Консультация к отдельным вопросам государственного экзамена бакалавриата ПМИ ВМК МГУ в 2020 году

4 курс ПМИ (основная часть)

Вопрос 16. Формализация понятия алгоритма. Машины Тьюринга, нормальные алгоритмы Маркова. Алгоритмическая неразрешимость. Задача останова. Задача самоприменимости.

См. Боресков и др. Методические материалы для подготовки к государственному экзамену по прикладной математике и информатике. – М.: Издательский отдел ВМК МГУ 2004, 294 с. http://sp.cs.msu.ru/info/gos.djvu

Корухова Л.С., Шура-Бура М.Р. Введение в алгоритмы. Учебное пособие для студентов I курса. – М.: Изд. отдел ф-та ВМК МГУ, 2010 – [http://sp.cs.msu.ru/info/1/vvedalg.pdf]

Вопрос 21. Базы данных.Основные понятия реляционной модели данных. Реляционная алгебра. Средства языка запросов SQL.

План ответа по вопросу

Перечислите и определите основные понятия реляционных баз данных: тип данных, домен, атрибут, кортеж, отношение, первичный ключ. Укажите фундаментальные свойства отношений: отсутствие кортежей-дубликатов, первичный и возможные ключи, отсутствие упорядоченности кортежей и атрибутов, атомарность значений атрибутов, первая нормальная форма отношения.

Алгебра Кодда. Перечислите и определите теоретико-множественные операции: объединение, пересечение, взятие разности, взятие декартова произведения. То же сделайте для специальных операций: ограничение, проекция, соединение, деление. Дайте пояснения по совместимости операндов.

Структура языка SQL, типы данных SQL. Перечислите и поясните на примерах средства определения, изменения определения и отмены определения доменов, базовых таблиц, ограничений целостности. Дайте общую характеристику оператора SELECT, опишите организация списка ссылок на таблицы в разделе FROM, предикаты раздела WHERE, группировка и условия раздела HAVING, порождаемые и соединенные таблицы. Желательно сопроводить всё это примерами. Перечислите и поясните на примерах средства манипулирования данными (INSERT, UPDATE, DELETE)

См.:
http://citforum.ru/database/advanced_intro/10.shtml#3
http://citforum.ru/database/advanced_intro/13.shtml#4
http://citforum.ru/database/advanced_intro/46.shtml#15
http://citforum.ru/database/advanced_intro/55.shtml#17
http://citforum.ru/database/advanced_intro/58.shtml#18
http://citforum.ru/database/advanced_intro/72.shtml#21

4 курс ПМИ (дополнительная часть, III поток)

Вопрос 5. Сортировка. Простейшие алгоритмы – сортировка выбором, вставками, обменом. Оценка сложности алгоритмов сортировки. Быстрая сортировка и ее сложность в среднем и в наихудшем случаях.

При подготовке по этому вопросу можно воспользоваться лекциями проф. С. А. Абрамова на ютьюбе: https://www.youtube.com/playlist?list=PLcsjsqLLSfNArTXIp-a3vPK3zeb9DcmTc

Литература:

[Абрамов, 2012] Абрамов С. А. Лекции о сложности алгоритмов. – М.: МЦНМО, 2012

[Шень, 2017] Шень А. Программирование: теоремы и задачи. – М.: МЦНМО, 2017. Глава 4. Сортировка. [https://www.mccme.ru/free-books/shen/shen-progbook.pdf]

План ответа по вопросу

Определение сортировки: Пусть есть массив a[1], ... , a[n], тип элементов которого таков, что в нём для любых значений v и w определена операция сравнения v $\leq$ w. результат операции истинен, если v в каком-то смысле «меньше либо равно» w, и ложь – иначе. Операция сравнения $\leq$ должна быть транзитивна. Тип элементов массива может быть нескалярным. Обычно сортировку рассматривают на числовых массивах. Так поступим и мы, выписывая код или псевдокод алгоритмов сортировки. *Сортировкой* массива a[1], ... , a[n] называется операция, результатом которой является массив b[1], ... , b[n] , содержащий те же элементы, что и исходный массив, взятые в тех же количествах, что и в исходном массиве, но расположенные так, что для любых $1 \leq i \leq j \leq n$ справедливо b[i] $\leq$ b[j]. Заметим, что операция сравнения $\leq$ может иметь различный смысл. Массив может быть отсортирован по неубыванию, по невозрастанию. Заметим также, что часто массив b[1], ... , b[n] записывается на место исходного массива a[1], ... , a[n].

