**Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики**

**Алгоритмы и структуры данных**

**Лабораторная работа №6**

**Хеширование**

Выполнил:

Бараканов Жаргал Мырзабекович

Факультет ИКТ

Группа K3121

Преподаватель:

Харьковская Татьяна Александровна

**Санкт-Петербург**

**19.12.2021**

**Задание 1.**

Реализуйте множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

* Формат входного файла (input.txt). В первой строке входного файла находится строго положительное целое число операций N, не превышающее 5 · 105. В каждой из последующих N строк находится одна из следующих операций:

– A x – добавить элемент x в множество. Если элемент уже есть в множестве, то ничего делать не надо;

– D x – удалить элемент x. Если элемента x нет, то ничего делать не надо;

– ? x – если ключ x есть в множестве, выведите «Y», если нет, то выведите «N». Аргументы указанных выше операций – целые числа, не превышающие по модулю 1018;

* Формат выходного файла (output.txt). Выведите последовательно результат выполнения всех операций «?»;
* Ограничение по времени. 2 сек.;
* Ограничение по памяти. 256 Мб

*Решение:*

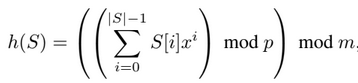
Изображение выглядит как текст

Автоматически созданное описание

**Задание 3.**

В этой задаче вы реализуете хеш-таблицу, используя схему цепочки. Хеширование с цепочками - один из самых популярных способов реализации хеш-таблиц на практике. Хеш-таблицу, которую вы создадите, можно использовать для реализации телефонной книги на вашем телефоне или для хранения таблицы паролей вашего компьютера или веб-службы (но не забывайте хранить хэши паролей вместо самих паролей, иначе вас могут взломать!).

В этой задаче ваша цель - реализовать хеш-таблицу с цепочкой списков. Вам дано количество сегментов (карманов) m и хеш-функция. Это полиномиальная хеш-функция



в которой S[i] - код ASCII i-го символа строки S, p = 1000000007 и x = 263. Ваш алгоритм должен поддерживать следующие типы запросов:

* add string – вставить строку string в таблицу. Если такая строка уже есть в хеш-таблице, то просто игнорируйте запрос;
* del string – удалить строку string из таблицы. Если такой строки нет в хэш-таблице, тогда просто игнорируйте запрос;
* find string – выведите «yes» или «no» (без кавычек) в зависимости от того, содержит ли таблица строку или нет;
* check i – вывести содержимое i-го списка в таблицу. Используйте пробелы для разделения элементов списка. Если i-й список пуст, вывести пустую строку. При вставке новой строки в цепочку вы должны вставить ее в начало цепочки.
* Формат ввода / входного файла (input.txt). В первой строке находится единственное целое число m - количество сегментов хэш-таблицы. Следующая строка содержит число запросов N, после которой идут еще N строк, каждая содержит один запрос в формате, заданном выше. Ограничения: 1 ≤ N ≤ 105, N/5 ≤ m ≤ N. Все строки состоят из латинских букв. Каждая из них не пустая и имеет длину не более 15 символов;
* Формат вывода / выходного файла (output.txt). Выведите результат каждого запроса find и check, по одному результату в строке, в том же порядке, в каком эти запросы указаны во входных данных;
* Ограничение по времени. 7 сек.;
* Ограничение по памяти. 512 мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, табличка, снимок экрана

Автоматически созданное описание

**Задание 4.**

Реализуйте прошитый ассоциативный массив. Ваш алгоритм должен поддерживать следующие типы операций:

* get x – если ключ x есть в множестве, выведите соответствующее ему значение, если нет, то выведите <none>
* prev x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до x, или <none>, если такого нет или в массиве нет x
* next x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после x, или <none>, если такого нет или в массиве нет x
* put x y – поставить в соответствие ключу x значение y. При этом следует учесть, что
  + – eсли, независимо от предыстории, этого ключа на момент вставки в массиве не было, то он считается только что вставленным и оказывается самым последним среди добавленных элементов – то есть, вызов next с этим же ключом сразу после выполнения текущей операции put должен вернуть <none>;
  + – если этот ключ уже есть в массиве, то значение необходимо изменить, и в этом случае ключ не считается вставленным еще раз, то есть, не меняет своего положения в порядке добавленных элементов
* delete x – удалить ключ x. Если ключа в ассоциативном массиве нет, то ничего делать не надо.
* Формат входного файла (input.txt). В первой строке входного файла находится строго положительное целое число операций N, не превышающее 5 · 105. В каждой из последующих N строк находится одна из приведенных выше операций. Ключи и значения операций - строки из латинских букв длиной не менее одного и не более 20 символов
* Формат выходного файла (output.txt). Выведите последовательно результат выполнения всех операций get, prev, next. Следуйте формату выходного файла из примера
* Ограничение по времени. 4 сек.
* Ограничение по памяти. 256 Мб.

*Решение:*

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание