****

**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«Владимирский государственный университет**

**им. А. Г. и Н. Г. Столетовых» (ВлГУ)**

А. В. ШУТОВ

Ю. А. МЕДВЕДЕВ

**СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ**

Рекомендации к самостоятельной работе студентов

по дисциплине «Структуры и алгоритмы компьютерной обработки данных» для студентов вузов, обучающихся по направлению 010500 «Математическое обеспечение и администрирование информационных систем»

ВЛАДИМИР 2013

УДК 004.31

ББК 32.988-5.я7

Ш 97

**Шутов А. В., Медведев Ю. А.**

Структуры и алгоритмы компьютерной обработки данных: рекомендации к самостоятельной работе студентов / Владимирский гос. университет им. А.Г. и Н.Г. Столетовых. – Владимир, 2013. – 36 с.

 Учебное издание предназначено для студентов вузов, обучающихся по направлению 010500 «Математическое обеспечение и администрирование информационных систем». Приведены методические рекомендации по изучению отдельных тем курса: алгоритмы на графах, алгоритмы комбинаторного перебора, общие методы разработки алгоритмов, теория сложности алгоритмов. Даны темы для докладов, рефератов и тесты по изучаемой дисциплине.

 Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

**Рецензенты:** доктор технических наук, профессор **Монахов М. Ю.**, зав. кафедрой информатикии защитыинформацииВлГУ;

кандидат физико-математических наук, доцент ВФ РУК

 **Красильщиков В. В.**

*Печатается по решению Редакционно-*

*издательского совета ВлГУ*

© ФГБОУ ВПО «Владимирский государственный университет», 2013

 © Шутов А. В., Медведев Ю. А., 2013

**ВВЕДЕНИЕ**

Предлагаемые рекомендации к самостоятельной работе адресованы студентам вузов, обучающихся по направлению 010500 «Математическое обеспечение и администрирование информационных систем». Они посвящены изучению отдельных тем дисциплины «Структуры и алгоритмы компьютерной обработки данных»:

– алгоритмы на графах,

– алгоритмы комбинаторного перебора,

– общие методы разработки алгоритмов,

– теория сложности алгоритмов.

Приведены темы для докладов, рефератов и тесты по изучаемой дисциплине.

 Отдельные разделы данного пособия могут быть использованы при изучении дисциплин «Программирование», «Теоретические основы информатики» студентами вузов, обучающимися на физико-математических факультетах по направлению «Педагогическое образование», а также в школах при проведении факультативов по информатике и при подготовке учащихся к олимпиадам по программированию.

 Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

**ОБЩИЕ РЕКОМЕНДАЦИИ**

**Целью самостоятельной работы** является формирование личности студента, развитие его способности к самообучению и повышению своего профессионального уровня.

Самостоятельная работа включает в себя следующие элементы:

– изучение содержания тем курса по конспектам лекций, учебникам и дополнительной литературе,

– подготовка к лабораторным и практическим занятиям,

– оформление отчетов по лабораторным работам,

– подготовка к рубежным рейтинг - контролям,

– самостоятельное изучение ряда вопросов, вынесенных на экзамен,

– подготовка доклада, посвященного углубленному рассмотрению одного из современных достижений в области алгоритмов компьютерной обработки данных;

– подготовка реферата на выбранную тему,

– самостоятельное решение задач, предлагаемых преподавателем в качестве домашних заданий;

– выполнение контрольной работы,

– работа с системой тестов по дисциплине,

– итоговая подготовка к экзамену.

Для облегчения самостоятельной работы студентом могут использоваться следующие материалы:

1. конспект лекций,
2. методические указания к практическим занятиям,
3. методические указания к лабораторным работам,
4. рекомендации по самостоятельной работе студентов,
5. система тестов,
6. учебники, учебные пособия и монографии из списка основной и дополнительной литературы,
7. материалы сайтов сети Интернет.

Список рекомендуемой учебной литературы по дисциплине приведен в рабочей программе, а также в конце конспекта лекций. Кроме того, в методических указаниях к практическим занятиям приводится более подробный анализ рекомендуемой литературы с разбиением по темам и указанием конкретных страниц.

Текущий контроль самостоятельной работы студентов осуществляется следующими способами:

– фронтальные опросы во время лекций и практических занятий,

– проверка домашних заданий,

– выступления с докладами во время практических занятий,

– проверка реферата на выбранную тему,

– защита лабораторных работ,

– проведение рейтинг-контролей,

– выполнение индивидуальных заданий преподавателя,

– проверка контрольной работы,

– проведение тестирования.

Итоговый контроль выполнения плана самостоятельной работы студента осуществляется во время экзамена.

Самостоятельная работа студентов должна проходить непрерывно на протяжении всего семестра. Она состоит из самостоятельной работы по изучению отдельных разделов курса, а также из подготовки к экзамену.

Планирование самостоятельной работы осуществляется студентом исходя из следующего объёма в часах.

Тема 1. АЛГОРИТМЫ НА ГРАФАХ – 16 часов.

Тема 2. АЛГОРИТМЫ КОМБИНАТОРНОГО ПЕРЕБОРА – 16 часов.

Тема 3. ОБЩИЕ МЕТОДЫ РАЗРАБОТКИ АЛГОРИТМОВ – 16 часов.

Тема 4. ТЕОРИЯ СЛОЖНОСТИ АЛГОРИТМОВ – 16 часов.

