Общие условия

Программа должна быть реализована на одном из языков: C/C++, Java, C#, Python, Haskell, Scala, Go, Kotlin. Прочие — по договоренности с преподавателем. Программа должна быть платформонезависимой (для C# есть Mono), не иметь зависимостей от нестандартных библиотек и выполнена в виде консольного приложения.

Программа принимает входные данные в виде одного или нескольких текстовых файлов и записывает результат работы в текстовый файл. Имена входных и выходных файлов задаются через аргументы командной строки. Программа неинтерактивная, пользовательский ввод не предусмотрен.

Программа должна быть покрыта тестами. Для проведения тестов студент может использовать готовое ПО или создать его самостоятельно. Тесты представляют собой набор пар «входные файлы»-«выходной файл» и должны содержать проверку корректности всех основных реализованных алгоритмов. Приветствуется предоставление отчета о покрытии разработанной программы тестами.

Программный код должен быть достойного качества, отформатирован и выполнен в едином стиле. Намеренная или неумышленная обфускация кода может привести к снижению баллов или к отклонению выполненной работы.

К программе должен прилагаться краткий отчет в формате PDF, содержащий:

- теоретическую часть, которая не зависит от реализации;
- краткую инструкцию по использованию (и компилированию) программы;
- краткое описание логики программы, описывающую специфику конкретной реализации;
- подробное описание формата входных и выходных данных;
- краткую инструкцию по использованию (установке) тестирующего ПО;
- описание каждого теста;
- оценку сложности основных алгоритмов программы с доказательством.
- оценку использования памяти основными алгоритмами с доказательством.

Домашнее задание сдается в три этапа. Результат каждого из этапов должен быть отправлен на e-mail преподавателя в установленный срок. Допускается досрочное выполнение этапов и изменение результатов (в разумных пределах) предыдущих этапов.

Первый этап содержит только теоретическую часть и должен быть выполнен до 1 октября. Баллы за него учитываются в 1 модуле.

Второй этап содержит:

1. прототип программы, в котором реализованы функции ввода/вывода, а вместо основных алгоритмов — заглушки;
2. описание формата входных и выходных данных;
3. набор тестов, не зависящих от конкретной реализации;
4. тестирующее ПО и инструкцию к нему.

Он должен быть выполнен до 1 ноября и учитывается во втором модуле.

Третий этап учитывается в 3 модуле и содержит полностью выполненное домашнее задание, включающее:

- исходный код программы;
- тесты;
- отчет.

Этап должен быть выполнен до 12 декабря. После проверки преподавателем домашнее задание подлежит защите, после которой выставляется итоговая оценка.

Студент может размещать результаты работы в публично доступном репозитории с веб-интерфесом (например, GitHub). В таком случае допускается оформление отчета в виде одного или нескольких файлов формата Markdown, а на почту преподавателю можно присылать только ссылку на репозиторий.

Не позднее, чем до 1 октября, студент может предложить свой вариант домашнего задания, позволяющий, на его взгляд, продемонстрировать его навыки использования алгоритмов и структур данных (в том числе на основе предлагаемых задач).

Структуры данных

В программе должна присутствовать реализация операций вставки, поиска, удаления, получения минимального и максимального элементов и других, специфичных для указанных структур данных. Реализация структуры данных должна быть инкапуслирована.

Входной файл содержит последовательность команд, т. е. представляет набор строк вида

command [key] [data]

где command — add, delete, search, min, max, print или спец. команда;

key — ключ, целое число;

data — данные, целое число.

Примеры:

add 10 20
search 15
print
min

Команда print должна отражать внутреннее строение структуры данных, а ее формат вывода должен быть описан в отчете.

Теоретическая часть отчета должна содержать типовые сценарии использования всех структур. В практической части для всех сценариев должна быть предоставлена информация о скорости работы сравниваемых структур данных.

1. Сравнить 2-3-дерево и красно-черное дерево.
2. Сравнить AA-дерево и АВЛ-дерево.
3. Сравнить дерево со штрафами (scapegoat tree) и косое дерево.
4. Сравнить рандомизированное бинарное дерева поиска (Randomized binary search tree) и красно-черное дерево.
5. Сравнить декартово дерево (treap) и АВЛ-дерево.
6. Сравнить B-дерево и косое дерево.
7. Сравнить список с пропусками и АВЛ-дерево.
8. Сравнить развёрнутый связный список с отсортированным списком и отсортированным массивом.
9. Сравнить фибоначчиеву кучу и двоичную кучу.
10. Сравнить биномиальную кучу и двоичную кучу.
11. Сравнить структуру данных Rope (веревка) и обычную строку.
12. Сравнить хэш-таблицу, отсортированный массив и любое из деревьев поиска.

