Оценочные материалы для промежуточной аттестации по дисциплине

Название дисциплины «Алгоритмы и структуры данных»

Код, направление подготовки	09.03.02 Информационные системы и технологии
Направленность (профиль)	Информационные системы и технологии
Форма обучения	Очная
Кафедра-разработчик	Информатики и вычислительной техники
Выпускающая кафедра	Информатики и вычислительной техники

Типовые задания для контрольной работы (2 семестр).

Задание: адаптируя стандартные структуры данных и алгоритмы написать программу, позволяющую заданную прикладную задачу, оценить ее эффективность и составить отчет по результатам ее применения.

Вариант 1. Тимофей ищет место, чтобы построить себе дом. Улица, на которой он хочет жить, имеет длину n, то есть состоит из n одинаковых идущих подряд участков. Каждый участок либо пустой, либо на нём уже построен дом.

Общительный Тимофей не хочет жить далеко от других людей на этой улице. Поэтому ему важно для каждого участка знать расстояние до ближайшего пустого участка. Если участок пустой, эта величина будет равна нулю — расстояние до самого себя.

Помогите Тимофею посчитать искомые расстояния. Для этого у вас есть карта улицы. Дома в городе Тимофея нумеровались в том порядке, в котором строились, поэтому их номера на карте никак не упорядочены. Пустые участки обозначены нулями.

Вариант 2. Тимофей решил организовать соревнование по спортивному программированию, чтобы найти талантливых стажёров. Задачи подобраны, участники зарегистрированы, тесты написаны. Осталось придумать, как в конце соревнования будет определяться победитель.

Каждый участник имеет уникальный логин. Когда соревнование закончится, к нему будут привязаны два показателя: количество решённых задач P_i и размер штрафа F_i. Штраф начисляется за неудачные попытки и время, затраченное на задачу.

Тимофей решил сортировать таблицу результатов следующим образом: при сравнении двух участников выше будет идти тот, у которого решено больше задач. При равенстве числа решённых задач первым идёт участник с меньшим штрафом. Если же и штрафы совпадают, то первым будет тот, у которого логин идёт раньше в алфавитном (лексикографическом) порядке.

Тимофей заказал толстовки для победителей и накануне поехал за ними в магазин. В своё отсутствие он поручил вам реализовать алгоритм быстрой сортировки (*англ.* quick sort) для таблицы результатов. Так как Тимофей любит спортивное программирование и не любит зря расходовать оперативную память, то ваша реализация сортировки не может

потреблять *O(n)* дополнительной памяти для промежуточных данных (такая модификация быстрой сортировки называется "in-place").

Как работает in-place quick sort

Как и в случае обычной быстрой сортировки, которая использует дополнительную память, необходимо выбрать опорный элемент (*англ.* pivot), а затем переупорядочить массив. Сделаем так, чтобы сначала шли элементы, не превосходящие опорного, а затем —— большие опорного.

Затем сортировка вызывается рекурсивно для двух полученных частей. Именно на этапе разделения элементов на группы в обычном алгоритме используется дополнительная память. Теперь разберёмся, как реализовать этот шаг *in-place*.

Пусть мы как-то выбрали опорный элемент. Заведём два указателя *left* и *right*, которые изначально будут указывать на левый и правый концы отрезка соответственно. Затем будем двигать левый указатель вправо до тех пор, пока он указывает на элемент, меньший опорного. Аналогично двигаем правый указатель влево, пока он стоит на элементе, превосходящем опорный. В итоге окажется, что что левее от *left* все элементы точно принадлежат первой группе, а правее от *right* — второй. Элементы, на которых стоят указатели, нарушают порядок. Поменяем их местами (в большинстве языков программирования используется функция *swap()*) и продвинем указатели на следующие элементы. Будем повторять это действие до тех пор, пока *left* и *right* не столкнутся.

Этап: проведение промежуточной аттестации по дисциплине

Семестр 2 (Экзамен)

Алгоритмическое мышление и машина Тьюринга

1. Что такое алгоритм? Перечислите основные свойства алгоритма.
2. Какие типы алгоритмов существуют? Опишите каждый тип подробно.
3. Чем отличаются линейные, ветвящиеся и циклические алгоритмы?
4. Как строится блок-схема согласно стандарту ГОСТ 19.701-90?
5. Что представляет собой машина Тьюринга? Объясните её устройство и принцип работы.
6. Для чего используется концепция машины Тьюринга в теории информатики?
7. Поясните значение понятия «алгоритм полиномиальной сложности».
8. Приведите пример простейшего алгоритма, реализуемого на машине Тьюринга.
9. Реализуйте алгоритм сложения двух натуральных чисел на машине Тьюринга.
10. Почему изучение машины Тьюринга важно для понимания современных компьютеров?

