МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ФГБОУ ВО «ИГУ»
Кафедра алгебраических и информационных систем

«Утверждаю»
Директор ИМЭИ ИГУ
_____Фалалеев М.В.
«30» марта 2016 г.

Рабочая программа дисциплины (модуля)

Б1.В.ДВ.4.2 Алгоритмы на графах

Направление подготовки 09.03.03 Прикладная информатика

Тип образовательной программы *академический бакалавриат*

Направленность (профиль) Информационная сфера

Квалификация (степень) выпускника – БАКАЛАВР

Форма обучения *заочная*

Иркутск 2016

Согласовано с УМК ИМЭИ

Протокол № 1

от 11 февраля 2016 г.

Председатель _____ Рожина Л.В.

Рекомендовано кафедрой

Протокол № 7

от 2 февраля 2016 г.

Ио зав. кафедрой

_____ Манцивода А.В.

Содержание

	стр.
1. Цели и задачи дисциплины (модуля)	3
2. Место дисциплины (модуля) в структуре ООП.	3
3. Требования к результатам освоения дисциплины (модуля)	3
4. Объем дисциплины (модуля) и виды учебной работы	3
5. Содержание дисциплины (модуля)	4
5.1 Содержание разделов и тем дисциплины (модуля)	4
5.2 Разделы дисциплины (модуля) и междисциплинарные связи с обеспечиваемыми (последующими) дисциплинами (модулями)	
5.3 Разделы и темы дисциплин (модулей) и виды занятий	
6. Перечень семинарских, практических занятий и лабораторных работ.	7
7. Примерная тематика курсовых работ (проектов) (при наличии)	-
8. Учебно-методическое и информационное обеспечение дисциплины (модуля) :	9
а) основная литература;	
б) дополнительная литература;	
в) программное обеспечение;	
9. Материально-техническое обеспечение дисциплины	11
10.Образовательные технологии	11
11.Оценочные средства. (ОС).	11

1. Цели и задачи дисциплины:

Цель дисциплины – обучение основам современной теории графов и сравнительный анализ алгоритмов, используемых при решении задач на графах.

Задачи дисциплины – познакомить студентов различными видами графов, методами их представления и обработки в программе, изучить и реализовать на практике базовые и усложненные алгоритмы на графах.

2. Место дисциплины в структуре ООП:

Дисциплина “Алгоритмы на графах” относится к дисциплинам по выбору цикла Б1. Дисциплина опирается на такие курсы как “Дискретная математика” и “Информатика и программирование”.

3. Требования к результатам освоения дисциплины (модуля):

Процесс изучения дисциплины (модуля) направлен на формирование следующих компетенций:

ОПК-4 способность решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

В результате изучения дисциплины студент должен:

Знать (ОПК-4) основные определения и понятия теории графов, методы представления графов в программе, базовые и усложненные алгоритмы на графах;

Уметь: (ОПК-4) применять изученные алгоритмы для решения конкретных практических задач, комбинировать на их основе более сложные алгоритмы;

Владеть: (ОПК-4) методами программирования изученных алгоритмов на языках программирования высокого уровня.

4. Объем дисциплины и виды учебной работы (разделяется по формам обучения)

Вид учебной работы	**Всего часов / зачетных единиц**	**Сессия**	
		В	С
Аудиторные занятия (всего)	**18**	**10**	**8**
Лекции			
Практические занятия (ПЗ)	8	4	4
Лабораторные работы (ЛР)	10	6	4

Контроль самостоятельной работы (КСР)			
Самостоятельная работа (всего)	153	80	73
Вид промежуточной аттестации	Экз, КР		Экз, КР
Контроль	9		9
Общая трудоемкость часы	180	90	90
зачетные единицы	5		

5. Содержание дисциплины

Раздел 1. Графы, их элементы и методы представления
Раздел 2. Общие вопросы теории алгоритмов
Раздел 3. Кратчайшие пути на графах
Раздел 4. Связность и паросочетания
Раздел 5. Сети и потоки
Раздел 6. Деревья и структуры данных

5.1. Содержание разделов и тем дисциплины.

Раздел 1. Графы, их элементы и методы представления
Граф, определение, полный, планарный, двудольный графы. Ориентированный граф, взвешенный граф. Числовые характеристики графа. Маршрут, путь, цепь, цикл, простые цепь и цикл. Ярус, диаметр, мост, точка сочленения. Обходы графа.