Подготовка к экзамену – 36 часов.

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ ОТДЕЛЬНЫХ ТЕМ КУРСА**

**Тема 1**. **АЛГОРИТМЫ НА ГРАФАХ** **– 16 часов.**

РАССМАТРИВАЕМЫЕ ВОПРОСЫ:

1. Понятие графа. Способы задания графов.
2. Поиск в глубину.
3. Поиск в ширину.
4. Эйлеровы циклы.
5. Гамильтоновы циклы.
6. Алгоритмы нахождения кратчайших путей.
7. Алгоритм Беллмана-Форда.
8. Алгоритм Дейкстры.
9. Алгоритм Флойда-Варшалла.
10. Минимальные остовные деревья.
11. Алгоритмы Крускала и Прима.
12. Потоки в сетях.
13. Алгоритм Форда-Фалкерсона.
14. Алгоритм Эдмондса-Карпа.

**Целью изучения данной темы** является рассмотрение базовых алгоритмов, связанных с математическими моделями теории графов, а также построение работающих компьютерных программ, реализующих рассматриваемые алгоритмы.

Для достижения поставленной цели студенту необходимо овладеть следующими основными понятиями теории графов (вопрос 1): граф, ориентированный граф, сеть, вершина, ребро, путь, цикл, расстояние, подграф, остовной подграф, остовное дерево, эйлеров цикл, гамильтонов цикл, степень вершины, поток в сети, разрез в графе.

Также необходимо овладеть основными способами компьютерного представления графов: *матрица смежности, матрица инцидентности, список ребер, списки смежности*.

По итогам изучения темы студент должен уметь решать следующие задачи:

– перечисление всех вершин заданного графа (вопросы 2 - 3);

– поиск циклов в графе, обладающих специальными свойствами (вопросы 4 - 5);

­– поиск кратчайших путей в графе (вопросы 6 - 9);

– поиск минимального остовного дерева (вопросы 10 - 11);

– поиск максимального потока в сети (вопросы 12 - 13).

При рассмотрении *конкретной задачи* студенту необходимо:

– изучить формулировку задачи,

– изучить основные алгоритмы, используемые для ее решения,

– провести сравнение различных алгоритмов для решения задачи,

– рассмотреть практические примеры применения изученных алгоритмов.

При рассмотрении *конкретного алгоритма* необходимо:

– изучить формулировку задачи,

– изучить алгоритм решения задачи,

– провести ручное исполнение изученного алгоритма на ряде примеров,

– выбрать язык для программной реализации алгоритма,

– реализовать изучаемый алгоритм на выбранном языке программирования, провести отладку и тестирование созданной компьютерной программы,

– изучить вычислительную сложность рассматриваемого алгоритма, а также время работы компьютерной программы.

Также студентам предлагается ряд задач для проверки освоения рассмотренных алгоритмов.

1. Дан ориентированный граф с *N* вершинами ( *N* =< 50 ). Вершины и дуги окрашены в цвета с номерами от 1 до *М* ( *М* =< 6 ) . Указаны две вершины, в которых находятся фишки игрока и конечная вершина. Правило перемещения фишек: игрок может передвигать фишку по дуге, если ее цвет совпадает с цветом вершины, в которой находится другая фишка; ходы можно делать только в направлении дуг графа; поочередность ходов необязательна. Игра заканчивается, если одна из фишек достигает конечной вершины. Написать программу поиска кратчайшего пути до конечной вершины, если он существует.

2. Некоторые школы связаны компьютерной сетью. Между школами заключены соглашения: каждая школа имеет список школ-получателей, которым она рассылает программное обеспечение всякий раз, получив новое бесплатное программное обеспечение (извне сети или из другой школы). При этом, если школа *В* есть в списке получателей школы *А*, то школа *А* может не быть в списке получателей школы *В*. *Требуется написать программу*, определяющую минимальное количество школ, которым надо передать по экземпляру нового программного обеспечения, чтобы распространить его по всем школам сети в соответствии с соглашениями. Кроме того, надо обеспечить возможность рассылки нового программного обеспечения из любой школы по всем остальным школам. Для этого можно расширять списки получателей некоторых школ, добавляя в них новые школы. Требуется найти минимальное суммарное количество расширений списков, при которых программное обеспечение из любой школы достигло бы всех остальных школ. Одно расширение означает добавление одной новой школы-получателя в список получателей одной из школ.

3. Задан неориентированный граф. При прохождении по некоторым ребрам отдельные (определенные заранее) ребра могут исчезать или появляться. Найти кратчайший путь из вершины с номером *q* в вершину с номером *w*.

4. Заданы два числа *N* и *М* ( 20 =< *М* =< *N* =< 150), где *N* — количество точек на плоскости. Требуется построить дерево из М точек так, чтобы оно было оптимальным. Дерево называется оптимальным, если сумма всех его ребер минимальна. Все ребра — это расстояния между вершинами, заданными координатами точек на плоскости.

5. Даны два числа *N* и *М*. Построить граф из *N* вершин и *М* ребер. Каждой вершине ставится в соответствие число ребер, входящих в нее. Граф должен быть таким, чтобы сумма квадратов этих чисел была минимальна.

6. Задан ориентированный граф с *N* вершинами, каждому ребру которого приписан неотрицательный вес. Требуется найти простой цикл, для которого среднее геометрическое весов его ребер было бы минимально.

