Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Курганский государственный университет»

Кафедра «Программное обеспечение автоматизированных систем»

УТВЕРЖДАЮ:
Ректор
_____ Н.В. Дубив
« 31 » августа 2020 г.

Рабочая программа учебной дисциплины

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

образовательной программы высшего образования –
программы бакалавриата

09.03.03 Прикладная информатика

Направленность
Интеллектуальные информационные системы и технологии

Форма обучения: очная

Курган 2020

Рабочая программа дисциплины «Алгоритмы и структуры данных» составлена в соответствии с учебным планом программы бакалавриата: «Прикладная информатика» (Интеллектуальные информационные системы и технологии), утвержденным для очной формы обучения 28 августа 2020 г.

Рабочая программа дисциплины одобрена на заседании кафедры «Программное обеспечение автоматизированных систем» 31.08.2020 года, протокол № 1.

Рабочую программу составил:
доцент кафедры ПОАС _____/А.М. Семахин/

Согласовано:

Заведующий кафедрой
«Программное обеспечение
автоматизированных систем» _____ /Т.Р. Змызгова/

Специалист
по учебно-методической работе
Учебно-методического отдела _____/Г. В Казанкова/

Начальник
Управления образовательной
деятельности _____/С.Н. Синицын/

1. ОБЪЕМ ДИСЦИПЛИНЫ

Всего: 4 зачетных единицы трудоемкости (144 академических часов)

Очная форма обучения

Вид учебной работы	На всю дисциплину	Семестр
		3
Аудиторные занятия (контактная работа с преподавателем), всего часов в том числе:	**48**	**48**
Лекции	16	16
Лабораторные работы	32	32
Практические занятия	-	-
Аудиторные занятия в интерактивной форме, часов	-	-
Самостоятельная работа, всего часов в том числе:	**96**	**96**
Курсовая работа	36	36
Подготовка к зачету	18	18
Другие виды самостоятельной работы (самостоятельное изучение тем (разделов) дисциплины)	42	42
Вид промежуточной аттестации	**зачет**	**зачет**
Общая трудоемкость дисциплины и трудоемкость по семестрам, часов	**144**	**144**

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина «Алгоритмы и структуры данных» относится к дисциплинам обязательной части блока 1 «Дисциплины (модули)», Программирование.

Изучение дисциплины базируется на знаниях, умениях и навыках, приобретенных студентами, при изучении следующих дисциплин:

- Информатика.
- Основы программирования.
- Дискретная математика.
- Вычислительная математика.

Результаты обучения по дисциплине необходимы для изучения дисциплин: «Теория информации», «Управление качеством и тестирование ПО», «Объектно-ориентированное программирование», «Методы и системы принятия решений» и выполнения выпускной квалификационной работы.

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Цель освоения дисциплины: формирование знаний и практических навыков разработки и анализа алгоритмов программ и выбора структур данных.

Задачи дисциплины: изучение основ теории структур, методов представления данных на логическом (абстрактном) и физическом (машинном) уровнях, методов разработки эффективных алгоритмов обработки структур данных.

Компетенции, формируемые в результате освоения дисциплины:

- способность использовать современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности (ОПК-2);
- способность разрабатывать алгоритмы и программы, пригодные для практического применения(ОПК-7).

В результате изучения дисциплины обучающийся должен

знать:

- современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности (ОПК-2);

уметь:

- использовать современные информационные технологии и программные средства, в том числе отечественного производства при решении задач профессиональной деятельности (ОПК-2);
- разрабатывать алгоритмы и программы, пригодные для практического применения (ОПК-7);

владеть:

- современными информационными технологиями и программными средствами, в том числе отечественного производства при решении задач профессиональной деятельности (ОПК-2);
- методами разработки алгоритмов и программ, пригодных для практического применения (ОПК-7).

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Учебно-тематический план.

Очная форма обучения

Рубеж	Номер раздела, темы	Наименование раздела, темы	Количество часов контактной работы с преподавателем	
			Лекции	Лабораторные работы
Рубеж 1	1	Типы данных и базовые структуры данных	2	4
	2	Линейные структуры данных	2	4
	3	Нелинейные структуры данных	4	4
		Рубежный контроль №1	-	2
Рубеж 2	4	Алгоритмы сортировки данных	2	4
	5	Алгоритмы поиска данных	2	4

	6	Алгоритмы на графах	4	8
		Рубежный контроль №2	-	2
Всего:			**16**	**32**

4.2. Содержание лекционных занятий

Тема 1. Типы данных и базовые структуры данных

Алгоритмы и данные. Свойства алгоритма. Анализ сложности алгоритма. Семантика, синтаксис, прагматика. Структура данных. Структуры хранения данных: вектор, список, сеть. Массивы. Структуры данных массивов. Структуры хранения массивов. Строки. Операции над строками. Записи. Операции над записями. Множества.

Тема 2. Линейные структуры данных

Списки. Структура и классификация списков. Операции над линейными списками. Применение списков. Стеки. Структура стека. Операции над стеками. Применение стеков. Очереди. Деки. Операции над очередями и деками. Применение очередей и деков.

Тема 3. Нелинейные структуры данных

Алгоритм преобразования m-арного дерева в бинарное дерево. Представление деревьев в памяти ЭВМ. Идеально-сбалансированное бинарное дерево. Бинарные (двоичные) деревья поиска. Сбалансированные деревья поиска. Сбалансированные АВЛ-деревья поиска. Рандомизированные деревья поиска. Оптимальные деревья поиска. Операции над деревьями. B-деревья. Операции над B-деревьями.

Тема 4. Алгоритмы сортировки данных

Основные понятия и классификация алгоритмов сортировки. Внутренняя сортировка. Метод прямого включения. Метод прямого выбора. Метод прямого обмена. Шейкерная сортировка. Быстрые (улучшенные) методы сортировки. Метод Шелла. Метод пирамиды. Метод Хоара. Поразрядная сортировка. Внешняя сортировка. Прямое слияние. Естественное слияние. Сбалансированное многопутевое слияние. Многофазная сортировка. Каскадная сортировка.

Тема 5. Алгоритмы поиска данных

Классификация алгоритмов поиска. Поиск в последовательно организованных структурах. Последовательный поиск. Двоичный и Фибоначчиев поиск. Интерполяционный поиск. Индексно-последовательный поиск. Поиск в деревьях. Случайные двоичные деревья поиска. Оптимальные двоичные деревья поиска. Сбалансированные деревья поиска. Хеширование. Хеш-функции. Разрешение коллизий методом цепочек. Разрешение коллизий методом открытой адресации. Идеальное хеширование.

Тема 6. Алгоритмы на графах

Основные понятия и определения. Представление графов. Матрица смежности. Векторы смежности. Списки смежности. Матрица инцидентности. Пути в графе. Путевая матрица (матрица достижимости). Кратчайшие пути. Алгоритм Дейкстры. Алгоритм Флойда. Остовные деревья графа. Обходы графов. Поиск в глубину. Поиск в ширину.Остовное дерево наименьшей стоимости. Алгоритм Прима. Алгоритм Крускала. Упорядочение графа

4.3. Лабораторные занятия

Номер раздела, темы	Наименование раздела, темы	Наименование лабораторной работы	Норматив времени, час.
			Очная форма обучения
1	Типы данных и базовые структуры данных	Структуры данных списки, очереди	4
2	Линейные структуры данных	Структуры данных стеки, деки	4
3	Нелинейные структуры данных	Двоичные деревья поиска	4
		Рубежный контроль №1	2
4	Алгоритмы сортировки данных	Методы упорядочивания данных.	4
5	Алгоритмы поиска данных	Поиск в последовательно организованных структурах.	4
6	Алгоритмы на графах	Алгоритм определения кратчайших путей на графах	8
		Рубежный контроль №2	2
		Всего:	**32**

4.4 Практические занятия

Практические занятия не предусмотрены учебным планом.