Раздел 2. Общие вопросы теории алгоритмов
Понятие алгоритма. Примеры алгоритмических процессов. Сложность и эффективность алгоритма.

Раздел 3. Кратчайшие пути на графах
Путь, кратчайший путь. Волновой алгоритм. Обход в ширину, очередь. Поиск выхода из лабиринта. Графы с неотрицательными ребрами. Алгоритм Дейкстры. Случай ребер разного знака. Циклы отрицательного веса. Алгоритм Форда-Беллмана. Алгоритм Флойда.

Раздел 4. Связность и паросочетания
Поиск компонент связности, обход в глубину. Алгоритм поиска мостов и точек сочленения. Двудольные графы и паросочетания. Алгоритм Куна поиска максимального паросочетания. Задача о назначениях, венгерский алгоритм решения задачи о назначениях.

Раздел 5. Сети и потоки
Сеть, поток в сети, пропускная способность, увеличивающий путь. Алгоритм Форда-Фалкерсона поиска максимального потока в сети.

Раздел 6. Деревья и структуры данных
Дерево, элементы и методы представления дерева. Бинарные деревья и их приложение в обработке и хранении данных. Бинарное дерево поиска. Бинарная куча. Дерево отрезков.

5.3. Разделы и темы дисциплин и виды занятий

№ п/п	Наименование раздела	Наименование темы	Виды занятий в часах				
			Лекц.	Практ. зан.	Лаб. зан.	СРС	Всего
1.	Раздел 1.	Графы, их элементы и методы представления		1	1	25	27
2.	Раздел 2.	Общие вопросы теории алгоритмов		1	1	25	27
3.	Раздел 3.	Кратчайшие пути на графах		1	2	25	28
4.	Раздел 4.	Связность и паросочетания		2	2	26	30
5.	Раздел 5.	Сети и потоки		2	2	26	30
6.	Раздел 6.	Деревья и структуры данных		1	2	26	29
	Всего			8	10	153	172

6. Перечень семинарских, практических занятий и лабораторных работ

№ п/п	№ раздела и темы дисциплины (модуля)	Наименование семинаров, практических и лабораторных работ	Трудоемкость (часы)	Оценочные средства	Формируемые компетенции
1	Раздел 1.	Графы, их элементы и методы представления	2	Индивидуальные задания	ОПК-4
2	Раздел 2.	Общие вопросы теории алгоритмов	2	Индивидуальные задания	ОПК-4
3	Раздел 3.	Кратчайшие пути на графах	3	Индивидуальные задания	ОПК-4
4	Раздел 4.	Связность и паросочетания	4	Индивидуальные задания	ОПК-4
5	Раздел 5.	Сети и потоки	4	Индивидуальные задания	ОПК-4
6	Раздел 6.	Деревья и структуры данных	3	Индивидуальные задания	ОПК-4

7. Примерная тематика курсовых работ (проектов) (при наличии)

1. Топологическая сортировка ациклического графа
2. Поиск компонент сильной связности, построение конденсации графа
3. Алгоритм Левита нахождения кратчайших путей от заданной вершины до всех остальных вершин
4. Минимальное остовное дерево. Алгоритм Прима
5. Минимальное остовное дерево. Алгоритм Крускала
6. Система непересекающихся множеств (DSU)
7. Код Прюфера. Формула Кэли.
8. Нахождение Эйлерова пути или Эйлерова цикла
9. Наименьший общий предок. (LCA)
10. Задача о быстром нахождении минимума на отрезке (RMQ)
11. Поток минимальной стоимости (min-cost-flow)
12. Дерево Фенвика
13. Декартово дерево

8. Учебно-методическое и информационное обеспечение дисциплины:

а) основная литература

1. О. Оре ; пер. с англ. И. Н. Врублевская ; ред. Н. Н. Воробьев. Теория графов. - 2-е изд. - М. : Либроком, 2009. - 352 с. ISBN 978-5-397-00044-4

2. К.Д. Кириченко, Языки программирования: учеб. пособие - Вост.-Сиб. гос. акад. образования. - Иркутск : Изд-во ВСГАО, 2014. - 105 с. - ISBN 978-5-91344-765

б) дополнительная литература

1. Н. Кристофидес ; ред. Гаврилов Г. П. Теория графов. Алгоритмический подход: пер. с англ. - М. : Мир, 1978. - 432 с.