**ТЕМА 2. АЛГОРИТМЫ КОМБИНАТОРНОГО ПЕРЕБОРА**

РАССМАТРИВАЕМЫЕ ВОПРОСЫ:

1. Основные комбинаторные объекты. Кортежи, перестановки, сочетания, размещения.
2. Генерация кортежей.
3. Коды Грея.
4. Генерация перестановок.
5. Генерация сочетаний.
6. Генерация размещений.
7. Генерация разбиений.
8. Генерация деревьев.

**Целью изучения данной темы** является рассмотрение базовых алгоритмов, связанных с генерацией комбинаторных объектов и различными вариантами реализации метода полного перебора, а также построение работающих компьютерных программ, реализующих рассматриваемые алгоритмы.

Для достижения поставленной цели студенту необходимо овладеть следующими основными понятиями комбинаторики (вопрос 1): кортеж, перестановка, сочетание, размещение, разбиение, подмножество, дерево, лексикографический порядок. Также необходимо знать формулы для числа рассматриваемых комбинаторных объектов.

По итогам изучения темы студент должен уметь решать задачи генерации всех рассмотренных комбинаторных объектов.

При рассмотрении *конкретной задачи* студенту необходимо:

– изучить формулировку задачи,

– изучить основные алгоритмы, используемые для ее решения,

– провести сравнение различных алгоритмов для решения задачи,

– рассмотреть практические примеры применения изученных алгоритмов.

При рассмотрении *конкретного алгоритма* необходимо:

– изучить формулировку задачи,

– изучить алгоритм решения задачи,

– провести ручное исполнение изученного алгоритма на ряде примеров,

– выбрать язык для программной реализации алгоритма,

– реализовать изучаемый алгоритм на выбранном языке программирования, провести отладку и тестирование созданной компьютерной программы,

– изучить вычислительную сложность рассматриваемого алгоритма, а также время работы компьютерной программы.

Также студентам предлагается ряд задач для проверки освоения рассмотренных алгоритмов.

1. Разработать программу генерации всех последовательностей длины *k* из чисел 1, 2, ... *N*. Первой последовательностью является 1, 1, ...,1, последней — *N*, *N*, ..., *N*.

2. Разработать программу генерации всех последовательностей длины *k*, у которых *i*-й элемент не превосходит значения *i*. Первой последовательностью является 1, 1, ...1, последней — 1, 2, ..., *k*.

3. Перечислить все разбиения натурального числа *N* на натуральные слагаемые ( разбиения, отличающиеся лишь порядком слагаемых, считаются за одно) в следующих порядках (пример, при *N* = 4):

 4, 3 + 1, 2 + 2, 2 + 1 + 1, 1 + 1 + 1 + 1;

 4, 2 + 2, 1 + 3, 1 + 1 + 2, 1 + 1 + 1 + 1;

 1 + 1 + 1 + 1, 1 + 1 + 2, 1 + 3, 2 + 2, 4.

4. Разработать программу генерации всех последовательностей длины 2\**N*, составленных из *N* единиц и *N* минус единиц, у которых сумма любого начального отрезка неотрицательна, т. е. количество минус единиц в нем не превосходит количества единиц.

5. Путем трассировки приведенной программы определите её назначение и логику работы.

*Const n=4 ;*

*Var A:Array[l..n] Of Integer;*

*i:Integer;*

*Function Place (i ,m: Integer) -.Integer;*

*Begin*

*If (m Mod 2=0) And (m>2) Then*

*If i<m-l Then Place:=i*

*Else Place:=m-2*

*Else Place:=m-l;*

*End;*

*Procedure Perm (m:Integer);*

*Var i,t:Integer;*

*Begin*

*If m=l Then Begin For i:=l To n Do Write (A[i] :3) ;*

*WriteLn;End*

*Else*

*For i:=J To m Do Begin*

*Perm (m-1) ;*

*If Km Then Begin t: =A[Place (i ,m) ] ;*

*AfPlace(i,m)]:=A[m];A[m]:=t;End;*

*End;*

*End;*

*Begin*

*For i:=l To n Do A[i]:=i;*

*Perm (n);*

*End.*

6. Путем трассировки приведенной программы определите её назначение и логику работы.

*Const n=4;*

*Var A:Array[1..п] Of Integer;*

*i,j,p,l:Integer;*

*Begin*

*For i:=l To n Do A[i]:=0;*

*i:=0;*

*Repeat*

*For 1:-1 To n Do Write (A[l] : 3) ;WriteLn ;*

*i : =i +1 ;p: =1 ;j : =i ;*

*While j Mod 2=0 Do Begin*

*j:=j Div 2;p:=p+l;*

*End;*

*If p<=n Then A[p] :=1-A[p] ;*

*Until p>n;*

*End.*

**ТЕМА 3. ОБЩИЕ МЕТОДЫ РАЗРАБОТКИ АЛГОРИТМОВ**

РАССМАТРИВАЕМЫЕ ВОПРОСЫ:

1. Динамическое программирование.
2. «Жадные» алгоритмы.
3. Перебор с возвратом.
4. Алгоритмы «Разделяй и властвуй».

**Целью изучения данной темы** является рассмотрение базовых приемов и методов разработки алгоритмов, а также приобретение умений и навыков по разработке алгоритмов рассматриваемыми методами.