4.5. Курсовая работа

Курсовая работа посвящена разработке визуального приложения, формализующего алгоритм решения задачи, представленной в варианте задания согласно методическим рекомендациям, указанным в разделе 8.

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Лекционный курс основывается на методе обучения, использующем технологию, при которой студенты конспектируют теоретический материал, участвуют в опросах и дискуссиях. В этом случае задействованы зрительная, слуховая, моторная и ассоциативная виды памяти.

При прослушивании лекций рекомендуется в конспекте отмечать все важные моменты, на которых заостряет внимание преподаватель, в частности те, которые направлены на качественное выполнение соответствующей лабораторной работы.

Преподавателем запланировано использование при чтении лекций технологии учебной дискуссии. Поэтому рекомендуется фиксировать для себя интересные моменты с целью их активного обсуждения на дискуссии в конце лекции.

Залогом качественного выполнения лабораторных работ и занятий является самостоятельная подготовка к ним накануне путем повторения материалов лекций. Рекомендуется подготовить вопросы по неясным моментам и обсудить их с преподавателем в начале занятия.

Лабораторные работы выполняются с применением интегрированной среды программирования Microsoft Visual Studio 2019 Community, объектно-ориентированного языка программирования Visual C++ и новых версий этих программных продуктов.

Преподавателем запланировано применение на лабораторных занятиях технологий развивающейся кооперации, коллективного взаимодействия, разбора конкретных ситуаций.

Для текущего контроля успеваемости по очной форме обучения преподавателем используется балльно-рейтинговая система контроля и оценки академической активности. Поэтому настоятельно рекомендуется тщательно прорабатывать материал дисциплины при самостоятельной работе, участвовать во всех формах обсуждения и взаимодействия, как на лекциях, так и на лабораторных занятиях в целях лучшего освоения материала и получения высокой оценки по результатам освоения дисциплины.

Выполнение самостоятельной работы подразумевает самостоятельное изучение разделов дисциплины, подготовку к лабораторным занятиям, к рубежным контролям, выполнение курсовой работы, подготовку к зачёту.

Рекомендуемая для очной формы обучения трудоемкость самостоятельной работы представлена в таблице:

Рекомендуемый режим самостоятельной работы

Наименование вида самостоятельной работы	**Рекомендуемая трудоемкость, акад. час.**
	очная форма обучения
Самостоятельное изучение тем дисциплины:	**22**
Алгоритмы решения задач выбора	2
Красно-черные деревья	3
Базовое восходящее скошенное дерево	3
Нисходящие скошенные деревья	3
Поиск кратчайшего отрицательно взвешенного пути на графе	3

Алгоритм нахождения максимального пути на графе	3
Алгоритм Беллмана-Мура нахождения кратчайшего пути на графе.	3
Определение экстремальных путей на графах. Метод Шимбелла	2
Подготовка к лабораторным занятиям (по 1 часу на каждое занятие)	**16**
Подготовка к рубежным контролям (по 2 часа на каждый рубеж)	**4**
Выполнение курсовой работы	**36**
Подготовка к зачёту	**18**
Всего:	**96**

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

6.1. Перечень оценочных средств

1. Балльно-рейтинговая система контроля и оценки академической активности студентов в КГУ.
2. Отчеты студентов по лабораторным работам.
3. Банк заданий к рубежным контролям № 1, № 2.
4. Вопросы к зачёту.
5. Курсовая работа.

6.2. Система балльно-рейтинговой оценки работы студентов по дисциплине

Очная форма обучения

<table>
<tr><td>№</td><td>Наименование</td><td colspan="6">Содержание</td></tr>
<tr><td rowspan="6">1</td><td rowspan="6">Распределение баллов за семестры по видам учебной работы, сроки сдачи учебной работы (доводятся до сведения студентов на первом учебном занятии)</td><td colspan="6">Распределение баллов,
3 семестр</td></tr>
<tr><td>Вид учебной работы:</td><td>Посещение лекций</td><td>Выполнение и защита отчетов по лабораторным работам</td><td>Рубежный контроль №1</td><td>Рубежный контроль №2</td><td>Зачёт</td></tr>
<tr><td>Балльная оценка:</td><td>1б*8=8б</td><td>9б*5+7б*1=52б</td><td>5</td><td>5</td><td>30</td></tr>
<tr><td colspan="6">Курсовая работа</td></tr>
<tr><td colspan="2">Качество пояснительной записки</td><td colspan="2">Качество программной части</td><td>Качество защиты</td><td>Всего</td></tr>
<tr><td></td><td></td><td colspan="2"></td><td></td><td></td></tr>
</table>

		До 40	До 30	До 30	100
2	Критерий пересчета баллов в традиционную оценку по итогам работы в семестре и зачёта	60 и менее баллов – незачтено; 61…100 – зачтено.			
3	Критерии допуска к промежуточной аттестации, возможности получения автоматического зачета (экзаменационной оценки) по дисциплине, возможность получения бонусных баллов	Для допуска к промежуточной аттестации (зачету) студент должен набрать по итогам текущего и рубежного контроля не менее 50 баллов, выполнить все лабораторные работы и курсовую работу. Для получения «автоматически» оценки «зачтено» студенту необходимо набрать 61 балл. По согласованию с преподавателем студенту могут быть добавлены дополнительные (бонусные) баллы за активность на консультациях, активное участие в научной и методической работе, оригинальность принятых решений в ходе выполнения практических работ, за участие в значимых учебных и внеучебных мероприятиях кафедры.			
4	Формы и виды учебной работы для неуспевающих (восстановившихся на курсе обучения) студентов для получения недостающих баллов в конце семестра	В случае если к промежуточной аттестации (зачету) набрана сумма менее 50 баллов, студенту необходимо набрать недостающее количество баллов за счет выполнения дополнительных заданий, до конца последней (зачетной) недели семестра. При этом необходимо проработать материал всех пропущенных лабораторных занятий. Формы дополнительных заданий (назначаются преподавателем): - выполнение и защита пропущенного лабораторного занятия (при невозможности проведения дополнительного занятия преподаватель самостоятельно устанавливает форму дополнительного задания по тематике пропущенного лабораторного занятия) – до 8 баллов. Ликвидация академических задолженностей, возникших из-за разности в учебных планах при переводе или восстановлении, проводится путем выполнения дополнительных заданий, форма и объем которых определяется преподавателем.			

6.3. Процедура оценивания результатов освоения дисциплины

Рубежные контроли проводятся в форме письменного тестирования, зачёт в устной форме виде ответов на вопросы в билетах к зачёту.

Перед проведением рубежного контроля преподаватель прорабатывает со студентами основной материал соответствующих разделов дисциплины в форме краткой лекции-дискуссии.

Варианты заданий для рубежных контролей № 1, № 2 состоят из 20 вопросов. Для определения баллов при проверке рубежных контролей используются интервальные оценки, представленные в таблице

Количество правильных ответов	1-5	6-8	9-11	12-14	15-17	18-20
Количество баллов	0	1	2	3	4	5

На каждую подготовку к рубежному контролю студенту отводится 1 академический час.

Преподаватель оценивает в баллах результаты рубежных контролей каждого студента по количеству правильных ответов и заносит в ведомость учета текущей успеваемости.

Билет к зачету состоит из 1 вопроса. Вопросы к зачёту доводятся до студентов на последней лекции в семестре. На подготовку ответа по вопросам билета к зачёту студенту отводится 1 астрономический час.

Результаты текущего контроля успеваемости и зачёта заносятся преподавателем в зачётную ведомость, которая сдаётся в организационный отдел института в конце зачётной недели, а также выставляется в зачетную книжку студента.

