**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №8 (Week 8 Openedu)

Студент Дунаев Алексей Игоревич

Группа P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 Множество 3](#_Toc6900769)

[Исходный код к задаче 1 3](#_Toc6900770)

[Бенчмарк к задаче 1 4](#_Toc6900771)

[Задача 2. Прошитый ассоциативный массив 5](#_Toc6900772)

[Исходный код к задаче 2 6](#_Toc6900773)

[Бенчмарк к задаче 2 7](#_Toc6900774)

[Задача 3 Почти интерактивная хеш-таблица 9](#_Toc6900775)

[Исходный код к задаче 3 10](#_Toc6900776)

[Бенчмарк к задаче 3 10](#_Toc6900777)

# Задача 1 Множество

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

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

#### Формат входного файла

В первой строке входного файла находится строго положительное целое число операций N, не превышающее . 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 . 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 .

#### Формат выходного файла

Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.

#### Пример

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

## Исходный код к задаче 1

#include **<vector>**

#include **<iostream>**

#include **<algorithm>**

#include **<unordered\_set>**

**using namespace** std;

*/\*\*/*

#include **"edx-io.hpp"**

#define **cout** io

#define **cin** io

*/\*\*/*

**int** main() {

**int** N;

**cin** >> N;

**char** action;

**long long** key, temp;

unordered\_set<**long long**> set;

**for** (**int** i = 0; i < N; i++) {

**cin** >> action;

**cin** >> key;

**switch** (action)

{

**case 'A'**:

set.insert(key);

**break**;

**case 'D'**:

set.erase(key);

**break**;

**case '?'**:

**if** (set.find(key) == set.end())

cout << **"N\n"**;

**else**

cout << **"Y\n"**;

**break**;

**default**:

**break**;

}

}

**return** 0;

}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.328 | 38440960 | 11189636 | 501237 |
| 1 | OK | 0.000 | 3506176 | 43 | 9 |
| 2 | OK | 0.031 | 3489792 | 8 | 3 |
| 3 | OK | 0.015 | 3497984 | 51 | 12 |
| 4 | OK | 0.031 | 3473408 | 542 | 99 |
| 5 | OK | 0.015 | 3485696 | 618 | 54 |
| 6 | OK | 0.046 | 3518464 | 5451 | 1038 |
| 7 | OK | 0.046 | 3489792 | 6436 | 957 |
| 8 | OK | 0.015 | 3497984 | 13382 | 957 |
| 9 | OK | 0.000 | 3481600 | 22394 | 981 |
| 10 | OK | 0.000 | 3522560 | 7030 | 465 |
| 11 | OK | 0.000 | 3543040 | 7020 | 411 |
| 12 | OK | 0.015 | 3502080 | 63829 | 10002 |
| 13 | OK | 0.000 | 3784704 | 80339 | 4947 |
| 14 | OK | 0.015 | 3788800 | 80203 | 5034 |
| 15 | OK | 0.031 | 3993600 | 545113 | 100323 |
| 16 | OK | 0.015 | 4091904 | 639485 | 99282 |
| 17 | OK | 0.062 | 4214784 | 738870 | 99558 |
| 18 | OK | 0.031 | 6328320 | 1338668 | 99636 |
| 19 | OK | 0.031 | 7208960 | 2237627 | 99540 |
| 20 | OK | 0.031 | 6455296 | 903052 | 50202 |
| 21 | OK | 0.031 | 6467584 | 902843 | 49536 |
| 22 | OK | 0.062 | 6168576 | 2725205 | 501237 |
| 23 | OK | 0.062 | 6651904 | 3196877 | 499713 |
| 24 | OK | 0.062 | 7196672 | 3694712 | 501051 |
| 25 | OK | 0.140 | 17367040 | 6694340 | 500355 |
| 26 | OK | 0.171 | 21897216 | 11189636 | 500040 |
| 27 | OK | 0.140 | 20041728 | 4902931 | 249012 |
| 28 | OK | 0.156 | 20045824 | 4902757 | 250305 |
| 29 | OK | 0.187 | 29462528 | 9687139 | 300000 |
| 30 | OK | 0.171 | 29417472 | 9687570 | 300000 |
| 31 | OK | 0.140 | 27717632 | 8000008 | 300000 |
| 32 | OK | 0.328 | 38440960 | 11000008 | 150000 |

# Задача 2. Прошитый ассоциативный массив

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | **3 секунды** |
| Ограничение по памяти: | 256 мегабайт |

Реализуйте прошитый ассоциативный массив.

#### Формат входного файла

В первой строке входного файла находится строго положительное целое число операций N, не превышающее . В каждой из последующих N строк находится одна из следующих 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 — поставить в соответствие ключу значение x. 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 . Если ключа x в ассоциативном массиве нет, то ничего делать не надо.

