# Финал - Бэкенд-разработка

1 июн 2019, 16:00:05
старт: 1 июн 2019, 12:00:00
финиш: 1 июн 2019, 16:00:00
длительность: 04:00:00
начало: 1 июн 2019, 12:00:00
конец: 1 июн 2019, 16:00:00

## D. Задача про потерянные данные

| | Все языки | Python 2.7 | Python 3.6 |
|---|---|---|---|
| Ограничение времени | 2 секунды | 4 секунды | 4 секунды |
| Ограничение памяти | 256Mb | 256Mb | 256Mb |
| Ввод | стандартный ввод или input.txt | | |
| Вывод | стандартный вывод или output.txt | | |

В распределённых системах одна из наиболее важных задач — это умение переживать сбои оборудования. Классическим подходом к решению проблемы сбоев в случае хранения данных является тройная репликация, то есть хранение каждого блока данных на трёх разных хостах в кластере. В таком случае данные не потеряются при одновременном выходе из строя двух дисков. Одновременная поломка трёх дисков считается крайне маловероятным событием и на практике практически никогда не случается.
Однако, тройная репликация приводит к тому, что для хранения $X$ данных на кластере надо иметь $3X$ свободного места, что достаточно расточительно. Для экономии места и сохранения гарантий надёжности используются коды, исправляющие ошибки (а точнее, в нашем случае, коды, исправляющие потерю части данных).

Рассмотрим один из таких кодов. Представим, что наши данные состоят из числа битов, кратного 20. Для простоты будем считать, что ровно из 20 бит и научимся кодировать эти 20 бит (для большего числа бит задача решается простым повторением данного алгоритма).

Итак, представим наши 20 бит в виде матрицы $\{a_{i,j}\}$ размером 4 на 5, то есть $i = 1 \dots 4, j = 1 \dots 5$. Мы дополним матрицу с данными двумя контрольными столбцами $\{a_{i,6}\}$ и $\{a_{i,7}\}$.

Шестой столбец будет просто представлять собой xor-сумму столбцов, то есть:

$$a_{i,6} = \bigoplus_{j=1}^{5} a_{i,j}, \quad i = 1 \dots 4$$

Седьмой столбец будет устроен более хитро. Он будет представлять собой определённые xor-суммы диагоналей. А именно, положим $S = \oplus_{k=1..4} a_{k,6-k}$ , тогда:

$$a_{i,7} = S \oplus \bigoplus_{k=1..4} a_{k,shift(6+i-k)}, \quad i = 1 \dots 4$$

Здесь $shift$ обозначает функцию, которая шагом величины 5 сдвигает число в диапазон $1 \dots 5$, (то есть $shift\,(x) \in \{1 \dots 5\}$ и $x - shift\,(x)$ кратно 5).

Изобразим вклад элементов в контрольные суммы 7-го столбца в виде матрицы:

$$\begin{pmatrix} a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & S \\ a_{7,2} & a_{7,3} & a_{7,4} & S & a_{7,1} \\ a_{7,3} & a_{7,4} & S & a_{7,1} & a_{7,2} \\ a_{7,4} & S & a_{7,1} & a_{7,2} & a_{7,3} \end{pmatrix}$$

После этого мы будем писать каждый столбец матрицы на отдельный хост. Утверждается, что при потере любых двух столбцов их можно восстановить из имеющихся, то есть полученный код в определённом смысле не менее надёжен, чем тройная репликация. При этом видно, что накладные расходы указанного кода составляют лишь 40%, в отличие от 200%, которые были у тройной репликации.

Заметим, что при подсчёте контрольных сумм используется только операция xor, поэтому можно в качестве $a_{i,j}$ использовать числа фиксированной битовой ширины, а не просто биты. В нашем случае мы будем кодировать 32-битные беззнаковые числа.

Опишем немного точнее, как мы будем кодировать данные. Для кодирования данные разбиваются на пять непрерывных частей одинакового размера, причём размер части должен быть кратен четырём. Каждая из частей соответствует фиксированному столбцу, при этом первые четыре числа в части относятся к первой матрице, которую мы кодируем, вторые четыре числа ко второй матрице и так далее.

Ваша задача состоит в том, чтобы написать алгоритм, который восстанавливает потерянные данные при использовании указанной схемы кодирования.

# Формат ввода

В первой строке входных данных указано число $n$ — количество кодируемых чисел в одной части и числа $e_1, e_2$, обозначающие номера потерянных частей ($1 \le e_1 < e_2 \le 7, i = 1, 2$). Число $n$ кратно четырём и не превосходит $2 \cdot 10^5$.
Далее в 5 строках заданы данные, отвечающие частям, которые остались в сохранности. В каждой из 5 строк записано $n$ чисел, разделенных пробелами, относящихся к очередному сохранившемуся столбцу в порядке возрастания индексов столбцов.

Например, если были потеряны 1-й и 4-й столбцы, то в пяти строках будут указаны данные, относящиеся к 2-му, 3-му, 5-му, 6-му и 7-му столбцам соответственно.

Гарантируется, что входные данные были получены алгоритмом, описанным в условии.

# Формат вывода

Выведите две строки по $n$ чисел в каждой, соответствующие потерянным данным.

## Пример 1

| Ввод | Вывод |
|---|---|
| 4 6 7<br>1 0 0 0<br>0 0 0 1<br>0 0 0 0<br>0 0 0 0<br>0 0 0 0 | 1 0 0 1<br>0 1 1 1 |

## Пример 2

| Ввод | Вывод |
|---|---|
| 8 1 2<br>0 0 0 0 2 2 2 2<br>0 0 0 0 2 2 2 2<br>0 0 0 0 2 2 2 2<br>1 0 0 1 2 2 2 2<br>0 1 1 1 0 0 0 0 | 1 0 0 0 2 2 2 2<br>0 0 0 1 2 2 2 2 |

# Примечания

Обратите внимание, что у задачи большой вход, поэтому в случае использования языка C++ рекомендуется выключить синхронизацию C/C++ способов чтения, то есть вызвать метод `std::ios::sync_with_stdio(false)`.

Язык Mono C# 5.2.0

Набрать здесь | Отправить файл

1

Отправить

Предыдущая | Следующая