

# Алгоритмы и структуры данных

## Задание к лабораторной работе №6. Хеширование. Хеш-таблицы

Лабораторная работа посвящена ознакомлению с такими понятиями, как множество, словари, хеш-таблицы и хеш-функции. Аналогично предыдущим лабораторным работам, есть **два** способа ее выполнения и защиты, однако присутствуют изменения:

Читать внимательно: изменения (отсутствие вариантов)!

1 **Базовый уровень.** Решается 3 задачи, причем должно выполняться два условия:

- Все три задачи не могут стоять **подряд** в задании. Т.е. решать набор (1,2,3) - нельзя, (1,2,4) - можно
- Номер одной из задач находится по следующей хэш-функции:

$$H(v) = (A \cdot v \bmod p) \bmod 9,$$

где  $A$  - последние 2 числа номера вашей группы,  $v$  - ваш номер в списке группы (вариант),  $p$  - следующее простое число, которое больше чем количество человек в вашей группе.

- (опционально) Номер второй задачи можете найти по принципу:

$$H(v) = ((A \cdot v + B) \bmod p) \bmod 9,$$

где  $B$  - это сумма кодов ASCII всех букв вашей фамилии ☺

В этом случае максимум за защиту можно получить 4 балла. В сумме с самой работой (0,5 балла) и отчетом (1 балл) получается 5,5 балла, что достаточно для зачета.

2 **Продвинутый уровень.** Решаются 5 задач или больше, причем принципы, описанные выше для базового случая тоже учитываются. В этом случае вы сможете получить максимальные 7,5 баллов.

## 1 задача. Множество

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  $5 \cdot 10^5$ . 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  $10^{18}$ .

- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
8	Y
A 2	N
A 5	N
A 3	
? 2	
? 4	
A 2	
D 2	
? 2	

- Примечание.

Эту задачу можно решить совершенно разными способами, включая использование различных средств стандартных библиотек (правда, не всех - в стандартных библиотеках некоторых языков программирования используются слишком предсказуемые методы хеширования). Именно по этой причине ее разумно использовать для проверки реализаций хеш-таблиц, которые понадобятся в следующих задачах этой работы.

## 2 задача. Телефонная книга

В этой задаче ваша цель - реализовать простой менеджер телефонной книги. Он должен уметь обрабатывать следующие типы пользовательских запросов:

- `add number name` – это команда означает, что пользователь добавляет в телефонную книгу человека с именем `name` и номером телефона `number`. Если пользователь с таким номером уже существует, то ваш менеджер должен перезаписать соответствующее имя.
- `del number` – означает, что менеджер должен удалить человека с номером из телефонной книги. Если такого человека нет, то он должен просто игнорировать запрос.
- `find number` – означает, что пользователь ищет человека с номером телефона `number`. Менеджер должен ответить соответствующим именем или строкой «not found» (без кавычек), если такого человека в книге нет.

- **Формат ввода / входного файла (input.txt).** В первой строке входного файла содержится число  $N$  ( $1 \leq N \leq 10^5$ ) - количество запросов. Далее следуют  $N$  строк, каждая из которых содержит один запрос в формате, описанном выше.

Все номера телефонов состоят из десятичных цифр, в них нет нулей в начале номера, и каждый состоит не более чем из 7 цифр. Все имена представляют собой непустые строки из латинских букв, каждая из которых имеет длину не более 15. Гарантируется при проверке, что не будет человека с именем «not found».