Разные реализации операции сортировки (разные методы сортировки или разные алгоритмы) работают быстрее или медленнее. Параметром для оценки их «скорости работы» является длина массива n. Характеристики реализаций, на которые обращают внимание – количество вычисления результатов операции сравнения $\leq$ (количество сравнений) и количество присваиваний элементов массива. Если элементы массива не скалярные, то сравнение не сводится к вычитанию двух чисел, а присваивание к записи числа. Обе операции могут быть довольно затратными. Иногда обращают внимание на то, требуется ли для сортировки дополнительная память, объем которой зависит от n.

Идея *сортировки выбором* заключается в том, чтобы осуществить n шагов, таких, что на i-ом шаге i-й элемент массива ставится на своё итоговое место. На первом шаге на 1-ое место ставится минимальный элемент всего массива. При этом прежний 1й элемент меняется с минимумом местами. На втором шаге мы повторяем то же самое, но с массивом a[2], ... , a[n]. И т. д. Псевдокод алгоритма дан в [Шень, 2017]. Алгоритм делает n-1 обмен (количество присваиваний линейно) и квадратичное количество сравнений (n-1 сравнение на 1м шаге, n-2 на втором, …, что даёт в итоге n*(n-1)/2). Оба количества не зависят от конкретных элементов массива (сложность в среднем и в худшем совпадает).

Идея *сортировки вставками* заключается в том, что мы на i-м шаге уже имеем отсортированные первые i элементов и нужно среди них *вставить* i+1й элемент так, чтобы получились отсортированные первые i+1й элемент. Вставки можно делать разными способами. При *простых* вставках, начиная либо с начала, либо с конца отсортированной части последовательно сравнивается i+1й с текущим элементом отсортированной части до тех пор, пока не будет обнаружено его место. При *бинарных* вставках место ищется способом «деления пополам». Когда место вставки найдено, «хвост» отсортированной части сдвигается на одну позицию, затем i+1й элемент ставится на освободившееся место. Сложность в среднем и в худшем в этом алгоритме разные. Оценка количества сравнений для алгоритма сортировки простыми вставками выведена в книге [Абрамов, 2012]. Она равна $n^2/4+O(n)$. Оценка количества присваиваний та же. В лучшем случае, когда массив уже отсортирован, при простых вставках требуется n-1 сравнение и 0 присваиваний. В худшем случае (при обратном порядке элементов по сравнению с требуемым) требуется n*(n-1)/2 сравнений и столько же присваиваний. Псевдокод простых вставок дан в [Шень, 2017].

Сортировка *простыми обменами* или сортировка *«пузырьком»* состоит из n-1 прохода по массиву. При проходе каждая пара подряд стоящих элементов a[i] и a[i+1] сравнивается и, если они не по порядку, обменивается местами. После 1го прохода максимальный элемент (при сортировке по неубыванию) оказывается на n-ом месте в массиве, а минимальный элемент сдвигается влево на 1 позицию. Следующий проход затрагивает n-1 элемент, т. к. на n-ом месте уже стоит нужный элемент. При проходе подсчитывается количество обменов, сделанных в течение прохода. Если обменов не было, то массив уже отсортирован и оставшиеся проходы не нужны. В худшем, при обратно отсортированном массиве количество сравнений и обменов $O(n^2)$