Наибольшее внимание должно уделяться применению рассматриваемого метода при решении конкретной задачи, а также выработке умений по выбору метода разработки алгоритма.

При рассмотрении конкретного метода разработки алгоритмов необходимо:

– изучить метод разработки алгоритмов и область его применения,

– изучить рассматриваемые в курсе лекций и в процессе практических занятий примеры применения метода,

– реализовать на компьютере программы для решения рассмотренных задач, провести отладку и тестирование созданных программ,

– самостоятельно применить рассматриваемый метод для решения предложенных преподавателем задач.

Также студентам предлагается ряд задач для проверки освоения рассмотренных алгоритмов.

1. На острове Новой Демократии каждый из жителей организовал партию, которую сам и возглавил. Любой из жителей острова может состоять не только в своей партии, но и в других партиях. К всеобщему удивлению, даже в самой малочисленной партии оказалось не менее двух человек. К сожалению, финансовые трудности не позволили создать парламент, куда вошли бы, как предполагалось по Конституции острова, президенты всех партий. Посовещавшись, островитяне решили, что будет достаточно, если в парламенте будет хотя бы один член каждой партии. Помогите островитянам организовать такой, как можно более малочисленный парламент, в котором будут представлены все партии.

2. Какое наименьшее число ферзей можно расставить на доске так, чтобы они держали под боем все ее свободные поля?

3. Расставить на доске *N* \* *N* ( *N* < 12) *N* ферзей так, чтобы наибольшее число ее полей оказалось вне боя ферзей.

4. Расставить на доске как можно больше ферзей так, чтобы при снятии любого из них появлялось ровно одно неатакованное поле.

5. Задача о коне Аттилы («Трава не растет там, где ступил мой конь!»). На шахматной доске стоят белый конь и черный король. Некоторые поля доски считаются «горящими». Конь должен дойти до неприятельского короля, повергнуть его и вернуться на исходное место. При этом ему запрещено становиться как на горящие поля, так и на поля, которые уже пройдены.

6. Вы победили в соревновании, организованном канадскими авиалиниями. Приз — бесплатное путешествие по Канаде. Путешествие начинается с самого западного города, в который летают самолеты, проходит с запада на восток, пока не достигнет самого восточного города, в который летают самолеты. Затем путешествие продолжается обратно с востока на запад, пока не достигнет начального города. Ни один из городов нельзя посещать более одного раза за исключением начального города, который надо посетить ровно дважды ( в начале и в конце путешествия ) . Вам также нельзя пользоваться авиалиниями других компаний или другими способами передвижения. *Задача состоит в следующем*: дан список городов и список прямых рейсов между парами городов; найти маршрут, включающий максимальное количество городов и удовлетворяющий вышеназванным условиям.

7. Имеется клеточное поле размером *N*\**M*. Из каждой клетки можно перемещаться в одну из соседних, если она есть (вверх , вправо , вниз , влево). Коммивояжер стартует из какой-то клетки. Может ли он обойти все клетки и вернуться в исходную клетку ?

8. Клетки доски 8\*8 раскрашены в два цвета: белый и черный. Необходимо пройти из левого нижнего угла в правый верхний так, чтобы цвета клеток перемежались. За один ход разрешается перемещаться на одну клетку по вертикали или горизонтали.

Программа должна (если путь существует):

• находить хотя бы один путь;

• находить путь минимальной длины.

**ТЕМА 4. ТЕОРИЯ СЛОЖНОСТИ АЛГОРИТМОВ**

РАССМАТРИВАЕМЫЕ ВОПРОСЫ.

1. Понятие сложности алгоритма.
2. Классы сложности.
3. Полиномиальные и экспоненциальные алгоритмы.
4. Класс *NP*.
5. *NP*-полные алгоритмы.
6. Проблема *P* < > *NP*.

**Целью изучения данной темы** является знакомство студентов с основными положениями математической теории сложности алгоритмов, включая главную проблему современной теоретической информатики: проблему совпадения или различия сложностных классов *P* и *NP*.

Студентам необходимо овладеть строгими математическими определениями понятия алгоритма и сложности алгоритма (включая тезис Черча ). При этом студенты могут использовать любую из существующих вычислительных моделей по своему выбору, однако рекомендуется использование моделей, основанных на классической машине Тьюринга, а также ( для определения класса *NP* ) машины Тьюринга с оракулом. При изучении рассматриваемой темы строгие математические доказательства могут быть опущены.

Далее необходимо изучить определения основных классов сложности алгоритма и примеры задач рассматриваемых классов. Для самопроверки рекомендуется самостоятельно определить класс сложности для всех ранее рассмотренных в курсе алгоритмов.

При изучении *NP*-полных задач необходимо изучить метод сведения, используемый для доказательства *NP*-полноты.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАЧИ

1. Дайте формальное определение задачи поиска самого длинного простого цикла в неориентированном графе. Сформулируйте соответствующую задачу принятия решения. Опишите язык, соответствующий этой задаче принятия решения.

2. Покажите, что алгоритм, который содержит не больше некоторого постоянного количества вызовов процедуры с полиномиальным временем работы, сам работает полиномиальное время. Если же алгоритм делает полиномиальное число вызовов такой процедуры, то общее время работы может быть экспоненциальным.

3. Докажите, что класс языков *NP* замкнут относительно операций объединения, пересечения, конкатенации и замыкания Клини. Обсудите замкнутость класса *NP* относительно дополнения.

