Практические задания по курсу «Теория информации и теория кодирования» (весна 2019–2020 учебного года)

И. В. Чижов

23 марта 2020 г.

1. Моделирование дискретных источников

Требуется реализовать программу, моделирующую работу дискретного источника.

1.1. Требования к программе

Программа должна работать в двух режимах.

Режим 1. На вход подаётся

1) файл с описанием дискретного источника (см. пример),
2) либо число N, либо $None$.

Программа выдаёт на экран реализацию источника длины N символов, если число N не равное $None$ и больше нуля. Во всех остальных случаях на экран выдаётся бесконечная последовательность символов до тех пор пока пользователем не будет нажата клавиша «q» (учесть русскую раскладку) на клавиатуре.

Режим 2. На вход подаётся

1) число $N > 0$,
2) $a_1, a_2, \ldots, a_k$ — последовательность символов дискретного вероятностного ансамбля.

Программа должна вычислить вероятность появления последовательности $a_1, a_2, \ldots, a_k$ на основе выборки из N символов.

Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную систему на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

Пример 1. Формат файла с описанием дискретного источника (формат называется json):

```
{
    "models": {
        "имя модели": {
            "символ": "вероятность появления",
        },
    },
    "switches": {
        "имя переключателя": {
                "имя модели": "вероятность",
        },
    },
    "source": [
        "имя переключателя", "имя переключателя",
    ]
}
```

Пример 2. Приведём пример файла:

```
{
    "models": {
        "монета_1": {
            "0": "1/3",
            "1":  "2/3"
        },
        "монета_2": {
            "0": "2/3",
            "1": "1/3"
        }
    },
    "switches":{
        "switch_1": {
            "монета_1": "1/2",
            "монета_2": "1/2"
        },
        "switch_2": {
            "монета_1": "0.3",
            "монета_2": "0.7"
        },
        "switch_3": {
            "монета_1": "0.4",
            "монета_2": "0.6"
```

```
        }
    },
    "source": ["switch_1", "switch_2", "switch_3", "switch_2"]
}
```

Пример означает, что в начальный момент времени источник выбирает переключатель «switch_1» и реализует модель выбора монеты, а потом выдаёт символ в соответствии с распределением для «монеты_1». Потом выбирается переключатель «switch_2» и реализуется его схема работы, потом «switch_3», затем «switch_2», и так далее.

1.2. Требования к выполнению задания и принцип его оценивания.

1) Предполагается, что к программе будет приложен файл пояснительной записки с обоснованием интерпретацией выходных данных, описания картины наблюдаемого процесс, постарайтесь привести нетривиальные примеры стационарного/нестационарного процесса, эргодического/неэргодического. Предложите развитие предложенного формата для описания источников с памятью (например, Марковской цепи заданной длины). Объём пояснительной записки — достаточный для интерпретации результатов. Влад в оценку — 45%.

2) Оценивается правильность работы программы (вклад в оценку — 50%) и её эргономика (вклад в оценку — 5%).

3) Плагиат кода или пояснительной записки — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию источника. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 20% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 50% и получается 30% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

4) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

2. Кодирование стационарных источников без памяти равномерными кодами.

Требуется реализовать программу, реализующую кодирование стационарных источников без памяти и эргодических источников равномерными кодами.

2.1. Требования к программе

Программа должна работать в двух режимах.

Режим 1. На вход подаётся описание стационарного источника без памяти в формате json.

Программа выдаёт значение собственной энтропии источника, заданного в файле.

Режим 2. На вход подаётся

1) описание стационарного источника,

2) скорость кодирования R,

3) граница ошибки декодирования ε,

4) мощность q алфавита кодера.

Программа должна вывести в файл в удобном разработчику формате q-ичный код, кодирующий заданный источник со скоростью передачи не выше R и с вероятностью ошибки декодирования ε, если такой код не существует. В том случае, если он не существует, то программа должна объяснить причину, по которой построить такой код невозможно.

Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную систему на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

Формат описания источника приведено в примере 1, возможно доработать формат для задания источника с памятью (например, для Марковского источника заданной глубины). Формат должен быть единым для источников всех типов.

2.2. Требования к выполнению задания и принцип его оценивания.

