массива выделяется память для хранения *n*+1 элемента (*n* — размерность массива). Все элементы массива записываются, начиная со второго элемента. Далее, на каждом шаге внешнего цикла в первую позицию записывается вставляемый элемент. После выполнения этих действий так же можно опустить второе сравнение во внутреннем цикле.

Сортировка Шелла. Сортировка Шелла — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки **вставками.** Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга. Иными словами — это сортировка вставками с предварительными «грубыми» проходами.

Сортировка Шелла была названа в честь её изобретателя — Да Шелла, который опубликовал этот алгоритм в 1959 году.

Выбор длин промежутков

– Первоначально используемая Шеллом последовательность длин промежутков: $d_1 = N/2, d_i = d_{i-1}/2, d_k = 1$ в худшем случае, сложность алгоритма составит $O(n^2)$.

– Предложенная Хиббардом последовательность: $2^i - 1 \leq N, i \in N$. Такая последовательность шагов приводит к алгоритму сложностью $O(n^{3/2})$, массив шагов заполняется перед сортировкой.

– Предложенная Седжвиком последовательность: $d_i = 9 \cdot 2^i - 9 \cdot 2^{i/2} + 1$, если i четное и $d_i = 8 \cdot 2^i - 6 \cdot 2^{(i+1)/2} + 1$, если i нечетное. При использовании таких приращений средняя сложность алгоритма составляет: $O(n^{7/6})$, а в худшем случае порядка $O(n^{4/3})$. Массив приращений заполняется перед сортировкой. Последнее значение массива шаг[s-1], если 3*шаг[s] > N (если размер массива меньше 3-х шагов).

– Наиболее часто используемая последовательность шагов - d_i изменяется по правилу $d_{i+1} = (d_i - 1)/2$ (для массивов, содержащих более 500 элементов) и $d_{i+1} = (d_i - 1)/3$(для массивов, содержащих менее 500 элементов). За d_0 принимается число элементов массива. Метод заканчивает работу, когда d_i становится меньше 1.

Комбинированная сортировка (сортировка «расческой»). Комбинация сортировки обменом и сортировки Шелла. На каждом шаге сравниваются значения отстоящие друг от друга на заданное значение шага $H_{i+1} = 8 \cdot H_i/11$, но такое сравнение происходит всего один раз. Как только значение смещения становится равным 1, выполняется сортировка до конца методом пузырька. За H_0 принимается число элементов массива.

Пирамидальная сортировка. Пирамида — это частично упорядоченное двоичное дерево, элементы которого расположены в узлах дерева по следующему правилу — каждый элемент родительского узла обязательно больше элементов, расположенных в дочерних узлах. На рис. 1 представлена пирамида из 15 элементов:

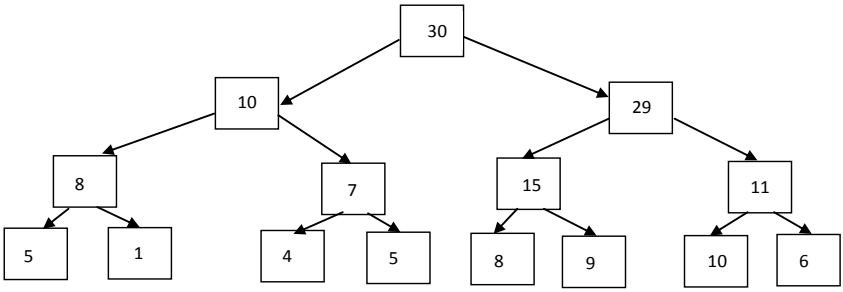

Рисунок 1 — Пирамидально упорядоченный массив

Элементы дерева легко представляются в виде массива — пусть родительский узел имеет индекс i, тогда дочерние узлы имеют индексы $2i$ и $2i + 1$. Рассмотренная пирамида может быть представлена массивом:

(30, 10, 29, 8, 7, 15, 11, 5, 1, 4, 5, 8, 9, 10, 6).