6.4 Примеры оценочных средств для рубежных контролей, зачёта

6.4.1 Примеры заданий для рубежного контроля №1

Вариант 1_1

1. Что называется алгоритмом?

1. Система формальных правил, четко и однозначно определяющая процесс решения поставленной задачи в виде конечной последовательности действий или операций.
2. Набор команд.
3. Последовательность действий.
4. Совокупность правил и инструкций, оформленных в соответствии стандарта.

2. Какие характеристики алгоритма?

1. Время работы алгоритма.
2. Количество операторов.
3. Количество переходов.
4. Объем памяти компьютера.

3. Что называется вычислительной сложностью (трудоемкостью) алгоритма T(n), где n – размер задачи?

1. Объем памяти компьютера, необходимый для его выполнения.
2. Количество операций, необходимых для его выполнения.
3. Количество переходов.
4. Количество операторов.

4. Что называется ёмкостной сложностью алгоритма?

1. Количество операций, необходимых для его выполнения.
2. Количество переходов.
3. Количество операторов.
4. Объем памяти компьютера, требуемый для реализации алгоритма.

5. Что является свойством алгоритма?

1. Финитность.
2. Динамичность.
3. Гибкость.
4. Эффективность.

6. Какие случаи проведения анализа алгоритма существуют?

1. Сравнение алгоритмов.
2. Оценка производительности.
3. Установка значений параметров алгоритма.
4. Просмотр алгоритма.

7. Что называется О-нотацией?

1. Математическое выражение.
2. Свойство функции.
3. Математическая запись, позволяющая отбрасывать подробности при анализе алгоритмов.
4. Алгоритмическая запись.

8. Каковы причины использования О-нотации при оценке функции вычислительной сложности алгоритма?

1. Не учитывать вклад малых слагаемых в математических формулах.
2. Удобство восприятия.
3. Простота записи.
4. Классифицировать алгоритмы согласно верхней границе их общего времени выполнения.

9. Что означает выражение O(1)?

1. Время выполнения алгоритма равно 1.
2. Константное время выполнения алгоритма, не зависящее от n.
3. Ошибка в алгоритме.
4. Нулевой алгоритм.

10. Что называется данными?

1. Сведения, полученные путем измерения наблюдения логических или арифметических операций и представленные в форме, пригодной для хранения, передачи и обработки.
2. Множество числовых величин.
3. Совокупность сообщений.
4. Характеристики случайных величин.

11. Что называется типом данных?

1. Набор данных, определяющий диапазон значений.
2. Набор данных, определяющий способ представления.
3. Набор данных, определяющий допустимые операции.
4. Набор данных, определяющий диапазон возможных значений, способ представления, допустимые операции над данными.

12. Что называется структурой данных?

1. Набор правил и ограничений, определяющих связи между отдельными элементами данных.

2. Набор правил и ограничений, определяющих связи между отдельными группами данных.

3. Набор правил и ограничений, определяющих связи между отдельными элементами и группами данных.

4. Некоторую иерархию данных.

13. Что называется динамической структурой данных?

1. Структура данных, размер и конфигурация которых изменяется во время выполнения программы.

2. Структура данных, отражающая способ физического представления данных в машинной памяти.

3. Структура данных, которая не может быть расчленена на составные части, большие, чем биты.

4. Структура данных, составными частями которых являются другие структуры данных.

14. Как называется линейная структура данных, для которой выполняется принцип LIFO?

1. Стек
2. Очередь.
3. Дек.
4. Линейный список.

15. Как называется линейная структура данных, для которой выполняется принцип FIFO?

1. Стек
2. Очередь.
3. Дек.
4. Линейный список.

16. Древовидная структура – иерархическая структура, состоящая из набора вершин и ребер, каждая вершина содержит определенную информацию и ссылку на вершину нижнего уровня?

1. Да.
2. Нет.

17. Как называется узел дерева, который не имеет потомков?

1. Корень.
2. Лист.
3. Промежуточный.
4. Конечный.

18. Что называется высотой дерева?

1. Максимальное количество узлов.
2. Максимальное количество связей.
3. Максимальная длина пути от корня до листа.
4. Максимальное количество листьев.

19. Какое дерево называется бинарным?

1. Дерево, узлы которого имеют не менее двух предков.
2. Дерево, из каждой вершины которого выходит по два ребра.
3. Дерево, у которого от корня до листа не более двух уровней.
4. Дерево, у которого от корня до листа не менее двух уровней.

20. Каким образом представляется бинарное дерево?

1. Посредством указателей.
2. Посредством массивов.
3. Посредством индексов.
4. Правильного ответа нет.

Вариант 1_2

1. Что называется динамической структурой данных?

1. Данные, размер которых изменяется во время выполнения программы, память выделяется по мере необходимости.
2. Данные, память под которые выделяется во время компиляции и сохраняется в течение всей работы программы.
3. Данные, которые предоставляют для работы с элементами данных определённый набор функций, возможность создавать элементы данных при помощи специальных функций.
4. Данные из которых строятся более сложные данные.

2. Что называется линейным списком?

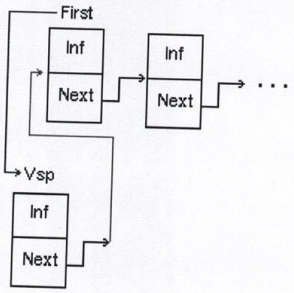
1. Иерархическая структура данных, состоящая из связанных между собой элементов.
2. Структура данных, состоящая из связанных между собой элементов, расположенных на одном уровне иерархии.
3. Структура данных, состоящая из связанных между собой элементов, расположенных на разных уровнях иерархии.
4. Линейная структура данных, состоящая из элементов одного типа, связанных между собой указателями.

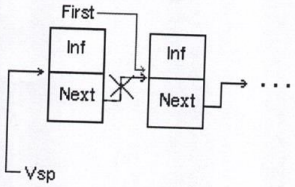
3. Какой тип данных используется для формализации элемента списка?

1. Структура.
2. Массив.
3. Объединение.
4. Множество.

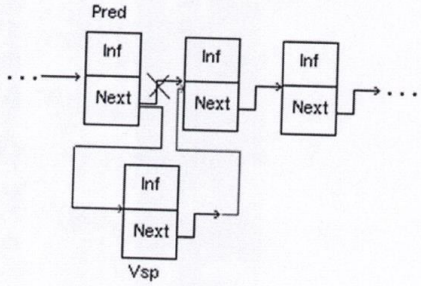
4. Добавление звена в начало списка

1. 2.

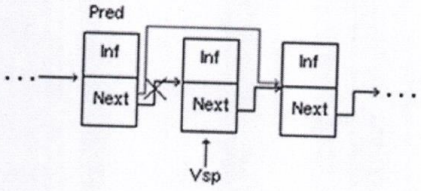




3.

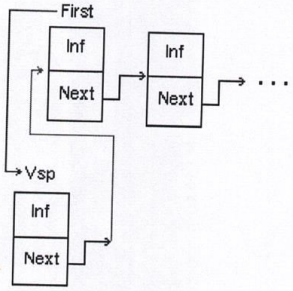


4.

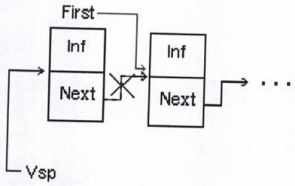


5. Удаление звена из начала списка

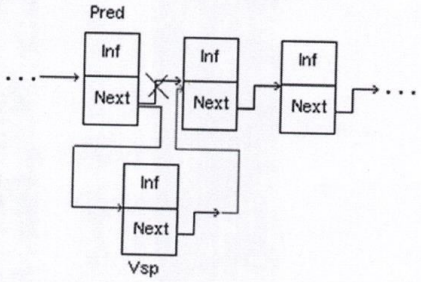
1.