1) Предполагается, что к программе будет приложен файл пояснительной записки с нетривиальными примерами стационарного процесса без памяти и эргодического источника. Для каждого эргодического источника необходимо доказать его эргодичность. Объём пояснительной записки — достаточный для обоснования эргодичности источников и пояснения выбора форматов. Вклад в оценку — 45%.

2) Оценивается правильность работы программы (вклад в оценку — 50%) и её эргономика (вклад в оценку — 5%).

3) Плагиат кода или пояснительной записки — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию источника. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 20% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 50% и получается 30% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

4) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

3. Архиватор данных "ZipMEHuffman".

Требуется реализовать программу сжатия данных по методу Хаффмана.

3.1. Требования к программе

Программа должна работать в двух режимах.

Режим 1. На вход подаётся бинарный файл.

Программа сжимает файл, сохраняя результат в файл с тем же именем, что и входящий, но с расширением zmh. Сжатие выполняется двухпроходным методом Хаффмена. В первый проход строится модель, а во второй проход выполняется сжатие данных. Формат сжатого файла определяется разработчиком программы.

Режим 2. На вход подаётся файл с расширением zmh.

Программа выполняет разархивирование файла. Результат записывается в файл с тем же именем, что и входной файл, но без расширения zmh. Предусмотреть возможно обработки файлов, которые повреждены или имеют некорректный формат. В этом случае надо уведомлять пользователя.

Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную систему на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

3.2. Требования к выполнению задания и принцип его оценивания.

1) Предполагается, что к программе будет приложен файл пояснительной записки с описанием формата zmh. Вклад в оценку — 25%.

2) Оценивается правильность работы программы (вклад в оценку — 70%) и её эргономика (вклад в оценку — 5%).

3) Плагиат кода или формата файла — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию функции сжатия. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 50% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 70% и получается 20% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

4) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

4. Реализация схемы декодирования на основе стандартного расположения

Требуется реализовать программу, реализующую схему декодирования на основе стандартного расположения.

4.1. Требования к программе

Программа должна работать в трёх режимах.

Режим 1 или режим генерации кода. На вход подаётся:

1) r — число проверочных символов;

2) n — желаемая максимальная длина блока сообщения, передаваемого по каналу связи;

3) p — вероятность ошибки в канале связи для двоичного симметричного канала.

По выходным параметрам генерирует случайный линейный код с максимально возможной скоростью передачи, строит его порождающую матрицу в систематическом виде, строит словарь для декодирования сообщений с границей числа ошибок t. Этот словарь имеет вид {«значение синдрома»: [список векторов ошибок веса не более t, имеющих этот синдром]}. Если такого кода не существует, то выдать объяснение того почему такой код не существует.

На выходе в файл записывается информация для кодера, а также информация для декодера. На экран выводится оценка вероятности ошибки декодера для этого кода на ошибках кратности не более t.

Число ошибок определяется исходя из входных параметров.

Режим 2 или режим кодирования. На вход подаётся:

1) файл с описанием кода;

2) сообщение m;

3) ошибка e — параметр не обязательный.

На экран выводится результат кодирования и наложения ошибки, а также значение вектора ошибки для контроля правильности работы программы. Если ошибка не задана, то она генерируется случайно.

Режим 3 или режим декодирования. На вход подаётся:

1) файл с описанием кода;

2) сообщение y, полученное из канала связи.

На экран выводится значение ошибки и декодированное сообщение (а не кодовый вектор), и вероятность ошибки декодирования.

Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную систему на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

4.2. Требования к выполнению задания и принцип его оценивания.

1) Оценивается правильность работы программы (вклад в оценку — 50%,) скорость построения стандартного расположения (вклад в оценку — 30%) и её эргономика (вклад в оценку — 20%).

2) Плагиат кода — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе. Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию функции кодирования. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 50% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 75% и получается 25% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

3) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».

5. исправление ошибок в каналах связи с помощью кодов БЧХ

Требуется реализовать программу эмуляции исправления ошибок в канале связи с помощью кода БЧХ.

5.1. Требования к программе

Программа должна работать в трёх режимах.

