Intern Backend Meetup 2024

5 мар 2024, 22:22:53
старт: 5 мар 2024, 19:44:07
финиш: 5 мар 2024, 23:44:07
до финиша: 01:21:08
...

Объявления жюри | Завершить

D. P2P обновление

✓ A. 2 к 1
✓ B. Плоттер
✓ C. Эмоциональные качели
✓ D. P2P обновление

Ограничение времени	15 секунд
Ограничение памяти	256Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

В системе умного дома под управлением голосового помощника Лариса n устройств, соединяющихся между собой по сети LoRaWAN. Устройство номер 1 подключено к интернету и на него было скачано обновление, которое необходимо передать на все устройства.
Сеть LoRaWAN очень медленная, поэтому для распространения протокола был придуман peer-to-peer (P2P) протокол. Файл обновления разбивается на k одинаковых по размеру частей, занумерованных от 1 до k.

Передача части обновления происходит во время таймслотов. Каждый таймслот занимает одну минуту. За один таймслот каждое устройство может получить и передать ровно одну часть обновления. То есть устройство во время таймслота может получать новую часть обновления и передавать уже имеющуюся у него к началу таймслота часть обновления, или совершать только одно из этих действий, или вообще не осуществлять прием или передачу. После приема части обновления устройство может передавать эту часть обновления другим устройствам в следующих таймслотах.

Перед каждым таймслотом для каждой части обновления определяется, на скольких устройствах сети скачана эта часть. Каждое устройство выбирает отсутствующую на нем часть обновления, которая встречается в сети реже всего. Если таких частей несколько, то выбирается отсутствующая на устройстве часть обновления с наименьшим номером.

После этого устройство делает запрос выбранной части обновления у одного из устройств, на котором такая часть обновления уже скачана. Если таких устройств несколько — выбирается устройство, на котором скачано наименьшее количество частей обновления. Если и таких устройств оказалось несколько — выбирается устройство с минимальным номером.

После того, как все запросы отправлены, каждое устройство выбирает, чей запрос удовлетворить. Устройство A удовлетворяет тот запрос, который поступил от наиболее ценного для A устройства. Ценность устройства B для устройства A определяется как количество частей обновления, ранее полученных устройством A от устройства B. Если на устройство A пришло несколько запросов от одинаково ценных устройств, то удовлетворяется запрос того устройства, на котором меньше всего скачанных частей обновления. Если и таких запросов несколько, то среди них выбирается устройство с наименьшим номером.

Далее начинается новый таймслот. Устройства, чьи запросы удовлетворены, скачивают запрошенную часть обновления, а остальные не скачивают ничего.

Для каждого устройства определите, сколько таймслотов понадобится для скачивания всех частей обновления.

Формат ввода

Вводится два числа n и k ($2 \le n \le 100$, $1 \le k \le 200$).

Формат вывода

Выведите $n-1$ число — количество таймслотов, необходимых для скачивания обновления на устройства с номерами от 2 до n.

Пример

Ввод	Вывод
3 2	3 3

Примечания

Для удобства будем пользоваться обозначениями устройств буквами A, B, C (соответствует устройствам с номерами 1, 2 и 3). На устройстве A есть обе части обновления, а на устройствах B и C — ни одной.
Перед первым таймслотом для каждой части определяется количество устройств, на которых скачана каждая часть обновления: и 1 и 2 часть обновления присутствуют только на одном устройстве.

Устройства B и C выбирают самую редкую отсутствующую у них часть обновления с минимальным номером: самая редкая часть с минимальным номером — это часть 1. Она отсутствует и на устройстве B, и на устройстве C. Они запрашивают ее у устройства A. Ценность устройств B и C для устройства A равна нулю. Количество имеющихся у устройств B и C частей обновления одинакова и равно нулю. Поэтому устройство A выбирает устройство с минимальным номером (B). Во время первого таймслота выполняется передача части 1 с устройства A на устройство B. Ценность устройства A для устройства B становится равной 1.

Перед вторым таймслотом для каждой части определяется количество устройств, на которых скачана каждая часть обновления: самой редкой оказывается часть 2 (присутствует только на устройстве A), следующая по редкости часть 1 (присутствует на устройствах A и B).

Устройства B и C выбирают среди отсутствующих у них частей обновления самую редкую: для обоих устройств выбирается часть 2. Каждое из них делает запрос части 2 у единственного обладателя этой части — устройства A. Ценность устройств B и C для устройства A одинакова и равна нулю. Количество имеющихся у устройства C частей (0) меньше, чем у устройства B (1), поэтому выбирается устройство C. Во время второго таймслота выполняется передача части 2 с устройства A на устройство C. Ценность устройства A для устройства C становится равной 1.

Перед третьим таймслотом для каждой части определяется количество устройств, на которых скачана каждая часть обновления: обе части 1 и 2 присутствуют на двух устройствах (часть 1 на устройствах A и B, часть 2 — на устройствах A и C)

Устройство B может сделать запрос недостающей части 2 у обладающей ей устройств A и C, но выбирает устройство C, т.к. на устройстве C скачано меньше частей (1), чем у устройства A (2).

Устройство C может сделать запрос недостающей части 1 у обладающей ей устройств A и B, но выбирает устройство B, т.к. на устройстве B скачано меньше частей (1), чем у устройства A (2).

Во время третьего таймслота оба запроса оказываются единственными запросами у устройств B и C и удовлетворяются. Часть 2 передается с устройства C на устройство B. Часть 1 передается с устройства B на устройство C. Ценность устройства B для устройства C становится равной 1. Ценность устройства C для устройства B становится равной 1.

Все части обновления оказываются на всех устройствах и на этом обновление заканчивается.

Язык Python 3.9 (PyPy 7.3.11)

Набрать здесь | Отправить файл

```
from collections import defaultdict

def get_next_part(dev, downloaded, finished):
    for _, next_part in downloaded:
        if next_part not in finished[dev]: return next_part

def get_next_source(part, finished):
    sources = []

    for i in range(len(finished)):
        if part in finished[i]: sources.append((i, len(finished[i])))

    min_finished = next_source = float('inf')
    for source, count in sources:
        if count < min_finished:
            min_finished = count
            next_source = source
        elif count == min_finished:
            if source < next_source: next_source = source

    return next_source

def get_next_worth_request(cur_dev, requests, worth, finished):
    if not requests: return None, None

    max_worth = float('-inf')
    for dev, _ in requests:
        if worth[cur_dev][dev] > max_worth:
            max_worth = worth[cur_dev][dev]

    min_finished = next_dev = float('inf')
    for dev, _ in requests:
        if worth[cur_dev][dev] == max_worth:
            if len(finished[dev]) < min_finished:
                min_finished = len(finished[dev])
                next_dev = dev
            elif len(finished[dev]) == min_finished:
```

Отправить | осталось 99 попыток

Предыдущая

Время посылки	ID	Задача	Компилятор	Вердикт	Тип посылки	Время	Память	Тест	Баллы	
5 мар 2024, 22:20:04	108907058	D	Python 3.9 (PyPy 7.3.11)	OK	-	3.406s	35.98Mb	-	-	отчёт