Программа учебной дисциплины «Алгоритмы дизайн и анализ»

Утверждена
Академическим советом ООП
Протокол № 8.1.2.1-54/02 от «27 » июня 2016 г.

Автор	Шутов А.А., старший преподаватель
Число кредитов	3
Контактная работа (час.)	-
Самостоятельная работа (час.)	114
Курс	2
Формат изучения дисциплины	с использованием онлайн курса

I. ЦЕЛЬ, РЕЗУЛЬТАТЫ ОСВОЕНИЯ ДИСЦИПЛИНЫ И ПРЕРЕКВИЗИТЫ

Настоящая дисциплина относится к циклу профессиональных дисциплин и блоку дисциплин по выбору. Изучается на 2-м курсе.

Целями освоения дисциплины «Алгоритмы дизайн и анализ» являются изучение основ алгоритмизации и структур данных, овладение методами разработки и описания различных алгоритмов, связанных с управлением данными и применение полученных знаний для работы в избранной сфере деятельности, обладать универсальными и предметно-специализированными компетенциями, способствующими его социальной мобильности и устойчивости на рынке труда.

Дисциплина представляет собой on-line курс:
Алгоритмы: дизайн и анализ /Algorithms: Design and Analysis
Платформа: coursera
Ссылка: https://www.coursera.org/learn/algorithm-design-analysis
Даты изучения: 15.10.2018-25.04.2019
Название высшего учебного заведения: Стэнфордский университет

Изучение дисциплины «Алгоритмы дизайн и анализ» базируется на следующих дисциплинах:

- Программирование;
- Теоретические основы информатики.

Для освоения дисциплины студент должен владеть современными методами и средствами программирования.

Основные положения дисциплины должны быть использованы в дальнейшем при изучении следующих дисциплин:

1 Управление качеством программного обеспечения;
2 Информационные процессы, системы и сети.

II. СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

1. Introduction; "big-oh" notation and asymptotic analysis.

1.1. Why Study Algorithms?
1.2. Integer Multiplication
1.3. Karatsuba Multiplication

1.4. Merge Sort: Motivation and Example
1.5. Merge Sort: Pseudocode
1.6. Merge Sort: Analysis
1.7. Guiding Principles for Analysis of Algorithms
1.8. The Gist
1.9. Big-Oh Notation
1.10. Basic Examples
1.11. Big Omega and Theta

2. Divide-and-conquer basics; the master method for analyzing divide and conquer algorithms.

2.1. O(n log n) Algorithm for Counting Inversions I
2.2. O(n log n) Algorithm for Counting Inversions II
2.3. Strassen's Subcubic Matrix Multiplication Algorithm
2.4. O(n log n) Algorithm for Closest Pair I [Advanced - Optional]
2.5. O(n log n) Algorithm for Closest Pair II [Advanced - Optional]
2.6. Motivation
2.7. Formal Statement

3. The QuickSort algorithm and its analysis; probability review.

3.1. Quicksort: Overview
3.2. Partitioning Around a Pivot
3.3. Correctness of Quicksort [Review - Optional]
3.4. Choosing a Good Pivot
3.5. Analysis I: A Decomposition Principle
3.6. Analysis II: The Key Insight
3.7. Analysis III: Final Calculations
3.8. Probability Review I
3.9. Probability Review II

4. Linear-time selection; graphs, cuts, and the contraction algorithm.

4.1. Randomized Selection - Algorithm
4.2. Randomized Selection - Analysis
4.3. Deterministic Selection - Algorithm [Advanced - Optional]
4.4. Deterministic Selection - Analysis I [Advanced - Optional]
4.5. Deterministic Selection - Analysis II [Advanced - Optional]
4.6. Omega(n log n) Lower Bound for Comparison-Based Sorting [Advanced - Optional]
4.7. Graphs and Minimum Cuts
4.8. Graph Representations
4.9. Random Contraction Algorithm
4.10. Analysis of Contraction Algorithm
4.11. Counting Minimum Cuts

5. Breadth-first and depth-first search; computing strong components; applications.

5.1. Graph Search - Overview
5.2. Breadth-First Search (BFS): The Basics
5.3. BFS and Shortest Paths
5.4. BFS and Undirected Connectivity
5.5. Depth-First Search (DFS): The Basics
5.6. Topological Sort
5.7. Computing Strong Components: The Algorithm
5.8. Computing Strong Components: The Analysis

