NAUMEN

Тестовое задание

Стажер Java-разработчик

Кэширующий сервер

Владимир работает в крупной компании, одно из направлений которой обработка больших данных. Данные находятся в распределённой системе. Количество уникальных данных в системе ограничено и каждый экземпляр данных имеет свой идентификатор (номер). Клиенты этой компании часто запрашивают эти данные и, так как время, затраченное на получение данных достаточно велико, то для оптимизации обращений Владимиру поручено написать middlware сервер (сервер посредник), через который будут проходить запросы данных. Распределённая система хранит огромное количество данных, поэтому все данные не могут поместиться на сервер полностью. Но сервер может кэшировать результаты запросов к распределённой системе и для этого выделена память, достаточная для хранения не более чем N запросов. В связи с тем, что запросы от клиентов проходят через наш сервер, клиент не может получить данных напрямую из распределённой системы. Это значит, что в любом случае данные должны быть загружены на middlware сервер.

К счастью Владимира, оказалось, что самые крупные и значимые клиенты всегда обращаются за одними и теми же данными в одной и той же последовательности, так что у него есть заранее определённый чёткий порядок запросов. Помогите Владимиру придумать алгоритм, чтобы как можно больше данных было получено из кэша сервера, без обращения к распределённой системе.

Для упрощения задачи считаем, что все полученные с помощью запроса данные занимают одинаковое количество байт, поэтому объём памяти для данных можно вынести из рассмотрения и оставить только количество кэшируемых запросов. Обращение к распределённой системе реализовывать не нужно, как и оформление программы в виде сервера, нужен только алгоритм в виде консольного приложения.

Требования к работе алгоритма (желательно)

- Ограничение по времени: 3 секунды
- Ограничение по памяти: 64 мегабайта

Формат входных данных

- Имя входного файла: input.txt
- На вход программы подаётся максимальное количество запросов $1 \le N \le 100\,000$, которое может быть закэшировано на сервере, количество запросов $1 \le M \le 100\,000$ и ровно M запросов с идентификаторами $0 \le R_i \le (2^{63}-1)$. Количество различных номеров запросов ограничено и не превосходит 100 000.

Формат выходных данных

- Имя выходного файла: output.txt
- Требуется вывести одно число: сколько раз пришлось обратиться к распределённой системе за данными, отсутствующими в кэше. В начале работы кэш пуст.

Пример

input.txt	output.txt
5 15 3 1 4 1 5 9 2 6 5 3 5 8 7 9 3	9

Комментарий к примеру

В первой строке файла input.txt содержится максимальное количество запросов (5), кэшируемых на сервере и количество запросов (15), разделенные пробелом.

В примере первые 3 запроса (номера 3, 1, 4) приведут к обращению к распределённой системе, так как их нет в кэше. Запрос с номером 1 есть в кэше, значит обращения к распределённой системе не будет. Запросы номеров 5 и 9 добавят их в кэш. Следующий запрос с номером 2 в кэше отсутствует, заменим номер 1 на номер 2 (так как у нас есть информация о будущих запросах, то мы видим, что запрос 1 больше не повторится и нет смысла хранить его дальше). Потом запрос номера 6 заменит неиспользуемый далее номер 2. Следующие 3 запроса будут изъяты из кэша (5, 3, 5). Далее произойдёт ещё 2 замены – 8 и 7. Итого 9 обращений к распределённой системе.

Файлы с решением необходимо разместить на облачном хранилище (например Google Диск) и прикрепить ссылку к анкете на стажировку. Перед отправкой проверь доступ по ссылке и не размещай файлы в открытом доступе.
Желаем удачи!