2. Кубенский, А. А., Создание и обработка структур данных в примерах на Java. - СПб. : БХВ-Петербург, 2001. - 321 с.

3. В. Е. Алексеев, В. А. Таланов, Графы и алгоритмы. Структуры данных. Модели вычислений. : учеб. для студ. . - М. : Интернет-Ун-т информ. технологий : Бином. Лаб. знаний, 2006. - 320 с.

в) программное обеспечение

1. CodeBlocks 16.01
2. NetBeans IDE
3. Far Manager 2.0

г) базы данных, поисково-справочные и информационные системы

1. acmp.ru/
2. acm.timus.ru/
3. codeforces.ru/
4. http://acm.mipt.ru/

9. Материально-техническое обеспечение дисциплины:

Компьютерная аудитория на 15 мест,

Программное обеспечение:
IDE CodeBlocks 16.01
IDE NetBeans
Far Manager 2.0

10. Образовательные технологии:

Лекционные занятия, практические занятия в форме группового и индивидуального решения задач при помощи средств автоматических систем тестирования, мастер-класс, разбор конкретных ситуаций.

11. Оценочные средства (ОС):

11.1. Оценочные средства текущего контроля

Текущий контроль осуществляется путем ручной или автоматической проверки решенных студентом задач.

11.2. Оценочные средства для промежуточной аттестации

Промежуточная аттестация проводится путем суммирования результатов студента по всем заданным в течение курса задачам.

Для получения оценки “удовлетворительно”, студент должен решить не менее 40% предложенных для проверки задач.

Для получения оценки “хорошо”, студент должен решить не менее 60% предложенных для проверки задач.

Для получения оценки “отлично”, студент должен решить не менее 80% предложенных для проверки задач.

Приложение 1.

Примеры типовых задач для практических занятий и самостоятельной работы.

Для успешной оценки обучающийся должен написать программу на одном из допустимых языков программирования (C++, java, python), которая получает на вход данные строго в описанном в условии формате и преобразует их в ответ, который подает на выход так же строго в заданном формате. Программы могут быть проверены как преподавателем непосредственно на занятии, так и автоматической проверяющей системой.

Алгоритм Флойда

(Время: 1 сек. Память: 16 Мб)

Полный ориентированный взвешенный граф задан матрицей смежности. Постройте матрицу кратчайших путей между его вершинами. Гарантируется, что в графе нет циклов

отрицательного веса.
Входные данные
В первой строке входного файла INPUT.TXT записано единственное число N (1 <= N <= 100) - количество вершин графа. В следующих N строках по N чисел - матрица смежности графа (j-ое число в i-ой строке соответствует весу ребра из вершины i в вершину j). Все числа по модулю не превышают 100. На главной диагонали матрицы - всегда нули.
Выходные данные
В выходной файл OUTPUT.TXT выведите N строк по N чисел - матрицу кратчайших расстояний между парами вершин. j-ое число в i-ой строке должно быть равно весу кратчайшего пути из вершины i в вершину j.
Пример

№	INPUT.TXT	OUTPUT.TXT
1	4 0 5 9 100 100 0 2 8 100 100 0 7 4 100 100 0	0 5 7 13 12 0 2 8 11 16 0 7 4

Алгоритм Форда-Беллмана
(Время: 1 сек. Память: 16 Мб)
Дан ориентированный граф, в котором могут быть кратные ребра и петли. Каждое ребро имеет вес, выражающийся целым числом (возможно, отрицательным). Гарантируется, что циклы отрицательного веса отсутствуют.
Требуется посчитать длины кратчайших путей от вершины номер 1 до всех остальных вершин.
Входные данные
Во входном файле INPUT.TXT записано сначала число N (1 <= N <= 100) - количество вершин графа, далее идет число M (0 <= M <= 10000) - количество ребер. Далее идет M троек чисел, описывающих ребра: начало ребра, конец ребра и вес (вес - целое число от -100 до 100).
Выходные данные
В выходной файл OUTPUT.TXT выведите N чисел - расстояния от вершины номер 1 до всех вершин графа. Если пути до соответствующей вершины не существует, вместо длины пути выведите число 30000.
Пример