- **Формат вывода / выходного файла (output.txt).** Выведите результат каждого поискового запроса `find` – имя, соответствующее номеру телефона, или «not found» (без кавычек), если в телефонной книге нет человека с таким номером телефона. Выведите по одному результату в каждой строке в том же порядке, как были заданы запросы типа `find` во входных данных.
- Ограничение по времени. 6 сек.
- Ограничение по памяти. 512 мб.
- Примеры:

1:	<b>input.txt</b>	
	12	
	add 911 police	
	add 76213 Mom	
	add 17239 Bob	
	find 76213	
	find 910	
	find 911	
	del 910	
	del 911	
2:	<b>input.txt</b>	
	8	
	find 3839442	
	add 123456 me	
	<b>output.txt</b>	
	not found	
	granny	
	me	
	not found	

Описание примера 1. 76213 - это номер Mom, 910 - нет в телефонной книге, 911 - это номер police, но затем он был удален из телефонной книги, поэтому второй поиск 911 вернул «not found». Также обратите внимание, что когда daddy был добавлен с тем же номером телефона 76213, что и номер телефона Mom, имя контакта было переписано, и теперь поиск 76213 возвращает «daddy» вместо «Mom».

### 3 задача. Хеширование с цепочками

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

$$h(S) = \left( \left( \sum_{i=0}^{|S|-1} S[i]x^i \right) \bmod p \right) \bmod m,$$

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 хэш-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:  $1 \leq N \leq 10^5$ ,  $N/5 \leq m \leq N$ . 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.
- Примеры:

1:	<b>input.txt</b>	2:	<b>input.txt</b>	3:	<b>input.txt</b>
	5		4		3
	12		8		12
	add world		add test		check 0
	add Hello		add test		find help
	check 4		find test		add help
	find World		del test		add del
	find world		find test		add add
	del world		del test		find add
	check 4		find test		find del
	del Hello		find Test		del del
	add luck		add Test		find del
	add Good		find Test		check 0
	check 2				check 1
	del good				check 2
	<b>output.txt</b>		<b>output.txt</b>		<b>output.txt</b>
	Hello world		yes		no
	no		no		yes
	yes		no		yes
	Hello		yes		no
	Good luck				add help

- Описание примера №1. Код ASCII для «w» - 119, для «o» - 111, для «г» - 114, для «l» - 108, а для «d» - 100. Таким образом,  $h(\text{«world»}) = (119 + 111 \cdot 263 + 114 \cdot 263^2 + 108 \cdot 263^3 + 100 \cdot 263^4 \bmod 1000000007) \bmod 5 = 4$ . Также хеш-значение «Hello» равно 4. Напомним, что мы всегда вставляем в начало цепочки, поэтому после добавления «world», а затем «Hello» в ту же цепочку с номером 4 сначала идет «Hello», а затем «world».

Далее, строка «World» не найдена, а строка «world» найдена, потому что строки чувствительны к регистру, а коды «W» и «w» различны. После удаления «world» в цепочке 4 будет найдено только «Hello». Наконец после добавления «luck» и «Good» к одной цепочке 2 сначала идет «Good», а затем «luck».

#### 4 задача. Прошитый ассоциативный массив

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
  - если, 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  $5 \cdot 10^5$ . 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.
- Примеры:

input.txt	output.txt
14	c
put zero a	b
put one b	d
put two c	c
put three d	a
put four e	e
get two	<none>
prev two	
next two	
delete one	
delete three	
get two	
prev two	
next two	
next four	

- P.s. Задача на [openedu](#), 8 неделя.

## 5 задача. Выборы в США

Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов — число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов. Вам известно за кого проголосовал каждый штат и сколько голосов было отдано данным штатом. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов.

- **Формат ввода / входного файла (input.txt).** Каждая строка входного файла содержит фамилию кандидата, за которого отдают голоса выборщики этого штата, затем через пробел идет количество выборщиков, отдавших голоса за этого кандидата.
- **Формат вывода / выходного файла (output.txt).** Выведите фамилии всех кандидатов в *лексикографическом* порядке, затем, через пробел, количество отданных за них голосов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.
- Примеры:



№	input.txt	output.txt
1	McCain 10 McCain 5 Obama 9 Obama 8 McCain 1	McCain 16 Obama 17

№	input.txt	output.txt
2	ivanov 100 ivanov 500 ivanov 300 petr 70 tourist 1 tourist 2	ivanov 900 petr 70 tourist 3