4. Покажите, что задача по определению выполнимости булевых формул в дизъюнктивной нормальной форме разрешима в течение полиномиального времени.

5. Предположим, кто-то разработал алгоритм с полиномиальным временем выполнения, позволяющий решить задачу о выполнимости формул. Опишите, как с помощью этого алгоритма в течение полиномиального времени находить выполняющие наборы.

6. Покажите, что задача о гамильтоновом пути *NP*-полная.

7. Задача о самом длинном простом цикле ( *longest-simple-cycle problem* ) — это задача, в которой в заданном графе находится простой цикл (без повторения вершин) максимальной длины. Покажите, что эта задача — *NP*-полная.

ТЕМЫ ДЛЯ ДОКЛАДОВ

1. Математические методы анализа алгоритмов.
2. Использование рекурсии для анализа алгоритмов.
3. Вероятностный анализ алгоритмов.
4. Амортизационный анализ алгоритмов.
5. Биномиальные пирамиды.
6. Фибоначчиевы пирамиды.
7. Алгоритмы быстрых вычислений.
8. Алгоритмы работы с матрицами.
9. Теоретико - числовые алгоритмы.
10. Алгоритмы вычислительной геометрии.
11. Приближенные алгоритмы.

Темы рефератов

1. Алгоритмы работы с деревьями.
2. Поиск кратчайших путей в графах.
3. Эйлеровы графы и их приложения.
4. Задача коммивояжера.
5. Задача о паросочетаниях.
6. Методы организации полного перебора.
7. Методы сокращения перебора в комбинаторных задачах.
8. Эвристические методы в задачах комбинаторного перебора.
9. Применение методов комбинаторного перебора в задачах искусственного интеллекта.
10. Метод ветвей и границ.
11. Метод динамического программирования.
12. «Жадные» алгоритмы и теория матроидов.
13. Математические методы анализа алгоритмов.
14. Методы разработки параллельных алгоритмов.
15. Метод Монте-Карло и его приложения в теории алгоритмов.
16. Классы сложности алгоритмов и их иерархия.
17. NP-полные задачи.
18. Методы доказательства результатов об NP-полноте.
19. Современные подходы к решению NP-полных задач.
20. Сложность по Колмогорову.

**Тест 1 по дисциплине "СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ"**

1. Структура данных представляет собой
2. набор правил и ограничений, определяющих связи между отдельными элементами и группами данных,
3. набор правил и ограничений, определяющих связи между отдельными элементами данных,
4. набор правил и ограничений, определяющих связи между отдельными группами данных,
5. некоторую иерархию данных.
6. Линейный список, в котором доступен только последний элемент, называется
7. стеком,
8. очередью,
9. деком,
10. массивом,
11. кольцом.
12. Структура данных работа с элементами которой организована по принципу *FIFO* («первый пришел» - «первый ушел») это –

a) стек,

b) дек,

c) очередь,

d) список.

1. Линейный последовательный список, в котором включение - исключение элементов возможно с обоих концов, называется
2. стеком,
3. очередью,
4. деком,
5. кольцевой очередью.
6. В чём особенности очереди ?
7. открыта с обеих сторон,
8. открыта с одной стороны на вставку и удаление,
9. доступен любой элемент.
10. В чём особенности стека ?
11. открыт с обеих сторон на вставку и удаление,
12. доступен любой элемент,
13. открыт с одной стороны на вставку и удаление.
14. Какую дисциплину обслуживания принято называть *FIFO* ?

a) стек,

b) очередь,

c) дек.

1. Какая операция читает верхний элемент стека без удаления ?

a) *pop*,

b) *push*,

c) *stackpop*.

**9**. Каково правило выборки элемента из стека ?
a) первый элемент,
b) последний элемент,
c) любой элемент.

**10**. Как освободить память от удаленного из списка элемента ?
a) *p* = *getnode*,
b) *ptr*(*p*) = *nil*,
c) *freenode* (*p*) ,
d) *p* = *lst*.

**11**. Как создать новый элемент списка с информационным полем *D* ?
a) *p* = *getnode*,
b) *p* = *getnode*; *info*(*p*) = *D*;c) *p* = *getnode*; *ptr*(*D*) = *lst*.

**12**. Как создать пустой элемент с указателем p?
a) *p* = *getnode*,
b)*info* (*p*),
c) *freenode* (*p*),
d) *ptr* (*p*) = *lst*.

**13.** Сколько указателей используется в односвязных списках?
a) 1 ,
b) 2 ,
c) сколько угодно.

**14**. В чём отличительная особенность динамических объектов ?
a) порождаются непосредственно перед выполнением программы,
b) возникают уже в процессе выполнения программы,
c) задаются в процессе выполнения программы.

**15**. При удалении элемента из кольцевого списка…
a) список разрывается,
b) в списке образуется дыра,
c) список становится короче на один элемент .

**16**. Для чего используется указатель в кольцевых списках ?
a) для ссылки на следующий элемент,
b) для запоминания номера сегмента расположения элемента,
c) для ссылки на предыдущий элемент ,
d) для расположения элемента в списке памяти.

**17**. Чем отличается кольцевой список от линейного ?
a) в кольцевом списке последний элемент является одновременно и первым,
b) в кольцевом списке указатель последнего элемента пустой,
c) в кольцевых списках последнего элемента нет ,d) в кольцевом списке указатель последнего элемента не пустой.