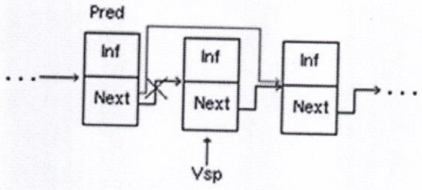
2.



3.

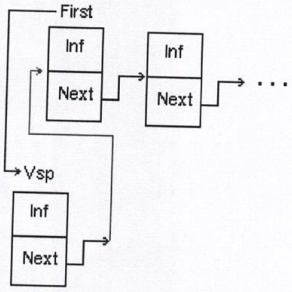


4.

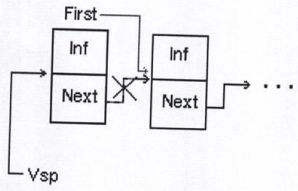


6. Добавление звена в произвольное место списка, отличное от начала (после звена, указатель на которое задан)

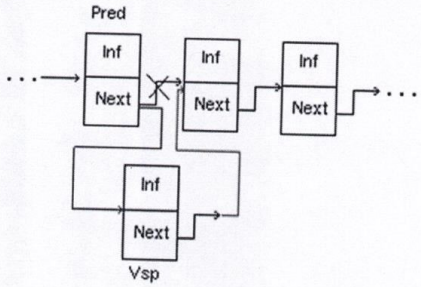
1.



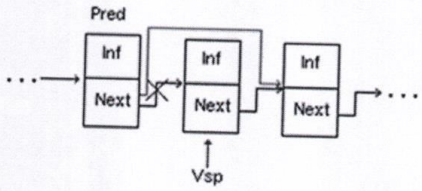
2.



3.

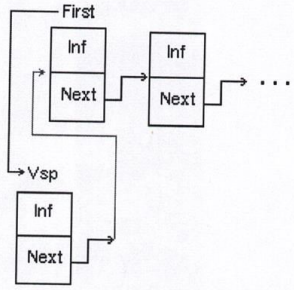


4.

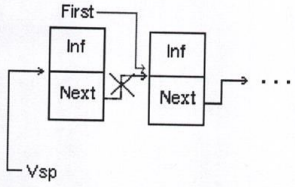


7. Удаление звена из произвольного места списка, отличного от начала (после звена, указатель на которое задан)

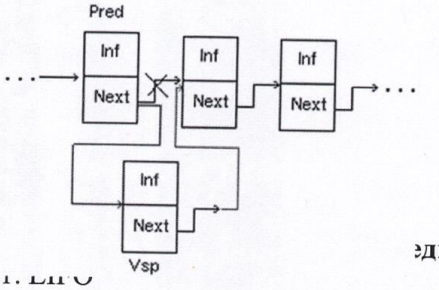
1.



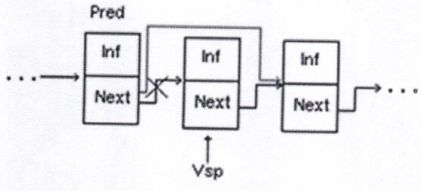
2.



3.



4.



еди?

1. LIFO

2. API
3. GUI
4. FIFO

9. Какой принцип работы стека?

1. . API
2. GUI
3. LIFO
4. FIFO

10. Что называется бинарным деревом поиска?

1. Упорядоченное дерево, вершина которого имеет не более двух поддеревьев. Ключи в левом поддереве больше значения данного узла, ключи правого поддерева меньше значения данного узла.

2. Упорядоченное дерево, вершина которого имеет не более двух поддеревьев. Ключи в левом поддереве меньше значения данного узла, ключи правого поддерева больше значения данного узла.

3. Упорядоченное дерево, вершина которого имеет не более двух поддеревьев. Все пути от корня до любого листа имеют одинаковую длину.

4. Упорядоченное дерево, вершина которого имеет не более двух поддеревьев. Высота его двух поддеревьев различается не более чем на 1.

11. Что называется высотой дерева?

1. Количество узлов дерева.
2. Количество связей дерева.
3. Количество листьев дерева.
4. Количество уровней, на которых располагаются узлы дерева.

12. Как определяется длина пути дерева?

1. Сумма уровней всех узлов дерева.
2. Максимальное количество ребер.
3. Сумма уровней всех внутренних узлов дерева.
4. Сумма уровней всех внешних узлов дерева.

13. Что называется глубиной дерева?

1. Сумма максимального уровня листа дерева и длины пути от корня дерева до данного узла.

2. Разность между длиной пути от корня дерева до данного узла и максимальным уровнем листа дерева.

3. Среднее значение максимального уровня листа дерева и длины пути от корня дерева до данного узла.

4. Максимальный уровень листа дерева или длина пути от корня дерева до данного узла.

14. Какой алгоритм удаления узла дерева?

1. На место удаляемого узла помещается потомок.

2. На место удаляемого узла помещается самый правый лист правого поддерева, или самый левый лист левого поддерева.

3. На место удаляемого узла помещается самый левый лист правого поддерева, или самый правый лист левого поддерева.

4. На место удаляемого узла помещается предок.

15. Какая структура узла бинарного дерева?

1. Узел состоит из четырех полей:
   - информационное поле (ключ вершины);
   - служебное поле;
   - указатель на левое поддерево;
   - указатель на правое поддерево.
2. Узел состоит из четырех полей:
   - информационное поле (ключ вершины);
   - служебное поле;
   - указатель на корень дерева;
   - указатель на лист дерева.
3. Узел состоит из трех полей:
   - информационное поле (ключ вершины);
   - служебное поле;
   - указатель на левое поддерево;
4. Узел состоит из трех полей:
   - информационное поле (ключ вершины);
   - служебное поле;
   - указатель на правое поддерево.

16. Что называется сбалансированным бинарным деревом (B-balanced)?

1. Бинарное дерево, высота левого поддерева каждого узла равна высоте правого поддерева.

2. Бинарное дерево, высота левого поддерева каждого узла отличается от высоты правого не более чем на 1.

3. Бинарное дерево, высота левого поддерева каждого узла отличается от высоты правого не более чем на 2.

4. Бинарное дерево, высота левого поддерева каждого узла отличается от высоты правого на произвольное число n>1.

17. В чем особенность алгоритма последовательного поиска?

1. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

2. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

3. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

4. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

18. В чем сущность интерполяционного поиска?

1. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

2. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

3. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

4. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

19. В чем особенность бинарного поиска?

1. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

2. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

3. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

4. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

20. Какая функция программы, формализующей двунаправленный линейный список с использованием структуры struct Node{ int d; Node *next; Node *prev; }; **и операции над элементами, формирует первый элемент?**

1. *Node * f1(int d){*
*Node *pv = new Node;*
pv->d = d; pv->next = 0; pv->prev = 0;
return pv;
}
2. *void f2(Node **pend, int d){*
*Node *pv = new Node;*
*pv->d = d; pv->next = 0; pv->prev = *pend;*
*(*pend)->next = pv;*
**pend = pv;*
}
3. *Node * f3(Node * const pbeg, int d){*
*Node *pv = pbeg;*
while (pv){
if(pv->d == d) break;
pv = pv->next;
}

```
return pv;
}
4. bool f4(Node **pbeg, Node **pend, int key){
if(Node *pkey = find(*pbeg, key)){
if (pkey == *pbeg){
*pbeg = (*pbeg)->next;
(*pbeg)->prev =0;}
else if (pkey == *pend) {
*pend = (*pend)->prev;
(*pend)->next =0; }
else{
(pkey->prev)->next = pkey->next;
(pkey->next)->prev = pkey->prev; }
delete pkey;
return true;
}
return false;
}
```

6.4.2 Примеры заданий для рубежного контроля №2

Вариант 2_1

1. Что такое функция сложности алгоритма?

