Дисциплина «Информационная безопасность»
Направление - 02.03.03 Математическое обеспечение и администрирование информационных систем
Профиль - Разработка и администрирование информационных систем

Лабораторная работа №4

Коды распознающие и исправляющие ошибки

В настоящее время приходится иметь дело с большими числами: с их помощью обозначают, например, номер страхового полиса, номер банковской карты, номер авиабилета, заводские номера различных устройств. Нам часто приходится вводить эти номера в формах монитора ЭВМ или писать на счетах и квитанциях. Как избежать ошибок? Их избежать нельзя, но можно обнаруживать, если к коду добавить один или несколько контрольных разрядов. При вводе в ЭВМ проверяется выполняется или нет определенное соотношение между информационной и контрольной частями. Код стал избыточным.

В 1958 году сотрудник фирмы IBM Ханс Петер Лун предложил метод, позволяющий значительно снизить вероятность ошибок. Это метод хеширования. Лун предложил к данной последовательности цифр добавлять в конце ещё одну контрольную цифру. Казалось бы, вероятность ошибок может только увеличиться. Для примера рассмотрим операцию, которую в математике называют сложением по модулю натурального числа N: если, складывая два числа, мы получили результат больше, чем N, то будем заменять этот результат его остатком от деления на N. Нам понадобится лишь сложение по модулю 10, то есть, складывая числа, мы всегда будем оставлять от результата его последнюю цифру. Например, 7+8=5, 9+4=3. Специального знака для такого сложения вводить не станем, ограничимся использованием привычного знака + и будем обозначать им именно такое сложение.

Идея Луна следующая. Пусть у нас есть длинные наборы цифр (например, номера банковских счетов), которые мы сообщаем клиентам банка. Чтобы при вводе этих номеров клиенты не допускали ошибки, в конце каждого счёта добавим ещё одну цифру — контрольную, равную сумме всех остальных цифр. Например, к десятизначному номеру 1234567890 добавляем в конце цифру 5 (потому что 1+2+3+4+5+6+7+8+9+0=5 по модулю 10) и сообщаем клиенту 11-значный номер 12345678905. Так у него окажется 11-значный номер счёта. Когда клиент вводит номер своего счёта, ЭВМ складывает первые десять введённых им цифр (по модулю 10 и сравнивает результат с 11-й цифрой. Если результаты получатся разные, мы сообщим клиенту, что он ошибся при вводе.

Изучение ошибок показало, что наибольшую частоту имеют однократные ошибки, двукратные – типа перестановок соседних разрядов, двукратных типа сдвига – 11 в 22, или 55 в 66 или в 44.

Алгоритм Луна. 1. Начиная с первой цифры последовательности слева и через одну цифру (то есть позиции 1, 3, 5, 7, 9, …) в случае, если количество цифр в последовательности нечетное (как в этом примере, где оно равно 15, 16 — контрольная), если же количество цифр четное, тогда, начиная со второй цифры последовательности через одну цифру (то есть позиции 2, 4, 6, 8, …), делается проверка: если $2 \cdot x > 9$, то из произведения вычитается 9, иначе произведение $2 \cdot x$ оставляем без изменения, где x — текущая цифра.

например:

```
4  5  6  1     2  6  1  2     1  2  3  4     5  4  6  4
8     12       4     2       2     6       10    12
```

```
8      3         4      2         2      6         1      3
```

2. Затем все числа, полученные на предыдущем этапе, складываются.

```
8+5+3+1 + 4+6+2+2 + 2+2+6+4 + 1+4+3+4 = 57
```

3. Полученная сумма должна быть кратна 10 (то есть равна 40, 50, 60, 70, …). В примере выше исходная последовательность некорректна.