6. Dijkstra's shortest-path algorithm.

6.1. Dijkstra's Shortest-Path Algorithm
6.2. Dijkstra's Algorithm: Examples

6.3. Correctness of Dijkstra's Algorithm
6.4. Dijkstra's Algorithm: Implementation and Running Time
7. Heaps; balanced binary search trees.
7.1. Heaps: Operations and Applications
7.2. Heaps: Implementation Details [Advanced - Optional]
7.3. Balanced Search Trees: Operations and Applications
7.4. Binary Search Tree Basics, Part I
7.5. Binary Search Tree Basics, Part II
7.6. Red-Black Trees
7.7. Rotations [Advanced - Optional]
7.8. Insertion in a Red-Black Tree [Advanced]
8. Hashing; bloom filters.
8.1. Hash Tables: Operations and Applications
8.2. Hash Tables: Implementation Details, Part I
8.3. Hash Tables: Implementation Details, Part II
8.4. Pathological Data Sets and Universal Hashing Motivation
8.5. Universal Hashing: Definition and Example [Advanced - Optional]
8.6. Universal Hashing: Analysis of Chaining [Advanced - Optional]
8.7. Hash Table Performance with Open Addressing [Advanced - Optional]
8.8. Bloom Filters: The Basics
8.9. Bloom Filters: Heuristic Analysis

III. ОЦЕНИВАНИЕ

Итоговый контроль предусматривает экзамен в четвертом модуле. На экзамене студент должен продемонстрировать владение терминологией, умение решать задачи, знание структур данных, умение работать с различными алгоритмами сортировки и сжатия данных.

IV. ПРИМЕРЫ ОЦЕНОЧНЫХ СРЕДСТВ

Оценочные средства для промежуточной аттестации

1. Какими параметрами характеризуется скорость работы сортировки (написать)
2. Минимальное количество перестановок для пузырьковой сортировки
3. В чем разница между внешней и внутренней сортировками.
4. Принцип сортировки слиянием
5. Напишите на языке C++ функцию, которая не рекурсивно считает факториал.
6. Почему желательно использовать итерационные алгоритмы, а не рекурсивные при реализации
7. Определение хэш-функции
8. Как будет вести себя программа, использующая поиск с хешированием, если из-за ошибки программирования значением хеш-функции всегда будет одна и та же константа?
9. Можно ли использовать хеширование в случае хранения данных на внешнем устройстве?
10. Почему требуется, чтобы хеш-функция принимала все значения из множества индексов I? Чем плохо, если она будет принимать, например, только четные значения?
11. Как можно реализовать удаление ключа из хешируемой таблицы? Для каких способов разрешения коллизий это сделать легче?

12. Почему значения функции двойного хеширования hh(x) должны быть взаимно просты с размером хеш-таблицы?
13. Разместить в хеш-таблице из 11 элементов следующие ключи: 60, 24, 77, 126, 100, 239, 2, 93. Использовать алгоритм деления и алгоритм линейных проб.
14. Напишите процедуру поиска трех минимальных элементов массива за один проход.
15. Напишите процедуру поиска k-й порядковой статистики при помощи неполного алгоритма QuickSort.
16. Выполните поиск медианы в массиве A = (48, 72, 3, 14, 35, 65, 88, 89, 95, 6, 5, 65, 21, 24, 77), используя неполный алгоритм QuickSort. В качестве разделяющего использовать первый элемент массива.
17. Дан массив дат A. Для упрощения принято, что номер года и номер дня лежат в пределах от 01 до 10. Выполнить поразрядную сортировку массива. Показать результаты каждого прохода. A = (06.12.03; 09.03.07; 01.01.01; 01.05.01; 10.10.10; 02.05.10; 06.01.03; 07.04.05; 07.11.05; 09.12.06; 09.12.02; 05.12.03).
18. Докажите, что длины всех ветвей ИС-дерева различаются не более, чем на 1.
19. Можно ли при удалении вершины дерева поиска, имеющей двух сыновей, вместо ближайшей вершины слева использовать ближайшую справа?
20. Докажите, что число листьев для наихудших АВЛ-деревьев при увеличении высоты дерева образует последовательность Фибоначчи.
21. Сформулируйте алгоритм вставки в B+-дерево
22. Известна модификация страничных деревьев поиска, называемая B++-деревьями. Для этих деревьев каждая страница, кроме корня, заполнена не менее, чем на 3/4. Объясните, как должна работать вставка в B++-дерево и сколько ключей может содержать корень такого дерева.
23. Чем отличается операция изменения значения ключевого поля записи базы данных от изменения неключевого поля?
24. Сколько сравнений выполняется при сортировке массива из n элементов по алгоритму пузырька?
25. При каком условии алгоритм пузырька может завершить сортировку массива за n–2 прохода? за n–3 прохода? На сколько операций сравнения меньше будет выполнено в этих случаях?
26. Предположим, что в алгоритме вставок значение переменной r сравнивается со значениями элементов массива не в обратном порядке (ak, ak–1, ak–2 …, a1), а в прямом (a1, a2, a3 …, ak). Повлияет ли это на правильность и эффективность алгоритма? Какой из двух вариантов больше подходит для сортировки Шелла?
27. Объясните, почему в нерекурсивном варианте QuickSort при занесении в стек более длинных отрезков глубина стека оказывается меньше, чем при занесении более коротких.
28. Объясните, почему в рекурсивном варианте QuickSort глубина используемого стека оказывается значительно больше, чем в нерекурсивном.
29. Запрограммируйте смешанный вариант QuickSort, в котором разделение массива и сортировка меньшего из получившихся отрезков выполняется в цикле, а для сортировки большего отрезка используется рекурсивный вызов (при этом нет необходимости явно использовать переменную-стек).
30. Какая глубина стека может потребоваться для реализации QuickSort, описанной в предыдущем упражнении?
31. Алгоритм сортировки называется устойчивым, если элементы массива, имеющие одно и то же значение ключа, сохраняют после сортировки свое взаимное положение. Какие из рассмотренных в разделе алгоритмов являются устойчивыми?