Алгоритмы

Формат входного(ых) файла студент выбирает самостоятельно. Выходной файл содержит результат работы алгоритма. В отчете необходимо описать используемые структуры данных и обосновать их выбор. Реализация алгоритма должна быть инкапуслирована. Если студент доказывает, что его задача является NP-полной, то допускается использование приближенных и/или эвристических алгоритмов для ее решения.

13. Вы очутились в подземелье, но у вас есть его карта. Постройте оптимальный маршрут движения к выходу. (В подземельях опытных студентов есть участки, которые можно пройти только в одну сторону, и запертые двери, ключи от которых разбросаны по локации).
14. Построить абстрактное синтаксическое дерево программы. Вход — исходный код программы на выбранном студентом языке.
15. На некотором квадратном поле каждая клетка поля может либо содержать одно число от 1 до 6, либо быть пустой. Выделены начальная и конечная клетки, в каждой обязательно есть число. С каждой клетки можно двигаться на 1 шаг по одному из 8 направлений. Задача заключается в том, чтобы из начальной точки попасть в конечную, задав в соответствие каждой цифре направление движения. При этом путь не должен содержать пустых клеток.

    Вход — поле с цифрами, координаты начальной и конечной точки.

    Выход — набор пар цифра-направление.
16. Написать программу для решения судоку.
17. Некое заведение общепита очень любит акции. Акции заключаются либо в назначении сниженной цены за комбинацию продуктов, либо в предоставлении бесплатного продукта при покупке определенной комбинации продуктов. Напишите алгоритмы для определения набора акций такого, чтобы
    a. купить желаемый набор продуктов максимально выгодно.
    b. максимально сытно поесть на фиксированную сумму.

    Вход — меню (с ценами и калорийностью), акции, набор продуктов или сумма.
18. Пусть у вас есть несколько плоских фигур различной формы. Каждая фигура вырезана из листочка в клеточку строго по линиям и не содержит отверстий. Напишите алгоритм укладки этих фигур в прямоугольник. Фигуры можно вращать на угол, кратный 90^{o} и отражать по горизонтали или вертикали.

19. Коллекционер животных отправился на охоту. Благодаря современным технологиям, он знает расположение всех животных на местности. Его интересуют только те виды, которых у него нет. Прочих он может пустить на мясо. Реализуйте алгоритм построения оптимального маршрута (Опытные студенты могут учитывать вероятность поимки).
20. Постройте программу для сборки кубика Рубика. На вход подается текущее состояние кубика, выходом является последовательность поворотов.
21. Одна из распространенных тактик в онлайн-шутерах и не только – rush, или быстрое перемещение в сторону противника. Пусть у вас есть некоторая карта, и две стартовые точки для двух противоборствующих команд. Все игроки, кроме вас, перемещаются с обычной скоростью, осторожно. Определите маршруты движения, которые обеспечат вам внезапность.
22. Напишите программу, решающую шахматные задачи.
23. Пусть есть некоторое закольцованное игровое поле и набор правил для перемещения по нему. Примеры правил: «переместиться на такую-то клетку», «отойти на три клетки назад». Начальное перемещение задается броском двух кубиков, при выпадении дубля можно ходить еще раз, но три дубля подряд завершают ход. Определите условия, при которых игрок пройдет максимальное количество клеток. В качестве примера можно взять игру «Монополия».
24. Реализуйте программу аналитического вычисления производной функции от нескольких переменных.
25. Ваш друг не так успешен как вы и работает курьером. Он попросил вас помочь ему с его работой. Каждый день он получает заказы в разных частях города. Ему нужно получить оптимальный маршрут передвижения по клиентам. Более того, начальство требует от него указать приблизительное время доставки для каждого заказчика.
26. Представьте, что вы являетесь разработчиком стратегии реального времени. Допустим, игрок может создавать на карте здания-приемники (амбары, лесопилки, кузницы и т. д.). В них поступают ресурсы с прилегающих территорий (зерно, лес, руда и т. д.). При имеющемся расположении приемников необходимо поделить всю карту на непересекающиеся области таким образом, чтобы в каждой области был ровно один приемник и он был ближайшим для всех ресурсов в этой области.
27. Реализуйте алгоритм, строящий и эмулирующий конечный автомат.