№	INPUT.TXT	OUTPUT.TXT
1	4 5 1 2 10 2 3 10 1 3 100 3 1 -10 2 3 1	0 10 11 30000

Алгоритм Дейкстры
(Время: 1 сек. Память: 16 Мб)
Дан ориентированный взвешенный граф. Для него вам необходимо найти кратчайшее расстояние от вершины S до вершины F.
Входные данные
В первой строке входного файла INPUT.TXT записаны три числа: N, S и F ($1 \le N \le 100$; $1 \le S, F \le N$), где N - количество вершин графа. В следующих N строках записаны по N

чисел - матрица смежности графа, где число в i-ой строке j-ом столбце соответствует ребру из i в j: -1 означает отсутствие ребра между вершинами, а любое неотрицательное целое число (от 0 до 100) - наличие ребра данного веса. На главной диагонали матрицы всегда записаны нули.
Выходные данные
В выходной файл OUTPUT.TXT необходимо вывести искомое расстояние или -1, если пути между указанными вершинами не существует.
Пример

№	INPUT.TXT	OUTPUT.TXT
1	3 2 1 0 1 1 4 0 1 2 1 0	3

Цикл отрицательного веса
(Время: 1 сек. Память: 16 Мб Сложность: 46%)
Дан взвешенный граф. Определить, есть ли в нем цикл отрицательного веса.
Входные данные
Во входном файле INPUT.TXT в первой строке записано число N (1 <= N <= 100) - количество вершин графа. В следующих N строках находится по N чисел - матрица смежности графа. Веса ребер не превышают по модулю 10000. Если ребра нет, соответствующее значение равно 100000.
Выходные данные
В выходной файл OUTPUT.TXT выведите "YES", если цикл существует, или "NO" в противном случае.
Пример

№	INPUT.TXT	OUTPUT.TXT
1	2 0 -1 -1 0	YES

Коммерческий калькулятор
(Время: 1 сек. Память: 16 Мб)
Фирма OISAC выпустила новую версию калькулятора. Этот калькулятор берет с пользователя деньги за совершаемые арифметические операции. Стоимость каждой операции равна 5% от числа, которое является результатом операции.
На этом калькуляторе требуется вычислить сумму N натуральных чисел (числа известны). Нетрудно заметить, что от того, в каком порядке мы будем складывать эти числа, иногда зависит, в какую сумму денег нам обойдется вычисление суммы чисел (тем самым, оказывается нарушен классический принцип «от перестановки мест слагаемых сумма не меняется»).
Например, пусть нам нужно сложить числа 10, 11, 12 и 13. Тогда если мы сначала сложим 10 и 11 (это обойдется нам в $1.05), потом результат — с 12 ($1.65), и затем — с 13 ($2.3), то всего мы заплатим $5, если же сначала отдельно сложить 10 и 11 ($1.05), потом — 12 и 13 ($1.25) и, наконец, сложить между собой два полученных числа ($2.3), то в итоге мы заплатим лишь $4.6.
Напишите программу, которая будет определять, за какую минимальную сумму денег можно найти сумму данных N чисел.
Входные данные
Во входном файле INPUT.TXT записано число N (2<=N<=100000). Далее идет N

натуральных чисел, которые нужно сложить, каждое из них не превышает 10000.
Выходные данные
В выходной файл OUTPUT.TXT выведите, сколько денег нам потребуется на нахождение суммы этих N чисел. Результат должен быть выведен с двумя знаками после десятичной точки без лидирующих нулей.
Примеры

№	INPUT.TXT	OUTPUT.TXT
1	4 10 11 12 13	4.60
2	2 1 1	0.10

Адаптивный поиск
(Время: 2 сек. Память: 16 Мб)
Ни для кого не секрет, что в основе реляционной базы данных лежит таблица, которую можно рассматривать как одномерный массив записей. Известно, что при поиске все записи таблицы просматриваются последовательно, начиная с самой первой и заканчивая найденной.
Технический отдел компании установил, что часто бывает так, что поиск одной и той же записи в таблице производится несколько раз. Основываясь на этом, программисты решили после каждого нового поискового запроса менять порядок следования записей в таблице. Другими словами, после поиска найденная запись перемещается на первое место в таблице. Очевидно, что чем чаще осуществляется поиск определенной записи, тем ближе она будет к началу таблицы и тем быстрее будет поиск этой записи.
Вашей задачей является написать программу, которая для каждого из M последовательно заданных поисковых запросов будет определять количество просмотренных записей при поиске заданной. Для простоты обозначения будем считать, что имеется таблица с N записями, где запись – это число от 1 до N. В начале все записи в таблице расположены в порядке возрастания, то есть на i-м месте в таблице находится число i. Для приведенного ниже примера при M = 2, N = 6 и запросах на поиск чисел «5» и «3» потребуется 5 и 4 просмотра записей соответственно.