**18**. Сколько указателей используется в односвязном кольцевом списке ?
a) 1,
b) 2 ,
c) сколько угодно.

**19**. В каких направлениях можно перемещаться в кольцевом двунаправленном списке ?
a) в обоих ,
b) влево,
c) вправо.

**20**. С помощью какой структуры данных наиболее рационально реализовать очередь ?
a) стек,
b) список ,
c) дек.

**21**. В памяти ЭВМ бинарное дерево удобно представлять в виде:
a) связанных линейных списков,
b) массивов,
c) связанных нелинейных списков .

**22**. Элемент *t* , на который нет ссылок:
a) корнем ,
b) промежуточным,
c) терминальным (лист).

**23**. Дерево называется полным бинарным, если степень исходов вершин равна:
a) 2 или 0 ,
b) 2,
c) *М* или 0;
d) *M*.

**24**. Даны три условия окончания просеивания при сортировке прямым включением. Найдите среди них лишнее.
a) найден элемент *a* (*i*) с ключом, меньшим чем ключ у *x*,
b) найден элемент *a* (*i*) с ключом, большим чем ключ у *x* ,
c) достигнут левый конец готовой последовательности.

**25**. Какой из критериев эффективности сортировки определяется формулой *M* = 0,01\*n\*n + 10\*n ?
a) число сравнений ,
b) время, затраченное на написание программы,
c) количество перемещений,
d) время, затраченное на сортировку.

**26**. Как называется сортировка, происходящая в оперативной памяти ?
a) сортировка таблицы адресов,
b) полная сортировка,
c) сортировка прямым включением,
d) внутренняя сортировка ,
e) внешняя сортировка.

**27**. Как можно сократить затраты машинного времени при сортировке большого объёма данных ?
a) производить сортировку в таблице адресов ключей ,
b) производить сортировку на более мощном компьютере,
c) разбить данные на более мелкие порции и сортировать их.

**28**. Существуют следующие методы сортировки. Найдите ошибку.
a) строгие,
b) улудшенные,
c) динамические .

**29**. Метод сортировки называется устойчивым, если в процессе сортировки…
a) относительное расположенние элементов безразлично,
b) относительное расположение элементов с равными ключами не меняется ,
c) относительное расположение элементов с равными ключами изменяется,
d) относительное расположение элементов не определено.

**30**. Улучшенные методы имеют значительное преимущество:
a) при большом количестве сортируемых элементов ,
b) когда массив обратно упорядочен,
c) при малых количествах сортируемых элементов,
d) во всех случаях.

**31**. Что из перечисленных ниже понятий является одним из типов сортировки ?
a) внутренняя сортировка ,
b) сортировка по убыванию,
c) сортировка данных,
d) сортировка по возрастанию.

**32**. Сколько сравнений требует улучшенный алгоритм сортировки ?
a) n\*log(n) ,
b) en,
c) n\*n/4.

**33**. Сколько сравнений и пeрестановок элементов требуется в пузырьковой сортировке ?
a) n\*lon(n),
b) (n\*n)/4 ,
c) (n\*n-n)/2.

**34**. Сколько дополнительных переменных нужно в пузырьковой сортировке помимо массива, содержащего элементы ?
a) 0 (не нужно),
b) всего 1 элемент ,
c) *n* переменных (ровно столько, сколько элементов в массиве).

**35**. Как рассортировать массив быстрее, пользуясь пузырьковым методом?
a) одинаково ,
b) по возрастанию элементов,
c) по убыванию элементов.

**36**. В чём заключается идея метода *QuickSort* ?
a) выбор 1,2,…*n* – го элемента для сравнения с остальными,
b) разделение ключей по отношению к выбранному ,
c) обмен местами между соседними элементами.

**37**. Массив сортируется “пузырьковым” методом. За сколько проходов по массиву самый “лёгкий” элемент в массиве окажется вверху ?
a) за 1 проход ,
b) за *n*-1 проходов,
c) за *n* проходов, где *n* – число элементов массива.

**38**. При обходе дерева слева направо получаем последовательность…
a) отсортированную по убыванию,
b) неотсортированную ,
c) отсортированную по возрастанию.

**39**. При обходе дерева слева направо его элемент заносится в массив…
a) при втором заходе в элемент ,
b) при первом заходе в элемент,
c) при третьем заходе в элемент.

**40**. Где эффективен линейный поиск ?
a) в списке,
b) в массиве,
c) в массиве и в списке.

**Тест 2 по дисциплине "СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ"**

**1**. Какой поиск эффективнее ?
a) линейный,
b) бинарный,
c) без разницы.

**2**. В чём суть бинарного поиска ?
a) нахожденние элемента массива x путём деления массива пополам каждый раз, пока элемент не найден,
b) нахождение элемента x путём обхода массива,
c) нахождение элемента массива х путём деления массива.

**3**. Как расположены элементы в массиве бинарного поиска ?
a) по возрастанию,
b) хаотично,
c) по убыванию.

**4**. В чём суть линейного поиска ?
а) производится последовательный просмотр от начала до конца и обратно через 2 элемента,
b) производится последовательный просмотр элементов от середины таблицы,
c) производится последовательный просмотр каждого элемента.

**5**. Где наиболее эффективен метод транспозиций ?
a) в массивах и в списках,
b) только в массивах,
c) только в списках.