Режим 1 или режим генерации кода. На вход подаётся:

1) n — желаемая максимальная длина блока сообщения, передаваемого по каналу связи;

2) p — вероятность ошибки в канале связи для двоичного симметричного канала.

На основе параметр p программа рассчитывает максимальную длину k исходного сообщения, а также количество ошибок, которое должен исправлять код БЧХ. Если код БЧХ с такими параметрами существовать не может, то длина блока n может быть скорректирована программой автоматически. В конце работы в файл записываются параметры системы связи:

1) t — число ошибок, которое будет исправлять код в блоке данных длины n;

2) n — длина кодового вектора;

3) k — длина блока сообщения;

4) $g(x)$ — порождающий полином кода БЧХ;

5) иная информация, необходимая для декодирования кода БЧХ.

Форматы данных и файлов определяются разработчиком программы.

Режим 2 или режим кодирования. На вход подаётся:

1) файл с описанием кода.

Программа запрашивает у пользователя сообщение в кодировке UTF8. Программа преобразует ввод в двоичную последовательность, разбивает её на блоки длины k. В конце фазы выводится на каждой строке блок сообщения и соответствующее ему кодовое слово. Далее программа для каждого кодового вектора генерирует вектор ошибок в соответствие с моделью двоичного симметричного канала связи с вероятностью ошибки p.

Вектор ошибок накладывается на кодовый вектор. В конце этой фазы для каждого кодового слова на экран выводится сам кодовый вектор, вектор ошибок и искажённый кодовый вектор.

Результат прохождения сообщения по каналу связи записывается в файл.

Режим 3 или режим декодирования. На вход подаётся:

1) файл с описанием кода;

2) файл с сообщением, прошедшим канал связи.

Для каждого искажённого блока применяется алгоритм декодирования Берлекэмпа–Месси и находится величина ошибки. На экран выводится искажённый вектор, величина ошибки, исправленный кодовый вектор, и статус декодирования (Успех/Неудача). Для каждого полученного кодового вектора восстанавливается исходный блок сообщения, а также блоки собираются в итоговый текст и преобразуется в кодировку UTF8. На экран при этом выводится для каждого кодового вектора соответствующее ему сообщение, а также восстановленное сообщение целиком.

Предпочтительный язык программирования — python. В случае использования другого языка нужно создать простую сборочную систему на основе make-файлов для трёх операционных системы: Windows 10, MacOS 10.15+, Linux.

5.2. Требования к выполнению задания и принцип его оценивания.

1) Оценивается правильность работы программы (вклад в оценку — 75%,) и её эргономика (вклад в оценку — 20%).

2) При работе с БЧХ кодами задание предполагает реализацию работы в конечных полях характеристики 2 (вклад в правильность работы программы — 30%). Заимствование возможно (см. следующий пункт), но при этом из общего вклада вычитается 30%. Примитивные многочлены для задания полей можно «зашить» в программу или в конфигурационный файл (что предпочтительнее). Таблицы примитивных многочленов можно взять здесь.

3) Плагиат кода — обнуляет выполнение задачи. И она не засчитывается. Плагиатом не является любое заимствование с указанием его авторства. Заимствованный участок кода не включается в оценивание. Например, Вы не смогли реализовать функцию и заимствовали её у друга, указав это в программе. Реализация этой функции исключается из вклада в оценку правильности работы программы на основе «важности» (принципиальности) этой функции в программе.

Важность вещь разумно-субъективная, определяемая лектором курса. Так, если Вы заимствовали реализацию функции кодирования. То, ясно, что это важная функция, т. к. она проверяет знания по теме курса. И её вклад может быть до 50% вклада кода программы в оценку. В этом случае, этот вклад вычитается из общего вклада в 75% и получается 25% вклада программы. Если же была заимствована функция реализации интерфейса командной строки, то она не относится к тем функциям, которые призваны проверить знания в области криптографии с открытым ключом, поэтому её принципиальность может быть оценена не более, чем в 5%.

4) Для получения оценки «отлично» нужно выполнить задание более, чем на 79%, оценка «хорошо» ставится за выполнение на 65% – 79%, за 50% – 65% — тройка, и при выполнении задания менее, чем на 50% ставится — «неудовлетворительно».