1. Функция, характеризующая структуру алгоритма.
2. Функция, характеризующая объем алгоритма.
3. Функция, выражающая относительную скорость алгоритма в зависимости от некоторой переменной (переменных).
4. Функция, характеризующая эффективность алгоритма.

2. Сортировка – упорядочение элементов множества в возрастающем или убывающем порядке?

1. Да.
2. Нет.

3. Какова сущность сортировки выбором?

1. В неупорядоченном списке выбирается и отделяется от остальных наименьший элемент. Исходный список оказывается измененным. Измененный список принимается за исходный и процесс продолжается до тех пор, пока все элементы не будут выбраны.
2. Из неупорядоченной последовательности элементов выбирается поочередно каждый элемент, сравнивается с предыдущим, уже упорядоченным, и помещается на соответствующее место.
3. Анализируются первые элементы обоих массивов. Меньший элемент переписывается в новый массив. Оставшийся элемент последовательно сравнивается с элементами из другого массива. В новый массив после каждого сравнения попадает меньший элемент. Процесс продолжается до исчерпания эле-

ментов одного из массивов. Затем остаток другого массива дописывается в новый массив.

4. Элементы последовательно сравниваются между собой и меняются местами в случае, если предшествующий элемент больше последующего.

4. Какова сущность сортировки вставкой?

1. В неупорядоченном списке выбирается и отделяется от остальных наименьший элемент. Исходный список оказывается измененным. Измененный список принимается за исходный и процесс продолжается до тех пор, пока все элементы не будут выбраны.

2. Из неупорядоченной последовательности элементов выбирается поочередно каждый элемент, сравнивается с предыдущим, уже упорядоченным, и помещается на соответствующее место.

3. Анализируются первые элементы обоих массивов. Меньший элемент переписывается в новый массив. Оставшийся элемент последовательно сравнивается с элементами из другого массива. В новый массив после каждого сравнения попадает меньший элемент. Процесс продолжается до исчерпания элементов одного из массивов. Затем остаток другого массива дописывается в новый массив.

4. Элементы последовательно сравниваются между собой и меняются местами в случае, если предшествующий элемент больше последующего.

5. Какова сущность сортировки слиянием?

1. В неупорядоченном списке выбирается и отделяется от остальных наименьший элемент. Исходный список оказывается измененным. Измененный список принимается за исходный и процесс продолжается до тех пор, пока все элементы не будут выбраны.

2. Из неупорядоченной последовательности элементов выбирается поочередно каждый элемент, сравнивается с предыдущим, уже упорядоченным, и помещается на соответствующее место.

3. Анализируются первые элементы обоих массивов. Меньший элемент переписывается в новый массив. Оставшийся элемент последовательно сравнивается с элементами из другого массива. В новый массив после каждого сравнения попадает меньший элемент. Процесс продолжается до исчерпания элементов одного из массивов. Затем остаток другого массива дописывается в новый массив.

4. Элементы последовательно сравниваются между собой и меняются местами в случае, если предшествующий элемент больше последующего.

6. Какова сущность сортировки обменом?

1. В неупорядоченном списке выбирается и отделяется от остальных наименьший элемент. Исходный список оказывается измененным. Измененный список принимается за исходный и процесс продолжается до тех пор, пока все элементы не будут выбраны.

2. Из неупорядоченной последовательности элементов выбирается поочередно каждый элемент, сравнивается с предыдущим, уже упорядоченным, и помещается на соответствующее место.

3. Анализируются первые элементы обоих массивов. Меньший элемент переписывается в новый массив. Оставшийся элемент последовательно сравнивается с элементами из другого массива. В новый массив после каждого сравнения попадает меньший элемент. Процесс продолжается до исчерпания элементов одного из массивов. Затем остаток другого массива дописывается в новый массив.

4. Элементы последовательно сравниваются между собой и меняются местами в случае, если предшествующий элемент больше последующего.

7. Внутренняя сортировка – сортировка данных, элементы которых располагаются в оперативной памяти?

1. Да.
2. Нет.

8. Какие сложные методы сортировки?

1. Метод фон Неймана.
2. Сортировка Шелла.
3. Сортировка Хоара.
4. Метод «пузырька».

9. Внешняя сортировка – сортировка данных, элементы которых располагаются на внешнем носителе?

1. Да.
2. Нет.

10. Какова сущность поиска в глубину?

1. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$, то поиск закончен);

2. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=u, то поиск закончен);

3. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$, то поиск закончен).

4. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u.

Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$ и v=u то поиск закончен).

11. Что является стандартным способом устранения рекурсии при поиске в глубину?

1. Использование массива.
2. Использование очереди.
3. Использование стека.
4. Использование циклического списка.

12. Что используется при поиске в ширину?

1. Массив.
2. Очередь
3. Стек.
4. Циклический список

13. Какой доступ к информации в последовательном файле?

1. Последовательный.
2. Последовательный и произвольный.
3. Произвольный.
4. Прямой.

14. В чем особенность алгоритма последовательного поиска?

1. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

2. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

3. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

4. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

15. В чем сущность интерполяционного поиска?

1. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

2. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

3. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

4. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

16. В чем особенность бинарного поиска?

1. Исходное множество не упорядочено. Отыскивается ключ, последовательно сравнивая со всеми элементами множества. Поиск заканчивается досрочно, если ключ найден.

2. Исходное множество упорядочено по возрастанию весов. Сравнение осуществляется на расстоянии шага d. Значение шага рассчитывается после каждого этапа. Алгоритм заканчивается при d=0.

3. Исходное множество упорядочено по возрастанию. Отыскиваемый ключ сравнивается с центральным элементом множества. Если он меньше центрального, то поиск продолжается в левом подмножестве, иначе – в правом.

4. Анализируются элементы, находящиеся в позициях, равных числам Фибоначчи. Поиск продолжается, пока не будет найден интервал между двумя ключами, где может располагаться отыскиваемый ключ.

17. Какие условия выполняются для пирамиды?

1. - Каждая конечная вершина имеет высоту h или h-1;
- каждая конечная вершина высоты h находится слева от любой конечной вершины высоты h-1;
- ключ любой вершины больше ключа следующей за ней вершины (потомка).

2. - Каждая конечная вершина имеет высоту h или h-1;
- каждая конечная вершина высоты h находится справа от любой конечной вершины высоты h-1;
- ключ любой вершины больше ключа следующей за ней вершины (потомка).

3. - Каждая конечная вершина имеет высоту h или h-1;
- каждая конечная вершина высоты h находится слева от любой конечной вершины высоты h-1;
- ключ любой вершины меньше ключа следующей за ней вершины (потомка).

4. - Каждая конечная вершина имеет высоту h или h+1;
- каждая конечная вершина высоты h находится слева от любой конечной вершины высоты h+1;
- ключ любой вершины больше ключа следующей за ней вершины (потомка).

18. В чем особенность поиска с использованием перемешанной таблицы (хэш-таблицы)?

1. Организация данных представляется в виде очереди. Ключ преобразуется в индекс соответствующего столбца таблицы.

2. Организация данных представляется в виде стека. Ключ преобразуется в индекс соответствующей строки в таблице.

3. Организация данных представляется в виде массива. Ключ преобразуется в индекс соответствующей строки в таблице.

4. Организация данных представляется в виде списка. Ключ преобразуется в индекс соответствующего столбца таблицы.

19. Какие требования предъявляются к функции преобразования?

1. Распределяет ключи как можно более неравномерно по шкале значений индексов.

2. Распределяет ключи как можно более равномерно по шкале значений индексов.

3. Распределяет ключи произвольно по шкале значений индексов.

4. Распределяет ключи выборочно по шкале значений индексов.

20. Как определяется эффективность хэш-поиска?