т. е. n*(n-1)/2. В лучшем, если массив уже отсортирован, n-1 сравнений, 0 обменов. оценка количества сравнений в среднем также $O(n^2)$ (точной формулы для количества нет).
Быстрая сортировка (или сортировка Хоара) состоит в том, что сначала мы выбираем какой-то *опорный* элемент массива и приводим массив к виду, такому, что выбранный элемент оказывается на своём месте, т. е. в начале массива находятся все элементы меньше *опорного*, затем идут элементы равные *опорному*, дальше – все, большие *опорного*. Затем то же преобразование применяется к началу (где находятся элементы меньше опорного) и к окончанию массива (где элементы большие опорного). Заметим, что тут нужны операции сравнения на «меньше», на «больше» и проверка на равенство. Для упрощения реализации массив разделяют не на три а на две части (небольше опорного и больше него). В книге [Абрамов, 2012] выведена оценка сложности в среднем O (n * log(n)) для быстрой сортировки. Худшим является случай, когда всякий раз опорным элементом выбирается максимум или минимум из рассматриваемой части массива. Сложность в этом случае составляет $O(n^2)$.

Известно, что составители задач для студенческих соревнований по программированию с автоматической проверкой решений специально добавляли тесты, на которых стандартные библиотечные реализации быстрой сортировки работают дольше всего.

Вопрос 8. Зависимости в реляционных отношениях: функциональные, многозначные, проекции/соединения. Проектирование реляционных БД на основе принципов нормализации отношений. Нормальные формы.
План ответа по вопросу
Дайте определение функциональной зависимости. Опишите декомпозицию без потерь и функциональные зависимости. Сформулируйте теорему Хита. Расскажите о диаграммах функциональных зависимостей. Сопроводите свой ответ примерами.

Расскажите о минимальной функциональной зависимости и второй нормальной форме.
Расскажите о нетранзитивной функциональной зависимости и третьей нормальной форме.
Расскажите о перекрывающихся возможных ключах и нормальной форме Бойса-Кодда.
Сопроводите свой ответ примерами.

Дайте определение многозначной зависимости. Сформулируйте теорему Фейджина.
Расскажите о многозначных зависимостях и четвертой нормальной форме. Сопроводите свой ответ примерами.

Дайте определение зависимости проекции/соединения. Расскажите о зависимости проекции/соединения и пятой нормальной форме. Сопроводите свой ответ примерами.

Приведите примеры аномалий обновления.
См.:
http://citforum.ru/database/advanced_intro/19.shtml#7
http://citforum.ru/database/advanced_intro/22.shtml#8
http://citforum.ru/database/advanced_intro/24.shtml#9

Вопрос 12. Классификация языков, определяемых конечными автоматами, регулярными выражениями и праволинейными грамматиками. Эквивалентность и минимизация конечных автоматов.
Материал для подготовки в книге «Теория и реализация языков программирования». Авторы: Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. http://sp.cs.msu.ru/info/trlp.pdf

Вопрос 13. Функции FIRST и FOLLOW. LL(l)-грамматики. Конструирование таблицы предсказывающего анализатора .
Материал для подготовки в книге Серебряков В. А., Галочкин М. П., Гончар Д. Р., Фуругян М. Г. «Теория и реализация языков программирования». http://sp.cs.msu.ru/info/trlp.pdf

Вопрос 14. Жизненный цикл программного обеспечения (ПО). Основные виды деятельности при разработке ПО. Каскадная и итерационная модели жизненного цикла.

Жизненный цикл ПО — интервал времени от появления идеи о создании этого ПО до вывода из эксплуатации последнего экземпляра.

Цикл — в связи с итеративным повторением некоторых видов деятельности: сбор и анализ требований (исходных или для изменений) – проектирование (исходное или вносимых изменений) – реализация – верификация/тестирование – ввод в эксплуатацию/развертывание.

Более 75% трудоемкости в ЖЦ сложного практически значимого ПО падает на сопровождение — исправление дефектов и внесение изменений.

