1. Описание известной задачи, решение которой поможет вам получить ответ на вашу задачу сразу или после некоторых манипуляций (либо вы как-то демонстируете, что на досуге я придумал какую-то новую область знаний, и ничего похожего никто никогда до вас не решал). Для этого имеет смысл формализовать текстовую задачу (с человеческого языка перевести на математический) - поэтому формальное описание задачи тоже должно быть в теоретической части.

2. Описание основных подходов к решению известной задачи, желательно с преимуществами и недостатками, ограничениями и т.п.

3. Описание метода, который вы будете использовать для решения задачи, или его модификации. Разумеется, с обоснованием, почему именно он. Это основная часть, в которой вы излагаете идею того, как вы будете решать задачу.  
4. Описание структур данных, которые вы планируете использовать. Формальный пункт, поскольку скорее всего ваш выбор изменится на втором или даже третьем этапе.

* 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 С# 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.

1

* 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

оценк

* 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 С# 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.

1

* 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

оценк

*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).*

Уточнения задания:

1. Каждый вид животных имеет свою вероятность поимки.
2. Во входных данных задается минимально приемлемая вероятность поимки животных каждого вида (см пункт 1)
3. Построение маршрута для охоты на животных ради мяса не планируется т.к. это противоречит условию: «…*его интересуют только те виды, которых у него нет…»*
4. Данную задачу можно перевести в задачу коммивояжера, которая принадлежит к классу NP-полных задач. В замкнутом варианте задачи коммивояжёра требуется посетить все вершины графа, после чего вернуться в исходную вершину. Незамкнутый вариант отличается от замкнутого тем, что в нём не требуется возвращаться в стартовую вершину.

Задача коммивояжера заключается в отыскании самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу с последующим возвратом в исходный город. Поскольку коммивояжер в каждом из городов встает перед выбором следующего города из тех, что он еще не посетил, всего существует (n - )!1 маршрутов. Таким образом, размер пространства поиска зависит экспоненциально от количества городов.

1. Некоторые приближенные способы решения задачи коммивояжера:
   * + - Жадные алгоритмы.
       - Двойная замена.
       - Метод имитации отжига.
       - Метод построения минимального остовного дерева.
       - Генетические алгоритмы.
       - Муравьиные алгоритмы.
       - Метод эластичной сети. (оптимален при малом количестве животных)

Способы для более точного решения задачи коммивояжера:

* + Полный перебор.
  + Метод ветвей и границ.

1. При большом количестве вершин, для получения лучшего результата, целесообразнее всего использовать алгоритм Литтла, т.е. результаты модифицированного алгоритма «Иди в ближний» и генетического алгоритма (турнирный метод отбора) с ростом числа вершин ухудшаются примерно с одной скоростью, ухудшение результатов алгоритма Литтла происходит приблизительно в 10 раз медленнее.