1. $L_{cp} = \frac{1-\alpha/2}{1-2*\alpha}$

2. $L_{cp} = \frac{1+\alpha/2}{1-\alpha}$

3. $L_{cp} = \frac{1-\alpha/2}{1+\alpha}$

4. $L_{cp} = \frac{1-\alpha/2}{1-\alpha}$

Вариант 2_2

1. Что называется графом?

1. Конечное множество вершин и конечное множество весов G(V, K).

2. Конечное множество вершин и конечное множество ребер G(V, E).

3. Конечное множество вершин и конечное множество путей G(V, P).

4. Конечное множество вершин и конечное множество инциденций G(V, I).

2. Что называется путем в графе?

1. Последовательность ребер простого орграфа, при которой весовой коэффициент дуги коррелирует с весовым коэффициентом следующей дуги.

2. Последовательность ребер простого орграфа, при которой конечная вершина любого ребра не является начальной вершиной следующего за ним ребра.

3. Последовательность ребер простого орграфа, при которой конечная вершина любого ребра является начальной вершиной следующего за ним ребра.

4. Последовательность ребер простого орграфа, при которой конечная вершина любого ребра циклически связана с начальной вершиной следующего за ним ребра.

3. Что называется матрицей смежности графа?

1. Матрица, столбцы и строки которой соответствуют вершинам графа. На пересечении строк и столбцов записывается число, определяющее наличие связи от вершины-строки к вершине-столбцу (или наоборот).

2. Матрица, столбцы и строки которой соответствуют вершинам графа. На пересечении строк и столбцов записывается число, определяющее количество строк.

3. Матрица, столбцы и строки которой соответствуют вершинам графа. На пересечении строк и столбцов записывается число, определяющее количество столбцов.

4. Матрица, столбцы и строки которой соответствуют вершинам графа. На пересечении строк и столбцов записывается число, равное среднему арифметическому значению весовых коэффициентов.

4. Что называется матрицей инцидентности?

1. Матрица, столбцы которой соответствуют вершинам графа, а строки соответствуют связям графа. В ячейку на пересечении i-ой строки j-м столбцом матрицы записывается 1 в случае, если связь j выходит из вершины i, -1, если связь входит в вершину, 0 во всех остальных случаях (т.е. если связь является петлей или связь не инцидентна вершине).

2. Матрица, строки которой соответствуют вершинам графа, а столбцы соответствуют связям графа. В ячейку на пересечении i-ой строки j-м столбцом матрицы записывается -1 в случае, если связь j выходит из вершины i, 1, если связь входит в вершину, 0 во всех остальных случаях (т.е. если связь является петлей или связь не инцидентна вершине).

3. Матрица, строки которой соответствуют вершинам графа, а столбцы соответствуют связям графа. В ячейку на пересечении i-ой строки j-м столбцом матрицы записывается 0 в случае, если связь j выходит из вершины i, 1, если связь входит в вершину, -1 во всех остальных случаях (т.е. если связь является петлей или связь не инцидентна вершине).

4. Матрица, строки которой соответствуют вершинам графа, а столбцы соответствуют связям графа. В ячейку на пересечении i-ой строки j-м столбцом матрицы записывается 1 в случае, если связь j выходит из вершины i, -1, если связь входит в вершину, 0 во всех остальных случаях (т.е. если связь является петлей или связь не инцидентна вершине).

5. Какой тип задач решает алгоритм Дейкстры?

1. Определение кратчайшего пути между начальной и остальными вершинами графа.

2. Определение кратчайшего пути между каждой парой вершин графа.

3. Определение кратчайшего пути между промежуточными вершинами графа.

4. Определение кратчайшего пути между двумя заданными вершинами графа.

6. Как определяется метка j узла при поиске кратчайшего пути в графе методом Дейкстры?

1. $[S_j, i] = [S_i - l_{ij}, i], l_{ij} \geq 0$

2. $[S_j, i] = [S_i + 2 * l_{ij}, i], l_{ij} \geq 0$

3. $[S_j, i] = [S_i + l_{ij}, i], l_{ij} \geq 0$

4. $[S_j, i] = [S_i - 2 * l_{ij}, i], l_{ij} \geq 0$

7. Какое условие прекращения процесса вычисления кратчайшего пути методом Дейкстры?

1. Все узлы имеют постоянные метки.
2. Все узлы имеют временные метки.
3. Все узлы имеют расчетные метки.
4. Все узлы имеют правильные метки.

8.Можно ли определить кратчайшее расстояние между любыми вершинами графа алгоритмом **Флойда?**

1. Да.
2. Нет

9. Что выполняет треугольный оператор?

1. $d_{ik} + d_{kj} > d_{ij}$, то $(d_{ik} + d_{kj})$ заменяет d_{ij}.
2. $d_{ik} + d_{kj} < d_{ij}$, то $(d_{ik} + d_{kj})$ заменяет d_{ij}.
3. $d_{ik} + d_{kj} < d_{ij}$, то d_{ij} заменяет $(d_{ik} + d_{kj})$.
4. $d_{ik} + d_{kj} > d_{ij}$, то d_{ij} заменяет $(d_{ik} + d_{kj})$.

10. Какие методы применяются для нумерации вершин графа?

1. Метод Фулкерсона.
2. Метод Форда.
3. Метод Дейкстры.
4. Метод Флойда.

11. Какие элементы таблицы расстояний проверяются треугольным оператором?

1. d_{ij}, $i \neq j$
2. d_{ik}
3. d_{kj}
4. d_{ij}, $i = j$

12. Какое условие окончания процесса определения кратчайшео пути на графе методом Флойда?

1. k>n
2. k=n
3. k<n
4. k≠n

13. Чему равно расстояние между вершинами i и j в матрице D_{n-1} после реализации n шагов алгоритма Флойда?

1. d_{ik}
2. d_{kj}
3. d_{jk}

4. d_{ij}

14. Какова последовательность вершин в матрице S_{n-1} между вершинами i и j если $S_{ij} = j$?

1. $i \to k \to j$
2. $i \to j$
3. $j \to i$
4. $j \to k \to i$

15. Какова сущность поиска в глубину?

1. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$, то поиск закончен);

2. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, u-v, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если v=u, то поиск закончен);

3. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$, то поиск закончен).

4. Поиск начинаем с некоторой фиксированной вершины v_0. Затем выбираем произвольную вершину u, смежную с v_0, и повторяем просмотр от u. Предположим, что находимся в некоторой вершине v. Если существует ещё не просмотренная вершина u, то рассматриваем её, затем продолжаем поиск с нее. Если не просмотренной вершины, смежной с v, не существует, то возвращаемся в вершину, из которой попали в v, и продолжаем поиск (если $v=v_0$ и v=u то поиск закончен).

16. Что является стандартным способом устранения рекурсии при поиске в глубину?

1. Использование массива.
2. Использование очереди.
3. Использование стека.
4. Использование циклического списка.

17. Что используется при поиске в ширину?

1. Массив.
2. Очередь
3. Стек.
4. Циклический список.

18. Что называется минимальным остовным деревом?

1. Подграф, не содержащий циклов, включающий все вершины исходного графа, для которого сумма весов ребер минимальна.
2. Подграф, не содержащий циклов, включающий все вершины исходного графа, для которого сумма весов ребер максимальна.
3. Подграф, содержащий циклы, включающий все вершины исходного графа, для которого сумма весов ребер минимальна.
4. Подграф, содержащий циклов, включающий все вершины исходного графа, для которого сумма весов ребер максимальна.

19. Какой алгоритм Прима?

1. а) Выбирается произвольно вершина, остальные (n-1) вершины графа отмечаются как невыбранные;

б) Определяются веса между выбранной вершиной и остальными невыбранными вершинами;

в) Выбираем вершину с наибольшим весом, фиксируем выбранное ребро и вес;

г) Выбранную вершину исключаем из перечня невыбранных, число выбранных вершин уменьшаем на 1;

д) Этапы а) – б) повторяем до тех пор, пока не будут выбраны все вершины, т. е. (n-1) раз.