В примере: последняя цифра — контрольная. Для того, чтобы номер был верен в соответствии с алгоритмом Луна, контрольная цифра должна быть равна 7.

```
4  5  6  1     2  6  1  2     1  2  3  4     5  4  6  7
8     12       4     2        2     6        10    12
8     3        4     2        2     6        1     3
8+5+3+1 + 4+6+2+2 + 2+2+6+4 + 1+4+3+7 = 60
```

Упрощённый алгоритм

1. Цифры проверяемой последовательности нумеруются справа налево.

2. Цифры, оказавшиеся на нечётных местах, остаются без изменений.

3. Цифры, стоящие на чётных местах, умножаются на 2.

4. Если в результате такого умножения возникает число больше 9, оно заменяется суммой цифр получившегося произведения — однозначным числом, то есть цифрой.

5. Все полученные в результате преобразования цифры складываются. Если сумма кратна 10, то исходные данные верны.

Программа проверки кода Луна

```
num = list(input("Пожалуйста, введите номер для проверки (без пробелов, без символов, только \
цифры): "))

num = list(map(int, num))[::-1] # преобразовать строку в int и перевернуть ее

for index in range(1,len(num),2):
    if num[index]<5:
        num[index] = num[index] *2
    else: #удвоение числа>=5 даст 2-значное число
        num[index] = ((num[index]*2)//10) + ((num[index]*2)%10)

checksum=sum(num)

print("контрольная сумма= {}".format(checksum))

if checksum%10 !=0:
    print('номер с ошибкой')
else:
    print('номер без ошибок!')
```

Алгоритм Якоба (Якобуса) Верхуффа хорошо известен специалистам по хешированию. В основе лежит структура алгебры группа. Группа в математике — множество, на котором определена ассоциативная бинарная операция, причём для этой операции имеется нейтральный

элемент (аналог единицы для умножения), и каждый элемент множества имеет обратный. Ветвь общей алгебры, занимающаяся группами, называется теорией групп. Один из примеров группы — множество целых чисел, снабжённое операцией сложения: сумма любых двух целых чисел также даёт целое число, роль нейтрального элемента играет ноль, а число с противоположным знаком является обратным элементом. Для наших операций введем специальный знак *. Чтобы множество получило право называться группой, мало ввести операцию, нужно, чтобы выполнялось три свойства:

1. Для любых трёх элементов a, b, c должно быть верно равенство a * (b * c) = = (a * b) * c. Такое свойство называют ассоциативностью.

2. Должен существовать такой элемент **e**, чтобы для любого другого элемента **a** было верно равенство: a * e = e * a = a. Такой элемент называют нейтральным элементом.

3. Для любого элемента a должен существовать такой элемент **a’**, чтобы было верно равенство: **a * a’ = a’ * a = e**. Такой элемент называют обратным элементом к элементу **a**. Из этих трёх свойств вытекает важное наблюдение: если (в любой группе) a * b = a * c , то b = c. Убедимся в этом на следующем примере. Найдём для элемента a элемент a’ (как в свойстве 3) и допишем его слева к нашему равенству. Получим, что a’ * (a * b) = a’ * (a * c). Теперь применим свойство 1 и получим, что (a’ * a) * b = (a’ * a) * c. Воспользуемся тем, что a’ * a = e, и получим, что e * b = e * c. Наконец, воспользуемся свойством 2 и получим, что b = c, чего мы и хотели. Наше наблюдение означает: абсолютно любая группа поможет находить ошибки в одной цифре (это уже удавалось с помощью сложения по модулю 10). А равенство a * b = b * a, между прочим, вовсе не обязательно должно выполняться. Это равенство называют коммутативностью, а группы, в которых оно верно, — коммутативными группами. Наша группа цифр со сложением по модулю 10 была именно такой, но это и помешало распознавать ошибки, состоящие в перестановке двух соседних цифр. Верхуфф предложил использовать для получения контрольной цифры другую группу.