Основные виды деятельности (в деталях могут быть различия в различных стандартах и описаниях процессов разработки)

- Сбор и анализ требований (анализ предметной области, выделение требований, систематизация требований, анализ области решений)
- Проектирование (проектирование архитектуры, детальное проектирование компонентов)
- Реализация (кодирование, построение ПО, отладка и исправление дефектов)
- Контроль качества (валидация и верификация, экспертизы, тестирование)
- Документирование (разработка проектной, пользовательской, эксплуатационной документации)
- Развертывание (установка, ввод в эксплуатацию, приемка)
- Управленческие виды деятельности (планирование, мониторинг и контроль работ, управление конфигурациями, управление персоналом, управление технологиями, управление рисками, информационное обеспечение)
- Поддержка разработки (создание и сопровождение репозиториев, переиспользуемых компонентов, средств разработки)

Один из основных современных стандартов на процесс разработки — ISO 12207.

Модель ЖЦ — общая схема организации и взаимосвязи работ.

Каскадная модель — виды деятельности выстроены в последовательность, от сбора требований до развертывания, каждый вид деятельности выполняется на некотором этапе проекта, пока не будет сделано все, что нужно, в требуемом качестве, возвращаться к предыдущим этапам нельзя.

Итеративная модель — этапы проекта формируют итерации, на каждой итерации происходит выполнение некоторой последовательности работ до достижения цели данной итерации, обычно в виде разработки значимой части ПО или создания значимой части проектных решений для дальнейшей реализации, по окончании очередной итерации можно вернуться к пересмотру или доработке ее решений в следующей.

Практически все сложные проекты — итеративные. Каскадная схема работоспособна для небольших систем, создаваемых за короткий период времени, во время которого не возникают значимые изменения в требованиях к создаваемым системам.

Разновидность итеративной модели — спиральная — предполагает одинаковую структуру итераций, на каждой итерации работы выстроены по практически одному и тому же сценарию.

В итеративной модели ЖЦ количество итераций устанавливается заранее при планировании ЖЦ. Этим она отличается от эволюционной модели ЖЦ, в которой вместо итераций -- периоды, завершающиеся выпусками новых версий ПО, количество которых не определено заранее. Эволюционная не относится к итеративным.

Материалы для подготовки: конспект 2-й лекции по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture02.pdf; книга Иан Соммервил. Инженерия программного обеспечения. 6-е издание. М. – СПб. – Киев: 2002. – 623 с.

Вопрос 15. Качество программного обеспечения и методы его контроля. Тестирование и другие методы верификации.

Качество ПО — его пригодность для решения тех задач, для которых оно предназначено и/или применяется на практике. Круг задач обычно четко не описан, пользователи часто придумывают новые способы как-то применить ПО. Поэтому для описания качества используют схему из

набора характеристик, комбинации которых можно использовать для оценки пригодности для очередной задачи.
Современная схема — стандарт ISO 25010, выделяются следующие характеристики.
- Функциональность — какие конкретно задачи и с какими характеристиками решений ПО умеет решать.
- Надежность — насколько устойчиво ПО работает в разных условиях, насколько быстро и корректно восстанавливает работу и данные при сбоях и перебоях в работе, вызванных внешними факторами.
- Защищенность — насколько ПО предотвращает доступ и вмешательство в свою работу и данные со стороны лиц и программ, которые такого доступа должны быть лишены.
- Совместимость — насколько ПО работоспособно при использовании данных других версий или от посторонних систем, и одновременной работе других программ в той же среде.
- Переносимость — насколько ПО работоспособно без пересборки (но, возможно, с изменением каких-то конфигурационных файлов) в разных средах.
- Производительность — сколько ПО использует ресурсов разного рода (процессор, разные виды памяти, сети и пр.) при решении определенных задач.
- Удобство использования — насколько быстро и эффективно пользователи обучаются работать с ПО и могут решать определенные задачи с его помощью.
- Удобство сопровождения — насколько быстро и эффективно разработки могут анализировать ПО, находить и исправлять в нем дефекты, вносить изменения.

Обеспечение качества — предотвращение, поиск и исправление ошибок. Поиск ошибок, он же контроль качества, разделяется на валидацию и верификацию.
Валидация — поиск ошибок, опирающийся на понимание разработчиков, пользователей и экспертов того, как ПО должно работать и как должны выглядеть проектные решения.
Верификация — поиск ошибок, опирающийся на выявление несоответствий между двумя или более артефактами разработки (документированными требованиями, проектными решениями, моделями работы, прототипами, кодом, тестами и пр.) и/или реальной работой ПО.
Методы верификации.
- Экспертиза — привлечение людей (разработчиков и экспертов) для выявления несоответствий.