Т.к. корневой элемент элемент пирамиды всегда является максимальным элементом, то процесс пирамидальной сортировки можно описать следующим образом: поменять верхний элемент пирамиды с нижним элементом и рассматривать в дальнейшем не n элементов исходного массива, а $n - 1$ элемент. При выполнении этих действий нарушается правило расположения элементов в пирамиде, поэтому после обмена необходимо перестроить пирамиду с $n - 1$ элементами и далее, повторять два этих шага, пока пирамида не останется пустой. Таким образом, необходимо написать процедуру, строящую пирамиду для произвольного массива размерности *n*, далее алгоритм пирамидальной сортировки очень прост. Для формального описания алгоритма назовем процедуру построения пирамиды из массива размерностью *N KeyDown(N)* — т.к. элемент, находящийся в корне пирамиды может быть и не самым большим, то необходимо опустить этот элемент на нижние уровни пирамиды, чтобы выполнялась частичная упорядоченность. Алгоритм может выглядеть следующим образом:

```
1. KeyDown(N,X); // Построение пирамиды на исходном массиве x.
2. X[1] ←→ X[N]; // Обмен первого элемента пирамиды с последним
3. L = N-1; // Изменение размерности пирамиды
4. Пока (L>1) // пока пирамида не пуста
          KeyDown(N-1,X);
          X[1] ←→ X[L];
          L = L-1;
5. Конец.
```

Рассмотрим процесс построения пирамиды на произвольном массиве (в скобках после элементов указаны индексы):

Элементы массива: (25(1), 11(2), 5(3), 11(4), 4(5), 8(6), 3(7), 28(8), 18(9), 10(10), 1(11), 5(12), 4(13), 2(14), 17(15)). Начальное расположение элементов массива в узлах пирамиды показано на рис. 2.

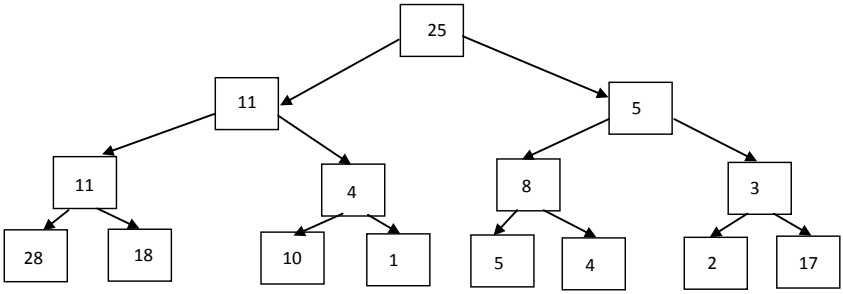

Рисунок 2 — Первый этап построения пирамиды

Размерность массива $N = 15$. Для элементов, находящихся на нижнем уровне не существует дочерних элементов, т.е. эти элементы могут не проверяться на выполнение правила пирамиды, индексы этих элементов от $N/2+1$ до N. Поэтому построение начинается с элемента с номером $N/2$, в рассматриваемом примере это x[7] = 3, сравним этот элемент с наибольшим из элементов x[14] и x[15], вторая нижняя пирамида остается без изменения, третья и четвертая пирамиды изменяются (см. рис. 3).

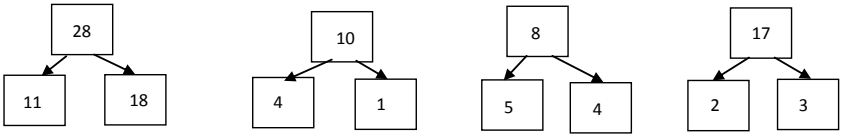

Рисунок 3 — Изменение порядка элементов нижнего уровня

Далее необходимо рассмотреть пирамиды с корневыми элементами во втором и третьем элементах (рис. 4):

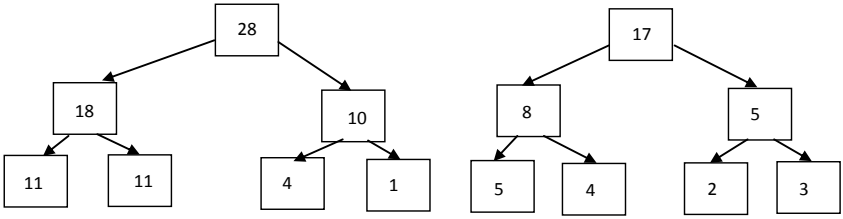

Рисунок 4 — Изменение пирамид среднего нижнего уровня

На рис. 5 изображен результат построения начальной пирамиды на массиве из 15 элементов.

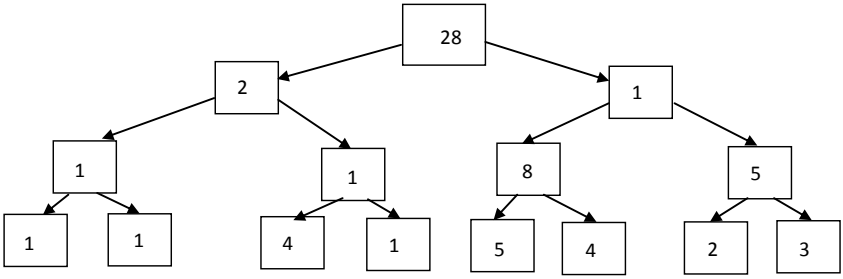

Рисунок 5 — Пирамидально упорядоченный массив

Очевидно, что процедура KeyDown(*N*,*X*) должна зависеть еще от одного параметра — номера элемента, для которого строится пирамида.