*Описание алгоритма:*

Процесс построения маршрута коммивояжера представляется в виде построения двоичного корневого дерева решений, в котором каждой вершине x соответствует некоторое подмножество M(x) множества всех маршрутов коммивояжера. Считается, что корню дерева решений поставлено в соответствие множество всех маршрутов коммивояжера. Правило ветвления. Пусть x – некоторая вершина дерева. Выберем дугу vw, которая входит ходя бы в один маршрут из M(x) . Тогда множество M(x) разбивается на два непересекающихся подмножества, в одно из которых можно отнести все маршруты, содержащие дугу vw, а в другое – не содержащие ее. Будем считать, что первое из этих подмножеств соответствует левому сыну вершины x , а второе – правому. Вершина дерева, для которой строятся сыновья называется активной (вершина, для которой построены оба сына, активной стать в дальнейшем не может). Пример такого дерева представлен на рис. 5. Главное достоинство метода Литтла по сравнению с полным перебором заключается в том, что активными объявляются лишь те вершины, в которых может содержаться оптимальный маршрут. Для этого существует правило активизации вершин, которое сводится к правилу подсчета границ. Если для вершин дерева решений вычислено значение f (x) такое, что вес любого маршрута из множества значений M(x) не меньше чем f (x) , то такое число f (x) называется нижней границей маршрутов множества M(x) . Правило активизации вершин заключается в том, что из множества вершин, не имеющих сыновей, в качестве активной выбирается вершина с наименьшей нижней границей. Вершина, для которой построены оба сына активной стать не может. Процесс построения дерева решений продолжается до тех пор, пока активной не будет объявлена вершина x , для которой множество M(x) состоит из одного единственного маршрута, а границы всех других вершин не меньше чем вес этого маршрута. Понятно, что тогда маршрут, содержащийся в M(x) , является оптимальным. Редукция строки (столбца) – процедура вычитания из каждого элемента строки (столбца) минимального элемента этой же строки (столбца). Правила вычисления нижних границ. Лемма 1 P – маршрут коммивояжера в сети G , c(P) (соответственно d(P) ) – вес этого маршрута, определяемый матрицей весов сети G (редуцированной матрицей). f – сумма всех констант, используемых при редукции. Тогда c(P) = d(P) + f . Так как редуцированная матрица содержит только неотрицательные элементы, то d(P) ³ 0 . Следовательно, c(P) ³ f . Лемма 2 Пусть вершина x является сыном вершины y в дереве решений, f (y) – нижняя граница вершины y и f – сумма всех констант, используемых при редукции матрицы, соответствующей вершине x . Тогда равенство f (x) = f (y) + f задает нижнюю границу f (x) вершины x .

Учитывая то, что желательно по условию дз учитывать так же вероятность поимки животного, выборка будет осуществляться по формуле:

L = l\*p-1, где L – учитывающееся расстояние, l – реальное расстояние, р - вероятность

1. Структуры данных:

Информация о животном.

Структура с координатами по оси x и y а также типом животного.

Множество животных.

Информация о всех животных будет храниться в классах списков т.к. кол-во видов животных известно изначально, а животные будут считываться по одному и добавляться в соответствующий класс.

Хранилище расстояний

Формат входных данных

• тип животного – целое беззнаковое число.

• Вероятности поимки для каждого из животного – числа в интервале (0, 1].

• Количество животных – целое беззнаковое число.

• Для каждого животного:

* + Координата x
  + Координата y
  + Тип животного
  + вероятность

Формат выходных данных

Последовательность строк с содержанием: type, (x , y)

Где x и y - координаты животного (число с плавающей точкой), type - тип данного животного (целое беззнаковое число).

Тестирующее ПО

Программа принимает в аргументах командной строки имя файла со списком пар

Результаты тестов выводятся в консоль.

Тесты

* 1. Исключить одинаковый ввод на 1 координату
  2. Ограничить вероятность на отрезке (0, 1];
  3. Исключить отрицательные координаты и координату (0,0)
  4. Проверка алгоритма, решающего задачу коммивояжера. Имеется по одному животному каждого типа, все вероятности выставляются равными единице.

ОПИСАНИЕ ЛОГИКИ

На вход подается последовательность чисел, которые считываются в массив, а после разбиваются, сначала в массив, а после в матрицу класса животное. Класс содержит в себе координаты х, у, тип, вероятность, переменную «дистанция» (впоследствии нужна для вычислений), переменная «существование» (впоследствии нужна для вычислений).

Матрица обрабатывается в функции, где ищется минимальные дистанции от I до j животного в строке, а потом в столбце. Данные о минимальных заносятся в матрицы для строк и для столбцов соответственно. Далее начинается «усечение»: если в строке и столбце совпадает минимальное, то мы находим тип этого минимального (j животное) и удаляем (а на самом деле делаем не действительными: «существование» становится false) все столбцы с таким типом в номере, а также все строки кроме строки для этого животного (строка с номером j). Так же это животное добавляется в массив для вывода в ответ.

Операция повторяется пока матрица существует.

ФОРМАТ ВХОДНЫХ/ВЫХОДНЫХ ДАННЫХ

На вход подается txt файл в котором содержатся только численного формата (0; +inf) данные в количестве, кратное 4м. При чем каждое четвертое лежит в пределах [1; 100] (вероятность).

На выход txt файл в котором по порядку выводятся животные.