Динамические структуры данных

1. Назовите ключевые отличия статического массива от динамического списка.
2. Что такое указатель? Где используются указатели в управлении памятью?
3. Охарактеризуйте операции над элементами односвязного и двусвязного списков.
4. Какие преимущества имеет динамический список перед статическим массивом?
5. Когда целесообразно применять списки, а когда — массивы?
6. Напишите программу на любом языке программирования, демонстрирующую работу с динамическим списком.
7. Проанализируйте ситуацию, когда лучше использовать массив, а когда — связанный список.
8. Какие структуры данных предпочтительнее выбирать для организации базы данных?

9. Расскажите о внутреннем устройстве связанного списка.
10. Объясните различие между одномерным и многомерным массивом.

Основные алгоритмы сортировки и поиска

1. Опишите основные этапы пузырьковой сортировки. Укажите её временную сложность.
2. Какой алгоритм сортировки применяется чаще всего и почему?
3. Проведите подробный разбор алгоритма быстрой сортировки («quicksort»).
4. Что означает запись O(n log n)? Приведите пример соответствующего алгоритма.
5. Определите разницу между методами поиска: линейным и двоичным поиском.
6. В чём состоит разница между временными сложностями O(log n) и O(n)?
7. По каким критериям оценивается эффективность алгоритма?
8. Покажите пошагово процесс сортировки методом выбора на примере массива целых чисел.
9. Какой алгоритм сортировки считается наиболее эффективным для небольших объёмов данных?
10. Представьте решение задачи сортировки массива чисел с использованием разных подходов и оцените их производительность.

Структуры данных типа дерево

1. Что такое бинарное дерево поиска? Как оно устроено?
2. Объясните три метода обхода деревьев: прямой, симметричный и обратный порядок.
3. Нарисуйте структуру бинарного дерева поиска, включающего элементы {10, 5, 15, 3, 7, 12, 18}, и покажите процесс поиска элемента 7.
4. Почему важна сбалансированность бинарного дерева поиска?
5. Какие виды деревьев являются балансированными? Приведите примеры.
6. Опишите алгоритм удаления узла из бинарного дерева поиска.
7. Реализуйте класс бинарного дерева поиска на вашем любимом языке программирования.
8. Решите задачу: добавьте элемент 11 в ваше бинарное дерево поиска и удалите узел 15.
9. Выполните символьный обход вашего бинарного дерева.
10. Подчеркните роль бинарных деревьев поиска в построении индексированных баз данных.

Хеш-таблицы

1. Что такое хеш-таблица? Зачем нужны хеш-функции?
2. В чём заключается проблема коллизий в хеш-таблице?
3. Перечислите методы разрешения коллизий в хеш-таблицах.
4. Какова эффективность поиска в хеш-таблице по сравнению с обычными массивами?
5. Создайте хеш-таблицу на выбранном вами языке программирования и продемонстрируйте её использование.
6. Применяйте разные стратегии устранения коллизий: открытый адрес и закрытые цепи.
7. Оцените среднюю производительность хеш-таблиц в зависимости от размера таблицы и качества хеширования.
8. Кратко поясните, почему использование хеш-таблиц ускоряет доступ к данным.
9. Сделайте выводы относительно эффективности поиска в структурах данных на основе результатов экспериментов.

10. Прочитав вашу реализацию хеш-таблицы, предложите улучшения для повышения её производительности.

Стеки и очереди

1. Что представляют собой стек и очередь как абстрактные структуры данных?
2. Приведите пример реального приложения, использующего стек.
3. Приведите пример реальной ситуации, где удобно применить очередь.
4. Реализуйте простую систему массового обслуживания, используя концепцию приоритетных очередей.
5. Разработайте собственный код для стэка и очереди на языке Python/Java/C++.
6. Почему важны ограничения на ввод и вывод данных в стеке и очереди?
7. Каковы отличительные черты между реализацией стека и очереди?
8. Постройте графическую иллюстрацию поведения стека и очереди при обработке входных данных.
9. Вычислите длину очереди и среднее время ожидания клиента в вашей системе массового обслуживания.
10. Поясните разницу между FIFO (first-in-first-out) и LIFO (last-in-first-out).

Динамическое программирование

1. Что такое динамическое программирование? Каковы его ключевые характеристики?
2. Что значит термин «мемоизация» и зачем она применяется?
3. Как реализуется метод динамического программирования для решения задачи о рюкзаке?
4. Объясните принцип оптимального подсчета наибольшей общей подпоследовательности (LCS).
5. В каком случае выгоднее использовать метод динамического программирования?
6. Запишите формулу состояния и переходов для задачи о числах Фибоначчи.
7. Опишите, как решается задача «самый большой общий делитель» методом динамического программирования.
8. Найдите оптимальный путь в матрице с минимальными значениями клеток.
9. Посчитайте количество способов добраться из точки A в точку B на прямоугольной сетке размером N x M.
10. Реализуйте решение классической задачи о путях через лабиринт с препятствиями, применяя динамическое программирование.