Ключи и значения — строки из латинских букв длиной не менее одного и не более 20 символов.

#### Формат выходного файла

Выведите последовательно результат выполнения всех операций get, prev, next. Следуйте формату выходного файла из примера.

#### Пример

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

## Исходный код к задаче 2

#include **<vector>**

#include **<iostream>**

#include **<algorithm>**

#include **<string>**

#include **<unordered\_map>**

#include **<list>**

**using namespace** std;

*/\*\*/*

#include **"edx-io.hpp"**

#define **cout** io

#define **cin** io

*/\*\*/*

**int** main() {

**int** N;

**cin** >> N;

string action;

*// string\* array = new string[500000];//****TODO N***

list<string> li;

string key, val;

unordered\_map<string, list<string>::iterator> set;

**for** (**int** i = 0; i < N; i++) {

**cin** >> action;

**cin** >> key;

**if** (action == **"get"**){

**if** (set.find(key) == set.end()) {

**cout** << **"<none>\n"**;

}

**else** {

**cout** << \*set[key] << **"\n"**;

}

}

**else if** (action == **"prev"** ) {

**if** (set.find(key) == set.end() || set[key] == li.begin()) {

**cout** << **"<none>\n"**;

}

**else** {

**cout** << \*(prev(set[key])) << **"\n"**;

}

}

**else if** (action == **"next"**) {

**if** (set.find(key) == set.end() || next(set[key]) == li.end()) {

**cout** << **"<none>\n"**;

}

**else** {

**cout** << \*(next(set[key])) << **"\n"**;

}

}

**else if** (action == **"put"**) {

**cin** >> val;

**if** (set.find(key) == set.end()) {

li.push\_back(val);

set[key] = (--li.end());

}

**else** {

(\*set[key]) = val;

}

}

**else if** (action == **"delete"**) {

**if** (set.find(key) == set.end()) {

}

**else** {

**auto** it = set[key];

set.erase(key);

li.erase(it);

}

}

**else**{

**cout** << action << **"\n"**;

}

}

**return** 0;

}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.265 | 169209856 | 23499808 | 10303658 |
| 1 | OK | 0.031 | 10846208 | 158 | 26 |
| 2 | OK | 0.031 | 10715136 | 12 | 8 |
| 3 | OK | 0.031 | 10825728 | 25 | 5 |
| 4 | OK | 0.046 | 10813440 | 25 | 8 |
| 5 | OK | 0.031 | 10846208 | 82 | 20 |
| 6 | OK | 0.046 | 10833920 | 1200 | 504 |
| 7 | OK | 0.031 | 10862592 | 1562 | 564 |
| 8 | OK | 0.031 | 11096064 | 12204 | 4617 |
| 9 | OK | 0.031 | 11116544 | 12058 | 4340 |
| 10 | OK | 0.078 | 11976704 | 960183 | 395964 |
| 11 | OK | 0.078 | 12066816 | 1318345 | 765350 |
| 12 | OK | 0.078 | 11988992 | 1420595 | 880052 |
| 13 | OK | 0.093 | 12083200 | 1079934 | 395020 |
| 14 | OK | 0.062 | 12070912 | 840022 | 332970 |
| 15 | OK | 0.109 | 12079104 | 1223121 | 889998 |
| 16 | OK | 0.109 | 21929984 | 3120970 | 486100 |
| 17 | OK | 0.125 | 21987328 | 3123298 | 486652 |
| 18 | OK | 0.109 | 21999616 | 3122193 | 479024 |
| 19 | OK | 0.078 | 12062720 | 900630 | 420456 |
| 20 | OK | 0.109 | 21950464 | 3121195 | 486718 |
| 21 | OK | 0.234 | 44470272 | 4199992 | 8 |
| 22 | OK | 0.234 | 44621824 | 4099993 | 8 |
| 23 | OK | 0.234 | 44867584 | 3999994 | 8 |
| 24 | OK | 0.250 | 44847104 | 3899995 | 8 |
| 25 | OK | 0.218 | 43405312 | 3799996 | 8 |
| 26 | OK | 0.218 | 42774528 | 3699997 | 8 |
| 27 | OK | 0.234 | 42168320 | 3599998 | 8 |
| 28 | OK | 0.234 | 42090496 | 3499999 | 8 |
| 29 | OK | 0.234 | 41189376 | 3400000 | 8 |
| 30 | OK | 0.218 | 40542208 | 3300001 | 8 |
| 31 | OK | 0.218 | 12115968 | 5399043 | 1973124 |
| 32 | OK | 0.218 | 12029952 | 4200443 | 1669405 |
| 33 | OK | 0.250 | 12017664 | 6099290 | 4429770 |
| 34 | OK | 0.546 | 41742336 | 15598672 | 2589784 |
| 35 | OK | 0.546 | 41619456 | 15589269 | 2586758 |
| 36 | OK | 0.500 | 42786816 | 15603830 | 2398360 |
| 37 | OK | 0.250 | 12054528 | 4499616 | 2110630 |
| 38 | OK | 0.531 | 41836544 | 15603381 | 2583188 |
| 39 | OK | 1.234 | 161107968 | 20999992 | 8 |
| 40 | OK | 1.234 | 159936512 | 20499993 | 8 |
| 41 | OK | 1.265 | 160509952 | 19999994 | 8 |
| 42 | OK | 1.218 | 160624640 | 19499995 | 8 |
| 43 | OK | 1.218 | 151867392 | 18999996 | 8 |
| 44 | OK | 1.218 | 153067520 | 18499997 | 8 |
| 45 | OK | 1.234 | 153944064 | 17999998 | 8 |
| 46 | OK | 1.203 | 153821184 | 17499999 | 8 |
| 47 | OK | 1.187 | 145936384 | 17000000 | 8 |
| 48 | OK | 1.187 | 144818176 | 16500001 | 8 |
| 49 | OK | 0.937 | 92385280 | 18500008 | 5499986 |
| 50 | OK | 1.250 | 169181184 | 23499808 | 220 |
| 51 | OK | 0.375 | 12038144 | 13500208 | 10303658 |
| 52 | OK | 0.703 | 47656960 | 15500008 | 8799944 |
| 53 | OK | 1.203 | 148049920 | 21500008 | 2200000 |
| 54 | OK | 0.937 | 92753920 | 18500008 | 5500000 |
| 55 | OK | 1.250 | 169209856 | 23499808 | 220 |
| 56 | OK | 0.406 | 12025856 | 13500208 | 10300130 |
| 57 | OK | 0.718 | 47710208 | 15500008 | 8799958 |
| 58 | OK | 1.234 | 146845696 | 21500008 | 2200000 |

