prototype3628800

Школа бэкенд-разработки 2021 (осень)

11 сен 2021, 15:12:10
старт: 11 сен 2021, 10:55:06
финиш: 11 сен 2021, 16:55:06
до финиша: 01:42:54
длительность: 06:00:00

F. Кэширование запросов

Ограничение времени	3 секунды
Ограничение памяти	512Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Сервисы Яндекса постоянно находятся под нагрузкой миллионов пользователей. Для эффективности их работы часть запросов кэшируется с помощью определенных алгоритмов.
Вам предлагается смоделировать алгоритм кэширования, при котором в памяти хранится информация о m наиболее поздних по времени вызова запросах. Важной деталью является тот факт, что порядок получения кэшом информации о запросах необязательно совпадает с порядком их вызова.

Для работы с кэшем есть 3 типа операций:

- `PUT` — положить информацию о запросе в кэш (если запроса нет в кэше).
- `UPDATE` — обновить информацию о запросе (если запрос уже есть в кэше).
- `DELETE` — удалить информацию о запросе из кэша, если необходимо освободить место.

Необходимо обработать список запросов и вывести список совершенных с кэшом операций, чтобы в любой момент соблюдались следующие условия:

- Хранимые в кэше запросы являются наиболее поздними по времени вызова среди уже обработанных.
- Количество запросов в кэше не превосходит m.
- Для каждого идентификатора запроса в кэше хранится самая поздняя по времени вызова информация.
- Операция `PUT` применяется только к запросам, которых нет в кэше на момент совершения операции, а `UPDATE` - только к уже находящимся в кэше.

Формат ввода

В первой строке заданы два целых числа n и $m (1 \le n, m \le 200\ 000)$ — количество запросов и максимальный размер кэша.
Каждая из следующих n строк содержит запрос в формате $id\ time \left(1 \le \left|id\right| \le 10; 1 \le time \le 10^{18}\right)$ — идентификатор и время вызова запроса. Идентификатор id состоит из строчных латинских букв.

Гарантируется, что все запросы вызывались в различные моменты времени ($time_i \ne time_j$ для $i \ne j$).

Формат вывода

Необходимо вывести список совершенных с кэшом операций в формате $index\ operation\ id$, где:

- $index$ $(1 \le index \le n)$ — номер запроса, при обработке которого были совершена операция;
- $operation$ — это одна из строк `PUT`, `UPDATE`, `DELETE`.

Выведенные операции должны удовлетворять следующим условиям:

- Хранимые в кэше запросы являются наиболее поздними по времени вызова среди уже обработанных.
- Количество запросов в кэше не превосходит m.
- Для каждого идентификатора запроса в кэше хранится самая поздняя по времени вызова информация.
- Операция `PUT` применяется только к запросам, которых нет в кэше на момент совершения операции, а `UPDATE` - только к уже находящимся в кэше.

Пример 1

Ввод	Вывод
6 3 status 1 history 2 status 3 price 4 name 5 card 6	1 PUT status 2 PUT history 3 UPDATE status 4 PUT price 5 DELETE history 5 PUT name 6 DELETE status 6 PUT card

Пример 2

Ввод	Вывод
5 2 status 4 history 2 history 10 price 7 status 3	1 PUT status 2 PUT history 3 UPDATE history 4 DELETE status 4 PUT price

Примечания

Рассмотрим первый тест из условия.
На момент обработки третьей строки `status 3` в кэше уже лежит запись о запросе `status`, поэтому производится операция UPDATE, а не PUT. Обратите внимание, что после данной операции в кэше лежит 2 записи, а не 3: (`status 3`; `history 2`).

При обработке пятой строки `name 5` из кэша необходимо удалить самую старую запись по времени вызова - `history 2`. В итоге после обработки пятой строки в кэше будут лежать записи (`status 3`; `price 4`; `name 5`).

Во втором тесте важно отметить два факта:

- запросы даются не в порядке времени вызова.

- последняя строка `status 3` никак не изменяет кэш, в котором на тот момент лежат записи (`history 10`; `price 7`), так как время запроса 3 ниже, чем оба присутствующих в кэше времён.

Язык Python 3.7 (PyPy 7.3.3)

Набрать здесь | Отправить файл

```
##
#   Author:     antikostya
#   Created:    2021-09-11 14:17:13
#   Modified:   2021-09-11 14:42:01
##

n, m = map(int, input().split())

h_t = dict()
t_h = dict()
t_s = set()
strings = dict()
for i in range(n):
    i += 1
```

Отправить

Предыдущая

```
        # update set
        t_s.remove(h_t[h])
        t_s.add(t)
        # update dict
        del t_h[h_t[h]]
        t_h[t] = h
        h_t[h] = t
    else:
        if len(t_s) == m:
            rm = min(t_s)
            if rm > t:
                continue
            print(i, "DELETE", strings[t_h[rm]])
            print(i, "PUT", req)
```