**6**. В чём суть метода транспозиции ?
a) перестановка местами соседних элементов ?
b) нахождение одинаковых элементов ?
c) перестановка найденного элемента на одну позицию в сторону начала списка.

**7**. Что такое уникальный ключ ?
a) если разность значений двух данных равна ключу,
b) если сумма значений двух данных равна ключу,
c) если  в таблице есть только одно данное с таким ключом.

**8**. В чём состоит назначение поиска ?
a) среди массива данных найти те данные, которые соответствуют заданному аргументу,
b) определить, что данных  в массиве нет,
c) с помощью данных найти аргумент.

1. Элемент дерева, который не ссылается на другие, называется
2. корнем,
3. листом,
4. узлом,
5. промежуточным.
6. Элемент дерева, на который не ссылаются другие, называется
7. корнем,
8. листом,
9. узлом,
10. промежуточным.
11. Элемент дерева, который имеет предка и потомков, называется
12. корнем,
13. листом,
14. узлом,
15. промежуточным.
16. Высотой дерева называется
17. максимальное количество узлов,
18. максимальное количество связей,
19. максимальное количество листьев,
20. максимальная длина пути от корня до листа.
21. Степенью дерева называется
22. максимальная степень всех узлов,
23. максимальное количество уровней его узлов,
24. максимальное количество узлов,
25. максимальное количество связей,
26. максимальное количество листьев.
27. Как определяется длина пути дерева
28. как сумма длин путей всех его узлов,
29. как количество ребер от узла до вершины,
30. как количество ребер от листа до вершины,
31. как максимальное количество ребер,
32. как максимальное количество листьев,
33. как длина самого длинного пути от ближнего узла до какого-либо листа.
34. Дерево называется бинарным, если
35. количество узловможет быть либо пустым, либо состоять из корня с двумя другими бинарными поддеревьями,
36. каждый узел имеет не менее двух предков,
37. от корня до листа не более двух уровней,
38. от корня до листа не менее двух уровней.
39. Бинарное дерево можно представить
40. с помощью указателей,
41. с помощью массивов,
42. с помощью индексов,
43. правильного ответа нет.
44. Какой метод поиска представлен в следующем фрагменте REPEAT I:=I+1 UNTIL (A[I]=X) OR (I=N);
45. последовательный,
46. двоичный,
47. восходящий,
48. нисходящий,
49. смешанный.
50. Какой метод поиска представлен в следующем фрагменте

REPEAT K:=(I+J)DIV 2; IF X>A[K] THEN I=K+1 ELSE J:=K-1;

 UNTIL (A[K]=X) OR (I>J);