# Задача 3 Почти интерактивная хеш-таблица

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | **5 секунд** |
| Ограничение по памяти: | 256 мегабайт |

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 . 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:

Требуется N раз выполнить следующую последовательность операций:

* Если X содержится в таблице, то установить , A <- (A + Ac) mod , B <- (B + Bc) mod
* Если X не содержится в таблице, то добавить в таблицу X и установить A <- (A + Ad) mod , B <- (B + Bd) mod
* Установить X <- (X \* A + B) mod

Начальные значения X, A и B, а также Ac, Bc, Ad, Bd даны во входном файле. Выведите значения X, A, B после окончания работы.

#### Формат входного файла

Во первой строке входного файла содержится четыре целых числа N, X, A, B. Во второй строке содержится еще четыре целых числа Ac, Bc, Ad и Bd, такие что , , .

#### Формат выходного файла

Выведите значения X, A, B после окончания работы.

#### Пример

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

## Исходный код к задаче 3

#include **<vector>**

#include **<iostream>**

#include **<algorithm>**

#include **<fstream>**

#include **<string>**

#include **<unordered\_map>**

#include **<list>**

**using namespace** std;

*/\*\*/*

#include **"edx-io.hpp"**

#define **cout** io

#define **cin** io

*/\*\*/*

**long** my\_hash(**long long** value, **long** ht\_size) {

**return** abs((value \* 47) ^ (value \* 31)) % ht\_size;

}

**bool** insert(**long long**\*& ht, **long long** value, **long** ht\_size) {

**long** h = my\_hash(value, ht\_size);

**while** (ht[h] != -1 && ht[h] != value) {

**if** (++h == ht\_size) {

h = 0;

}

}

**if** (ht[h] == value) {

**return false**;

}

**else** {

ht[h] = value;

**return true**;

}

}

**int** main() {

**long** N;

**int** A, Ac, Ad;

**long long** X, B, Bc, Bd;

**cin** >> N >> X >> A >> B >> Ac >> Bc >> Ad >> Bd;

**long long**\* ht = **new long long**[N \* 2];

**for** (**long** i = 0; i < N \* 2; i++) {

ht[i] = -1;

}

**for** (**long** i = 0; i < N; i++) {

**if** (insert(ht, X, N \* 2)) {

A = (A + Ad) % 1000;

B = (B + Bd) % 1000000000000000;

}

**else** {

A = (A + Ac) % 1000;

B = (B + Bc) % 1000000000000000;

}

X = (X \* A + B) % 1000000000000000;