Входные данные
Первая строка входного файла INPUT.TXT содержит два целых числа N и M ($1 \le N, M \le 65535$) — количество записей в таблице и количество запросов на поиск соответственно. Числа разделены одиночным пробелом.
Вторая строка содержит M натуральных чисел Ai ($1 \le Ai \le N$), разделенных одиночными пробелами, где Ai — запрос на поиск числа Ai в таблице. Запросы на поиск выполняются последовательно в порядке их ввода.
Выходные данные
Единственная строка выходного файла OUTPUT.TXT должна содержать M натуральных чисел, разделенных одиночными пробелами, i-е число – это количество просмотренных записей при поиске числа Ai.
Примеры

№	INPUT.TXT	OUTPUT.TXT
1	6 2 5 3	5 4
2	10 10	10 10 10 10 10 10 10 10 10 10

3	10 9 8 7 6 5 4 3 2 1 3 14 3 2 3 3 1 1 2 2 1 1 1 1 2 1	3 3 2 1 3 1 3 1 2 1 1 1 2 2

Размещение данных

(Время: 2 сек. Память: 32 Мб)

Телекоммуникационная сеть крупной IT-компании содержит n серверов, пронумерованных от 1 до n. Некоторые пары серверов соединены двусторонними каналами связи, всего в сети m каналов. Гарантируется, что сеть серверов устроена таким образом, что по каналам связи можно передавать данные с любого сервера на любой другой сервер, возможно с использованием одного или нескольких промежуточных серверов.

Множество серверов A называется отказоустойчивым, если при недоступности любого канала связи выполнено следующее условие. Для любого не входящего в это множество сервера X существует способ передать данные по остальным каналам на сервер X хотя бы от одного сервера из множества A.

На рис. 1 показан пример сети и отказоустойчивого множества из серверов с номерами 1 и 4. Данные на сервер 2 можно передать следующим образом. При недоступности канала между серверами 1 и 2 – с сервера 4, при недоступности канала между серверами 2 и 3 – с сервера 1. На серверы 3 и 5 при недоступности любого канала связи можно по другим каналам передать данные с сервера 4.

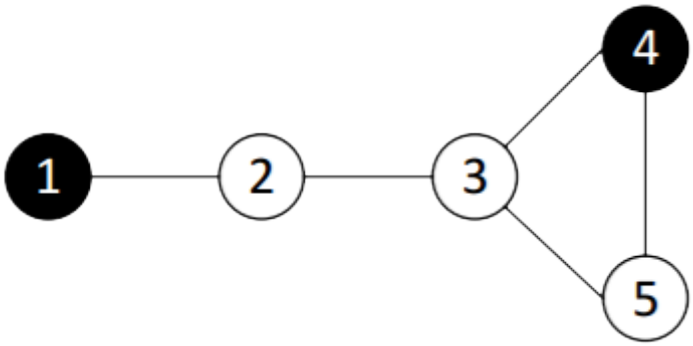

Рис. 1. Пример сети и отказоустойчивого множества серверов.

В рамках проекта группе разработчиков компании необходимо разместить свои данные в сети. Для повышения доступности данных и устойчивости к авариям разработчики хотят продублировать свои данные, разместив их одновременно на нескольких серверах, образующих отказоустойчивое множество. Чтобы минимизировать издержки, необходимо выбрать минимальное по количеству серверов отказоустойчивое множество. Кроме того, чтобы узнать, насколько гибко устроена сеть, необходимо подсчитать количество способов выбора такого множества, и поскольку это количество способов может быть большим, необходимо найти остаток от деления этого количества способов на число 10^9+7.

Требуется написать программу, которая по заданному описанию сети определяет следующие числа: k – минимальное количество серверов в отказоустойчивом множестве серверов, c – остаток от деления количества способов выбора отказоустойчивого множества из k серверов на число $10^9 + 7$.