В общем случае для построения пирамиды с корнем в *L*-том элементе необходимо итеративно выполнить продвижение элемента по дереву вниз (при этом, элемент с номером *L* меняется местами с большим из своих потомков), продвижение элемента заканчивается, когда значение элемента в позиции *L* становится больше значений элементов-потомков, или когда достигнут нижний уровень.

Полный алгоритм пирамидальной сортировки выглядит следующим образом:

```
1. L = (N/2) +1;
2. Пока L>1
        L = L-1;
        KeyDown(L,N,X);
3. N1 = N;
4. Пока N1>1
        V = x[1]; x[1] = x[N1]; x[N1] = v;
        N1 = N1 – 1;
        KeyDown(L,N1,X);
5. Конец.
```

Сортировка Хоара. Значение произвольного элемента, обычно центрального, принимается за значение опорного элемента. Организуется просмотр элементов массива. При движении по массиву слева направо ищется элемент больше или равный опорному. При движении справа налево ищется элемент меньше или равный опорному элементу. Найденные элементы меняются местами, далее продолжается встречный поиск. После этих действий массив окажется разделенным на две части. В первой части будут расположены элементы меньше либо равные опорному элементу, а справа - больше либо равные. Далее алгоритм рекурсивно выполняется для правой и левой частей.

Сортировка Хоара с выбором медианного элемента. Можно улучшить быструю сортировку, выбирая средний элемент таким образом, чтобы его значение было бы действительно близким к серединному значению массива. Для этого можно воспользоваться двумя стратегиями:

Выбор среднего значения осуществляется случайным образом (с использованием датчиков случайных чисел и информации о размерности массива). Т.к. разделяющий элемент выбирается при каждом вызове процедуры, случайный выбор может быть наиболее правильным и оградит от появления наихудшего случая — когда медианный элемент оказывается наименьшим или наибольшим.

Вторая стратегия состоит *в случайном выборе трех элементов*, по одному из начального, конечного и среднего интервалов сортируемого подмассива. Как разделяющий элемент используется среднее из этих трех чисел.

Сортировка слиянием. Необходимо изучить теоретический материал, содержащий описание алгоритма и изложенный в [1]. Раздел пособия 2.4 (стр. 97 — 122).

Поразрядная сортировка. Необходимо изучить теоретический материал, содержащий описание алгоритма и изложенный в [2]. В главе 5 рассматриваются основные принципы поразрядных сортировок (стр. 147 — 155).

Пример индивидуального варианта контрольной работы

Задание 1. Сортировка вставками со сторожевым элементом.

Задание 2. Сортировка Хоара с выбором медианного элемента.

Задание 3. Нисходящая сортировка слиянием, метод слияния — прямой.

Задание 4. Поразрядная двоичная быстрая сортировка массива целых чисел.

4.4 Самостоятельное изучение тем теоретической части курса

4.4.5 Поразрядные сортировки

Перечень вопросов, подлежащих изучению

1. Поразрядная MSD-сортировка.
2. Поразрядная LSD-сортировка.
3. Двоичная быстрая сортировка.

Методические рекомендации по изучению

При изучении этой темы обратите внимание на область применения таких сортировок. Изучите способ сортировки подсчетом — применение этого метода сортировки внутри разряда улучшает оценку сложности алгоритма. Попробуйте оценить сложность поразрядных сортировок. Попытайтесь реализовать изученные способы сортировки.

Рекомендуемые источники

Мещеряков, Р.В. Методы программирования [Электронный ресурс] : учебно-методическое пособие / Р.В. Мещеряков. — Электрон. дан. — Москва : ТУСУР, 2007. — 237 с. — Режим доступа: https://e.lanbook.com/book/11631. — Загл. с экрана. Стр. 147 — 151, 153 — 155.

Потопахин, В. Искусство алгоритмизации [Электронный ресурс] / В. Потопахин. — Электрон. дан. — Москва : ДМК Пресс, 2011. — 320 с. — Режим доступа: https://e.lanbook.com/book/1269. — Загл. с экрана. Стр. 186— 191.

4.4.6 Динамические структуры

Перечень вопросов, подлежащих изучению

1. Линейные однонаправленные списки.
2. Список — стек, список — очередь.
3. Добавление элементов в список.
4. Удаление элемента из списка.
5. Создание сортированных списков.