}

**cout** << X << **" "** << A << **" "** << B;

**return** 0;

}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 2.187 | 170532864 | 87 | 37 |
| 1 | OK | 0.015 | 10182656 | 18 | 7 |
| 2 | OK | 0.015 | 10141696 | 19 | 7 |
| 3 | OK | 0.031 | 10219520 | 21 | 7 |
| 4 | OK | 0.015 | 10248192 | 21 | 7 |
| 5 | OK | 0.031 | 10186752 | 21 | 7 |
| 6 | OK | 0.015 | 10240000 | 21 | 15 |
| 7 | OK | 0.031 | 10174464 | 21 | 7 |
| 8 | OK | 0.031 | 10268672 | 21 | 9 |
| 9 | OK | 0.046 | 10166272 | 21 | 9 |
| 10 | OK | 0.015 | 10174464 | 30 | 28 |
| 11 | OK | 0.031 | 10190848 | 30 | 28 |
| 12 | OK | 0.031 | 10199040 | 35 | 35 |
| 13 | OK | 0.015 | 10223616 | 47 | 32 |
| 14 | OK | 0.031 | 10178560 | 63 | 35 |
| 15 | OK | 0.031 | 10207232 | 81 | 37 |
| 16 | OK | 0.031 | 10203136 | 82 | 37 |
| 17 | OK | 0.031 | 11816960 | 23 | 7 |
| 18 | OK | 0.031 | 11825152 | 23 | 7 |
| 19 | OK | 0.031 | 11780096 | 23 | 7 |
| 20 | OK | 0.031 | 11927552 | 23 | 21 |
| 21 | OK | 0.046 | 11849728 | 23 | 7 |
| 22 | OK | 0.031 | 11784192 | 23 | 9 |
| 23 | OK | 0.031 | 11784192 | 23 | 9 |
| 24 | OK | 0.031 | 11780096 | 32 | 30 |
| 25 | OK | 0.031 | 11829248 | 32 | 30 |
| 26 | OK | 0.031 | 11784192 | 37 | 35 |
| 27 | OK | 0.046 | 11792384 | 51 | 35 |
| 28 | OK | 0.078 | 11792384 | 64 | 34 |
| 29 | OK | 0.031 | 11825152 | 84 | 37 |
| 30 | OK | 0.046 | 11833344 | 84 | 36 |
| 31 | OK | 0.062 | 26169344 | 24 | 7 |
| 32 | OK | 0.078 | 26218496 | 24 | 7 |
| 33 | OK | 0.078 | 26255360 | 24 | 7 |
| 34 | OK | 0.171 | 26185728 | 24 | 24 |
| 35 | OK | 0.078 | 26189824 | 24 | 7 |
| 36 | OK | 0.062 | 26169344 | 24 | 9 |
| 37 | OK | 0.062 | 26198016 | 24 | 9 |
| 38 | OK | 0.171 | 26181632 | 33 | 16 |
| 39 | OK | 0.140 | 26193920 | 33 | 30 |
| 40 | OK | 0.187 | 26165248 | 38 | 35 |
| 41 | OK | 0.171 | 26222592 | 52 | 35 |
| 42 | OK | 0.187 | 26206208 | 66 | 35 |
| 43 | OK | 0.187 | 26189824 | 84 | 37 |
| 44 | OK | 0.187 | 26173440 | 85 | 37 |
| 45 | OK | 0.500 | 170463232 | 25 | 7 |
| 46 | OK | 0.562 | 170438656 | 25 | 7 |
| 47 | OK | 0.546 | 170450944 | 25 | 7 |
| 48 | OK | 1.968 | 170446848 | 25 | 27 |
| 49 | OK | 0.500 | 170487808 | 25 | 7 |
| 50 | OK | 0.500 | 170446848 | 25 | 9 |
| 51 | OK | 0.500 | 170463232 | 25 | 9 |
| 52 | OK | 1.562 | 170426368 | 34 | 16 |
| 53 | OK | 1.343 | 170504192 | 34 | 30 |
| 54 | OK | 1.968 | 170487808 | 39 | 35 |
| 55 | OK | 1.968 | 170459136 | 51 | 35 |
| 56 | OK | 2.093 | 170450944 | 67 | 35 |
| 57 | OK | 2.015 | 170520576 | 87 | 37 |
| 58 | OK | 2.015 | 170467328 | 87 | 37 |
| 59 | OK | 2.015 | 170438656 | 87 | 35 |
| 60 | OK | 2.015 | 170491904 | 86 | 37 |
| 61 | OK | 2.000 | 170434560 | 87 | 37 |
| 62 | OK | 2.187 | 170455040 | 86 | 37 |
| 63 | OK | 2.015 | 170459136 | 86 | 37 |
| 64 | OK | 2.031 | 170459136 | 86 | 37 |
| 65 | OK | 2.015 | 170491904 | 87 | 37 |
| 66 | OK | 2.000 | 170532864 | 85 | 35 |
| 67 | OK | 2.000 | 170434560 | 85 | 36 |
| 68 | OK | 2.015 | 170450944 | 87 | 36 |