№	input.txt	output.txt
3	bur 1	bur 1

## 6 задача. Фибоначчи возвращается

Вам дается последовательность чисел. Для каждого числа определите, является ли оно числом Фибоначчи. Напомним, что числа Фибоначчи определяются, например, так:

$$\begin{aligned}
 F_0 &= F_1 = 1 \\
 F_i &= F_{i-1} + F_{i-2} \text{ для } i \geq 2.
 \end{aligned}
 \tag{1}$$

- **Формат ввода / входного файла (input.txt).** Первая строка содержит одно число  $N$  ( $1 \leq N \leq 10^6$ ) - количество запросов. Следующие  $N$  строк содержат по одному целому числу. При этом соблюдаются следующие ограничения при проверке:
  1. Размер каждого числа не превосходит 5000 цифр в десятичном представлении.
  2. Размер входа не превышает 1 Мб.
- **Формат вывода / выходного файла (output.txt).** Для каждого числа, данного во входном файле, выведите «Yes», если оно является числом Фибоначчи, и «No» в противном случае.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 128 мб. **Внимание:** есть вероятность превышения по памяти, т.к. сами по себе числа Фибоначчи большие. Делайте проверку на память!
- Примеры:

input.txt	output.txt
8	Yes
1	Yes
2	Yes
3	No
4	Yes
5	No
6	No
7	Yes
8	

## 7 задача. Драгоценные камни

В одной далекой восточной стране до сих пор по пустыням ходят караваны верблюдов, с помощью которых купцы перевозят пряности, драгоценности и дорогие ткани. Разумеется, основная цель купцов состоит в том, чтобы подороже продать имеющийся у них товар. Недавно один из караванов прибыл во дворец одного могущественного шаха.

Купцы хотят продать шаху  $n$  драгоценных камней, которые они привезли с собой. Для этого они выкладывают их перед шахом в ряд, после чего шах оценивает эти камни и принимает решение о том, купит он их или нет. Видов драгоценных камней на Востоке известно не очень много всего 26, поэтому мы будем обозначать виды камней с помощью строчных букв латинского алфавита. Шах обычно оценивает камни следующим образом. Он заранее определил несколько упорядоченных пар типов камней:  $(a_1, b_1), (a_2, b_2), \dots, (a_k, b_k)$ . Эти пары он называет красивыми, их множество мы обозначим как  $P$ . Теперь представим ряд камней, которые продают купцы, в виде строки  $S$  длины  $n$  из строчных букв латинского алфавита. Шах считает число таких пар  $(i, j)$ , что  $1 \leq i < j \leq n$ , а камни  $S_i$  и  $S_j$  образуют красивую пару, то есть существует такое число  $1 \leq q \leq k$ , что  $S_i = a_q$  и  $S_j = b_q$ .

Если число таких пар оказывается достаточно большим, то шах покупает все камни. Однако в этот раз купцы привезли настолько много камней, что шах не может посчитать это число. Поэтому он вызвал своего визиря и поручил ему этот подсчет. Напишите программу, которая находит ответ на эту задачу.

- **Формат ввода / входного файла (input.txt).** Первая строка входного файла содержит целые числа  $n$  и  $k$  ( $1 \leq n \leq 100000, 1 \leq k \leq 676$ ) – число камней, которые привезли купцы и число пар, которые шах считает красивыми. Вторая строка входного файла содержит строку  $S$ , описывающую типы камней, которые привезли купцы.

Далее следуют  $k$  строк, каждая из которых содержит две строчных буквы латинского алфавита и описывает одну из красивых пар камней.

- **Формат вывода / выходного файла (output.txt).** В выходной файл выведите ответ на задачу – количество пар, которое должен найти визирь.
- Ограничение по времени. 1 сек.

- Ограничение по памяти. 64 мб.

- Примеры:

№	input.txt	output.txt
1	7 1 abacaba aa	6
2	7 3 abacaba ab ac bb	7

## 8 задача. Почти интерактивная хеш-таблица

В данной задаче у Вас не будет проблем ни с вводом, ни с выводом. Просто реализуйте быструю хеш-таблицу.