Методические рекомендации по изучению

Изучение теоретического материала по этой теме желательно сопровождать зарисовками представления списков на бумаге. Понимание физического смысла создаваемой динамической структуры — основная за-

дача самостоятельной работы над этой темой и успешное освоение темы напрямую зависит от решения этой задачи. При изучении рекомендуемых источников помните, что программирование — творческий процесс, поэтому одна и та же задача в разных пособиях может иметь различные варианты реализации в виде программного кода.

Рекомендуемые источники

Потопахин, В. Искусство алгоритмизации [Электронный ресурс] / В. Потопахин. — Электрон. дан. — Москва : ДМК Пресс, 2011. — 320 с. — Режим доступа: https://e.lanbook.com/book/1269. — Загл. с экрана. Стр. 224 — 236.

Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана. Стр. 175 — 190.

4.4.7 Двоичные деревья поиска (BST-деревья)

Перечень вопросов, подлежащих изучению

1. Способы добавления элементов в дерево.
2. Фундаментальные операции над деревьями.
3. Методы печати элементов дерева.
4. Разделение дерева относительно *k*-того наименьшего.
5. Удаление элементов из дерева.

Методические рекомендации по изучению

Изучение этой темы желательно выполнять после знакомства с динамическими структурами данных. Рекомендуется составить план самостоятельной работы, так как это перечислено выше. Для закрепления материала попробуйте выполнить изученные алгоритмы на бумаге, на примере произвольной последовательности данных. После этого можно приступить к программной реализации алгоритмов. Обратите внимание — большинство функций работы с деревьями реализованы рекурсивно. В качестве дополнительного задания попробуйте выполнить не рекурсивную реализацию.

Рекомендуемые источники

Мещеряков, Р.В. Методы программирования [Электронный ресурс] : учебно-методическое пособие / Р.В. Мещеряков. — Электрон. дан. —

Москва : ТУСУР, 2007. — 237 с. — Режим доступа: https://e.lanbook.com/book/11631. — Загл. с экрана. Стр. 81 — 108.

Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана. Стр. 191 — 255.

4.4.8 Поиск

Перечень вопросов, подлежащих изучению

1. Поиск элементов в массиве.
2. Бинарный поиск.
3. Интерполяционный поиск.
4. Поиск подстроки в строке.

Методические рекомендации по изучению

Алгоритмы поиска встречаются в программировании повсеместно. При изучении этой темы студенты должны выделить три способа организации поиска в линейной структуре данных. После изучения всех трех способов сравните оценки сложности рассматриваемых алгоритмов. Вторая часть темы посвящена различным алгоритмам поиска подстроки в строке (в тексте). К таким алгоритмам относятся алгоритм поиска Боуера-Мура, алгоритм Кнута, Морриса и Пратта, прямой алгоритм. Ознакомьтесь с этими алгоритмами, попробуйте выполнить реализацию. Сравните алгоритмы по сложности.

Рекомендуемые источники

Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана. Стр. 54 — 64.

4.5 Подготовка к зачету

Необходимым условием для получения зачета является выполнение всех лабораторных работ, предусмотренных рабочей программой дисци-

плины. Зачет проводится в устной и письменной форме. Ниже приведен пример зачетного билета.

Для получения оценки *«зачтено»* необходимо выполнить 4 задания из шести.

Пример билета

Билет 1

Теоретическая часть

1. Запишите алгоритм сортировки обменом.

2. Продемонстрируйте работу сортировки Шелла на массиве
   9 8 0 6 13 7 18 11 1 13 16 13 12 1 4 16 3 10 1 3 с шагом 3

3. Нарисуйте дерево разбиений, которое строит нисходящая сортировка слиянием при обработке массива из 25 элементов.

4. Продемонстрируйте алгоритм MSD сортировки на массиве:
40464
18401
61242
74330
75675
99500
12042
13384
25745
33713

5. Постройте таблицу сдвигов (КМП-алгоритм) для образа «тригонометрия»

6. Постройте дерево методом вставки в лист, если элементы в дерево добавлялись в следующем порядке 13 12 8 18 22 0 16 25 29 13 6 17 13 17 15 16. Продемонстрируйте ротацию влево в узле 18.

5 Рекомендуемые источники

1.Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс] : учебное пособие / Н. Вирт. — Электрон. дан. — Москва : ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана.

2.Мещеряков, Р.В. Методы программирования [Электронный ресурс] : учебно-методическое пособие / Р.В. Мещеряков. — Электрон. дан. — Москва : ТУСУР, 2007. — 237 с. — Режим доступа: https://e.lanbook.com/book/11631. — Загл. с экрана.