2. а) Выбирается произвольно вершина, остальные (n-1) вершины графа отмечаются как невыбранные;

б) Определяются веса между выбранной вершиной и остальными невыбранными вершинами;

в) Выбираем вершину с наименьшим весом, фиксируем выбранное ребро и вес;

г) Выбранную вершину исключаем из перечня невыбранных, число выбранных вершин увеличиваем на 1;

д) Этапы а) – б) повторяем до тех пор, пока не будут выбраны все вершины, т. е. (n-1) раз.

3. а) Выбирается произвольно вершина, остальные (n-1) вершины графа отмечаются как невыбранные;

б) Определяются веса между выбранной вершиной и остальными невыбранными вершинами;

в) Выбираем вершину с наименьшим весом, фиксируем выбранное ребро и вес;

г) Выбранную вершину исключаем из перечня невыбранных, число выбранных вершин уменьшаем на 1;

д) Этапы а) – б) повторяем до тех пор, пока не будут выбраны все вершины, т. е. (n-1) раз.

4. а) Выбирается произвольно вершина, остальные (n+1) вершины графа отмечаются как невыбранные;

б) Определяются веса между выбранной вершиной и остальными невыбранными вершинами;

в) Выбираем вершину с наименьшим весом, фиксируем выбранное ребро и вес;

г) Выбранную вершину исключаем из перечня невыбранных, число выбранных вершин уменьшаем на 1;

д) Этапы а) – б) повторяем до тех пор, пока не будут выбраны все вершины, т. е. (n-1) раз.

20. Какой алгоритм Крускала?

1. а) Определяем граф, имеющий n вершин без ребер.

б) Множество рёбер сортируется в порядке возрастания весового коэффициента.

в) Выбираем ребро с наибольшим весом и включаем в граф.

г) Выбираем следующее ребро с наибольшим весовым коэффициентом. Ребро добавляется в граф, если оно связывает две вершины из разных компонент.

д) Построение остовного дерева минимального веса заканчивается, если все вершины графа принадлежат одной компоненте.

2. а) Определяем граф, имеющий n вершин без ребер.

б) Множество рёбер сортируется в порядке возрастания весового коэффициента.

в) Выбираем ребро с наименьшим весом и включаем в граф.

г) Выбираем следующее ребро с наименьшим весовым коэффициентом. Ребро добавляется в граф, если оно связывает две вершины из разных компонент.

д) Построение остовного дерева минимального веса заканчивается, если все вершины графа принадлежат одной компоненте.

3. а) Определяем граф, имеющий n вершин без ребер.

б) Множество рёбер сортируется в порядке возрастания весового коэффициента.

в) Выбираем ребро с наименьшим весом и включаем в граф.

г) Выбираем следующее ребро с наименьшим весовым коэффициентом. Ребро добавляется в граф, если оно связывает две вершины из одинаковых компонент.

д) Построение остовного дерева минимального веса заканчивается, если все вершины графа принадлежат одной компоненте.

4. а) Определяем граф, имеющий n вершин без ребер.

б) Множество рёбер сортируется в порядке возрастания весового коэффициента.

в) Выбираем ребро с наибольшим весом и включаем в граф.

г) Выбираем следующее ребро с наибольшим весовым коэффициентом. Ребро добавляется в граф, если оно связывает две вершины из одинаковых компонент.

д) Построение остовного дерева минимального веса заканчивается, если все вершины графа принадлежат одной компоненте.

6.4.3 Таблица ответов

№ вопроса	Правильные ответы			
	Вариант 1_1	Вариант 1_2	Вариант 2_1	Вариант 2_2
1	1	1	3	2
2	1, 4	4	1	3
3	2	1	1	1
4	4	1	2	4
5	1, 4	2	3	1
6	1, 2, 3	3	4	3
7	3	4	1	1
8	1, 4	4	2, 3	1
9	2	3	1	2
10	1	2	1	1, 2
11	4	4	3	1
12	3	1	2	2
13	1	4	1	4
14	1	3	1	2
15	2	1	4	1
16	1	2	3	3
17	2	1	1	2
18	3	4	3	1
19	2	3	2	3
20	1, 2	1	4	2

6.4.4 Примерный перечень вопросов для зачёта

1. Алгоритм. Свойства, виды и структуры алгоритмов. Этапы решения задач на ЭВМ.

2. Данные. Типы данных. Структуры хранения данных: вектор, список и сеть.

3. Массивы. Структуры данных и структуры хранения массивов.

4. Сложность алгоритма. Функции сложности алгоритмов. Оценка эффективности алгоритмов.

5. Линейный список. Операции над линейным списком. Применение линейного списка.

6. Стек. Структура хранения стека. Операции над стеками. Применение стека.

7. Очереди. Структура хранения очереди. Операции над очередями. Применение очереди.

8. Нелинейные структуры данных. Представление m-арного дерева бинарным деревом. Идеально сбалансированное бинарное дерево.

9. Бинарные деревья поиска. Сбалансированные деревья поиска.

10. Операции над деревьями. Алгоритмы обхода дерева.

11. B- деревья. B+ – деревья. Операции над B – деревьями.

12. Простые методы сортировки. Методы прямого включения, прямого обмена, прямого выбора.

13. Сложные методы сортировки. Метод Шелла. Сортировка с помощью дерева (пирамиды). Быстрая сортировка Хоара.

14. Внешняя сортировка. Методы прямого и естественного слияния.

15. Методы поиска, основанные на сравнении ключей: последовательный, бинарный, интерполяционный.

16. Методы поиска, основанные на цифровых свойствах ключей: хеширование.

17. Основные определения теории графов. Представление графов.

18. Алгоритмы поиска в глубину и ширину на графах.

19. Алгоритм построения остовного дерева методом Крускала.

20. Алгоритм построеия остовного дерева методом Прима.

21. Алгоритмы нахождения кратчайшего пути на графе. Алгоритм Дейкстры.

22. Алгоритмы нахождения кратчайшего пути на графе. Алгоритм Флойда.

6.4.5 Примерный перечень тем курсовой работы

6.4.5.1 Назначение, цели и задачи курсовой работы

Проектируемое визуальное приложение курсовой работы формализует разработанный студентом алгоритм задачи, проанализированный посредством функции оценки сложности.

Основная учебная цель выполнения курсовой работы – повышение теоретических знаний и практических навыков в проектировании, анализе и программной реализации сложных структур данных и алгоритмов их обработки.

Основные задачи, решаемые студентом в процессе выполнения курсовой работы:

- программная реализация алгоритма;
- анализ алгоритма решения задачи;
- проведение экспериментального исследования алгоритма и анализ его результатов;
- документирование курсовой работы в соответствии с установленными требованиями.

6.4.5.2 Требования к курсовой работе

6.4.5.2.1 Требования к функциональным характеристикам

Проектируемая система должна обеспечивать выполнение следующих основных функций:

- ввод исходных данных задачи;
- расчет параметров;
- оценка сложности реализации алгоритма по временным и объемным параметрам;
- хранение исходных данных и результатов расчетов с возможностью их загрузки для повторной обработки;
- вывод данных.

6.4.5.2.2 Требования к эксплуатационным характеристикам

- модульность;
- расширяемость.

6.4.5.2.3 Требования к программному обеспечению

- среда разработки MS Visual C++ версии не ниже 6.0.

6.4.5.2.4 Требования к содержанию курсовой работы

К защите курсовой работы должны быть представлены визуальное приложение и альбом, включающий проектные, программные и эксплуатационные документы:

- Опись альбома.
- Пояснительная записка (состав основных разделов документа):
- Аналитический обзор.
- Описание алгоритмов решения задачи.
- Оценка сложности алгоритма.
- Описание структуры программного комплекса.
- Описание структур данных.
- Описание методики проведения экспериментального исследования.
- Описание и анализ результатов проведенного исследования.
- Выводы по результатам проведенного анализа.

