

# Е. Секрет Пазырыкского ковра - Yandex Cup 2025 — Backend



В 1949 году при раскопках пятого Пазырыкского кургана был обнаружен старейший из ныне существующих ворсовых ковров. Учёные датируют находку V веком до нашей эры, а споры о цивилизации-авторе идут до сих пор. После обнаружения была проделана работа по изучению пряжи и техники изготовления данного ковра, во время которой ученым удалось сделать удивительное открытие.

Все ковры того периода объединяет особый вид техники изготовления, когда нити для изготовления сплетаются между собой одним из двух способов и дальше продолжают как одно целое, до тех пор пока не получается одна нить-узел в самом центре ковра. Учёные предполагают, что, подавая как-либо сигналы на исходные нити, каждое переплетение преобразует все входящие сигналы в выходной и передаёт на следующее переплетение, пока оно не дойдёт до последнего узла в центре. Таким образом, ковёр мог выступать первым процессором.

Чтобы подтвердить свою гипотезу, учёные придумали компьютерную систему для моделирования передачи сигналов через ковер. В качестве двух возможных способов переплетения нитей были выбраны

битовые операции конъюнкции (&) и дизъюнкции (|), а в качестве входных сигналов принято решение использовать 32-битные числа в десятичном представлении.

Система состоит из следующих частей:

1. Карта плетения ковра — JSON-строка, описывающая ковёр.
2. Программа, которая на основе карты плетения моделирует преобразование сигнала ковром и выводит результат обработки в последнем узле.
3. Множество серверов-генераторов 32-битных чисел, которые можно получить через HTTP-интерфейс.

Ваша задача — написать программу для моделирования преобразования сигнала.

К сожалению, проект изначально имел очень ограниченный бюджет, поэтому серверы-генераторы оказались не очень стабильными. Практика показала, что если сервер начинает отвечать дольше 1 секунды, то отправлять ему запросы дальше бесполезно.

## Формат ввода

Всё взаимодействие с решением участника происходит через стандартный ввод и вывод.

**Первой строкой** в стандартном вводе решение получает карту плетения ковра в JSON-формате **одной строкой**.

Карта плетения — это валидный JSON следующего формата (переносы строк добавлены для наглядности):

```
{
  "result": {
    "operation": "&",
    "children": [
      {
```

```

    "operation": "|",
    "children": [
      {
        "operation": "call",
        "backend": "http://server1:123/"
      },
      {
        "operation": "call",
        "backend": "http://server2:124/"
      }
    ]
  },
  {
    "operation": "call",
    "backend": "http://server3:125/"
  }
]
}
}

```

Поле `operation` может принимать одно из трёх значений:

- `&` или `|` — в таком случае результат вычисления всех операций из массива `children` (наличие гарантируется) должен объединяться через битовое И или ИЛИ соответственно. В этом случае поле `backend` отсутствует.
- `call` — терминальная нить, на которую необходимо подать сигнал. Значение сигнала необходимо получить, вызвав `GET /code` запрос по адресу, указанному в поле `backend` (URL оканчивается на `/`). В качестве ответа запрос в теле возвращает знаковое целочисленное 32-битное число в формате `text/plain`.

После обработки строки с картой плетения ковра проверяющая программа ожидает, что решение запишет строку `OK` в стандартный вывод.

Далее следует серия экспериментов. Перед каждым экспериментом решение получает команду NEXT в стандартном вводе, а далее на основе карты плетения и ответов серверов-генераторов должна записать в стандартный вывод полученное после преобразований 32-битное число в десятичном формате. Если ситуация со стабильностью серверов-генераторов не позволяет произвести расчет, то вместо результата преобразования необходимо вернуть строку ERROR.

Моделирующая система гарантирует, что в рамках одного эксперимента один и тот же сервер-генератор всегда будет возвращать один и тот же код, но с каждым следующим экспериментом он может измениться.

Последней командой после серии экспериментов является команда SHUTDOWN, на которую необходимо также ответить OK и корректно завершить программу-решение.