Входные данные

Первая строка входного файла INPUT.TXT содержит целые числа n и m – количество

серверов и количество каналов связи соответственно ($2 \le n \le 200\,000$, $1 \le m \le 200\,000$). Следующие m строк содержат по два целых числа и описывают каналы связи между серверами. Каждый канал связи задается двумя целыми числами: номерами серверов, которые он соединяет.
Гарантируется, что любые два сервера соединены напрямую не более чем одним каналом связи, никакой канал не соединяет сервер сам с собой, и для любой пары серверов существует способ передачи данных с одного из них на другой, возможно с использованием одного или нескольких промежуточных серверов.

Выходные данные
В выходной файл OUTPUT.TXT выведите два целых числа, разделенных пробелом: k – минимальное число серверов в отказоустойчивом множестве серверов, c – количество способов выбора отказоустойчивого множества из k серверов, взятое по модулю $10^9 + 7$.
Пример

№	INPUT.TXT	OUTPUT.TXT
1	5 5 1 2 2 3 3 4 3 5 4 5	2 3

Пояснение к примеру
В приведенном примере отказоустойчивыми являются следующие множества из двух серверов: {1, 3}, {1, 4}, {1, 5}.

Построение последовательности
(Время: 2 сек. Память: 256 Мб)
Вам дано целое число m, а также список из n различных целых чисел от 0 до m - 1.
Вы хотите построить последовательность, удовлетворяющую следующим требованиям:
Каждый элемент последовательности — целое число от 0 до m - 1, включительно.
Все префиксные произведения (то есть, произведения положительного числа первых элементов последовательности) различны по модулю m.
Никакое префиксное произведение по модулю m не является элементом заданного списка.
Длина последовательности максимально возможна.
Постройте любую последовательность, удовлетворяющую описанным свойствам.
Входные данные
Первая строка содержит два целых числа n и m ($0 \le n < m \le 200\,000$) — число элементов в заданном списке и модуль, соответственно.
Если n ненулевое, то следующая строка содержит n различных целых чисел от 0 до m - 1 — запрещенные префиксные произведения. Если n равно нулю, то второй строки нет.

Выходные данные
В первой строке выведите число k — длину построенной последовательности.
Во второй строке выведите k целых чисел — вашу последовательность.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	0 5	5 1 2 4 3 0

2	3 10 2 9 1	6 3 9 2 9 8 0

Пояснение к примеру
В первом примере префиксные произведения построенной последовательности по модулю m равны [1, 2, 3, 4, 0].
Во втором примере префиксные произведения построенной последовательности по модулю m равны [3, 7, 4, 6, 8, 0].

Приложение 2.

Примерные вопросы к экзамену

1. Граф, определение, полный, планарный, двудольный графы. Ориентированный граф, взвешенный граф. Числовые характеристики графа.
2. Маршрут, путь, цепь, цикл, простые цепь и цикл. Ярус, диаметр, мост, точка сочленения.
3. Обходы графа.
4. Понятие алгоритма. Примеры алгоритмических процессов.
5. Сложность и эффективность алгоритма.
6. Путь, кратчайший путь. Волновой алгоритм.
7. Обход в ширину, очередь. Поиск выхода из лабиринта.
8. Графы с неотрицательными ребрами. Алгоритм Дейкстры.
9. Случай ребер разного знака. Циклы отрицательного веса.
10. Алгоритм Форда-Беллмана.
11. Алгоритм Флойда.
12. Поиск компонент связности, обход в глубину.
13. Алгоритм поиска мостов и точек сочленения.
14. Двудольные графы и паросочетания. Алгоритм Куна поиска максимального паросочетания.
15. Задача о назначениях, венгерский алгоритм решения задачи о назначениях.
16. Сеть, поток в сети, пропускная способность, увеличивающий путь.
17. Алгоритм Форда-Фалкерсона поиска максимального потока в сети.
18. Дерево, элементы и методы представления дерева.
19. Бинарные деревья и их приложение в обработке и хранении данных.
20. Бинарное дерево поиска.
21. Бинарная куча.
22. Дерево отрезков.

Разработчик: к.ф.-м.н, доцент Зубков О.В.

Настоящая программа не может быть воспроизведена ни в какой форме без предварительного письменного разрешения кафедры-разработчика программы.