32. Дан массив чисел A = (10, 50, 30, 32, 11, 40, 20, 5, 16, 37, 12, 1). Выполнить сортировку массива по алгоритму ShellSort, используя значения h = 7, 3, 1. Показать состояние массива после каждого прохода.
33. Дан массив чисел A = (20, 13, 5, 25, 16, 18, 40, 32, 21, 11, 1, 30). Выполнить сортировку массива по алгоритму QuickSort. В качестве разделяющего выбирать первый элемент отрезка. Показать состояние массива после каждой операции разделения.
34. Дан массив чисел A = (35, 8, 24, 12, 42, 15, 31, 40, 14, 50). Выполнить преобразование массива в пирамиду (1-я фаза алгоритма HeapSort) и два первых прохода второй фазы алгоритма. Показать состояние массива после каждой операции просеивания.
35. Как следует изменить алгоритм барьерного поиска, если свободная позиция находится не в конце, а в начале массива?
36. Есть такая математическая игра. Один человек задумывает число от 1 до 1 000, а другой должен определить это число, задав десять вопросов, на которые первый отвечает «Да» или «Нет». Какие вопросы следует задавать? Сколько потребуется вопросов, если задумано число от 1 до 1 000 000?
37. Дан массив целых чисел A = (–5, –2, 3, 8, 12, 12, 15, 20, 30, 35, 40, 41, 41, 49, 50). Выполните вручную алгоритм бинарного поиска для ключа x = 12 и для ключа x = 42, выписывая значения i, j, q, A[q].
38. Приведите пример “плохого” изображения для алгоритма Хаффмана.

V. РЕСУРСЫ

5.1 Программное обеспечение

№ п/п	Наименование	Условия доступа
1.	Microsoft Office 2013 Prof +	*Государственный контракт*

5.2 Профессиональные базы данных, информационные справочные системы, интернет-ресурсы (электронные образовательные ресурсы)

№ п/п	Наименование	Условия доступа
	Интернет-ресурсы (электронные образовательные ресурсы)	
1.	Coursera	URL: https://www.coursera.org/learn/algorithm-design-analysis

5.3 Материально-техническое обеспечение дисциплины

Учебные аудитории для лекционных занятий по дисциплине обеспечивают использование и демонстрацию тематических иллюстраций, соответствующих программе дисциплины в составе:

– ПЭВМ с доступом в Интернет (операционная система, офисные программы, антивирусные программы);

– мультимедийный проектор с дистанционным управлением.

Учебные аудитории для лабораторных и самостоятельных занятий по дисциплине оснащены необходимым оборудованием, с возможностью подключения к сети Интернет и доступом к электронной информационно-образовательной среде НИУ ВШЭ.