Спецификация.

Описание программы.

Текст программы (на машинном носителе).

Руководство пользователя.

Руководство программиста.

Требования к структуре документов определены соответствующими стандартами ЕСПД.

Пояснительная записка оформляется в соответствии с требованиями методического указания к оформлению документации курсовых и дипломных проектов.

6.4.5.3 Варианты заданий курсовой работы

1. Транспортная задача. Метод потенциалов.
2. Эвристические методы поиска решения на графах. Метод минимальной стоимости.
3. Динамическое программирование. Метод обратной прогонки.
4. Сортировка файлов простым слиянием.
5. Сортировка файлов естественным слиянием.
6. Транспортная задача. Распределительный метод.
7. Остовное дерево наименьшей стоимости. Алгоритм Прима.
8. Остовное дерево наименьшей стоимости. Алгоритм Борувки.
9. Алгоритм определения максимального потока сети.
10. Алгоритм минимизации стоимости потока в сети с ограниченной пропускной способностью.
11. Алгоритм определения критического пути.
12. Методы поиска решения на графах. Методы поиска в ширину.
13. Методы поиска решения на графах. Методы поиска в глубину.
14. Эвристические методы поиска решения на графах. Метод экстремума.
15. Алгоритм поиска оптимального кода. Алгоритм Хаффмана.
16. Динамическое программирование. Метод прямой прогонки.
17. Алгоритм поиска оптимального кода. Алгоритм Шеннона-Фено.
18. Алгоритм сжатия данных. Алгоритм «арифметическое кодирование».
19. Структуры данных и алгоритмы для внешней памяти. Внешние деревья поиска. B дерево (B tree).
20. Алгоритм метода ветвей и границ. Задача расшифровки криптограмм.
21. Остовное дерево наименьшей стоимости. Алгоритм Крускала.
22. Алгоритмы внешней сортировки. Многофазная сортировка.
23. Алгоритмы внешней сортировки. Каскадная сортировка.
24. Алгоритм внешней сортировки. Сбалансированное многопутевое слияние.
25. Алгоритмы на графах. Задача поиска кратчайшего отрицательно взвешенного пути при одном источнике (алгоритм Беллмана-Форда).
26. Алгоритм Йена.
27. Алгоритм Форда.
28. Оптимальное двоичное дерево поиска. Алгоритм Гарсия-Воча.
29. Поиск в деревьях. Красно-черные деревья (RB tree).
30. Структуры данных и алгоритмы для внешней памяти. Внешние деревья поиска. B+ дерево (B+ tree).
31. Алгоритм Джонсона нахождения всех пар кратчайших путей.
32. Алгоритм с возвратом. Задача Гамильтона.
33. Алгоритм метода ветвей и границ. Задача коммивояжера.

34. Алгоритм Литтла. Задача коммивояжера.

6.5. Фонд оценочных средств

Полный банк заданий для текущего, рубежных контролей и промежуточной аттестации по дисциплине, показатели, критерии, шкалы оценивания компетенций, методические материалы, определяющие процедуры оценивания образовательных результатов, приведены в учебно-методическом комплексе дисциплины.

7. ОСНОВНАЯ И ДОПОЛНИТЕЛЬНАЯ УЧЕБНАЯ ЛИТЕРАТУРА

7.1. Основная учебная литература

1. Мясникова Н. А. Алгоритмы и структуры данных. Учебное пособие. – Москва: Кнорус, 2020. – 186 с. – (сер. Бакалавриат).

2. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ. – Москва: Вильямс, 2018. – 1328 с.

3. Сэджвик Р. Алгоритмы на C++. – Москва: Изд-во «Вильямс», 2016. – 1056 с.

7.2. Дополнительная учебная литература

1. Анашкина Н.В. Технологии и методы программирования. – М.: Издательский центр «Академия», 2012. – 384 с.

2. Давыдов В.Г.Технологии программирования. C++. – Санкт-Петербург: БХВ – Петербург, 2005. – 672 с.

3. Уайс, М. А. Организация структур данных и решение задач на C++ [Текст] / Уайс М. : А. пер. с англ. – М. : ЭКОМ Паблишерз, 2008. – 896 с.

4. Хусаинов Б.С. Структуры и алгоритмы обработки данных. Примеры на языке Си: Учебное пособие. – Финансы и статистика, 2004. – 464 с

5. Аксёнова Е.А., Соколов А.В. Алгоритмы и структуры данных на C++: Учебное пособие. - Петрозаводск: Изд-во ПетрГУ, 2008.- 81 с. URL: http://window.edu.ru/resource/576/63576 (дата обращения: 25.05.2020).

8. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

1. Семахин А.М. Алгоритмы и структуры данных. Методические указания к выполнению лабораторных и курсовых работ для студентов направления 231000.62«Программная инженерия». Курган, КГУ, 2012. – 64 с.

2. Алгоритмы и структуры данных. Методические указания к выполнению курсовых работ для студентов направления 231000.62«Программная инженерия». Курган, КГУ, 2014. – 74 с.

9. РЕСУРСЫ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫЕ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1. Федеральный портал «Российское образование» URL: http://www.edu.ru/
2. Сайт дистанционного обучения в НОУ «ИНТУИТ». URL: http://www.intuit.ru/

10. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ И ИНФОРМАЦИОННЫЕ СПРАВОЧНЫЕ СИСТЕМЫ

При чтении лекций используются слайдовые презентации.

Минимальные требования к операционной системе и программному обеспечению компьютера, используемого при показе слайдовых презентаций: Windows, Foxit Reader.

11. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Материально-техническое обеспечение включает в себя учебные лаборатории и классы, оснащенные современными компьютерами (рабочими станциями локальной вычислительной сети) с доступом в Интернет, мультимедийное оборудование (переносной персональный компьютер, мультимедийный проектор, мультимедийный экран).

Программные средства обеспечения учебного процесса включают лицензионное программное обеспечение: операционную систему Windows XP, интегрированную среду программирования Microsoft Visual C++ 2010 Professional.

ПРИЛОЖЕНИЕ 1

Аннотация к рабочей программе дисциплины

«Алгоритмы и структуры данных»

образовательной программы высшего образования –
программы бакалавриата

09.03.03 – Прикладная информатика

Направленность:

Интеллектуальные информационные системы

Трудоемкость дисциплины: 4 ЗЕ (144 академических часа)
Семестр: 3 (очная)
Форма промежуточной аттестации: зачёт

Содержание дисциплины

Алгоритмы и данные. Свойства алгоритма. Анализ сложности алгоритма. Семантика, синтаксис, прагматика. Структура данных. Структуры хранения данных: вектор, список, сеть. Массивы. Структуры данных массивов. Структуры хранения массивов. Строки. Операции над строками. Записи. Операции над записями. Множества.

Списки. Структура и классификация списков. Операции над линейными списками. Применение списков. Стеки. Структура стека. Операции над стеками. Применение стеков. Очереди. Деки. Операции над очередями и деками. Применение очередей и деков.

Алгоритм преобразования m-арного дерева в бинарное дерево. Представление деревьев в памяти ЭВМ. Идеально-сбалансированное бинарное дерево. Бинарные (двоичные) деревья поиска. Операции над деревьями. B-деревья. Операции над B-деревьями.

Основные понятия и классификация алгоритмов сортировки. Внутренняя сортировка.

Классификация алгоритмов поиска. Поиск в последовательно организованных структурах. Последовательный поиск. Двоичный Поиск в деревьях.

Представление графов. Кратчайшие пути. Алгоритм Дейкстры. Алгоритм Флойда. Остовные деревья графа.