В этой хеш-таблице будут храниться целые числа из диапазона  $[0; 10^{15} - 1]$ . Требуется поддерживать добавление числа  $x$  и проверку того, есть ли в таблице число  $x$ . Числа, с которыми будет работать таблица, генерируются следующим образом. Пусть имеется четыре целых числа  $N, X, A, B$  такие что:

- $1 \leq N \leq 10^7$
- $1 \leq X \leq 10^{15}$
- $1 \leq A \leq 10^3$
- $1 \leq B \leq 10^{15}$

Требуется  $N$  раз выполнить следующую последовательность операций:

- Если  $X$  содержится в таблице, то установить  $A \leftarrow (A + A_C) \bmod 10^3$ ,  $B \leftarrow (B + B_C) \bmod 10^{15}$ .
- Если  $X$  не содержится в таблице, то добавить  $X$  в таблицу и установить  $A \leftarrow (A + A_D) \bmod 10^3$ ,  $B \leftarrow (B + B_D) \bmod 10^{15}$ .
- Установить  $X \leftarrow (X \cdot A + B) \bmod 10^{15}$ .

Начальные значения  $X, A$  и  $B$ , а также  $N, A_C, B_C, A_D$  и  $B_D$  даны во входном файле. Выведите значения  $X, A$  и  $B$  после окончания работы.

- **Формат входного файла (input.txt).** В первой строке входного файла содержится четыре целых числа  $N, X, A, B$ . Во второй строке содержится еще четыре целых числа  $A_C, B_C, A_D$  и  $B_D$  такие что  $0 \leq A_C, A_D < 10^3$ ,  $0 \leq B_C, B_D < 10^{15}$ .
- **Формат выходного файла (output.txt).** Выведите значения  $X, A$  и  $B$  после окончания работы.

- Ограничение по времени. 5 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

input.txt
4 0 0 0
1 1 0 0
output.txt
3 1 1

## 9 задача. Убийца хешей

При хешировании строк широко используются полиномиальные хеши. Код, вычисляющий полиномиальный хеш, приведен на листинге ниже. В этом коде `int` - это 32-битное знаковое число. Множитель `multiple` обычно либо жестко фиксируется, либо генерируется каким-либо образом один раз на весь сеанс работы со строками.

```
int multiple = ???;
public int hashOf(String s) {
    int rv = 0;
    for (int i = 0; i < s.length(); ++i) {
        rv = multiple * rv + s.charAt(i);
    }
    return rv;
}
```

Вычисление полиномиальных хешей можно производить эффективно (и даже параллельно), кроме того, они могут быть эффективно пересчитаны при добавлении и удалении символов (в начало, в конец и даже в произвольное место строки), а также при конкатенации и разрыве строк.

Однако у полиномиальных хешей есть и недостаток: для них легко подобрать строки с совпадающими хешами. Ваша задача – сделать это для *всех множителей* от 2 до 1023 включительно. Сгенерируйте  $N$  строк таким образом, чтобы для каждого из этих множителей хеши *всех* этих строк совпадали.

Более формально, пусть  $H_m(S)$  – полиномиальный хеш строки  $S$  с множителем  $m$ . Требуется сгенерировать набор из  $N$  строк  $S_1, S_2, \dots, S_N$ , такой что для всех  $2 \leq m \leq 1023$  выполнялось  $H_m(S_1) = H_m(S_2) = \dots = H_m(S_N)$ .

- **Формат входного файла (input.txt).** Входной файл содержит одно целое число  $N$ ,  $1 \leq N \leq 10^4$ .
- **Формат выходного файла (output.txt).** Выведите  $N$  строк. Строки должны быть непусты, должны состоять из строчных латинских букв. Длина каждой из строк не должна превосходить 2500. **Все строки должны быть различными!**
- Ограничение по времени. 2 сек.

- Ограничение по памяти. 256 мб.

- Примеры:

input.txt	output.txt
1	hello