предложил алгоритм вычисления контрольной цирфы, который по сей день используется во многих ситуациях. Якоб Верхуфф родился в Гааге, окончил университет в Амстердаме и в 1969 году защитил диссертацию по теме: «Десятичные коды с обнаружением ошибок», где был представ ря, идея хеширования в том и состоит, что-бы выбрать какую-нибудь группу, элемен¬ты которой — цифры, и последовательно применять операцию из этой группы к данным цифрам, а результат записывать в конце в качестве контрольной цифры. Операцию в нашей группе мы обозначали знаком +, потому что она похожа на сложе-ние. Операции, которые мы изучим, ни на что похожи не будут, поэтому введём для них специальный знак *. Чтобы множество получило право называться группой, мало ввести опе¬рацию, нужно, чтобы выполнялось три свойства:

1.Для любых трёх элементов a, b, c должно быть верно равенство a * (b * c) = = (a * b) * c. Такое свойство называют ас-социативностью.

2.Должен существовать такой эле¬мент e, чтобы для любого другого элемен¬та a было верно равенство: a * e = e * a = a. Такой элемент называют нейтральным элементом.

3.Для любого элемента a должен существовать такой элемент a’, чтобы было верно равенство: a * a’ = a’ * a = e. Такой элемент называют обратным элементом к элементу a.Из этих трёх свойств вытекает важное наблюдение: если (в любой группе) a * b = a * c , то b = c. Убедимся в этом на следующем примере. Найдём для элемента a элемент a’ (как в свойстве 3) и допишем его слева к нашему равенству. Получим, что a’ * (a * b) = a’ * (a * c). Теперь применим свойство 1 и получим, что (a’ * a) * b = (a’ * a) * c. Воспользуемся тем, что a’ * a = e, и получим, что e * b = e * c.Наконец, воспользуемся свойством 2 и получим, что b = c, чего мы и хотели. Наше наблюдение означает: абсолютно любая группа поможет находить ошибки в одной цифре (это уже удавалось с помощью сложения по модулю 10). А равенство a * b = b * a, между прочим, вовсе не обязательно должно выполняться. Это равенство называют коммутативностью, а группы, в которых оно верно, — коммутативными группами. Наша группа цифр со сложением по модулю 10 была именно такой, но это и помешало распознавать ошибки, состоящие в перестановке двух соседних цифр. Верхуфф предложил использовать для получения контрольной цифры другую группу. Но как её задать? Как вообще задают группы, а точнее — операции в них? Для бесконечных групп этот вопрос действительно трудный, но для групп с конечным числом элементов (как у нас — всего десять) можно воспользоваться простой таблицей. На что это похоже? Конечно же на таблицу умножения, которую изучают в школе. Здесь в клетках таблицы «Таблица умножения для диэдральной группы D5» написаны тоже цифры.

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	0	6	7	8	9	5
2	2	3	4	0	1	7	8	9	5	6
3	3	4	0	1	2	8	9	5	6	7
4	4	0	1	2	3	9	5	6	7	8
5	5	9	8	7	6	0	4	3	2	1
6	6	5	9	8	7	1	0	4	3	2
7	7	6	5	9	8	2	1	0	4	3
8	8	7	6	5	9	3	2	1	0	4
9	9	8	7	6	5	4	3	2	1	0

Пользоваться этой таблицей очень просто: если вы хотите посчитать результат операции a * b, то найдите в левом столбце a, в верхней строчке b, а на пересечении определяемых ими строчки и столбца стоит результат операции. Например, 2 * 3 = 0, 6 * 8 = 3.

Дальнейшие действия — те же самые. Берём многозначное число (например, номер авиабилета), вычисляем по таблице результат последовательного применения операций к цифрам, то есть по номеру, например 54321, вычисляем 5 * 4 * 3 * 2 * 1 = 5 и дописываем полученную цифру в конце. В таком виде сообщаем номер клиенту и, после того как он его введёт (зачем бы то ни было), вычисляем по первым пяти цифрам контрольную цифру и сверяем её с введённой.

Таким образом, мы распознаём больше ошибок: во-первых, все замены любой одной цифры на другие будут найдены, как и раньше. Но не только. Операция, предложенная Верхуффом, позволяет находить 60 из 90 возможных перестановок соседних цифр. Таких

перестановок ровно 90, потому что пару различных цифр можно выбрать 90 способами: десятью способами первую цифру и девятью способами — вторую.

Теперь рассмотрим такой объект, как квазигруппа. Это «почти» группа, то есть операция тоже задана, однако требуется выполнение лишь одного её свойства, а именно: для любых a и b существуют такие элементы квазигруппы c и d, что a * c = b и d * a = b.

И всё! С точки зрения таблицы умножения это означает, что в каждой строке и каждом столбце таблицы встречаются все элементы квазигруппы. Оказывается, только этого свойства достаточно, чтобы контрольная цифра, посчитанная с помощью данной группы, находила все замены одной цифры на ошибочную. Доказательство этого факта сложнее вышеуказанного, так как пользоваться ассоциативностью и существованием нейтрального элемента теперь нельзя, поэтому мы не будем на нём останавливаться.

Полностью антисимметричная квазигруппа Дамма. Это открытие — одно из значительных математических достижений XXI века. До него долгое время считалось, что такие группы не существуют! Что же значат слова «полностью антисимметричная»? Они означают выполнение ещё двух свойств, а именно: Таблица умножения для полностью антисимметричной группы из 10 элементов.

- если для некоторых a и b оказалось, что a * b = b * a, то a = b;
- если для некоторых a, b и c из квазигруппы оказалось, что (a * b) * c =

= (a * c) * b, то b = c.

Для нас важным является первое свойство. Оно означает, что в этой группе совсем нет коммутирующих элементов.

Таким образом, квазигруппа Дамма позволяет легко достичь цели. Мы нашли способ вычисления контрольной цифры так, чтобы распознавались все единичные замены одной цифры на другую и также все перестановки соседних цифр. Сам алгоритм такой же, как и раньше. Если есть длинная последовательность цифр (например, 56789), то мы вычисляем с помощью группы Дамма контрольную цифру, просто, как и раньше, применяя операцию к цифрам слева направо. В нашем случае 5 * 6 * 7 * 8 * 9 = 1, поэтому мы дописываем в конец последовательности 1 и сообщаем пользователю последовательность 567891. На главной диагонали группы Дамма стоят нули. Это значит, что можно просто вычислить результат операции с данными шестью цифрами и, если получится 0, то всё в порядке. Действительно, 5 * 6 * 7 * 8 * 9 * 1 = 0, так что число 567891 корректно.

Талица (группа) Дамма.

	0	1	2	3	4	5	6	7	8	9
0	0	3	1	7	5	9	8	6	4	2
1	7	0	9	2	1	5	4	8	6	3
2	4	2	0	6	8	7	1	3	5	9
3	1	7	5	0	9	8	3	4	2	6
4	6	1	2	3	0	4	5	9	7	8
5	3	6	7	4	2	0	9	5	8	1
6	5	8	6	9	7	2	0	1	3	4
7	8	9	4	5	3	6	2	0	1	7
8	9	4	3	8	6	1	7	2	0	5
9	2	5	8	1	4	3	6	7	9	0

ЗАДАНИЯ РАБОТЫ

1. Создать проект в среде Visual Studio 2019 с использование языка программирования Python.
2. Сформировать необходимое окружения языка Python из библиотек, необходимых для выполнения лабораторной работы.
3. Создать два файла-программы в языке python для генерации и проверки от 1 до12 разрядных номеров с одним контрольным разрядом групп Верхуфф и Дамма.
4. Подготовить 4 примера с ошибками и без них, 2 – группа Верхуфф, 2 – Дамма.
5. Оформить отчет по работе с указанием описания алгоритма кодирования, описания программ шифровки и дешифровки, описание примеров и указания недостатков и достоинств алгоритма.