Инспекция кода или проектных документов, экспертиза защищенности или удобства использования, экспертные оценки надежности и пр.
- Статические методы — автоматизированный поиск ошибок по некоторым шаблонам и правилам без выполнения ПО.

Применение разнообразных инструментов, наиболее эффективные техники анализа включаются в семантику языков и компиляторы.
- Динамические методы — поиск ошибок в результатах реальной работы ПО (или в рамках некоторой симуляции).

Мониторинг, профилирование, тестирование.
- Формальные методы — формализация проверяемых артефактов в виде математических моделей (спецификации требований и реализации) и формальная проверка определенного математического соответствия между ними.

Дедуктивная верификация — модели логико-алгебраические, соответствие означает выводимость спецификации из реализации. Проверка моделей — обычно спецификация логико-алгебраическая, реализация исполнимая или автоматная, соответствие означает выполнимость спецификации на реализации.
Тестирование — проверка соответствия реальной работы системы требованиям на заранее выбранном наборе ситуаций. Наиболее широко используемый метод контроля качества. Основная проблема — выбор небольшого набора ситуаций, достаточного, чтобы более-менее адекватно судить о поведении системы в целом. Выбранные ситуации преобразуются в тесты — каждый тест содержит способ формирования нужной ситуации и проверку корректности работы системы в ней. Выделяются виды тестирования

- По проверяемым свойствам (тестирование функциональности, надежности, производительности, и т.д., на соответствие, сертификационное, регрессионное)
- По источнику данных для тестов (тестирование черного ящика, структурное тестирование, тестирование на отказ, включая тестирование работоспособности, нагрузочное и стрессовое)
- По уровню проверяемой части системы (модульное, компонентное, интеграционное, тестирование характеристик, системное)
- По исполнителю (тестирование при разработке, альфа, бета, независимое, приемочное).

Материалы для подготовки: лекция 5 по курсу «Основы программной инженерии» http://se-course.narod.ru/Lecture05.pdf исключая описания стандартов ISO 9000, 9001, 9003, 9004, 90003 и раздел об ошибках. Обзор методов верификации http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf, разделы 1, начало раздела 3 (до 3.1), разделы 3.2, 3.3.5, 3.3.6, начало 3.4 (до 3.4.1) и 3.4.2-3.4.5.

Вопрос 22. Синхронизация в распределенных системах. Синхронизация времени. Логические часы. Выборы координатора. Взаимное исключение. Координация процессов.

При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 3-4-MPI-Sync.doc Раздел 4

Вопрос 23. Отказоустойчивость в распределенных системах. Типы отказов. Фиксация контрольных точек и восстановление после отказа. Репликация и протоколы голосования. Надежная групповая рассылка.

При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 7-Fault_T.doc Раздел 7

Вопрос 24. Распределенные файловые системы. Доступ к директориям и файлам. Семантика одновременного доступа к одному файлу нескольких процессов. Кэширование и размножение файлов.

При подготовке ответа следует воспользоваться текстами лекций: В.А. Крюков, В.А. Бахтин. «Распределенные системы» [http://sp.cs.msu.su/courses/os/distr-systems-2018.zip]. Файл 5-DistrFS.doc Раздел 5

Вопрос 25. Промежуточные представления программы: абстрактное синтаксическое дерево; последовательность трехадресных инструкций. Базовые блоки и граф потока управления.

[...]

Вопрос 26. Локальная оптимизация при компиляции программы. Ориентированный ациклический граф и метод нумерации значений.

[...]

Вопрос 27. Глобальная оптимизация при компиляции программы. Построение передаточных функций базовых блоков. Монотонные и дистрибутивные передаточные функции. Метод неподвижной точки и его применение для нахождения достигающих определений.

[...]