28. Разработать алгоритм определения геолокации объекта: попадание точки в один из пересекающихся интервалов. Необходимо вернуть интервал с наибольшим весом.
29. У вас есть здание, подлежащее защите. Вам необходимо обеспечить видеонаблюдение на прилегающей территории, используя минимальное количество камер (обзор камеры — сектор, угол которого является параметром задачи).
30. Напишите программу, решающую задачи о расстановке фигур на шахматной доске (8 ферзей, не бьющих друг друга; 5 ферзей, чтобы под ударом были все клетки поля и т.п.)
31. Реализовать алгоритм для решения задачи формирования портфеля акций на основе метода ветвей и границ. Каждый тип акции имеет цену, вероятность получения дохода и размер возможного дохода. Число акций каждого типа ограничена. Необходимо максимизировать ожидаемую прибыль.
32. Реализовать один из алгоритмов кластеризации графов – например, label propogation или infomap. (Этот вариант могут взять несколько студентов при условии, что алгоритмы будут разными).
33. Реализовать алгоритм сжатия данных. Учтите, что на вход может поступать несколько файлов, поэтому вам необходимо сохранить структуру расположения файлов (дерево папок). Результатом работы алгоритма является ровно один файл. Для проверки корректности работы алгоритма должна быть реализована функция распаковки.
34. Вы устроились на работу специалистом по ИБ. Руководство поставило задачу обеспечения ИБ в предприятии. Известно, что при повышении безопасности страдает удобство пользователей. У вас есть набор средств и мер ИБ, каждая из которых защищает от одной из нескольких угроз. При этом каждое средство снижает удобство пользователя. Постройте алгоритм выбора средств, которые с одной стороны защищают от всех известных угроз, а с другой — наименее неудобны для пользователя.
35. Постройте алгоритм генерации лабиринтов с нужным уровнем проходимости (лабиринт гарантированно должен быть проходим).
36. И снова вы разработчик онлайн-стратегии. Игрок может объединять своих воинов в группы. Постройте алгоритм передвижения группы на местности такой, чтобы суммарно воины прошли наименьшее расстояние (иначе они могут устать), прибыли в конечный пункт одновременно (иначе их перебьют там по одному) и двигались по оптимальному маршруту (иначе перебьют других воинов игрока). Воины двигаются с

одинаковой скоростью, но могут «подождать» свою группу. Движение осуществляется по полю с шестиугольными ячейками («хексами»).

37. Сравните производительность алгоритмов разбиения графов на компоненты связности, основанные на различных алгоритмах поиска (например, BFS), и алгоритма использующего систему непересекающихся множеств (DSU).
38. Постройте алгоритм для нахождения «лестницы слов»: последовательности слов, в которой каждой слово отличается от соседнего одной буквой. Начальное и конечное слово подается на вход программы.
39. Пусть на некоторой местности расположено несколько объектов, между которыми необходимо проложить дороги. Постройте алгоритм, который принимает на вход расположение этих объектов и вычисляет такую карту дорог, в которой общая протяженность дорожного полотна минимальна.
40. Дана сцена с 2D или 3D объектами и наблюдатель, который смотрит на сцену из своей точки обзора. Нужно отрисовать на сцене видимые наблюдателю части объектов. Постройте алгоритм для определения видимых частей.
41. Напишите программу, эмулирующую простой процессор (можно взять за основу, например, MSP430).

Приложение

Список абстракций, задач, методов, алгоритмов и структур данных, которые могут пригодиться при выполнении домашнего задания.

- Алгоритм A*
- Алгоритм Дейсктры
- Алгоритм LZW
- Алгоритм Барроуза-Уилера
- Задача об упаковке в контейнеры (и ее вариант offline-2DSP)
- Диаграммы Вороного
- Метод ветвей и границ
- Задача о рюкзаке
- Задача коммивояжёра
- Интервальное дерево
- Задача ЦЛП
- Алгоритм Прима
- Алгоритм Крускала
- Задача Штейнера
- BSP-дерево
- Конечный автомат
- Генетические алгоритмы