1. последовательный,
2. бинарный,
3. восходящий,
4. нисходящий,
5. смешанный.
6. Реализация поиска в линейном списке выглядит следующим образом:
7. WHILE (P<>NIL) AND (P^.KEY<>X) DO P:=P^.NEXT,
8. WHILE (P<>NIL) DO P:=P^.NEXT,
9. WHILE AND (P^.KEY<>X) DO P:=P^.NEXT,
10. WHILE (P<>NIL) AND (P^.KEY<>X) P:=P^.NEXT,
11. WHILE (P<>NIL P^.KEY<>X) DO P:=P^.NEXT.
12. Как называются предки узла, имеющие уровень на единицу меньше уровня самого узла
13. детьми,
14. родителями,
15. братьями.
16. Стандартным способом устранения рекурсии при поиске в глубину является использование:
17. массива,
18. очереди,
19. стека,
20. циклического списка.
21. При поиске в ширину используется:
22. массив,
23. очередь,
24. стек,
25. циклический список.
26. В последовательном файле доступ к информации может быть
27. только последовательным,
28. как последовательным, так и произвольным,
29. произвольным,
30. прямым.
31. Граф – это
32. нелинейная структура данных, реализующая отношение «многие ко многим»,
33. линейная структура данных, реализующая отношение «многие ко многим»,
34. нелинейная структура данных, реализующая отношение «многие к одному»,
35. нелинейная структура данных, реализующая отношение «один ко многим»,
36. линейная структура данных, реализующая отношение «один ко многим».
37. Узлам (или вершинам) графа можно сопоставить:
38. отношения между объектами,
39. объекты,
40. связи,
41. типы отношений,
42. множества.
43. Рёбрам графа можно сопоставить:
44. связи,
45. типы отношений,
46. множества,
47. объекты,
48. отношения между объектами.
49. Граф, содержащий только ребра, называется
50. ориентированным,
51. неориентированным,
52. простым,
53. смешанным.
54. Граф, содержащий только дуги, называется
55. ориентированным,
56. неориентированным,
57. простым,
58. смешанным,
59. Граф, содержащий дуги и ребра, называется
60. ориентированным,
61. неориентированным,
62. простым,
63. смешанным.
64. Есть несколько способов представления графа в ЭВМ. Какой из способов приведенных ниже не относится к ним
65. матрица инциденций,
66. матрица смежности,
67. список ребер,
68. массив инцидентности.
69. Если последовательность вершин *v*0, *v*1, …*v*p определяет путь в графе *G*, то его длина определяется:
70. ,
71. ,
72. ,
73. .
74. Каким образом осуществляется алгоритм нахождения кратчайшего пути от вершины *s* до вершины *t* ?
75. нахождение пути от вершины *s* до всех вершин графа,
76. нахождение пути от вершины *s* до заданной вершины графа,
77. нахождение кратчайших путей от вершины *s* до всех вершин графа,
78. нахождение кратчайшего пути от вершины *s* до вершины *t* графа,
79. нахождение всех путей от каждой вершины до всех вершин графа.
80. Суть алгоритма Дейкстры - нахождение кратчайшего пути от вершины *s* до вершины *t* – заключается в
81. вычислении верхних ограничений *d* [*v*] в матрице весов дуг *a* [*u*,*v*] для *u*, *v* ,
82. вычислении верхних ограничений *d* [*v*],
83. вычислении верхних ограничений в матрице весов дуг *a* [*u*,*v*],
84. вычислении нижних ограничений *d* [*v*] в матрице весов дуг *a* [*u*,*v*] для *u*, *v*.
85. Улучшение *d*[*v*] в алгоритме Форда- Беллмана производится по формуле
86. *D* [*v*] : = *D* [*u*] + *a* [*u*,*v*] ,
87. *D* [*v*] : = *D* [*u*] - *a* [*u*,*v*] ,
88. *D* [*v*] : = *a* [*u*,*v*] ,
89. *D* [*v*] : = *D* [*u*] .
90. Строка представляет собой
91. конечную линейно-упорядоченную последовательность простых данных символьного типа,
92. конечную последовательность простых данных символьного типа,
93. конечную последовательность простых данных,
94. последовательность данных символьного типа.
95. Граф, содержащий только ребра, называется
96. ориентированным,
97. неориентированным,
98. простым,
99. связным.
100. Граф, содержащий только дуги, называется
101. ориентированным,
102. неориентированным,
103. простым,
104. связным.
105. Граф, содержащий ребра и дуги, называется
106. неориентированным,
107. простым,
108. смешанным,
109. связным.
110. Путь (цикл), который содержит все ребра графа только один раз, называется
111. Эйлеровым,
112. Гамильтоновым,
113. Декартовым,
114. замкнутым.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. *Кормен*, *Т*. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. - М.: Вильямс, 2007. 1296 с.
2. *Ахо*, *А*. Структуры данных и алгоритмы / А. Ахо, Д. Хопкрофт, Д. Ульман. - М.: Вильямс, 2007. 384 с.
3. *Окулов*, *С.М.* Программирование в алгоритмах / С.М. Окулов. - М.: Бином, 2007. 341с.
4. *Кнут, Д*. Искусство программирования. Т. 4 Вып.2. Генерация всех кортежей и перестановок / Д. Кнут. - М.: Вильямс, 2008. 160 с.
5. *Кнут, Д*. Искусство программирования. Т. 4 Вып.3. Генерация всех сочетаний и разбиений / Д. Кнут. - М.: Вильямс, 2008. 208 с.
6. *Кнут, Д.* Искусство программирования. Т. 4 Вып.4. Генерация всех деревьев. История комбинаторной генерации / Д. Кнут. - М.: Вильямс, 2008. 160 с.
7. *Седжвик, Р*. Фундаментальные алгоритмы на С++ / Р. Седжвик. - М.: Вильямс, 2011. 1056 с.
8. *Ху, Т.Ч*. Комбинаторные алгоритмы / Т.Ч. Ху, М.Т. Шинг. - Нижний Новгород: Изд-во Нижегородского госуниверситета им. Н.И. Лобачевского, 2004. 330 с.
9. *Новиков, Ф.А.* Дискретная математика для программистов / Ф.А. Новиков. - СПб.: Питер, 2012. 304 с.

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ……………………………………………………………………4**

**ОБЩИЕ РЕКОМЕНДАЦИИ………………………………………………..5**

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ИЗУЧЕНИЮ**

**ОТДЕЛЬНЫХ ТЕМ КУРСА………………………………………………...7**

**Тема 1**. **АЛГОРИТМЫ НА ГРАФАХ……………………………………...7**

**Тема 2**. **АЛГОРИТМЫ КОМБИНАТОРНОГО ПЕРЕБОРА…………..11**

**Тема 3**. **ОБЩИЕ МЕТОДЫ РАЗРАБОТКИ АЛГОРИТМОВ…………..15**

**Тема 4**. **ТЕОРИЯ СЛОЖНОСТИ АЛГОРИТМОВ………………………18**

**ТЕМЫ ДЛЯ ДОКЛАДОВ…………………………………………………..20**

**ТЕМЫ ДЛЯ РЕФЕРАТОВ…………………………………………………21**

**Тест 1 по дисциплине «СТРУКТУРЫ И АЛГОРИТМЫ**

**КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ»…………………………22**

**Тест 2 по дисциплине «СТРУКТУРЫ И АЛГОРИТМЫ**

**КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ»…………………………28**

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК…………………………………….35**

ШУТОВ Антон Владимирович

МЕДВЕДЕВ Юрий Алексеевич

СТРУКТУРЫ И АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ДАННЫХ

( рекомендации к самостоятельной работе студентов )

Издается в авторской редакции

Подписано в печать Формат 60х84 1/16

Усл. печ. л. 2,25 Уч. изд.л.- 2,45

Заказ № 10-13 Тираж 50 экз.

Отпечатано в отделе оперативной полиграфии ВлГУ

600014 г. Владимир, ул. Университетская, д. 2

Телефон: (4922) 33-87-40